Components of an Automatic Single Document Summarization System in the News Domain

Inaugural-Dissertation

zur

Erlangung des Doktorgrades der

Mathematisch-Naturwissenschaftlichen Fakultät

der Heinrich-Heine-Universität Düsseldorf

vorgelegt von

Pashutan Modaresi Chahardehi

aus Teheran

Februar 2017

Aus dem Institut für Informatik
der Heinrich-Heine Universität Düsseldorf

Gedruckt mit der Genehmigung der
Mathematisch-Naturwissenschaftlichen Fakultät der
Heinrich-Heine-Universität Düsseldorf

Referent: Prof. Dr. Stefan Conrad
Koreferent: Prof. Dr. Markus Kollmann
Tag der mündlichen Prüfung: 11.04.2017

Dedicated to

Sara

Ich versichere an Eides Statt, dass die Dissertation von mir selbstständig und ohne unzulässige fremde Hilfe unter Beachtung der *Grundsätze zur Sicherung guter wissenschaftlicher Praxis an der Heinrich-Heine-Universität Düsseldorf* erstellt worden ist.

Die hier vorgelegte Dissertation habe ich eigenständig und ohne unerlaubte Hilfe angefertigt. Die Dissertation wurde in der vorgelegten oder in ähnlicher Form noch bei keiner anderen Institution eingereicht. Ich habe bisher keine erfolglosen Promotionsversuche unternommen.

Düsseldorf, Deutschland   Pashutan Modaresi Chahardehi
Februar, 2017

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisor Prof. Dr. Conrad for the continuous support for my Ph.D. study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. Besides my advisor, I would like to thank Prof. Dr. Kollmann for accepting to be my second reviewer.

Being an observational learner, I want to thank my colleagues at the University of Düsseldorf, who indirectly (and directly) taught me valuable lessons: Alexander Askinadze (a smart researcher whose ideas inspired me always), Kirill Bogomasov (a fresh researcher I wished I had the chance to work with), Daniel Braun (my dear roommate who tolerated me on a daily basis, and who is obsessed with mountains), Janine Golov (a structural thinker with discipline), Ludmila Himmelspach (a specialist in fuzzy clustering with whom I had very constructive discussions on ethics in machine learning), Matthias Liebeck (an analytical thinker I had the chance to work closely with, and who taught me so many lessons that I can't even count), Magdalena Rischka (a former colleague, who was one of the reasons I got interested in text mining), Julia Romberg (a conceptual thinker with great questions in mind) and Michael Singhof (a computer scientist with the attitudes of a mathematician). I am also greatly indebted to Mani Dardashti and Matthias Liebeck for their proofreading, which has profoundly improved the composition of this thesis.

Furthermore, I want to thank Guido Königstein and Sabine Freese for their excellent technical and administrative support.

My sincere thanks also go to Fady El-Murr (who provided me an opportunity to join their team at pressrelations GmbH) and Philipp Gross (a great mathematician who was always on my side). Moreover, I want to express my warmest thanks to Jens Schmitz, Siavash Sefidroodi, Stefan Bungter, Mirja Eckhoff and Joachim de Bruin from the pressrelations GmbH. Without their valuable support, it would not have been possible to conduct this research.

Finally, and most importantly, I would like to thank my parents (Roya and Ali), my sister (Nakat) and my wife (Sara). I could fill a thousand pages with Thank You, and it would not be enough. You are the meaning of my life.

Düsseldorf, Germany
February, 2017 *Pashutan Modaresi Chahardehi*

Abstract

The increasing amount of available information on the Internet has necessitated the creation of tools and algorithms to automatically manage and summarize them. This problem is even more compelling when dealing with unstructured data such as textual documents. An enormous amount of textual information is created on a daily basis originating from sources such as blogs, Twitter, Facebook and online news. To manage this amount of information, automatic approaches are required to summarize them in a compact form. For this reason, *automatic text summarization* has gained lots of attention by researchers in the field of *natural language processing* and *artificial intelligence*.

Specifically, in the news domain, a large number of companies and organizations are interested in using sophisticated algorithms to summarize news articles automatically. Although the field of automatic text summarization has been investigated by researchers for almost sixty years, there still exists enormous potential to improve existing approaches. In general, the existing summarization approaches are categorized into *single document* (one single document has to be summarized) and *multidocument* (multiple documents have to be summarized). In this work, we focus on the problem of automatic single document text summarization in the news domain, and investigate the several components and elements that we claim to be critical in the design and development of the summarization algorithms.

In our work, we follow a bottom-up approach. We start with an attempt to formally define the problem of automatic text summarization and propose a definition that we use as a guideline in our entire work. Next, we propose approaches to automatically collect training data for summarization algorithms that incorporate machine learning. As a critical component in many summarization systems, we propose a machine learning approach to summarize a textual document in the form of keywords and keyphrases.

Moreover, we propose components to automatically detect and remove redundant sentences in a textual document and order the remaining ones in such a manner that the resulting summary text is linguistically coherent. We also propose automatic and manual approaches to evaluate the quality of the created summaries. Specifically, we present the results of an extensive study in the domain of media monitoring and media responsive analysis and show the impressive financial benefits of incorporating automatic summarization systems.

Another contribution of this work is the attempt to establish a connection between the fields of automatic text summarization and *digital text forensics* where various techniques in the field of digital text forensics such as *author verification*, *author profiling* and *plagiarism alignment detection* will be used to improve the quality of the summaries and assure that the documents and their automatically created summaries obey the same writing style.

Zusammenfassung

Die zunehmende Menge verfügbarer Informationen im Internet erfordert die Schaffung von Werkzeugen und Algorithmen, um diese automatisch zu verwalten und zusammenzufassen. Dieses Problem ist sogar noch größer, wenn es sich um unstrukturierte Daten wie Textdokumente handelt. Die große Menge an Informationen in Textform wird auf einer täglichen Basis aus Quellen wie Blogs, Twitter, Facebook und Online-News erstellt. Um diese immense Menge an Informationen in einer kompakten Form zusammenzufassen, sind automatische Ansätze zur Verwaltung der Informationen erforderlich. Aus diesem Grund hat *automatische Textzusammenfassung* die Aufmerksamkeit vieler Forscher auf dem Gebiet der *natürlichen Sprachverarbeitung* und *künstliche Intelligenz* erregt.

Insbesondere in der Nachrichten-Domäne sind eine große Anzahl von Unternehmen und Organisationen an anspruchsvollen Algorithmen, um Nachrichtenartikel automatisch zusammenzufassen, interessiert. Im Allgemeinen werden die existierenden Ansätze in *Einzeldokument* (ein einziges Dokument muss zusammengefasst werden) und *Multidokument* (mehrere Dokumente müssen zusammengefasst werden) kategorisiert. In dieser Arbeit konzentrieren wir uns auf das Problem der automatischen Einzeldokument-Textzusammenfassung im Nachrichtenbereich und untersuchen verschiedene Komponenten und Elemente, die wir für die Konstruktion und Entwicklung der Zusammenfassungsalgorithmen kritisch beanspruchen.

In unserer Arbeit folgen wir einem Bottom-up-Ansatz. Wir beginnen mit dem Versuch, das Problem der automatischen Textzusammenfassung formal zu definieren und eine Definition vorzuschlagen, die stets als Leitlinie verwendet wird. Als nächstes schlagen wir Ansätze vor, um automatisch Trainingsdaten für Zusammenfassungsalgorithmen zu erfassen, die das maschinelle Lernen verwenden. Als kritische Komponente in vielen Zusammenfassungssystemen schlagen wir einen maschinellen Lernansatz vor, um ein Textdokument in Form von Schlüsselwörtern und Schlagwörtern zusammenzufassen.

Darüber hinaus ziehen wir Komponenten heran, um redundante Sätze in einem Textdokument automatisch zu erkennen und zu entfernen und die übrigen so zu ordnen, dass der resultierende zusammengefasste Text linguistisch kohärent ist. Wir schlagen auch automatische und manuelle Ansätze vor, um die Qualität der erstellten Zusammenfassungen zu bewerten.

In der vorliegenden Arbeit wird ferner versucht, eine Verbindung zwischen den Gebieten der automatischen Textzusammenfassung und der *digitalen Textforensik* herzustellen, wo verschiedene Techniken im Bereich der digitalen Textforensik wie *Authorenverifikation*, *Author Profiling* und *Plagiatserkennung* verwendet werden, um die Qualität der Zusammenfassungen zu verbessern und sicherzustellen, dass die Dokumente und ihre automatisch erstellten Zusammenfassungen demselben Schreibstil folgen.

Components of An Automatic Single Document Summarization System in the News Domain

Pashutan Modaresi Chahardehi

Contents

1
Introduction

We are living in the age of big data, where due to the digitization revolution, the amount of data is rapidly increasing. A considerable amount of the existing data is in an unstructured form that needs to be analyzed and processed. Specifically, textual data are a target to this problem.

The production sources of textual data are enormous: Wikipedia[1], Twitter[2], blogs, News websites, and Facebook[3] are some examples to be named. A possible and intuitive solution to manage this huge amount of data is to summarize them. Summarization is indeed a solution that is applied in many disciplines and workflows to manage the amount of information and ease the communication between the people. Summaries emphasize the most important aspects of a text and save us a tremendous amount of time by perceiving, processing and communicating the information.

Although manual summarization of the information is an easy task for humans, it is not scalable. As already mentioned, due to the rapid increase of data, it is not feasible to summarize the available information manually. At the same time, parallel to the growth of data, the digital revolution requires us to access and perceive more and more information to be competitive in the society. As in other disciplines, a natural solution would be to automate the summarization process.

At first glance, automating the summarization process seems to be a simple solution that can be quickly realized. At least, this was the impression 58 ago as Luhn published his seminal work [Luh58] on automatic summarization of literature abstracts. Meanwhile, Luhn's work has been cited almost 3000 times in other publications, and we have still a long way to go to produce high quality summaries. But, why is automatic

[1]https://www.wikipedia.org/
[2]https://twitter.com/
[3]https://www.facebook.com/

summarization such a challenging task?

As soon as we want to automate a task, we need a (formal) definition of the underlying problem. What exactly is a text summary? Many researchers have attempted to (formally) define a summary or the summarization process. Some define a summary as a reduced form of a textual document that contains the most salient information of the original document [SSJ01], some as an abstraction of the original document containing the topical indicators [Luh58], and some as the reduced form of an original document containing the aspects related to a user query [MC15b]. In Section 2.2, we address this problem. We shortly review the existing definitions of automatic text summarization, investigate their advantages and disadvantages, and propose a new definition of automatic text summarization, which addresses the drawbacks of the previous definitions.

Text summaries can be used in an extrinsic or intrinsic manner. By extrinsic, we refer to applications such as document indexing, text classification or text clustering, where automatic summaries can be used instead of the original documents to reduce computational cost and to increase the performance of the system. By intrinsic, we mean applications in which summaries are used as end products to be delivered to the users. As soon as a user is directly involved in an automation system, the needs of the user have an enormous impact on the design and architecture of the system. Having various users with heterogeneous requirements leads to different summarization systems. A user might need to summarize documents from different information sources (scientific publications, interviews, news, tweets, etc.) or with various cardinalities (single-document, multi-document). The next question that arises is regarding the ultimate intention of the user. Are the summaries used to filter the information (indicative summaries) or will they be used to provide the most relevant information in a concise form (informative summaries)? In this work we focus on the task of single document summarization and in Section 2.1 we will shortly review the available single-document summarization systems designed based on various user requirements. We start with the early works in the 1950s and cover the modern approaches of automatic text summarization in recent years.

The initial approaches in the field of automatic summarization were mainly heuristic-based. Heuristics were calculated to extract the most relevant information of the documents. Later on, machine learning approaches have gained popularity and became the standard way to tackle the summarization problem. The primary intention of using machine learning methods was to learn and mimic the principles incorporated (mostly unconsciously) by humans to summarize textual documents. Machine learning algorithms require annotated data (by human experts) to learn a strategy to mimic the human behavior. The need for high-quality data sets was recognized by researchers which led to the release of several high-quality data sets commonly used in the context of automatic text summarization. The DUC (Document Understanding Conference) and TREC (Text Analysis Conference) conferences have played a significant role in pushing the field of automatic text summarization forward and collecting valuable data sets to train and evaluate the machine learning algorithms. Despite these efforts, data and its quality and size remain one of the critical problems to develop sophisticated summarization systems. In Sections 3.1 and 3.2, we will shortly review the existing corpora in the field of automatic text summarization and introduce approaches to automatically (without any human effort) construct training corpora for machine learning algorithms.

In an intrinsic summarization scenario, the primary goal is usually to reduce the time to read and comprehend the textual documents. However, in the extrinsic scenario, summarization systems are mainly used as an intermediate step in the pipeline to increase the performance or the quality of the system. In both scenarios, the main question is how to store or represent the summaries? In the literature, there exist multiple approaches for representing the summaries. One possibility is to represent the summaries in the form of sentences. This means that a summary text will be a concatenation of multiple sentences extracted or generated from the original document. Moreover, the summaries can also be represented in an abstract notation, probably not easily readable by humans but suitable for the algorithms. Information items [GL11] belong to this category, where significant information are represented by a triple consisting of a subject, an object, and a verb. Another interesting representation method for summaries is the use of keywords and keyphrases.

Using the keywords or keyphrases to represent or store a summary has multiple advantages. In an indexing application, keywords are a highly condensed representation of the documents and lead to faster retrievals and decrease the index size. In classification tasks, extracting the keywords of the documents can be considered as a dimensionality reduction technique to accelerate training and avoid overfitting. Additionally, in an intrinsic summarization system, they can be used to assist the user to gain a rapid understanding of the topics of the underlying documents. Although keyword and keyphrase extraction does not fit the definition of automatic text summarization proposed in this work, they play a major role in the production of quality summaries. In Sections 4.1 and 4.2 we shortly review the existing keyphrase extraction approaches and propose a novel approach based on the fuzzy set theory to extract and rank the keyphrases of a document.

According to their type, automatic summarization systems are typically classified into extractive and abstractive approaches. As the name implies, extractive methods select or extract the sentences from the original document to form the summary, whereas abstractive approaches reformulate the sentences in the original document, and in some sense, *generate* the summaries. For a long time, abstractive text summarization was considered as a challenging task and researchers have mainly focused on extractive approaches. Today, with the popularity of deep learning methods, this is not the case anymore, and the field of abstractive text summarization is gaining more attention.

If we want to summarize the previous research in the field of extractive text summarization, three successive steps become apparent:

1. Preprocessing: this includes all the activities required to gain a clean representation of the textual data to be used by the summarization algorithms. Some common operations in the preprocessing step are tokenization, stopword elimination, stemming, named entity recognition and POS [4] tagging.

2. Content Selection: finding an answer to the question *what should be said?* is the primary intention of this step. Incorporating heuristics, supervised or unsupervised algorithms, strategies have to be defined to identify the salient information in a text or in multiple textual documents.

3. Postprocessing: finding an answer to the question *how should it be said?* is the main intention of this step. Given the outputs of the content selection step, the output format of the summaries has to be determined. Typical operations in this step are text normalization, coreference resolution, and text realization [5].

A critical component in any text summarization system is the content selection. The philosophy behind content selection is to select the important information from a document. Although this approach has been shown to work in practice, it has a major disadvantage: Even a perfect content selection module will result in some information loss. This means that to reduce the amount of text, some information from the original document has to be eliminated. In this work, we approach differently. To create extractive summaries, we do not incorporate a content selection module, but instead a content elimination module. This means that instead of selecting the most important information in a document, we eliminate the less important and redundant information from a document. Although both approaches might seem to be very similar and result in the same summaries, incorporating content elimination instead of content selection has an enormous impact on the design decisions of an automatic summarization system.

To realize a content elimination module, we propose a paraphrase detection algorithm to detect sentences that are semantically similar to each other (Section 5.1). In every extractive summarization approach, selection or elimination of specific sentences may lead to an incoherent text. A typical example is the existence of anaphora, referring to sentences in the original document that do not exist in the summary anymore. To handle this problem, we propose a neural architecture to automatically determine

[4]Part of Speech Tagging

[5]A subtask of natural language generation, which transforms abstract representations to actual texts in natural language

the order of sentences (Section 5.2). In this way, in every sentence elimination step, it can be decided whether the operation could lead to the incoherence of the text or not.

Given an automatically created text summary, a critical question is how to determine the quality of the summary? As in other machine learning applications, appropriate measures have to be defined to assess the quality of the summarization systems. In the case of extractive text summarization, a possible solution is to compare the automatically created summaries with the gold-standard summaries created by humans and report the quality of the system based on their similarities or intersections. This kind of automatic evaluation is indeed widely used in the research community. However, there exist several problems associated with this type of evaluation:

1. Summarization is a subjective task and different people may produce different summaries. Even the same person could produce different summaries at different time points. Comparing the automatically created text summaries with a single gold-standard will lead to a high bias in the evaluation and make the results less reliable. To solve this problem, several gold-standards could be used. Although this is a reasonable solution, the manual creation of summaries is a costly process and it does not scale.

2. In the case of abstractive text summarization, a direct comparison of the automatic and manually created summaries is not reasonable. Abstractive summaries are typically reductive reformulations of the original texts and might not have a common vocabulary. This will lead to lower scores in automatic evaluation metrics that might not reflect the real quality of the summaries.

In Chapter 6 we focus on the automatic and manual evaluation of the text summaries. In Section 6.1 we experiment with the manual and automatic measures for evaluating obfuscation systems. Although the experiment is performed in the obfuscation domain, the findings are transferable to the domain of automatic text summarization. In Section 6.2, we study the problem of evaluating summarization systems from a commercial point of view and state the question whether summarization systems are commercially profitable.

In our work, we also study the relationship between the fields of automatic text summarization and digital text forensics: Given a perfect abstractive summarization system, how can it be assured that the writing style of the original author of the document remains unaffected? Modern abstractive text summarization systems are typically trained on lots of data from various sources and completely ignore the writing style of the authors. We claim that author profiling and author verification are suitable mechanisms to retain the writing style of the authors in the summaries.

In author profiling, the primary goal is to predict the profile (age, sex, personality, etc.) of an author based on her or his written documents. An author profiling module can be used together with a summarization system to ensure that the predicted profile of the original documents and the predicted profile of the summaries are almost identical. Similarly we claim that author verification can be used to verify whether the original documents and the summaries are written by the same author. In Chapter 7 we describe the above ideas in more details.

Additionally, in Section 7.3 we propose a plagiarism detection algorithm to measure the contextual similarity between a document and its corresponding summary.

2

Overview and Definition

We begin our work by providing an overview of the existing summarization approaches. The field of automatic text summarization is already almost 60 years old, and various approaches have been introduced to target this problem.

In Section 2.1, we review the most influential works done in this field and shortly describe them. We also provide a taxonomy of the existing summarization approaches according to their function (indicative vs. informative), their input cardinality (single- vs. multi-document), their genre (news, interview, etc.) and their type (extractive vs. abstractive) [TM14].

As in every scientific discipline, we need a formal definition of the underlying problem to be solved. In the case of automatic text summarization, providing a precise definition of the task is quite a challenging problem. Summarization is considered a subjective task, and this leads to a subjective definition of the problem. This is the reason why there are many available definitions of automatic text summarization in the literature which can differ from each other dramatically.

In Section 2.2 we study various available definitions of automatic text summarization in the literature and investigate their advantages and drawbacks. Based on our findings we present a new definition for it and claim that the introduced definition is more suitable than the existing definitions to be used as the definition of automatic text summarization.

By providing an overview of the existing summarization approaches and introducing a definition of the summarization task, we build the foundation of our work and gain a deeper understanding of the underlying problem. The introduced definition of text summarization will be used in the entire work as a thread, and all the approaches will be oriented on it.

2.1 An Overview of Automatic Single Document Text Summarization Techniques

Pashutan Modaresi and Stefan Conrad. An Overview of Automatic Single Document Text Summarization Techniques.
Contributions: The research and the preparation of this manuscript was done entirely by the main author under the supervision of Prof. Conrad. **Status**: Unpublished.

In this section, we provide a short overview of the existing work in the field of automatic single document summarization. In Section 2.1.1, we introduce a taxonomy of the available summary types based on different criteria. Although the proposed taxonomy is by no means complete, it includes the most important types of the summaries available in the literature.

According to the cardinality, we categorize summaries into the single document and multidocument summaries. In this work, we focus on the single document summaries and review the existing summarization approaches to this type of summaries.

We start our overview with the classical methods in the field of single document summarization (Section 2.1.2). Although most of the traditional summarization strategies are heuristic-based and incorporate simple features, the same heuristics and features are still used in the proposed complex summarization methods.

In the 1990s, the machine learning algorithms gained popularity and were employed in the context of automatic text summarization. Various machine learning algorithms such as the Naive Bayes algorithm, classification trees, support vector machines and matrix factorization methods were applied to the field of text summarization. In Section 2.1.3, we review the most influential works in this area.

Although machine learning approaches have partially managed to tackle the automatic summarization problem, a significant progress has been made in this field in the last two years with the popularity of deep learning methods. More specifically, in the field of abstractive text summarization, deep learning approaches have revealed very satisfying results.

In Section 2.1.4, we review the most influential works in the field of neural text summarization. We start with the first published work in this area by Rush et al. [RCW15] and review some other neural summarizers that improve on the work of Rush et al.

As already stated, in this work we focus on the task of single document summarization and review the existing work in this subfield. For a more comprehensive review of the literature, we refer the reader to [GG16].

2.1.1 Taxonomy of Summaries

Different criteria can be used to categorize summaries. In this work we follow the work of Torres-Moreno [TM14] and categorize the summaries according to the following criteria:

- Function: According to their function, summaries can be categorized into indicative and informative summaries.
  - Indicative: These kinds of summaries contain indications to the relevant information of the original document, and their main purpose is to represent the information in a concise manner, such that the reader can easily filter and organize the required information. [SL02].
  - Informative: As the name suggests, informative summaries reflect the most relevant information in a document in a concise manner, such that the user is provided with all the required information, without the need to refer to the original document for further information. [HL98].
- Cardinality: According to the cardinality of the source documents, summaries can be categorized into single document and multidocument.
  - Single document: In single document summarization, there exists a single document to be summarized and the user is only interested in the relevant information of the underlying document [DBK16].
  - Multidocument: In this case, there exist more than one document to be summarized. Identifying redundant information or contradictions are challenging problems for these types of summaries [GMCK00].
- Genre: Summarization systems can be designed for specific document genres. This can include news articles [LJH05], tweets [XGMR13], blogs [HSL07], medical documents [AKS05], or meeting transcripts [WC12].
- Type: According to type, summaries can be categorized into extractive and abstractive ones.
  - Extractive: In this type of summaries, the summary document is constructed by selecting or extracting the relevant information from the source document [KMTD14].
  - Abstractive: Generative or abstractive summaries are produced by reformulating the content of a document in a concise manner [Gre11].
- Context: According to context, summaries can be categorized into generic and query-based summaries:

- Generic: This type of summaries are constructed from the author's perspective and completely ignore the user's needs [SSJ01].
- Query-based: Different from generic summaries, query-based summaries consider the information needs of the user, and contain the information relevant to the user's queries [AIAA15].

Although summaries could be categorized based on other criteria, in this work we follow the above-introduced criteria. We focus on the task of single document summarization and review the existing literature in this field.

2.1.2 Classical Approaches

We start our overview with a review of the classical approaches in the field of automatic single document summarization. The work of Luhn [Luh58] is considered to be the first published work in the field of automatic single document extractive summarization. Although the ideas presented in Luhn's work may seem trivial, those ideas were ahead of their time, and even today can be considered as a revolution in the field of natural language processing and automatic text summarization.

The main idea behind Luhn's algorithm was that the significance of a word in a document could be determined based on its frequency. Of course, stopwords are commonly the most frequent words in a document and should not be considered as significant words. For this, Luhn manually created a list of stopwords containing some common English words such as articles and pronouns. Additionally, Luhn proposed that normalizing the words in the document could be advantageous in discovering significant words in the document. Luhn's intention of normalizing was to group words that are similar to each other with respect to their orthographic features. Luhn's normalizing approach can be considered as one of the first automatic attempts to what we call *stemming* today.

Moreover, Luhn's approach also eliminated lexical groups with a low frequency (with lexical groups we mean all words with the same stemmed form). In this way, in the end, Luhn constructed a list of most frequent words in the document. Sentences in the document were then ranked based on the ratio of significant keywords and a total number of keywords, where higher ratios were a significance indicator for the sentences.

In the same year (1958), Baxendale [Bax58] analyzed a collection of documents and concluded that the positions of the sentences in the document play an important role in determining their significance. In the case of paragraphs, Baxendale claimed that in 85% of the cases, the most significant sentence of a paragraph is its first sentence. He also showed that in 7% of the cases, the last sentence of a paragraph is its most significant sentence. The positions of the sentences were also used as significant features in an early work of Edmundson and Wyllys [EW61].

In another work, in addition to the frequency and position features, Edmundson used the *cue words* as an additional feature to determine the significance of the sentences [Edm69]. By *cue words*, we mean the presence of specific words such as *significant* or *hardly* in a sentence, which might be an indicator that the sentence should be considered as important. Moreover, Edmundson also incorporated structural features that took the structure of the document (words occurring in the headlines or subtitles) into account.

2.1.3 Machine Learning Approaches

Although classical approaches introduced valuable ideas for automatic summarization of documents, they were mainly based on heuristics optimized for a particular domain and were not generalizable. A solution to this problem was the incorporation of machine learning algorithms. In the 1990s, the Naive Bayes algorithm was a popular machine learning algorithm, and it has since been used to train automatic summarization systems.

In 1995, Kupiec and his colleges [KPC95] introduced a trainable document summarizer based on the Naive Bayes algorithm. They claimed that the document summaries, consisting of roughly 20% of the original documents, could be as informative as the full text of the documents. In their work, they used five features to train the algorithm:

- Sentence Length Cut-off Feature: A binary feature indicating whether the sentence length is less than a pre-defined threshold (five words).
- Fixed-Phrase Feature: A binary feature indicating whether a sentence contains any of the 26 predefined keyphrases. The list of keyphrases included phrases and words such as: in conclusion, results, summary and discussion.
- Paragraph Feature: A ternary feature indicating whether a sentence is at the beginning of a paragraph (paragraph-initial), in the middle of it (paragraph-middle) or at the end of a paragraph (paragraph-final).
- Thematic Word Feature: A binary feature indicating whether a sentence is a high score or low score sentence. The score of a sentence was calculated based on the portion of content words in it.
- Uppercase Word Feature: A feature indicating whether a sentence contains proper names or not. Proper names were automatically determined based on the capitalization of their first letter.

The probability that a sentence s was in the summary S was expressed using the Bayes rule as stated in Equation 2.1.

$$P(s \in S|F_1, F_2, \cdots, F_k) = \frac{P(F_1, F_2, \cdots, F_k|s \in S)P(s \in S)}{P(F_1, F_2, \cdots, F_k)} \tag{2.1}$$

The probability $P(s \in S|F_1, F_2, \cdots, F_k)$ was interpreted as the probability of the sentence s being included in the summary S, given the k features $F_1, \cdots, F_k$. Assuming the statistical independence of the features, the probability was be calculated as follows:

$$P(s \in S|F_1, F_2, \cdots, F_k) = \frac{\prod_{j=1}^{k} P(F_j|s \in S)P(s \in S)}{\prod_{j=1}^{k} P(F_j)} \tag{2.2}$$

In Equation 2.2, $P(s \in S)$ is constant and $P(F_j|s \in S)$ and $P(F_j)$ can be estimated from the training corpus.

Another example of a summarization system that was based on the Bayesian classifier was the DimSum [AOGL99]. Different from the previous systems, Aone et al. addressed an interesting problem in text summarization. They claimed that the frequency-based approaches were based on a single word string as the unit for counting and completely ignore the semantic content of the words and their potential membership in multiword phrases. For this, they provided an example for the word *bill* where it represented a different instance in *Bill Clinton* than in *bill reform.* They criticized the previous frequency-based systems and claimed that ignoring the ambiguity in the word *bill* would introduce noise in the frequency counting.

Their proposed solution was to consider not only the significant words in a document, but also the important phrases (key concepts). To identify the important words and phrases in a document, they used both document-based and corpus-based statistics. More precisely, the term frequency was used as the text statistic, and inverse document frequency was used as the corpus statistic [SM86].

Different from the previous approaches that mostly assumed the independence of the features, in [Lin99], Lin introduced a summarization system based on decision trees. The system was called SUMMARIST, and it was designed to extract sentences from multilingual texts such as English, Arabic, Japanese and Spanish. Some of the features of the SUMMARIST system were:

- Position: The highest score was assigned to the first sentence and the lowest score to the last sentence of a document. Scores were normalized between zero and one.

- Title: The words in the document that occurred in the headline were assigned a positive score 1. Other words in the document were assigned a zero score.

- Frequency: Term frequency and inverse document frequency were used to score the words.

- Query Signature: In the case of query-based summarization, sentences that contained query words were scored higher.
- Sentence Length: Length of sentences, normalized by the length of the longest sentence in the document.
- Average Lexical Connectivity: Number of terms shared by the other sentences, divided by the total number of sentences in the document.

In the previous approaches, the documents were mostly considered as a bag of words and the sequential information in the text were neglected. In 2001, Conroy and O'leary [CO01] approached the summarization problem as a sequential modeling task using Hidden Markov Models (HMMs). In comparison to the Naive Bayes methods, where it is assumed that the features are independent, the HMM approach had several advantages. In general, the probability that the sentence i is in the summary is not independent of whether sentence $i-1$ is in summary. Additionally, using an HMM, a joint distribution is used for the set of features that considers the dependencies between them.

In total, three features were used in the Hidden Markov Model. The position of the sentences in the document, number of terms in the sentences, and the likelihood of the sentence terms given the document. To model the extraction of up to $s-1$ summary sentences, the authors proposed a Hidden Markov Model with s summary states and $s+1$ non-summary states. The transition matrix was then obtained by training the model on 1304 documents taken from TREC [1] data set.

Knight and Marcu [KM02] followed a different approach to summarize textual documents. They claimed that extraction is not the natural way of summarizing textual documents by humans. Rather, they read and comprehend the document and then create new sentences that are grammatically correct and coherent. The authors reformulated the summarization task as a sentence compression problem and proposed a noisy-channel and a decision tree approach to solve it. To train the algorithms, pairs of sentences with their corresponding human-written compressions were used.

In 2002, Neto et. al [NFK02] proposed a summarization approach that attracted the attention of the research community. Their approach can be summarized in four steps:

1. Preprocessing: Stopword removal, case folding and stemming [Por97].
2. Vectorization: Converting sentences to their vector representation [SB88].
3. Feature Extraction: Including 13 features.

[1]http://trec.nist.gov/

4. Training: Two classical machine learning algorithms were employed. C4.5 [Qui93] and Naive Bayes.

A large number of the incorporated features were borrowed from the exiting summarization approaches. Some lesser-known features included:

- Sentence-to-Sentence Cohesion: For each sentence s, its similarity to all other sentences in the document was computed. The similarities were then summed up and normalized in the range $[0, 1]$. The normalized values closer to one indicated sentences with larger cohesion.
- Sentence-to-Centroid Cohesion: First, the centroid vector of the document was computed. Then, for each sentence, the similarities between the sentences and the centroid of the document were computed and normalized in the range $[0, 1]$.
- Occurrence of Proper Names: A binary feature indicating whether a sentence contains proper names.
- Occurrence of Anaphora: A binary feature indicating whether the first six words of a sentence contain an anaphora or not. It was considered that anaphora indicated the presence of non-essential information in a text.

Graph-based approaches have also been popular in the field of automatic text summarization. The basic idea behind graph-based approaches is very straightforward. In the case of single document summarization, the documents are mainly represented as a graph $G = (V, E)$ with the set of vertices V and edges E. Each sentence in the document is typically represented by a vertex in the graph. An edge between two vertices is typically used to model the similarity or the overlap of two sentences. Using the above schema, in the end, we have a highly connected weighted graph, to which, a graph-based ranking algorithm can be applied to select the most salient sentences in the document [Mih04].

Graph-based ranking algorithms are used to identify the importance of the vertices in a graph structure. Let $I(v_i)$ be the set of vertices that point to v_i and $O(v_i)$ be the set of vertices that v_i points to. The PageRank [BP98] algorithm is considered as one of the most influential methods to rank the vertices in a graph. Based on the incoming and outgoing edges of a vertex, the PageRank algorithm calculates the score of the vertex v_i as follows:

$$PR(v_i) = (1 - d) + d \sum_{v_j \in I(v_i)} \frac{PR(v_j)}{|O(v_j)|} \tag{2.3}$$

LexRank [ER04] and TextRank [MT04a] are two of the other popular graph ranking algorithms that have been used to summarize textual documents.

Latent Semantic Analysis (LSA) is another popular machine learning approach that has been widely used in the context of automatic text summarization. LSA is a statistical method to extract the hidden semantic structure of the sentences. Latent semantic analysis can be summarized as follows [YKYM05]:

- Matrix Representation: In the first step, LSA transforms the input document into a matrix, where the columns represent the sentences, and the rows represent the words in the document. There exist various strategies to fill the cells of the matrix. A given cell c_{ij} can be filled with the frequency of the i-th token in the j-th sentence. TF-IDF (term frequency - inverse document frequency) is another popular choice to fill the entries of the matrix.
- Singular Value Decomposition: It is a matrix factorization method used to model the relationships between the words and the sentences. Given the input matrix $A \in \mathbb{R}^{m \times n}$, this method factorizes the input matrix into 3 matrices $A = U\Sigma V^T$, where $U \in \mathbb{R}^{m \times n}$, $V \in \mathbb{R}^{n \times n}$ and $\Sigma \in \mathbb{R}^{n \times n}$ is a diagonal matrix.

Most summarization approaches used the results of LSA and applied different selection strategies to extract the important sentences from a document. In [GL01], the authors used the matrix V^T to identify the relevant sentences in the document. Each row in the matrix V^T was interpreted as a concept and each column as a sentence. In V^T, rows (concepts) are placed according to their importance in the document, meaning that the first row represents the most important concept and the last row represents the least important concept in the document. Moreover, the entry v_{ij} in V^T indicates the relatedness of the j-th sentence to the i-th concept. To extract the sentences, the algorithm began with the most important concepts and selected the most related sentence to this concept. This process continued until a predefined number of sentences were reached. Different from this approach, in [SJ04], the authors use both V and Σ to extract the important sentence from the document.

A considerable amount of the work in the field of automatic text summarization has focused on extractive methods. On the other hand, abstractive summarization was considered a very challenging problem, and this subfield of text summarization has been less studied. In [GL11] the authors propose to generate the content of the summary not from the sentences in the text, but from an abstract representation of the document. The concept of *information items* was used to define the abstract representation of a document. An information item is defined as *the smallest element of coherent information in a text or a sentence.* The authors chose the subject-verb-object (SVO) triples in the document as the candidate information items and used a list of manually created rules to filter specific information items. The eliminated information items included the ones with verbs in infinite form and those that are part of a conditional clause. To realize (generate) the summaries from the selected information items, the original parse trees of the sentences, their corresponding information items and a text realizer called SimpleNLG [GR09] was used.

Rule-based approaches have also been popular to tackle the abstractive single document summarization problem. In [GL12], Genest and Lapalme introduced *abstraction schemas* consisting of information extraction rules, content selection heuristics and generation patterns. The abstraction schemas were all created by hand. In Table 2.1, an example for an abstraction schema is provided.

Table 2.1: Abstraction Schema for *killing*

Information Extraction	SUBJ(kill, X) $\rightarrow$ WHO(X)
Information Extraction	OBJ(kill, Y) $\rightarrow$ WHO_AFFECTED(Y)
Generation	X kill verb Y

The main problem with the rule-based approaches is that their creation and maintenance are costly and they do not generalize to other domains. In contrast to the classical machine learning approaches, deep learning method achieved great success in the area of abstractive text summarization. In the following section, we shortly review some of the existing approaches in this area.

2.1.4 Deep Learning Approaches

Deep neural networks are artificial neural networks composed of several hidden layers that are particularly used to learn distributed representations of the data [IGC16]. The first published work in the field of neural text summarization was by Rush et al. [RCW15]. The authors tackle the problem as a sequence generation problem, where the input consists of a sequence of M words $x_1, \cdots, x_M$ and the output (summary) is a sequence of length $N < M$, represented by $y_1, \cdots, y_N$. For simplicity, it was assumed that both input and output sequences come from the same vocabulary V. To generate the summaries, the distribution of interest was defined as $p(y_{i+1}|x, y_c; \theta)$, where y_c was a fixed window of previous words defined as $y[i-c+1, \cdots, i]$. This distribution is a conditional language model based on the input sequence. In their work, the authors directly parametrize this distribution as a neural network that consists of a neural language model and an encoder.

The neural language model was incorporated to estimate the probability of the next word [BDVJ03]. As the encoder, the authors use a convolutional encoder [KSH12] that allows local interactions between words. Although the convolutional encoder outperforms the bag-of-words encoder, it has the disadvantage of encoding the entire input sequence into a single vector. To solve this problem, the authors use an attention-based encoder that takes the generation context into account.

To train the neural architecture, the negative log-likelihood cost function was used (Equation 2.4) that was minimized using the stochastic gradient descent [IGC16] method.

$$\begin{aligned} NLL(\theta) &= -\sum_{j=1}^{J} \log p(y^{(i)}|x^{(j)};\theta) \\ &= -\sum_{j=1}^{J}\sum_{i=1}^{N} \log p(y_{i+1}^{(i)}|x^{(j)}, y_c;\theta) \end{aligned} \tag{2.4}$$

In [NZZ16] the authors treated the problem of extractive summarization as a sequence classification problem, where each sentence in the document was visited sequentially and a binary decision had to be made to determine whether the sentence should be kept or not. An important point in this process was that the binary decisions were not independent, and for each decision, the previous decisions were also taken into account.

The sequence classifier was realized using a specific type of recurrent neural networks called *Gated Recurrent Unit (GRU)*. A GRU is formally defined as follows:

$$\begin{aligned} u_j &= \sigma(W_{ux}x_j + W_{uh}h_{j-1} + b_u) \\ r_j &= \sigma(W_{rx}x_j + W_{rh}h_{j-1} + b_r) \\ h'_j &= tanh(W_{hx}x_j + W_{hh}(r_j \odot h_{j-1}) + b_h) \\ h_j &= (1-u_j)\odot h'_j + u_j \odot h_{j-1} \end{aligned} \tag{2.5}$$

In Equation 2.5, W's are the weights, b's are the bias vectors, h_j represents the hidden-state vector and $\odot$ is the Hadamard product. In their approach the authors use a bi-directional GRU. By bi-directional we mean two GRUs where the first one reads the inputs from left to right and the second one from right to left. Using the above architecture, the entire document can be represented as a vector d:

$$d = tanh(W_c \frac{1}{N_d}\sum_{j=1}^{N_d}[j_j^f, h_j^b] + b), \tag{2.6}$$

where h_j^f and h_j^b are the hidden-state vectors obtained from the forward and backward pass, N_d is the number of sentences in the document and $[\cdot]$ is used to represent the concatenation of two vectors. Finally, the binary classification can be defined as a conditional probability based on the hidden-state vectors and the dynamic representation of the summary at the j-th sentence position.

The encoder-decoder models discussed so far suffer from two shortcomings. First, the encoder part considers only the words it has read so far. This leads to a suboptimal vector representation of the document. Second, to avoid the out-of-vocabulary problem, the decoder utilizes large vocabularies that make it slow. In [ZLFU16] two mechanisms were introduced to tackle the afore mentioned problems. To avoid the first issue, a mechanism was introduced that first reads the entire input sequence and then commits to a representation of each word. For the second issue, a copy mechanism was introduced to handle even very small vocabularies.

The idea behind the encoder (called Read-Again model) is very intuitive and is inspired by how humans perform the summarization task. The model reads the input text twice and uses the information gained in the first pass to bias the second read. Using a GRU, the input sequence is read for the first time:

$$h_i^1 = GRU^1(x_i, h_{i-1}^1) \tag{2.7}$$

Given an input sequence of length n, the vector h_n^1 is computed by the first pass. Then an importance vector α_i is computed for each word in the input sequence in the second pass. This will result in the following update rule for the second pass:

$$h_i^2 = (1 - \alpha_i) \odot h_{i-1}^2 + \alpha_i \odot GRU^2(x_i, h_{i-1}^2) \tag{2.8}$$

In Equation 2.9, $\odot$ represents the element-wise product. Additionally, the decoder has access to the vocabulary of the input sequence that leads to a smaller vocabulary size for the decoder. A Long short-term memory was used as the decoder [HS97].

Another interesting approach in the field of query-based summarization is the work of [CLLW16]. Cao et al. propose a neural architecture called AttSum to learn query relevance and sentence saliency ranking jointly. The main idea behind the AttSum algorithm is to mimic the human behavior in the case of query-based summarization. Given a query, humans mainly focus and pay attention to the parts of the text that are related to the query. To mimic this behavior, the AttSum algorithm utilizes the following layers:

- CNN[2] Layer: This layer is used to obtain distributed representations for both input sentences s and the query q.

$$v(s) = CNN(s)v(q) = CNN(q) \tag{2.9}$$

  The output of the convolutional neural network is a vector $\hat{c}^h$ that is a distributed representation for both input sentences and the query.

[2]Convolutional Neural Network

- Pooling Layer: This layer is used to simulate the human attentive reading behavior. Using a mechanism called *pooling weight*, the relevance of the queries to the input documents are learned. If a sentence and a query are significantly related to each other, the pooling weight will be high.
- Ranking Layer: This layer in intended to rank the sentences in a document according to their relevance to the query which is based on the cosine similarity.

The research in the field of neural text summarization is still in its infancy. In this section we reviewed some of the main contributions in this field. In summary, the recurrent neural networks and more specifically the sequence to sequences models belong to the most popular approaches in this field. They mainly consist of an encoder part to construct a distributed representation of the input document, and a decoder part to generate the summary document based on the output of the encoder.

2.2 On Definition of Automatic Text Summarization

Pashutan Modaresi and Stefan Conrad. On Definition of Automatic Text Summarization. In *Proceedings of The Second International Conference on Digital Information Processing, Data Mining, and Wireless Communications (DIPDMWC2015)*, pages 33–40, 2015.
Contributions: The research and the preparation of this manuscript was done entirely by the main author under the supervision of Prof. Conrad. **Status**: Published.

Providing a formal definition of automatic text summarization is rather a challenging task. There exist various definitions of automatic text summarization in the literature and it is very common in the summarization community that authors come up with their own definition of text summarization and adapt the definition to their introduced approach.

It is understandable that due to the subjective nature of the summarization task, there might exist multiple definitions of it, but on the other side the rapid increase in the number of definitions and interpretations of the summarization task results in a high amount of discrepancy in the community and prevents the researchers from gaining a common understanding of the summarization problem.

This section describes our attempt to provide a formal definition of automatic text summarization. We review the available definitions in the literature and evaluate them based on five criteria: universality (is the definition valid for all known summary types?), generality (does the definition apply implementation restrictions on the summarization component?), minimality (are the number of properties and constraints in the definition minimal?), exclusivity (does the definition apply only for text summarization?) and repeatability (does the definition also hold for an already summarized document?).

Furthermore, we propose our own definition of automatic text summarization and claim that it possesses all the five properties introduced above. We construct our definition based on the concept of *readability* and claim that it is a more appropriate element to be used in the context of text summarization. We also claim that the *query* is a critical element in summarization systems that is mainly neglected by researchers working on generic summarization systems.

Notice that we do not claim that our proposed definition is flawless. But we claim that it does not possess the drawbacks of the previous definitions of automatic text summarization and encourage the community to use this definition as the baseline definition for further improvement. It should be clear that with further improvement of the field, the definition should also be updated accordingly.

Proceedings of Second International Conference on Digital Information Processing, Data Mining, and Wireless Communications (DIPDMWC2015), Dubai, UAE, 2015

On Definition of Automatic Text Summarization

Pashutan Modaresi Stefan Conrad
Heinrich-Heine-University of Düsseldorf
Institute of Computer Science, Düsseldorf, Germany
{modaresi, conrad}@cs.uni-duesseldorf.de

ABSTRACT

Research in the continuously growing field of automatic text summarization is branched into extractive and abstractive approaches. Over the past few decades, major advances have occurred in extractive summarization and a smooth transition from extractive to abstractive approaches can be observed in recent years. Despite advances, a proper definition of automatic text summarization has been mainly neglected by researchers. In this work we emphasize on the importance of an appropriate definition of automatic text summarization. We review previous definitions on text summarization, investigate their properties and propose our own definition.

KEYWORDS

Text Summarization, Scientific Definition, Content Selection, Readability, Text Mining

1 INTRODUCTION

Modern research on automatic text summarization began almost 60 years ago with the work of Luhn [1] on automatic creation of literature abstracts. Over the years, much progress has been done in the development of algorithms to automatically summarize documents.
Among the two major approaches of extractive and abstractive summarization, the first one has been investigated extensively in the literature. Despite satisfactory results in extractive summarization, researchers are focusing more and more on abstractive summarization in the recent years. Extractive summaries are usually created by concatenation of fragments of the source document with the addition of some post-processing. On the other hand, abstractive summaries are the result of rewriting or paraphrasing the source document, where a one-to-one mapping between the sentences of the source and target document is not always possible. Abstractive summarization is considered to be the natural way of summarizing performed by humans, which is one of the reasons for its popularity in the recent years.
As in other scientific disciplines, the first step to approach the automatic summarization problem is to define the problem itself. Previous research on extractive text summarization has revealed a disagreement in the community regarding the understanding and definition of the summarization problem. This can be acknowledged by the discrepancy between the various definitions of the problem proposed in the literature so far. In like manner, a similar flaw can also be observed in the field of abstractive text summarization.
With this in mind, a need for a proper definition of automatic text summarization is being felt in the research community. This has been mainly neglected by researchers and consequently led to inconsistent foundations of this field. By inconsistent we mean that there is no single definition of a summary which has been agreed upon by researchers.
We impose several requirements on a proper definition of automatic text summarization:

- **Universality**: The definition should be valid for the known types of automatic text summarization. This includes indicative [2] and informative (according to functionality), single- [3] and multi-document [4] (according to input cardinality), hierarchical and flat [5] (according to output cardinality), extractive [6] and abstractive [7] (according to type), as well as generic, update [8] and query-guided (according to context).

- **Generality**: The definition should not apply any restrictions on the implementation details of the various stages of automatic text summarization. This includes particularly the representation type of the source document, content selection, scoring, lexical selection or text realization.

- **Minimality**: The definition should be minimal, meaning that only a minimal number of properties or characteristics to *reconstruct* a text summary should be mentioned.

- **Exclusivity**: The definition should be exclusive, meaning that the definition allows degenerate cases that one may wish to exclude.

- **Repeatability**: The definition should be repeatable, meaning that applying the definition to a summary document as the input document, should either return a valid summary document, or prevent us from creating a new summary document if it is not possible.

In Section 2 we provide an overview of the existing definitions of automatic text summarization and investigate their properties. A commonly used concept in automatic text summarization is *compression rate*. The usage of this concept will be criticized in Section 3 and the concept of *readability* will be suggested as a substitution. Content selection as an important part of any summarization system will be discussed in Section 4. In Section 5 we propose our own definition of automatic text summarization and finally in Section 6 we will conclude our work.

2 RELATED WORK

Various definitions of automatic text summarization have been proposed in the literature. Despite some commonalities, these also include contradictions in some cases. Furthermore the proposed definitions are mostly applicable to a certain type of automatic text summarization and lack the properties introduced in Section 1. The lack of a proper definition of automatic text summarization can be due to a conservative attitude in the community, as Das and Martin state: "...it seems from the literature that any attempt to provide a more elaborate definition for the task [of automatic text summarization] would result in disagreement within the community"[3].

In this section we investigate the previous definitions of automatic text summarization proposed in the literature. We study their flaws and inspect the properties that make them inappropriate definitions of automatic text summarization.

In fact Luhn did not propose a definition of text summarization, but rather he mentioned the purpose of a summary in the context of literature abstracts as: "the purpose of abstracts in technical literature is to facilitate quick and accurate identification of the topic of published papers. The objective is to save a prospective reader time and effort in finding useful information in a given article or report"[1].

Although this cannot be considered as a definition of automatic text summarization, but Luhn's statement points to two important properties of a text summary. The first property is that the time and effort for reading a summary should be less than the one being consumed in reading the original document, and the second property is that a summary should accurately reflect the topic of the original document.

In 1995 Maybury defined an effective summary as "[a text that] distills the most important information from a source (or sources) to produce an abridged version of the original information for a particular user(s) and task(s)."[9].

By mentioning that a summary is produced from a source (or sources), Marbury covers the cases for single document and multi-document text summarization. Moreover the property that a summary is produced for a particular user(s) and task(s) can be interpreted as if the definition also covers the query-guided and generic cases. The most important property that Mybury's definition lacks is exclusivity. The same definition could also be applied to the task of *keyword extraction*. Although keyword extraction is occasionally also considered to be a text summarization task, but in general it is a distinct branch of text mining, as different from text summarization, the target document in keyword extraction is a collection of keywords and not a text document consisting of coherent sentences.

In his book *Automatic Summarization*, states Mani in 2001 that "a summary is a document containing several text units (words, terms, sentences or paragraphs) that are not present in the source document."[10]. Consider a source document d and a target document $t = d \cup \{s_1, \ldots, s_n\}$ constructed by addition of n sentences to the source document. Clearly t is not a summary document and thus Mani's definition lacks the universality property of a proper definition.

The same problem also applies to the in the 2001 proposed definition of Sakai and Spark-Jones where they define a summary to be "a reductive transformation of a source text into a summary text by extraction or generation" [11]. The above definition, cannot be applied to the query-guided summaries and thus lacks the universality property of a definition.

In 2002, Radev et al. defined a summary to be "a text that is produced from one or more texts, that conveys important information in the original text(s), and that is no longer than half of the original text(s) and usually significantly less than that"[12]. This definition lacks the generality property of a proper definition, as a restriction on the size of the output document is applied.

An example of a recent attempt in defining automatic text summarization is the work of Torres-Moreno in 2014 where he defines an automatic summary as "a text generated by a software, that is coherent and contains a significant amount of relevant information from the source text. Its compression rate τ is less than a third of the length of the original document"[13].

Torres-Moreno's definition points to an important property of the summary text, which is its coherence. The definition does not concretize *relevant information* and is not applicable to query-quided summaries, resulting in a lack of universality property. It also lacks the generality property by introducing compression rate as a part of the definition.

In Table 1 the properties of the discussed definitions are summarized. For the sake of readability, following abbreviations are used in the table: U: Universality, G: Generality, M: Minimality, E: Exclusivity, R: Repeatability

Table 1. Properties of Previous Definitions on Text Summarization

	U	G	M	E	R
Luhn [1]	✗	✓	✓	✗	✗
Maybury [9]	✗	✓	✓	✗	✗
Mani [10]	✗	✗	✓	✗	✗
Sakai [11]	✗	✗	✓	✗	✗
Radev et al. [12]	✗	✗	✓	✓	✗
Torres-Moreno [13]	✗	✗	✓	✓	✗

In the following section a commonly observed property of summaries, namely *compression rate* will be discussed and criticized as an unnecessary part of automatic summarization definition.

3 READABILITY

Usually in the context of automatic summarization we speak about *compression rate* τ which is defined as the ratio between the length of the summary and the length of the source document [13]:

$$\tau = \frac{|summary|}{|source|}, \tag{1}$$

where $|\bullet|$ is the length of the document in characters, sentences or words.

Various thresholds in the literature have been suggested for τ. In [14] a summary is defined to be a text which is not longer than the half of the source document. At the same time, in [15] the optimal compression ratio for a summary is defined to be between 15% and 30%.

The use of compression ratio as an essential part of the definition has several disadvantages. The first one is that a direct comparison of the length of a summary and the length of the source document is not always possible. This comparison is commonly made based on the character, word, or sentence length. The choice between words, characters or sentences is normally made arbitrarily without any specific reasoning and typically depending on the underlying data set.

Proceedings of Second International Conference on Digital Information Processing, Data Mining, and Wireless Communications (DIPDMWC2015), Dubai, UAE, 2015

By using the count of sentences in source and summary document for the calculation of compression rate, the length of a sentence is completely ignored. So two sentences s_1 and s_2 with $|s_1| \gg |s_2|$ in the summary document will contribute the same amount to the computation of τ.

On the other hand, computing the compression rate based on the length of documents in characters does not always return reliable information. As an example, consider two summaries t and $t' = t + "."$ where in t' a punctuation mark is inserted at the end of t. According to the formula of compression rate, the compression rate of the first summary t is less than the compression rate of the second one t', although the summaries do not differ significantly and a human evaluator would not even notice the difference between the summary documents.

Perhaps the most reliable measure among the introduced ones is the length of documents in words. However, to only consider the length of documents in word has its own drawbacks. One main drawback is that the complexity of the words will be completely ignored. This does not cause remarkable problems in the case of extractive summarization, but in the context of abstractive summarization where paraphrasing and lexical selection are typical procedures, this may make the measure inconsistent. From the other side, depending compression rate on the words (tokens), makes the comparison between the compression rates of different algorithms a tedious task. Different approaches use different techniques for tokenization of the underlying text and this results in the fact that compression rate will be highly dependent on the underlying tokenization algorithms.

The second problem with the concept of compression rate is the need to define a specific threshold. This threshold is usually selected without any specific reasoning and mostly the selection is done in a way that the algorithm will return the best possible results for the underlying data set. As already discussed various thresholds are suggested in the literature and this arises the question whether specifying a hard-coded threshold is reasonable and consistent with the natural way humans perform summarization?

Alternatively, *readability* of a text seems to be a suitable substitution for the compression rate (note that in this work we do not use *readability* in its common sense meaning but rather we define it as a measure). In general, we expect from a summary to be more readable than the source document. The concept of readability can capture various dimensions such as the consumed time for reading the summary, its cohesiveness or the complexity of the vocabulary in it.

The introduction of the readability may bring from one side more vagueness into the definition of the automatic text summarization, but from the other side, it will distract the focus from the compression rate that in the late researches was heavily regarded as a key factor, resulting in the ignorance of the other dimensions connected to the readability of a summary. Much research is already done to measure the readability of a text. This includes for example the *Flesch Reading-Ease Score* [16] that considers the average sentence length and the average word length in syllables:

$$\begin{aligned} FRES =& 206.835 - 1.015\left(\frac{\text{total words}}{\text{total sentences}}\right) \\ & - 84.6\left(\frac{\text{total syllables}}{\text{total words}}\right), \end{aligned} \tag{2}$$

or the *Gunning Fog Index* [17] that considers the average sentence length and a list of hard words (words with more than two syllables) in the text:

$$\begin{aligned} GFI =& 0.4\Big[\left(\frac{\text{words}}{\text{sentences}}\right) \\ & + 100\left(\frac{\text{complex words}}{words}\right)\Big]. \end{aligned} \tag{3}$$

Of course a direct takeover of the existing readability scores in the field of automatic text summarization is not appropriate and more sophisticated multi-factorial scores considering reading time or cohesion of the summaries have to be designed.

Some of the properties that should be required for a readable summary document are:

- Time: The amount of consumed time for reading a summary document should be

Proceedings of Second International Conference on Digital Information Processing, Data Mining, and Wireless Communications (DIPDMWC2015), Dubai, UAE, 2015

less that the amount of time that has to be consumed for reading the original document.

- Length: The length of a summary document should be less than the length of the original document (this can be modelled using the compression rate).
- Cohesiveness: A summary document should be at least as cohesive as the original document.
- Word Complexity: The average complexity of the words used in the summary text should be less than the average complexity of the word in the original document.

Note that the proposed properties for a readable summary are a subset of all possible properties that a readable summary document can exhibit. The intention of introducing properties for readability is to emphasize that length of a summary document (as considered in compression rate) is only one of the crucial dimensions in text summarization and other dimensions (specifically amount of time consumed for reading the summary document in comparison to the original document) have to be considered too.

Finally instead of compression ratio we suggest the use of readability ratio $\varrho = \frac{\rho(s)}{\rho(t)}$ to compare the ease of reading of the source document $\rho(s)$ to the summary document $\rho(t)$.

Beside the readability, content selection is another vital part of any definition on automatic text summarization. This will be discussed in more details in the following section.

4 CONTENT SELECTION

The decision which content to include in a summary is a critical one. The reason for this is that a summary will be finally read by a reader or a group of readers with diverse expectations from the content of the summary. Each reader has its own subjective preferences and expectations and creating a summary that fulfills all these subjective expectations is in practice impossible. With this is mind, summaries are commonly classified into generic and query-guided summaries. Generic summaries are the ones that estimate user's information need. In contrast guided summaries include the user's information need and ignore other irrelevant parts of the source document [18].

The user queries form a set Q of concepts, aspects, keywords or entities formulated by the user, representing the user's needs. Generic summaries shall be considered as a subcategory of query-guided summaries where the user query is an empty set $Q = \emptyset$.

We claim that a generic summary is of lesser use and almost impossible to evaluate manually. Consider a document d with its corresponding summary t, where $Q = \emptyset$. Now consider two users U_1 and U_2 aiming to manually evaluate the quality of content selection in the summary t. At this stage users will answer the question whether the summary contains the most relevant (significant) information of the source document or not. This is exactly the place where the subjective preferences of U_1 and U_2 will be formulated on the fly. Thus for U_1 we will have $Q_1 = \{q_1, \ldots, q_n\}$ and for U_2 we will have $Q_2 = \{q'_1, \ldots, q'_m\}$. Having two different sets of queries (probably formulated unconsciously by the users) will make the comparison of the scores calculated for t by U_1 and U_2 inconsistent.

By letting Q to be predefined, the vagueness of the phrases in the definition, such as *a text containing significant amount of information*, or *a text containing important information in the original text* will automatically disappear.

The question is now how to handle the case $Q = \emptyset$? More specifically, how should phrases such as *important* or *significant* information be interpreted, although no user preferences are pre-defined?

It is of course very restrictive to let the set of queries Q to be a non-empty set. This will lead to the lack of the universality property in the definition as the generic summaries can not be covered by such a restriction. By letting the set Q to be an empty set, the concretization of the keywords such as "important" or "significant" in the definition

will by postponed to the implementation of the algorithm itself. As an example, consider the work of Luhn [1]. For selecting the most important sentences in the document, Luhn followed an approach based on the frequencies of the words. Although Luhn's approach is classified under the generic summaries, but a query such as $q = \{$*sentences containing most frequent words in document*$\}$, could also be used. Another example is the work of Edmundson et al. [19] that used the presence of cue words for content selection. This can also be formulated as a query set such as $q = \{"significant", "impossible", "hardly"\}$.
In this way the evaluation of the summaries by multiple human evaluators will also be possible, as the evaluators will all be using the same set of queries.

5 DEFINITION OF AUTOMATIC TEXT SUMMARIZATION

Having told the drawbacks of the previous definitions of automatic text summarization and after discussing important aspects of any automatic text summarization system, we propose our own definition:

Definition. *Given a set Q of queries and a set K representing a knowledge base, automatic text summarization is a reductive transformation of a collection of documents D with $|D| > 0$ into a single or multiple target documents, where the target document(s) are more readable that the documents in D and contain the relevant information of D according to Q and K.*

The above definition exhibits the required properties of a proper definition as discussed in Section 1.
The definition can be applied to indicative and informative summaries. Furthermore the set D is defined as a collection of source documents with cardinality greater than or equal to one which covers both single and multi-document summarization. The set Q consists of queries in the form of phrases, entities, sentences or keywords and it can also be an empty set. By this, both query-guided and generic summaries are covered. This all results in that the proposed definition has the universality property.
The introduction of a knowledge base K in the definition, covers the case of update summaries where the users need is only to get update information about a specific topic in a summarized manner. Note that K can also be an empty set leading to the case of generic and query-guided summaries.
In the proposed definition it is also allowed to output multiple documents as the result of the summarization process. By this, the case of hierarchical summaries can be covered where the summary documents are ordered from more general and abstract ones, to more specific and detailed ones.
In the proposed definition, no restriction is applied to the implementation of the algorithms. Summarization is defined as a reductive transformation, meaning that the target document should be always shorter than the source document and by the introduction of the concept of readability and elimination of the concept of compression rate, the generality property of the definition is guaranteed.
The proposed definition also has the minimality property, as the elimination of any property in the definition will cause to the failure of the reconstruction of a text summary in a specific scenario.
We claim that the proposed definition has also the exclusivity property as the definition is not applicable to relevant fields such as keyword extraction, natural language translation or topic detection.
By use of the concept of readability, also the repeatability property of a proper definition is guaranteed. It is stated that the target document is more readable than the source document. Given a readable source document, it is always guaranteed that the target document is also readable. Assuming a source document consisting of two sentences and a target document consisting of one sentence produced from the source document, the question is now if the definition is still valid if we apply it to the target document consisting of one sentence? In other words, is it possible to summarize the target doc-

ument one more time, assuming the proposed definition? The requirement that the target document should always be more readable than the source document, results in the desired situation that some documents are not summarizable. This is especially the case where the source document is very short and compact enough that only a paraphrasing but not a summarization is possible.

6 CONCLUSION AND OUTLOOK

In this work we focused on a proper definition of automatic text summarization that has been neglected by many researchers in the community. We proposed the properties universality, generality, minimality, exclusivity, and repeatability (Section 1). Based on these properties, various existing definitions of automatic summarization in the literature have been investigated and criticized (Section 2).

We also discussed important aspects of a proper definition of text summarization such as readability (Section 3) and content selection (Section 4).

Finally in Section 5 we proposed our own definition of automatic text summarization and showed that the proposed definition exhibits all the properties of a proper definition introduced in Section 1. The proposed definition is by no means considered as a gold standard, however in the opinion of the authors it lacks many of the drawbacks of the previous definitions in the community.

Many other features such as the language of a summary, its coherence or the way it has to be evaluated have not been discussed in this work as a part of a proper definition. A more detailed investigation of the existing work on automatic text summarization is needed to examine the need for a more complex definition of automatic summarization.

Similar to any other definition, our proposed definition is also volatile in time and with respect to the community's feedback.

REFERENCES

[1] H. P. Luhn, "The automatic creation of literature abstracts," *IBM J. Res. Dev.*, vol. 2, pp. 159–165, April 1958.

[2] M. Kan, K. R. McKeown, and J. L. Klavans, "Applying natural language generation to indicative summarization," in *Proceedings of the 8th European Workshop on Natural Language Generation - Volume 8*, EWNLG '01, pp. 1–9, 2001.

[3] D. D. and A. F. T. M., "A survey on automatic text summarization," 2007.

[4] N. Ketui, T. Theeramunkong, and C. Onsuwan, "An edu-based approach for thai multi-document summarization and its application," *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, vol. 14, pp. 4:1–4:26, January 2015.

[5] J. Christensen, S. Soderland, G. Bansal, and Mausam, "Hierarchical summarization: Scaling up multi-document summarization," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 902–912, June 2014.

[6] Y. Meena and D. Gopalani, "Analysis of sentence scoring methods for extractive automatic text summarization," in *Proceedings of the 2014 International Conference on Information and Communication Technology for Competitive Strategies*, ICTCS '14, pp. 53:1–53:6, 2014.

[7] S. Banerjee, P. Mitra, and K. Sugiyama, "Abstractive meeting summarization using dependency graph fusion," in *Proceedings of the 24th International Conference on World Wide Web Companion*, WWW '15 Companion, pp. 5–6, 2015.

[8] R. McCreadie, C. Macdonald, and I. Ounis, "Incremental update summarization: Adaptive sentence selection based on prevalence and novelty," in *Proceedings of the 23rd ACM International Conference*

on Conference on Information and Knowledge Management, CIKM '14, pp. 301–310, 2014.

[9] M. T. Maybury, "Generating summaries from event data," *Inf. Process. Manage.*, vol. 31, pp. 735–751, September 1995.

[10] I. Mani, *Automatic Summarization*, vol. 3 of *Natural Language Processing*. John Benjamins Publishing Company, 2001.

[11] T. Sakai and K. Sparck-Jones, "Generic summaries for indexing in information retrieval," in *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pp. 190–198, 2001.

[12] D. Radev and A. Winkel, "Multi document centroid-based text summarization," in *In ACL 2002*, 2002.

[13] J. M. Torres Moreno, *Automatic Text Summarization*. Wiley-ISTE, 2014.

[14] E. Hovy and C. Lin, "Automated text summarization and the summarist system," in *Proceedings of a Workshop on Held at Baltimore, Maryland: October 13-15, 1998*, TIPSTER '98, pp. 197–214, 1998.

[15] C. Lin, "Training a selection function for extraction," in *Proceedings of the Eighth International Conference on Information and Knowledge Management*, CIKM '99, pp. 55–62, 1999.

[16] J. P. Kincaid, R. P. Fishburne, R. L. Rogers, and B. S. Chissom, *Derivation of New Readability Formulas (Automated Readability Index, Fog Count and Flesch Reading Ease Formula) for Navy Enlisted Personnel*. Research Branch report, Defense Technical Information Center, 1975.

[17] R. Gunning, *The Technique of Clear Writing*. McGraw-Hill, 1952.

[18] O. You, W. Li, S. Li, and Q. Lu, "Applying regression models to query-focused multi-document summarization.," *Inf. Process. Manage.*, vol. 47, no. 2, pp. 227–237, 2011.

[19] H. P. Edmundson, "New methods in automatic extracting," *J. ACM*, vol. 16, pp. 264–285, April 1969.

3 Data Acquisition

Data acquisition is considered as the first and one of the most important steps in machine learning [She11]. It is a time-consuming task, and the performance of the machine learning algorithms rely heavily on the quality of the gathered data. The large amount of the available data on the Internet motivates the researchers to automate the process of data acquisition. In this section, we introduce two approaches to automatically acquire data for training text mining algorithms and specifically automatic text summarization models.

In Section 3.1, we propose a machine learning approach to automatically identify news pages consisting of multiple pages. Although this is a fairly simple task for humans, due to the lack of visual hints, identifying the next page hyperlinks of a news page is not a trivial machine learning problem. The primary motivation here is to collect more data to train our intended machine learning algorithms. Additionally, by automatically identifying the next pages of a new article, we will gain access to the rest of the news. At the same time, this will help us to have a tool to retrieve full news articles without missing any salient information.

Furthermore, in Section 3.2, we present a tool called Simurg to automatically construct large corpora of news articles without any manual effort. The primary intention of this approach is to minimize the manual effort of the user to build an appropriate corpus and also solve problems regarding licensing of the corpora which can be a big obstacle for researchers to access the data they need. Additionally, the multilingualism is also kept in mind so that the corpora can also be easily constructed for any language requested by the user.

The constructed corpora using the approaches described in this section will then be used in the methods described later in this work for training the algorithms.

3.1 Automatic Identification of Multipage News: A Machine Learning Approach

Pashutan Modaresi. Automatic Identification of Multipage News: A Machine Learning Approach. In *Proceedings of the LWA 2015 Workshops: KDML, FGWM, IR, and FGDB*, page 75, 2015.
Contributions: The research and the preparation of this manuscript was done entirely by the main author under the supervision of Prof. Conrad. **Status**: Published.

Automatic content extraction is an active field of research that aims to automatically discriminate between clutter (unnecessary information) and content (required information) [GKG+05]. The definition of clutter and content is always domain and task dependent. For instance, in the case of *author extraction* of news articles, content is the name of the author or authors of the news article, and all the remaining elements of the news page (such as images, news body, headline, and banners) are considered as clutter. The HTML pages have a dynamic nature and their layout and attributes constantly change. This makes the design and development of robust automatic content extraction algorithms a challenging problem.

Although there exist generic approaches for extracting any kind of information (headlines, authors, images, etc.) from HTML pages, in practice, methods trained to extract a specific kind of information are more popular [WLF15]. This includes methods to extract headlines [MGSH12] and news body [WHW+09]. To the best knowledge of the author, there exists no previous work to identify multi-page news (e.g. extracting the next page link of the news).

In this section, we introduce a machine learning approach to identify multi-page news and extract the links to the subsequent pages. More precisely, we treat this task as a binary classification problem, where for each hyperlink it has to be decided whether it points to the next page of an article or not. In the context of automatic text summarization, being able to identify multi-page news has multiple advantages. From one side, the complete content of the news can be retrieved (mostly the following pages of news are ignored). This will lead to an increased size of the corpus. From the other side, the algorithms can access the complete content of the news that makes the inference in many cases easier.

In our work, we use the values of the class and id attributes of the HTML tags as the main features and label each HTML tag as either pointing to the next page of a news or not. We use the Naive Bayes algorithm to train a binary classification model and report results in terms of precision and recall.

Automatic Identification of Multipage News: A Machine Learning Approach

Pashutan Modaresi

Heinrich-Heine-University of Düsseldorf
Institute of Computer Science, Düsseldorf, Germany
modaresi@cs.uni-duesseldorf.de

Online news contain valuable information that can be utilized for private or commercial purposes. In the commercial context, online media monitoring services provide other companies or individuals with their required information in a systematic manner. This is accomplished by crawling plenty of news websites. Numerous news websites follow the strategy of pagination to split the stories into multiple pages. Given that, to identify multipage stories, manual rules have to be defined. On the other hand, the dynamic nature of the HTML pages requires a tremendous amount of effort in maintaining these rules. With this in mind, in this work we propose an automatic approach to identify multipage news stories.

We collected a list of web-pages in which the news were splitted in multiple pages and manually annotated them. To each link on the page a label has been assigned. That is, a link either points to the next page of the news or not. As the number of links which do not point to the next pages significantly dominates the number of link pointing to the next page of a news, the data set is highly imbalanced. Moreover, in order to design a language independent algorithm, news pages originating from different countries have been considered.

For each link, the *class* and *id* attributes of the corresponding anchor element, together with the text content of the anchor have been concatenated and fed into a Naive Bayes classifier. The same set of features extracted from the parent elements of an underlying link has been fed into another Naive Bayes classifier. Moreover, the relative position of a link on the news page (calculated by means of a heuristic) has been used to train a regression model. Additionally, some other features such as the structure of the *href* attribute of an anchor or the length of its text content have been integrated. Intentionally, the similarity between the content of the base page and the one of the target page has be ignored, as the calculation of this feature requires network availability that is not always given.

By cause of various learning algorithms being used, the final binary decision has to be performed by combining the results of the single constructed models. For this we use a *stacking* technique where we train a learning algorithm to combine the predictions of the constructed models.

Our first experimental results have revealed very high precision and recall values (≥ 0.9) for both labels under analysis.

3.2 Simurg: An Extendable Multilingual Corpus for Abstractive Single Document Summarization

Pashutan Modaresi and Stefan Conrad. Simurg: An Extendable Multilingual Corpus for Abstractive Single Document Summarization. In *Proceedings of the 8th Annual Meeting of the Forum on Information Retrieval Evaluation*, FIRE '16, pages 24–27. ACM, 2016.
Contributions: The research and the preparation of this manuscript was done entirely by the main author under the supervision of Prof. Conrad. **Status**: Published.

In Section 3.1, we introduced an approach to automatically identify the multi-page news articles. Although it was a critical component in the data acquisition process, the main problem is still not solved. How do we obtain a corpus consisting of news articles?

Most of the available corpora in the field of automatic text summarization are intended for evaluating summarization algorithms. They usually consist of a limited number of documents together with their corresponding summaries. TREC [1] and DUC [2] conferences have significantly contributed in providing evaluation datasets for automatic text summarization.

The available corpora for automatic text summarization have several drawbacks. The corpora are provided in a fixed size (containing a fixed number of documents), and the users do not have an impact on determining the size of the corpora. Moreover, many authors do not make publicly available the used corpora in their publications. Licensing is another issue that makes it hard for a researcher to gain access to the available corpora. Another issue is multilingualism, meaning that the corpora are not always available in the required language. Additionally, due to the change of the language, many corpora are not suitable for machine learning tasks any more.

We propose an approach to address the aforementioned problems of the available corpora for automatic text summarization. Our approach incorporates a distributed scrapper/crawler to automatically fetch news articles from the news aggregator of Google and automatically construct the headlines and bodies of the fetched HTML pages. The corpus consists of two parts: a sharable part that only contains the necessary information to extract the content of the news pages, and a second part that can be constructed and populated based on the information in the shareable part. To extract the headlines and bodies of the news articles, we used an existing automatic content extraction algorithm called Dragnet [PL13].

[1]Text REtrieval Conference
[2]Document Understanding Conference

Simurg: An Extendable Multilingual Corpus for Abstractive Single Document Summarization

Pashutan Modaresi
Institute of Computer Science
Heinrich Heine University of Düsseldorf
D-40225 Düsseldorf, Germany
modaresi@cs.uni-duesseldorf.de

Stefan Conrad
Institute of Computer Science
Heinrich Heine University of Düsseldorf
D-40225 Düsseldorf, Germany
conrad@cs.uni-duesseldorf.de

ABSTRACT

Abstractive single document summarization is considered as a challenging problem in the field of artificial intelligence and natural language processing. Meanwhile and specifically in the last two years, several deep learning summarization approaches were proposed that once again attracted the attention of researchers to this field.

It is a well-known issue that deep learning approaches do not work well with small amounts of data. With some exceptions, this is, unfortunately, the case for most of the datasets available for the summarization task. Besides this problem, it should be considered that phonetic, morphological, semantic and syntactic features of the language are constantly changing over the time and unfortunately most of the summarization corpora are constructed from old resources. Another problem is the language of the corpora. Not only in the summarization field, but also in other fields of natural language processing, most of the corpora are only available in English. In addition to the above problems, license terms, and fees of the corpora are obstacles that prevent many academics and specifically non-academics from accessing these data.

This work describes an open source framework to create an extendable multilingual corpus for abstractive single document summarization that addresses the above-mentioned problems. We describe a tool consisted of a scalable crawler and a centralized key-value store database to construct a corpus of an arbitrary size using a news aggregator service.

CCS Concepts

•**Computing methodologies → Artificial intelligence; Natural language processing;**

Keywords

Abstractive text summarization; single document summarization; extendable corpora; multilingual corpora

FIRE '16, December 08 - 10, 2016, Kolkata, India
© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ISBN 978-1-4503-4838-6/16/12…$15.00
DOI: http://dx.doi.org/10.1145/3015157.3015163

1. INTRODUCTION

Research in the field of automatic text summarization has a long story. Nearly 60 years ago, Luhn [12] published a work on the automatic creation of literature abstracts. He followed an *extractive* approach to summarize textual documents. Today, research is still done in the field of extractive text summarization to obtain more coherent and informative summaries [4].

Another approach to summarize textual documents is called *abstractive* text summarization that is considered to be the natural way humans generate text summaries and also a difficult problem in natural language processing [14].

Until recently, due to its difficulty, abstractive text summarization has been partly neglected by researchers. This has been changed in recent years using GPU-accelerated machine learning applications and the introduction of deep learning approaches [16].

One known property of deep learning approaches is their need for huge amounts of data. In some areas such as automatic image colorization [26] or image caption generation [25], accessing or generating this huge amount of data can be easily automated. However, in the case of abstractive text summarization it is rather a non-trivial problem.

The available corpora in the field of abstractive text summarization have several drawbacks regarding their size, accessibility, licensing, multilinguality, language change and extendability. With *size* we mean the number of documents in the corpus that should be high enough to train modern summarization algorithms. With *accessibility*, we mean whether the corpus is made publicly available or not. *Licencing* refers to the licensing terms and fees of the corpora and whether academics and non-academics can easily obtain it. A corpus is called *multilingual* if it provides data in more than one language. *Language change* is the variation or change of the language over time [5]. This is a property mainly neglected by the available corpora. We also call a corpus *extendable* when the user can easily extend the corpus on demand. In the following, we provide some examples of the available summarization corpora and shortly discuss their drawbacks.

One of the largest corpora recently used in the field of abstractive text summarization is called the English Gigaword corpus [7] published by Linguistic Data Consortium. In [21] and [15], attentional encoder-decoder recurrent neural networks are trained on the English Gigaword corpus to generate abstractive summaries. Although the Gigaword corpus is also available in some other languages such as Arabic, French and Spanish, due to its high costs (specifically for non-academics), obtaining a license is rather difficult.

Several summarization corpora are also made available in the scope of the Document Understanding Conference [6] and Text Retrieval Conference [2]. These corpora contain a limited number of documents and are mainly used for automatic evaluation purposes by means of the ROUGE [11] or BLUE [18] evaluation measures. As an example, the approach introduced in [3] is mentionable where the authors evaluate the goodness of a convolutional neural network on the DUC 2005-2007 datasets.

Other corpora such as the BC3 corpus [23] (violating the size and multilinguality properties), the SummBank corpus [20] (violating size, licensing and language change properties) and the PCSC corpus [1] (violating the size, licensing and multilinguality properties) are also mentionable.

In this work we introduce our methodology to automatically create a corpus (called Simurg) for abstractive single document summarization, keeping the properties mentioned above in mind. The corpus size can be arbitrary extended to millions of documents, and new languages can be automatically added on demand. The main idea is not to publish the corpus itself, but to provide the end users with a tool to create and populate the corpus locally on their machines. This tool is publicly available on GitHub[1].

In the remainder of this work, we introduce our methodology consisting of two steps: creating a shareable corpus called the template corpus (Section 2) and populating it with the required information (Section 3). We conclude our work in Section 4.

2. TEMPLATE CORPUS

The *temporal corpus* is the shareable part of the *Simurg* corpus. By *sharable* we mean that the corpus does not contain any information that can violate the license terms of the news publishers and it can be easily shared among the end users. At the same time, the *template corpus* contains all the relevant information required by a user to construct the final corpus.

To construct the *template corpus*, the news aggregator from Google[2] is used. The reason for this choice is the vast range of languages covered by the service (70 regional editions) and also the huge number of included sources (50,000 worldwide).

For the Simurg corpus, we only consider the news in the *Top Stories* section. The reason for this is that the news found in the *Top Stories* section usually cover international events and can be found in several regional editions. In this way, by collecting news from several regional editions, the corpus will contain multilingual stories about the same event. This property makes the corpus also interesting for translation tasks.

For each regional edition, we detect the top stories on the main page and for each top story, the URLs and the headlines of the news listed below that specific story will be detected (headlines will not be stored). In addition to the URLs of the news, we also store the URLs retrieved by *Internet Archive Wayback Machine* which is a digital archive of the World Wide Web (this idea is also used be Hermann et al. in the context of QA systems [8]). The *Wayback Machine* stores several snapshots of a web page across time, from which we retrieve the latest version (URLs also contain a timestamp of the form `YYYYMMDDhhmmss`). Often (and specifically in the news domain) the content of the web pages change and get updated. Using the Wayback URLs, each instance in the corpus will point to a resource by which it is guaranteed that its content will not be changed. This is, especially in the case of test corpora, of great importance to assure that all researchers report their results on the same corpus. It should also be noted that the Wayback URLs cannot be retrieved for all pages. In such cases, we only store the common URLs of the pages, as they could still be used for training purposes.

Storing and publishing the headlines may violate the license terms of the publishers. To avoid this, we proceed as follows. First, we fetch the content of the pages. The Wayback URL will be used to get the HTML content of the pages. Otherwise we use the common URLs. After parsing the HTML content of the page, tags such as `<style>`, `<script>` and `<title>` are eliminated. We then use the detected headline as a hint to find the HTML element containing the headline of the news article. For this, we traverse the whole DOM tree and perform an exact match search. The result could be a list of HTML elements. For each candidate HTML element, we automatically construct a CSS selector, by means of the element attributes such as `class`, `id` or `itemprop`. The final CSS selector will be the one which is valid and returns a unique HTML element containing the headline. The headline, as well as the fetched HTML content, will both be ignored and only the constructed CSS selector will be stored in the corpus.

As an additional field, we also store a timestamp representing the insertion time of the articles in the database. Although the publication date would be a more attractive candidate, experiments with state of the art solutions [17] did not show reliable results. The timestamps could specifically be used to derive corpora for tasks such as *topic tracking* [9] and *event detection* [22].

To persist the *template corpus*, we use Redis[3], which is an in-memory data structure store, used as a database, cache, and message broker. Each entry in the database is a `(key, value)` pair where the key is the URL of the news, and the value is a data structure in Redis called a *Hash* which is composed of fields associated with values. An example of a possible `(key, value)` pair can be seen in Table 1.

Table 1: Example of an Instance in Redis

key	http://www.sacbee.com/news/...
id	3409c881-8856-49fd-a1c0
url	http://www.sacbee.com/news/...
wayback_url	not found
headline_selector	h1.title
timestamp	2016-06-22T00:55:25

For each language, we define a separate database in Redis. To create the *template corpus* we provide a tool powered by Docker[4] that automatically setups the Redis server and databases locally and stores the collected news. Using this tool the corpus can be extended at will and if necessary new languages can be added (4 languages are supported as

[1]https://github.com/pasmod/simurg
[2]https://news.google.com/
[3]http://redis.io
[4]https://www.docker.com/

default: English, German, Italian and French). At the time of this publication the *template corpus* contains approximately 100,000 documents (25,000 documents for each language).

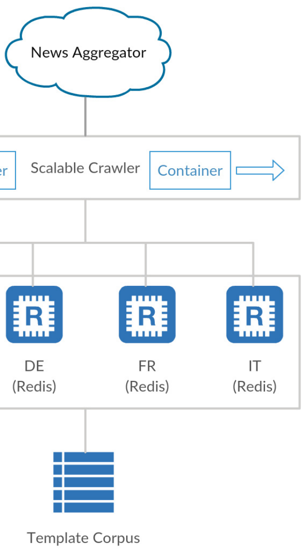


Figure 1: Architecture of the Crawler

This tool also contains a periodic scheduler that triggers a crawl every t seconds (default value $t = 300$). As the URLs are globally unique, we use them to check duplicates in each crawl and ignore the URLs that are already added to the database. This improves the performance of the system significantly.

3. CORPUS POPULATION

The *template corpus* contains all the required information to construct the final corpus. As the final corpus contains information that could violate the license agreement of the news publishers, we encourage the end users not to publish this corpus but instead only the *template corpus*. This means that the final corpus should always be constructed locally using of the *template corpus*.

All documents in the final corpus are stored as JSON documents consisting of several fields (See Table 2 for an example).

Fields such as `id`, `timestamp`, `url` and `wayback_url` are taken one-to-one from the *template corpus*. The field `lang` is determined based on the database in Redis from which the JSON file is being constructed and the field `headline` is populated by applying the exiting `headline_selector` field from the *template corpus* on the fetched HTML document.

To extract the content of the news articles we use an automatic approach introduced in [19]. This approach can be summarized as follows. First the HTML content is split into visual blocks using specific tags such as `<div>`, `<p>` or `<h1>`. Two kinds of features are then used to train a logistic regression [13] classifier: model features and semantic features.

Model features include text density [10] (average number of tokens per 80 character line), link density [10] (anchor text percent) and smoothed tag ratio [24] (ratio of the text length to the number of tags in the block).

Semantic features consist of the tokens in the `id` and `class` attributes of the tags in each block. Using a trainings dataset, for each feature its content to no-content odds ratio is calculated. Finally the ones with an odds ratio larger than 2.5 that appear in more than 1% of the blocks are selected. Using this approach, features such as `menu`, `top` and `header` were classified as non-content and features such as `comment`, `author` and `thread` were classified as content features.

Table 2: Example of a JSON Document

id	3409c881-8856-49fd-a1c0
timestamp	2016-07-11T12:09:11
lang	en
url	http://www.sacbee.com/news/...
wayback_url	None
headline	Natomas office park asks pastor who praised Orlando massacre to move out
body	The Natomas office park that houses Verity Baptist Church, whose pastor praised the recent massacre at a gay nightclub in Orlando, Fla., has asked the church to move out...

For the above-introduced approach, a F_1 score of nearly 0.9 is reported [19] which is sufficient for our use case.

It is also mentionable that in the tool provided to populate the *template corpus*, if the process of populating the template corpus is interrupted, it can be easily resumed from the interrupted point.

4. CONCLUSIONS

In this work, we introduced our approach to automatically create a corpus for abstractive single document summarization. Our methodology consisted of two steps. First, we created a template corpus as the shareable part of the corpus by crawling the news aggregator from Google, and in the second phase, we used the information in the template corpus to create the final corpus. For each document in the corpus, we defined the headline of the news article as the abstractive summary of the document.

The provided corpus can be used for the task of automatic single document summarization but at the same time it can be used as a base corpus to derive other corpora in fields such as sentence compression (for instance by clustering similar headlines where in each pair the source headline is always longer than the target headline) or topic tracking (for instance by collecting documents belonging to the same topic in a predefined time interval).

For future work, we plan to support more languages, perform an extrinsic evaluation of the corpus using existing summarization techniques and publish derived corpora for sentence compression and sentence level paraphrasing using the Simurg corpus.

5. ACKNOWLEDGMENTS

This work was funded by the German Federal Ministry of Economics and Technology under the ZIM program (Grant No. KF2846504).

6. REFERENCES

[1] M. B. Almeida, M. S. C. Almeida, andré F. T. Martins, H. Figueira, P. Mendes, and C. Pinto. Priberam Compressive Summarization Corpus: A New Multi-Document Summarization Corpus for European Portuguese. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, may 2014.

[2] J. A. Aslam, F. Diaz, M. Ekstrand-Abueg, R. McCreadie, V. Pavlu, and T. Sakai. TREC 2015 Temporal Summarization Track Overview. In *Proceedings of The Twenty-Fourth Text REtrieval Conference, TREC 2015, Gaithersburg, Maryland, USA, November 17-20, 2015*, 2015.

[3] Z. Cao, W. Li, S. Li, and F. Wei. AttSum: Joint Learning of Focusing and Summarization with Neural Attention. *CoRR*, abs/1604.00125, 2016.

[4] N. Chatterjee and P. K. Sahoo. Random Indexing and Modified Random Indexing based Approach for Extractive Text Summarization. *Computer Speech & Language*, 29(1):32–44, 2015.

[5] T. Crowley and C. Bowern. *An Introduction to Historical Linguistics.* OUP USA, 2010.

[6] H. T. Dang. Overview of DUC 2006. In *Proc. Document Understanding Workshop*, page 10, 2006.

[7] D. Graff and C. Cieri. English Gigaword. Linguistic Data Consortium, Philadelphia, 2003.

[8] K. M. Hermann, T. Kočiský, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. Teaching Machines to Read and Comprehend. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[9] J. Huang, M. Peng, H. Wang, J. Cao, W. Gao, and X. Zhang. A Probabilistic Method for Emerging Topic Tracking in Microblog Stream. *World Wide Web*, pages 1–26, 2016.

[10] C. Kohlschütter, P. Fankhauser, and W. Nejdl. Boilerplate Detection Using Shallow Text Features. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 441–450, New York, NY, USA, 2010. ACM.

[11] C.-Y. Lin. ROUGE: A Package for Automatic Evaluation of summaries. In *Proc. ACL workshop on Text Summarization Branches Out*, page 10, 2004.

[12] H. P. Luhn. The Automatic Creation of Literature Abstracts. *IBM J. Res. Dev.*, 2(2):159–165, Apr. 1958.

[13] P. McCullagh and J. Nelder. *Generalized Linear Models, Second Edition.* Taylor & Francis, 1989.

[14] P. Modaresi and S. Conrad. On Definition of Automatic Text Summarization. In *The Second International Conference on Digital Information Processing, Data Mining, and Wireless Communications (DIPDMWC2015)*, page 33, 2015.

[15] R. Nallapati, B. Xiang, and B. Zhou. Sequence-to-Sequence RNNs for Text Summarization. *CoRR*, abs/1602.06023, 2016.

[16] J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, Q. V. Le, and A. Y. Ng. On Optimization Methods for Deep Learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 265–272, 2011.

[17] L. Ostroumova Prokhorenkova, P. Prokhorenkov, E. Samosvat, and P. Serdyukov. Publication Date Prediction Through Reverse Engineering of the Web. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, WSDM '16, pages 123–132, New York, NY, USA, 2016. ACM.

[18] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318. Association for Computational Linguistics, 2002.

[19] M. E. Peters and D. Lecocq. Content Extraction Using Diverse Feature Sets. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13 Companion, pages 89–90, New York, NY, USA, 2013. ACM.

[20] D. Radev, S. Teufel, H. Saggion, W. Lam, J. Blitzer, A. Celebri, E. Drabek, D. Liu, H. Qi, and T. Allison. SummBank 1.0. 2003.

[21] A. M. Rush, S. Chopra, and J. Weston. A Neural Attention Model for Abstractive Sentence Summarization. In L. Màrquez, C. Callison-Burch, J. Su, D. Pighin, and Y. Marton, editors, *EMNLP*, pages 379–389. The Association for Computational Linguistics, 2015.

[22] M. Schinas, S. Papadopoulos, G. Petkos, Y. Kompatsiaris, and P. A. Mitkas. Multimodal Graph-based Event Detection and Summarization in Social Media Streams. In *Proceedings of the 23rd ACM International Conference on Multimedia*, MM '15, pages 189–192, New York, NY, USA, 2015. ACM.

[23] J. Ulrich, G. Murray, and G. Carenini. A Publicly Available Annotated Corpus for Supervised Email Summarization. In *AAAI08 EMAIL Workshop*, Chicago, USA, 2008. AAAI.

[24] T. Weninger, W. H. Hsu, and J. Han. CETR: Content Extraction via Tag Ratios. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 971–980, New York, NY, USA, 2010. ACM.

[25] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. *arXiv preprint arXiv:1502.03044*, 2(3):5, 2015.

[26] R. Zhang, P. Isola, and A. A. Efros. Colorful Image Colorization. *arXiv preprint arXiv:1603.08511*, 2016.

4 Automatic Keyphrase Extraction

Automatic keyphrase extraction aims to automatically extract the most salient keyphrases/keywords of a textual document. Although, by definition, it might not be considered as an automatic text summarization problem, it can be incorporated as a critical component in extractive and abstractive approaches to identify the most salient information items of the underlying text. Meanwhile, keyphrase and keyword extraction have been established as essential components in almost every machine learning API [1], such as the Alchemy API [2] or the Google Natural Language API [3].

Automatic keyphrase extraction has been applied to many different domains such as paper abstracts [Hul03], scientific papers [NK07], technical reports [WPF$^+$99], news articles [WX08], web pages [GGL09], meeting transcripts [LPLL09], emails [DWPP08] and live chats [HN14].

A keyphrase extraction algorithm usually consists of two steps. In the first step, a list of candidate keyphrases is extracted from the document, and, in the second step, the candidates are ranked according to some predefined measure and are defined as the final keyphrases of the document. Typically, selecting the candidate keyphrases is accomplished by defining heuristics [MT04b] or training a binary classifier [Tur00] that classifies a keyphrase as either a candidate or not.

In Section 4.1 we introduce a keyphrase extraction algorithm that is different from the existing approaches that define a keyphrase either as a candidate or not a candidate, and which represents the set of keyphrases in a document as a fuzzy set where each keyphrase has a degree of being a candidate keyphrase. An extended version of this work will be presented in Section 4.2.

[1]Application Programming Interface
[2]http://www.alchemyapi.com/
[3]https://cloud.google.com/natural-language/

4.1 From Phrases to Keyphrases: An Unsupervised Fuzzy Set Approach to Summarize News Articles

Pashutan Modaresi and Stefan Conrad. From Phrases to Keyphrases: An Unsupervised Fuzzy Set Approach to Summarize News Articles. In *Proceedings of the 12th International Conference on Advances in Mobile Computing and Multimedia*, MoMM '14, pages 336–341, 2014.
Contributions: The research and the preparation of this manuscript was done entirely by the main author under the supervision of Prof. Conrad. **Status**: Published.

The important keywords/keyphrases of a document are suitable indicators for the topic of a document. Although the salient information in a document cannot be captured using a set of keyphrases, they can provide a concise topical summary of a document. Probably, the most known application of keywords/keyphrases is in the context of scientific publications where the authors manually present a set of important keyphrases for their publications. In the domain of scientific publications, keyphrases are mainly used to index the publications for search applications, and they also provide some hints to the readers about the topics of the articles.

Extracting the keywords/keyphrases of a document manually is a time-consuming task. For this, the document has to be read and understood, and then significant keywords have to be extracted from the document and ranked according to their importance in the text. In the context of news articles, keyphrases are mainly used in applications such as *topic detection and tracking* (detecting the appearance of new topics and tracking the reappearance and evolution of them) [ACD+98] and text summarization [Sar14]. In the context of text summarization, keyphrases can either be represented in their raw form to the user, or they can be used as a part of extractive or abstractive approaches.

Previous work in automatic keywords/keyphrases extraction has mainly treated the set of keyphrases of a document as a crisp set, meaning that a phrase in a document was either a keyphrase or not. In this work, we model the set of phrases in a document as a fuzzy set, where each phrase has a specific membership value to this set. The ones with higher membership values are then regarded as keyphrases or keywords of the document.

Moreover, to assess the quality of our proposed approach, we introduce a simple evaluation technique inspired by the Turing test, where a rater decides in a binary fashion whether the extracted keyphrases are extracted by a machine or a human. As in the Turing test, the misjudgments of the rater are an indicator of the sound quality of the extracted keyphrases.

From Phrases to Keyphrases: An Unsupervised Fuzzy Set Approach to Summarize News Articles

Pashutan Modaresi
pressrelations GmbH
Klosterstraße 112
40211 Düsseldorf, Germany
pashutan.modaresi@pressrelations.de

Stefan Conrad
Heinrich-Heine University
Institute of Computer Science
40225 Düsseldorf, Germany
conrad@cs.uni-duesseldorf.de

ABSTRACT

Automatic keyphrase extraction aims at extracting a compact representation of a single document which can be used for various applications such as indexing, classification or summarization. Existing methods for keyphrase extraction usually define the set of phrases of a document as a crisp set and by scoring the phrases, they select the keyphrases of the document. In this work we define the set of phrases inside a document to be a fuzzy set, and based on the membership values of the phrases, we select the ones with higher membership values as the keyphrases of the document. Moreover we propose a novel evaluation method inspired by the Turing test which can be used for extractive summarization tasks.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Summarization*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Text analysis*

General Terms

Algorithms, evaluation, text mining

Keywords

Keyphrase extraction, fuzzy set theory, text summarization, summary evaluation

1. INTRODUCTION

A Phrase is a group of words acting as a single part of speech. In automatic keyphrase extraction the goal is to automatically extract the most important and significant phrases of a document. Depending on the use case, the extracted phrases can then be used for searching inside a collection of documents, summarizing single or multiple documents [23], document classification [5] and indexing [6]. In some publications the terms *keyword* and *keyphrase* are used interchangeably. It should be noticed that a keyword consists of only one token, but keyphrases have a length equal or greater than one.

Automatic Text Summarization is an active field of research which aims to deal with the problem of immense available textual information and find a solution to deal with this massive amount of data. Particularly in media branch, summarization plays a significant role and results in great time-savings, while increasing the work efficiency. Extracting keyphrases from a document, can help the reader to instantly attain an overview of the subject matter and contents of a document [4]. Moreover, the extracted keyphrases can be used to extract the most important sentences in the document.

Many approaches for automatic keyphrase extraction define the set of keyphrases of a document as a crisp set. This means that each candidate phrase inside the document will either belong to the class *keyphrase* or to the class *not-keyphrase*. We believe that a fuzzy set containing all the phrases of the document with different membership values is a more appropriate mean to represent the phrases of a document. In this way, the decision whether a phrase belongs to the set of keyphrases or not, can be easily made based on its membership value to the fuzzy set.

In this work we use an unsupervised fuzzy set approach to automatically extract the keyphrases from a single document. In Section 2 we will shortly introduce the related work. A formal definition of phrases and keyphrases based on the fuzzy sets will be introduced in Section 3. For each phrase in a document we will define a membership degree that determines its significance in the document. The calculation of membership values will be explained in Section 4. To evaluate our algorithm we will introduce a novel evaluation method (Section 5) inspired by the Turing test, and we will represent our evaluation results in Section 6. Finally, we conclude our work in section 7.

2. RELATED WORK

Automatic keyphrase extraction has been applied to various sources like scientific papers [10], web pages [8] and news articles [20]. Usually the algorithms extract a set of phrases in the document as the candidate keyphrases and using a scoring algorithm the importance of the candidate keyphrases will be determined.

For extracting the candidate keyphrases many approaches such as extraction of noun phrases [1], elimination of stopwords to obtain phrases [18] or extraction of words with specific part-of-speech tags [13] have been proposed. After extracting candidate keyphrases, they will be assigned

MoMM '14, December 08 - 10 2014, Kaohsiung, Taiwan
Copyright 2014 ACM 978-1-4503-3008-4/14/12 ...$15.00
http://dx.doi.org/10.1145/2684103.2684117

a score either using a supervised or an unsupervised approach. In supervised approaches the problem of extraction of keyphrases is usually formulated as a binary classification problem [22]. However, this approach has the disadvantage that the phrases will either be classified as *keyphrase* or *not-keyphrase* and their importance will not be comparable to each other. In order to overcome this problem supervised algorithms have been designed which are able to learn a phrase rater [9].

Several approaches have also been suggested for unsupervised keyphrase extraction. Graph-based approaches represent the content of the document as a graph and based on the edges connecting the nodes of the graph, determine the importance of the words inside the document [17]. Additionally clustering approaches are used to cluster the candidate phrases based on their semantic relationship and in this way it will be ensured that the extracted keyphrases reflect the content of the document [14].

Many other unsupervised phrase ranking algorithms use heuristics based on various features like length and frequency of the phrases [2] or the co-occurrence statistics of the terms inside the phrases [15].

3. PHRASES AS A FUZZY SET

In linguistics, a *phrase* is a group of words acting as a single part of speech. Phrases usually contain a keyword, through which their type and category can be identified. This word is known as the *head* of the phrase. Phrases where their head is a noun are called *noun phrases* [16].

Identifying the phrases in a document is not the main focus of this work and can be fulfilled in various ways. For summarizing news articles through keyphrases, understandable and syntactically correct phrases are needed to be extracted. For this purpose, we use a *Maximum Entropy Model* to extract the chunks inside the documents and from the extracted chunks the ones that are noun phrases are selected.

The extracted phrases, form now our set of phrases $\mathcal{K}$ which we define as a fuzzy set:

$$\mathcal{K} = \big\{\big(x, \mu_{\mathcal{K}}(x)\big) \big| x \in K, \;\; \mu_{\mathcal{K}}(x) \in [0,1]\big\}. \tag{1}$$

In Equation 1, K is a classical set that contains all the phrases in the document, and $\mu_{\mathcal{K}}(x)$ specifies the grade or degree to which any element $x \in K$ belongs to the fuzzy set $\mathcal{K}$. Larger values of $\mu_{\mathcal{K}}(x)$ indicate higher degrees of membership [11].

Intuitively, the membership value of each phrase will represent its significance inside the document. As already mentioned, we will only consider the noun phrases. Formally, this means that for each $x \in K$ where x is not a noun phrase, we will have $\mu_{\mathcal{K}}(x) = 0$.

In order to find the keyphrases more easily using a threshold and also to make the membership values among different documents comparable, we use the normalized membership values of the noun phrases. For this we compute the *height* of the fuzzy set $\mathcal{K}$ [11]:

$$h(\mathcal{K}) = \sup_{x \in K} \mu_{\mathcal{K}}(x), \tag{2}$$

and divide $\mu_{\mathcal{K}}(x)$ for all $x \in K$ by height of the fuzzy set $\mathcal{K}$. In this way we obtain the normalized fuzzy set with $h(\mathcal{K}) = 1$.

We define the set of keyphrases of a document to be the *α-cut* of the fuzzy set $\mathcal{K}$, defined as follows:

$$[\mathcal{K}]^{\alpha} = \begin{cases} \{x \in X |\; \mu_{\mathcal{K}}(x) \geq \alpha\} & 0 < \alpha \leq 1 \\ cl(supp(\mathcal{K})) & \alpha = 0 \end{cases}, \tag{3}$$

where $cl(supp(\mathcal{K}))$ is the closure of the support of $\mathcal{K}$ (We will not go into details of this definition as the case $\alpha = 0$ does not occur in our application).

Using the above definitions, the problem of finding the keyphrases of a document will be reduced to computing the membership values for the elements of the set K and determining the value of α.

4. MEMBERSHIP FUNCTIONS

In our work we analyze 3 features of the phrases in order to determine their significance within the documents. Length of keyphrases, their token frequency and skip-bigram co-occurrence frequency. Based on these features a final membership value (which is a weighted sum of membership values) will be assigned to each phrase. At the end, phrases with higher membership values will be selected as the keyphrases of the document.

4.1 Length of Keyphrases

During our experiments we have noticed that human annotators usually tend to select phrases of token length 2 or 3 as keyphrases. There are of course also cases where keyphrases of token length one or greater than 3 are selected.

In order to force the system to select keyphrases of a specific length, a size threshold can be hard-coded into the algorithm [21]. However, this will have the disadvantage that some keyphrases will be completely ignored (although they might be significant).

In order to assure the before mentioned length property of keyphrases, we use a *generalized bell membership function* which is defined as follows [7]:

$$f_1(x) = bell(x; a, b, c) = \frac{1}{1 + |\frac{x-c}{a}|^{2b}} . \tag{4}$$

In Equation 4, c and a represent the center and width of the membership function respectively. Moreover b is a positive number that controls the slope at the crossover point of the function. To understand the reason behind this function choice, the graph of the function represented in Figure 1 will be helpful. The solid curve represents the bell function $bell(x; 2, 2, 2.5)$. The dotted curves represent the impact of the change of parameter a, and the dashed curves show the impact of the change of the parameter b.

By selecting $c = \frac{2+3}{2} = 2.5$ which is the average of the desired length, phrases with length 2 or 3 will be assigned higher membership values than the other phrases. Note that bigger values of b will increase the slope at the crossover point of the function and will cause in greater differences between the membership values of phrases with length 2 or 3, and phrases of other lengths. Due to this property, we recommend small values for b such as $b = 1$ or $b = 1.5$, which will reduce the amount of punishment for very small or very long phrases.

Notice that the parameters a, b and c are not dependent of the document under analysis and have to be set only once. Given a large enough corpus, these parameters can also be learned using an appropriate learning algorithm.

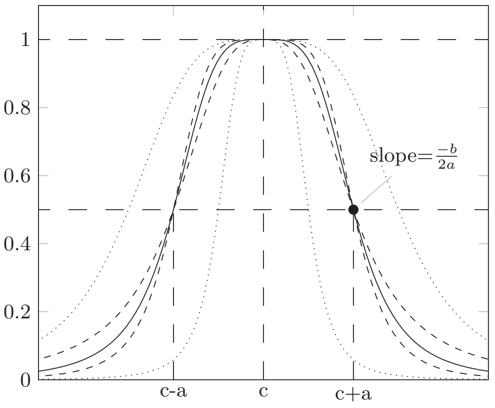


Figure 1: Graph of $bell(x, 2, 2, 2.5)$

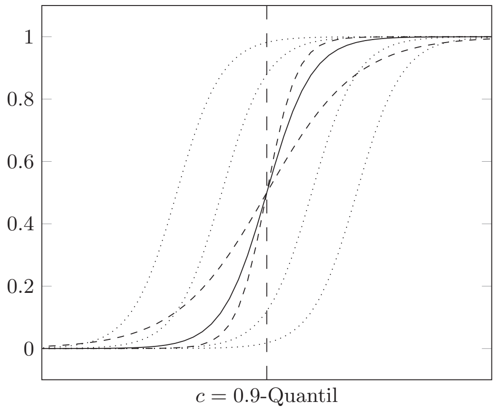


Figure 2: Graph of $sigmf(x, 2, 15)$

4.2 Token Frequency

Another property of phrases that plays a role in considering them as a keyphrase is their frequency. Phrases that contain tokens which occur frequently in the document are more likely to be keyphrases.

Let $p = t_1 \dots t_m$ be a phrase consisting of m distinct tokens. We define the frequency of a phrase inside a document as the sum of the frequencies of its constituting tokens. Formally this will be written as:

$$n_p = \sum_{i=1}^{m} n_{t_i}. \tag{5}$$

Note that n_{t_i} is actually the frequency of token t_i inside all available phrases in the document and not the document itself. In this way the effect of stop-words resulting in high frequencies for phrases will be avoided.

In order to model the property that phrases with frequent tokens are more likely than the ones with less frequent tokens to be keyphrases, we use a *sigmoid membership function* defined as follows [11]:

$$f_2(x) = sigmf(x; a', c') = \frac{1}{1 + e^{-a'(x-c')}}. \tag{6}$$

The parameter a' can be specified by the user ($a' = 1$ or $a' = 2$ is recommended). Depending on the sign of the parameter a', the function can be open right or left and thus will be appropriate for representing quantities such at *very frequent* or *very rare*. In order to determine the value of the parameter c', first the frequencies of all phrases (n_p) have to be computed. We compute then the value of c' as the p-Quantil of the calculated frequencies, with $p = 0.9$.

Note that we did not consider the effect of length of phrases in the definition of the membership function. This will of course result in higher membership values for longer phrases. However this effect will be canceled by means of the generalized bell function which we already defined in section 4.1.

In Figure 2 the solid curve represents the sigmoid function $sigmf(x; 2, 15)$. The dotted curves represent the impact of the change of parameter c' and the dashed curves show the impact of the change of the parameter a'.

As already stated, the parameters a' and c' can also be learned, given a large enough data set.

4.3 Skip-Bigram Co-Occurrence Frequency

Another feature which we believe to play a significant role in identifying the keyphrases of a document is the *skip-bigram co-occurrence frequency*. Assuming a phrase $p = t_1 \dots t_m$ consisting of m tokens, *skip-bigrams* are any pair of tokens inside the phrase, with arbitrary gaps [12]. As an example the phrase $p = t_1t_2t_3$ will consist of the following skip-bigrams: $(t_1, t_2), (t_1, t_3), (t_2, t_3)$. The total number of skip-bigrams inside a phrase of length m can be computed as the *2-combination* of the set of its tokens which is equal to $\binom{m}{2} = \frac{m!}{(m-2)!2!}$.

For each skip-bigram inside the document we calculate the frequency of it among the set of skip-bigrams of available phrases. Finally, the skip-bigram co-occurrence frequency of a phrase will be computed as the sum of the frequencies of all its constituting skip-bigrams. This will be formally written as:

$$n_p^{skip} = \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} n_{t_i,t_j}, \tag{7}$$

where n_{t_i,t_j} denotes the frequency of the skip-bigram (t_i, t_j). After computing the n_p^{skip} values for all phrases inside the document, we again apply the sigmoid membership function to the computed values. As in the previous section, we select the parameter c'' to be the p-Quantil of the n_p^{skip} values with $p = 0.9$. We also recommend the parameter a'' to be equal to 1 or 2. The skip-bigram co-occurrence frequency of a phrase x will be denoted as $f_3(x)$.

4.4 Final Membership Value Computation

As already mentioned in Section 3, our goal was to assign a membership value to each noun phrase inside the document. We have splitted this problem into 3 sub-problems and using 3 features, namely the phrase length f_1, token frequency f_2 and skip-bigrams co-occurrence frequency f_3, 3 membership functions have been defined. Finally, we define the membership value of a phrase x inside the set K to be the weighted sum of the 3 introduced membership functions:

$$\begin{aligned} \mu_{\mathcal{K}}(x) = \omega_1 \cdot bell(f_1(x); a, b, c) \\ + \omega_2 \cdot sigmf(f_2(x); a', c') \\ + \omega_3 \cdot sigmf(f_3(x); a'', c''), \end{aligned} \tag{8}$$

Table 1: Confusion Matrix of a Rater

		Predicted Labels		
		HG	MG	Total
Actual Labels	HG	A	B	$A+B$
	MG	C	D	$C+D$
	Total	$A+C$	$B+D$	$2n$

with $\sum_{i=1}^{3} w_i = 1$.

In our experiments we use $w_1 = 0.5$, $w_2 = 0.25$ and $w_3 = 0.25$. The keyphrases from the document can now be selected using the Equation 3. For larger values of α the algorithm will select the phrases which are most likely to be a keyphrase inside a document. It should be also clear that by setting a higher value for α the number of extracted keyphrases from the document will be decreased and vice versa.

5. IMITATION GAME INSPIRED EVALUATION

In this section we introduce our evaluation method inspired by the imitation game suggested by Alan Turing in 1950 to decide whether a machine is intelligent or not [19].

Assuming a collection $D = \{d_1, \ldots, d_n\}$ of n documents, our goal is to compare the automatically extracted keyphrases to the ones extracted by the humans. For each document d_i with $1 \leq i \leq n$, let $K_i^{\mathcal{M}} = \{x_{i1}, \ldots, x_{im}\}$ and $K_i^{\mathcal{A}} = \{x'_{i1}, \ldots, x'_{im}\}$ be the set of manually and automatically extracted keyphrases respectively. Note that the set of automatically and manually extracted keyphrases, both have the same cardinality, namely m.

The evaluation process consists of n iterations, and two human raters $\mathcal{R}_1$ and $\mathcal{R}_2$. In each iteration a document d_i with its associated keyphrases $K_i^{\mathcal{M}}$ and $K_i^{\mathcal{A}}$ is shown to the raters (The raters are not told which of the keyphrases are human-generated and which are machine-generated). The raters are asked to classify the keyphrases either to the class *human-generated (HG)* or to the class *machine-generated (MG)*.

In this way the set of labels of the classification problem will be $C = \{\text{human-generated}, \text{machine-generated}\}$ and for each set of keyphrases K_i the raters have to determine its label $c(K_i)$. Note that the raters are not allowed to assign the same label for both sets of keyphrases. This means that assuming the sets $K_i^{\mathcal{A}}$ and $K_i^{\mathcal{M}}$, we will have $c(K_i^{\mathcal{M}}) \neq c(K_i^{\mathcal{A}})$. Finally, for each rater a confusion matrix will be constructed (see Table 1).

Notice that on the bottom right corner of the Table 1, the total number of classifications is written as $2n$ and not n. For each document d_i a rater will be encountered with two sets of keyphrases, namely $K_i^{\mathcal{M}}$ and $K_i^{\mathcal{A}}$. The rater will classify one of the sets as *machine-generated* and the other one as *human-generated*. This means that in each iteration a rater performs two classifications. As we have a total number of n documents, the total number of classification at the end will be equal to $2n$.

In order to assess the agreement on the classification task, we use the *Cohen's Kappa Coefficient* [3] which is a mean to measure the inter-rater agreement and is defined as follows:

$$\kappa = \frac{Pr(a) - Pr(e)}{1 - Pr(e)}. \tag{9}$$

In Equation 9, $Pr(a)$ denotes the observed percentage of agreement and $Pr(e)$ is the expected percentage of agreement. We consider $\kappa \geq 0.60$ to be a good amount of agreement between the raters. For the case $\kappa < 0.60$ more documents have to be added to the collection, until the desired value is reached.

Table 2: Confusion Matrix $\mathcal{R}_1$

		Predicted Labels		
		HG	MG	Total
Actual Labels	HG	12	18	30
	MG	18	12	30
	Total	30	30	60

Table 3: Confusion Matrix of $\mathcal{R}_2$

		Predicted Labels		
		HG	MG	Total
Actual Labels	HG	14	16	30
	MG	16	14	30
	Total	30	30	60

Finally, we compute the average accuracy of the raters $\tilde{\alpha}_{\mathcal{H}}$ as follows:

$$\tilde{\alpha}_{\mathcal{H}} = \frac{1}{4n} \sum_{i=1}^{2} A_i + D_i. \tag{10}$$

In Equation 10, A_i represents the number of times that rater $\mathcal{R}_i$ correctly classified a human-generated set and D_i denotes the number of times that a machine-generated set has been correctly classified by $\mathcal{R}_i$.

Based on the average accuracy of the raters, the accuracy of the keyphrase extraction algorithm can be determined. Intuitively this means that in the cases where a rater classifies a set of keyphrases as human-generated, but the set is actually machine-generated, this would mean that the keyphrase extraction algorithm was successful in imitating the humans and has extracted keyphrases, even better than the human-generated ones. Based on this idea we define the accuracy of the keyphrase extraction algorithm $\tilde{\alpha}_{\mathcal{M}}$ to be the *misclassification rate* of the human raters:

$$\tilde{\alpha}_M = 1 - \tilde{\alpha}_{\mathcal{H}}. \tag{11}$$

The lower the average accuracy of human raters, the higher the accuracy of the algorithm will be.

6. EVALUATION RESULTS

For our experiments we have collected 30 English news articles from *BBC News* and for each document inside our collection we have manually extracted the top 10 keyphrases from the documents. The extraction process has been performed by two annotators, and from the extracted keyphrases of each annotator, an intersection set of size 10 has been selected. In this way we assure the reliability of the extracted keyphrases. Additionally we used our algorithm to extract the top 10 keyphrases from the documents.

For each document d_i, with $1 \leq i \leq 30$ we asked the raters to decide which element of the pair $(K_i^{\mathcal{H}}, K_i^{\mathcal{M}})$ is machine-generated and which one is human-generated (the actual labels of the keyphrases are of course not shown to the raters).

The confusion matrices of the raters $\mathcal{R}_1$ and $\mathcal{R}_2$ can be seen in Table 2 and Table 3 respectively.

To compute the *Cohen's Kappa Coefficient*, we collected the following information about the ratings:

Act now on climate change or see costs soar, White House says
(Reuters) - Putting off expensive measures to curb climate change will only cost the United States more in the long run, the White House said on Tuesday in a report meant to bolster a series of actions President Barack Obama has proposed to address global warming.
"Each decade we delay acting results in an added cost of dealing with the problem of an extra 40 percent," said Jason Furman, chairman of Obama's Council of Economic Advisers.
"We know way more than enough to justify acting today," Furman told reporters.
The report drew its conclusions from 16 economic studies that modeled the costs of climate change. It was released as the U.S. Environmental Protection Agency holds public hearings on its plan to cut carbon emissions from power plants - the centerpiece of Obama's climate action plan.
Business groups have said the EPA's plan would hurt jobs in the coal sector and harm the U.S. economy.
The White House and environmental groups have pushed back against that argument.

Figure 3: A snippet of a news article

1. Number of times that $\mathcal{R}_1$ and $\mathcal{R}_2$ classified the same set of keyphrases as human-generated = 24
2. Number of time that $\mathcal{R}_1$ and $\mathcal{R}_2$ classified the same set of keyphrases as machine-generated = 24
3. Number of times that $\mathcal{R}_1$ classified a set of keyphrases as human-generated but $\mathcal{R}_2$ classified the same set as machine-generated = 6
4. Number of times that $\mathcal{R}_2$ classified a set of keyphrases as human-generated but $\mathcal{R}_1$ classified the same set as machine-generated = 6

In total the number of observed agreements will be 48 (80% of the observations) and the number of expected agreements by chance is 30 (50% of the observations). Thus for *Cohen's Kappa Coefficient* we will have $\kappa = 0.6$ which is considered to be *good*.

Using the information contained in Table 2 and Table 3 we also compute the average accuracy of the raters which is $\tilde{\alpha}_{\mathcal{H}} = 0.4\overline{3}$. Finally, the accuracy of the algorithm will be $\tilde{\alpha}_{\mathcal{M}} = 1 - 0.4\overline{3} = 0.5\overline{6}$.

As an example for the output of our algorithm we extracted the top 10 keyphrase of a news article[1] from *Reuters*. A snippet of the news article can be seen in Figure 3. The top 10 automatically extracted keyphrases can be seen in Figure 4.

7. CONCLUSIONS

In this work we have introduced a fuzzy set approach to extract keyphrases from news articles. We used sentence chunking to extract phrases from the documents and used a weighted membership function to determine the importance of the phrases. Finally we introduced a novel evaluation approach inspired by the Turing test.

[1]http://www.reuters.com/article/2014/07/29/us-usa-climatechange-idUSKBN0FY0V820140729

energy and climate change, White House and environmental groups, natural gas transmission and distribution system, climate costs, White House, former New York City Mayor Michael Bloomberg, agriculture and food production, U.S. Environmental Protection Agency, Energy Secretary Ernie Moniz, methane emissions

Figure 4: Extracted keyphrases from the news article

Using the introduced approach we were able to partly imitate the humans in the way they extract keyphrases from documents. This led to the situation where in some cases our raters were unable to differentiate between the human-generated and machine-generated keyphrases. Also in many cases the machine-generated keyphrases were wrongly classified by the judges as human-generated keyphrases which is an indicator of the acceptable performance of our algorithm.

For future work, we plan to combine our introduced approach with a rule-based approach to assure the quality of the extracted keyphrases. Furthermore we intend to perform an in-depth analysis of our evaluation method and generalize it to further types of summarization such as *sentence extraction* or even *abstractive summarization*.

8. ACKNOWLEDGMENTS

The authors would like to thank Philipp Gross for his valuable comments and suggestions to improve the quality of the paper. They are also very grateful to Sara Shahidi, Ben Zimmermann and Arman Khaleghi for helping them to perform the experiments.

9. REFERENCES

[1] K. Barker and N. Cornacchia. Using noun phrase heads to extract document keyphrases. In H. Hamilton, editor, *Advances in Artificial Intelligence*, volume 1822 of *Lecture Notes in Computer Science*, pages 40–52. Springer Berlin Heidelberg, 2000.
[2] K. Barker and N. Cornacchia. Using noun phrase heads to extract document keyphrases. In *Proceedings of the 13th Biennial Conference of the Canadian Society on Computational Studies of Intelligence: Advances in Artificial Intelligence*, AI '00, pages 40–52, London, UK, UK, 2000. Springer-Verlag.
[3] J. Carletta. Assessing agreement on classification tasks: The kappa statistic. *Comput. Linguist.*, 22(2):249–254, June 1996.
[4] E. D. Avanzo, B. Magnini, and A. Vallin. Keyphrase extraction for summarization purposes: The lake system at duc-2004.
[5] L. Dolamic and C. Boyer. Key-phrase based classification of public health web pages. In C. U. Lehmann, E. Ammenwerth, and C. Nuhr, editors, *MedInfo*, volume 192 of *Studies in Health Technology and Informatics*, page 1133. IOS Press, 2013.
[6] N. Erbs, I. Gurevych, and M. Rittberger. Bringing order to digital libraries: From keyphrase extraction to index term assignment. *D-Lib Magazine*, 19(9/10), 2013.

[7] G. Feng. *Analysis and Synthesis of Fuzzy Control Systems: A Model-Based Approach*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 2010.

[8] M. Grineva, M. Grinev, and D. Lizorkin. Extracting key terms from noisy and multitheme documents. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 661–670, New York, NY, USA, 2009. ACM.

[9] D. Kelleher and S. Luz. Automatic hypertext keyphrase detection. In L. P. Kaelbling and A. Saffiotti, editors, *IJCAI*, pages 1608–1609. Professional Book Center, 2005.

[10] S. N. Kim, O. Medelyan, M.-Y. Kan, and T. Baldwin. Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pages 21–26, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[11] G. J. Klir and B. Yuan. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1995.

[12] S. Li, Y. Zhang, W. W. 0013, and C. Wang. Using proximity in query focused multi-document extractive summarization. In W. Li and D. M. Aliod, editors, *ICCPOL*, volume 5459 of *Lecture Notes in Computer Science*, pages 179–188. Springer, 2009.

[13] F. Liu, D. Pennell, F. Liu, and Y. Liu. Unsupervised approaches for automatic keyword extraction using meeting transcripts. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 620–628, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[14] Z. Liu, P. Li, Y. Zheng, and M. Sun. Clustering to find exemplar terms for keyphrase extraction. In *In Proceedings of EMNLP*, pages 257–266, 2009.

[15] Y. Matsuo and M. Ishizuka. Keyword extraction from a single document using word co-occurrence statistical information. *International Journal on Artificial Intelligence Tools*, 13:2004, 2004.

[16] P. H. Matthews. *The concise Oxford dictionary of linguistics*. Oxford paperback reference. Oxford University Press, Oxford, England, New York, 2007.

[17] R. Mihalcea and P. Tarau. TextRank: Bringing order into texts. In *Proceedings of EMNLP-04and the 2004 Conference on Empirical Methods in Natural Language Processing*, July 2004.

[18] S. Rose, W. Cowley, V. Crow, and N. Cramer. Rapid automatic keyword extraction for information retrieval and analysis, Mar. 6 2012. US Patent 8,131,735.

[19] A. M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.

[20] X. Wan and J. Xiao. Single document keyphrase extraction using neighborhood knowledge. In D. Fox and C. P. Gomes, editors, *AAAI*, pages 855–860. AAAI Press, 2008.

[21] L. Wang and F. Li. Sjtultlab: Chunk based method for keyphrase extraction. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pages 158–161, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[22] I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning. Kea: Practical automatic keyphrase extraction. In *Proceedings of the Fourth ACM Conference on Digital Libraries*, DL '99, pages 254–255, New York, NY, USA, 1999. ACM.

[23] Y. Zhang, E. Milios, and N. Zincir-heywood. A comparative study on key phrase extraction methods in automatic web site summarization. *Journal of Digital Information Management, Special Issue on Web Information Retrieval*, 2007.

4.2 Identification and Evaluation of Keyphrases: Fuzzy Set based Scoring and Turing Test Inspired Evaluation

Pashutan Modaresi and Stefan Conrad. Identification and Evaluation of Keyphrases: Fuzzy Set based Scoring and Turing Test Inspired Evaluation. In *The Second International Conference on Digital Information Processing, Data Mining, and Wireless Communications (DIPDMWC2015)*, page 107, 2015.
Contributions: The research and the preparation of this manuscript was done entirely by the main author under the supervision of Prof. Conrad. **Status**: Published.

In Section 4.1 we introduced an automatic approach to extract significant keyphrases from a textual document. We defined the set of phrases in a document as a fuzzy set and defined various membership functions to determine the significance of the phrases in the document.

We used the generalized bell function to measure the significance of the phrases based on their token length. The generalized bell function was defined as:

$$bell(x;a,b,c) = \frac{1}{1+|\frac{x-c}{a}|^{2b}} \tag{4.1}$$

where c and a represented the center and width of the membership function and b was a positive number to control the slope at the crossover point of the function. Based on our experiments on the training dataset, we manually defined the appropriate values for the parameters of the generalized bell function. This approach has the disadvantage that the estimated parameters are hard-coded and they might not be generalized to other datasets or domains. In this section, to solve this problem, we construct a small dataset to train a linear regression model that automatically learns the parameters of the generalized bell function.

The same problem also holds for the sigmoid membership functions that were used for token frequencies and skip-bigram co-occurrence frequencies of the phrases. The sigmoid membership function has also parameters that have to be learned. As in the case of generalized bell function, we use a linear regression model to learn the parameters of the sigmoid membership functions.

Moreover, in this section, we discuss our Turing test inspired evaluation method in more detail and report our results for a test set consisting of 30 English news documents evaluated by two raters. We also report the inter-rater agreement between the raters in terms of Cohen's kappa.

Proceedings of Second International Conference on Digital Information Processing, Data Mining, and Wireless Communications (DIPDMWC2015), Dubai, UAE, 2015

Identification and Evaluation of Keyphrases: Fuzzy Set based Scoring and Turing Test Inspired Evaluation

Pashutan Modaresi Stefan Conrad
Heinrich-Heine-University of Düsseldorf
Institute of Computer Science, Düsseldorf, Germany
{modaresi, conrad}@cs.uni-duesseldorf.de

ABSTRACT

Automatic keyphrase extraction aims at extracting a compact representation of a single document, which can be used for numerous applications such as indexing, classification or summarization. Existing keyphrase extraction approaches typically consist of two steps. An extraction step to select the *candidate phrases* using some heuristics and a scoring phase for ranking the extracted candidate phrases based on their importance in the text. Existing approaches to automatic keyphrase extraction mainly define the set of phrases of a document as a crisp set and by scoring and ranking the phrases, they select the keyphrases of the document. In this work we define the set of phrases in a document to be a fuzzy set, and based on the membership values of the phrases, we select the ones with higher membership values as the keyphrases of the document. Moreover we propose a novel evaluation method inspired by the Turing test, which can be used for extractive summarization tasks.

KEYWORDS

Automatic Summarization, Keyphrase Extraction, Keyword Extraction, News Summarization, Fuzzy Sets

1 INTRODUCTION

A Phrase is a group of words acting as a single part of speech. In automatic keyphrase extraction the goal is to automatically extract the most important and significant phrases of a document. Depending on the use case, the extracted phrases can then be used for searching inside a collection of documents, summarizing single or multiple documents [1], document classification [2] and indexing [3]. In some publications the terms *keyword* and *keyphrase* are used interchangeably. It should be noticed that a keyword consists of only one token, but keyphrases have a length equal or greater than one.

Automatic Text Summarization is an active field of research, which aims to deal with the problem of immense available textual information and find a solution to deal with this massive amount of data. Particularly in media branch, summarization plays a significant role and results in great time-savings, while increasing the work efficiency. Extracting keyphrases from a document, can help the reader to instantly attain an overview of the subject matter and contents of a document. Moreover, the extracted keyphrases can be used to extract the most important sentences in the document.

Many approaches to automatic keyphrase extraction define the set of keyphrases of a document as a crisp set. This means that each candidate phrase in the document will either belong to the class *keyphrase* or to the class *not-keyphrase*. We believe that a fuzzy set containing all the phrases of the document with different membership values is a more appropriate mean to represent the phrases of a document. In this way, the decision whether a phrase belongs to the set of keyphrases or not, can be easily made based on its membership value to the fuzzy set.

The process of keyphrase extraction typically consists of two steps. A *candidate selection* phase and a *keyphrase extraction* step. In the first step a set of candidate phrases will be selected from the text. This is normally done using some heuristics to determine the appropriate phrases in the text. Having the set of candidate phrases, a scoring algorithm (either supervised or unsupervised) ranks the candidate phrases regarding their importance in the context of the text and based on a threshold (either user-specified or automatically determined)

returns the ranked list of keyphrases.
Different from existing extraction approaches that introduce a unified algorithm consisting of both phases of keyphrase extraction, in this work we focus on the second step of keyphrase extraction, namely the scoring algorithm. We will assume a given set of candidate phrases that are selected either manually by human annotators or automatically using candidate selection algorithms.
We use an unsupervised fuzzy set approach to automatically score the candidate phrases of a single document. In Section 2 we will shortly introduce the related work. A formal definition of phrases and keyphrases based on the fuzzy sets will be introduced in Section 3. For each phrase in a document we will define a membership degree that determines its significance in the document. The calculation of membership values will be explained in Section 4. To evaluate our algorithm we will introduce a novel evaluation method (Section 5) inspired by the Turing test, and we will represent our evaluation results in Section 6. Finally, we conclude our work in section 7.

2 RELATED WORK

Automatic keyphrase extraction has been applied to various sources like scientific publications [4], web pages [5] and news articles [6]. Usually the algorithms extract a set of phrases in the document as the candidate keyphrases and using a scoring algorithm, the importance of the candidate keyphrases will be determined. For extracting the candidate keyphrases many approaches such as extraction of noun phrases [7], elimination of stop-words to obtain phrases [8] or extraction of words with specific part-of-speech tags [9] have been proposed. We investigate some of these approaches in more detail.
As already mentioned, heuristics are usually applied in order to extract candidate keyphrases. In [10] the document under analysis is first tokenized and from the set of available tokens, stop words will be eliminated. The remaining tokens will form the set of candidate keyphrases. In further steps of the algorithm, the tokens will be merged together to form longer candidate keyphrases. Similar to this approach is the *TextRank* [11] algorithm, which also uses tokens as candidate keyphrases. In TextRank the document is tokenized first and POS tags will be added to the tokens. Using a syntactic filter, specific tokens will be added as vertices to a graph. Finally an edge will be added between vertices that co-occur within a window of N words. In a later phase of the algorithm, longer keyphrases will be constructed based on the results of the ranking algorithm. Similar to the TextRank algorithm is the CollabRank [12] algorithm, which is designed to extract keyphrases from a set of documents (multi-document keyphrase extraction).
In [13] noun phrases are used as candidate keyphrases. Using a base noun phrase skimmer, the algorithm proceeds through a text word-by-word looking for sequences of nouns and adjectives ending with a noun and surrounded by non-noun/adjectives. This approach has the advantage that no detailed parse of sentences is needed. In a well known approach, called *KEA* [14], the extraction of candidate keyphrases is performed in three steps. In the first step, the document text will be cleaned by removing punctuation and similar non-relevant characters. In the second step the candidate keyphrases will be identified from the cleaned text, using several rules, such as:

- Candidate keyphrases have a certain maximum length
- Candidate keyphrases can not be proper names
- Candidate keyphrases can not begin or end with a stop word

In a final step, case folding and stemming will be applied to the selected candidate keyphrases. After extracting candidate keyphrases, they will be assigned a score either using a supervised or an unsupervised approach. In supervised approaches the problem of extraction of keyphrases is usually formulated as a binary classification problem [14]. However, this approach has the disadvantage that the phrases will either be classified as *keyphrase* or *not-keyphrase* and their importance will not be comparable to each other. In order to overcome this problem supervised algorithms have been designed which are able to learn a phrase rater [15].

As in all supervised algorithms, specific features have to be selected to train the algorithms. Examples of such features are:

- TF-IDF [16]: Frequency of the candidate keyphrase in the given document and its inverse document frequency
- Structural features [17]: Location of keyphrases in the document.
- Syntactic features [18]: Syntactic patterns inside the candidate keyphrases.

Several approaches have also been suggested for unsupervised keyphrase extraction. Graph-based approaches represent the content of the document as a graph and based on the edges connecting the nodes of the graph, determine the importance of the words in the document [11]. Additionally clustering approaches are used to cluster the candidate phrases based on their semantic relationship and in this way it will be ensured that the extracted keyphrases reflect the content of the document [10].

Many other unsupervised phrase ranking algorithms use heuristics based on various features like length and frequency of the phrases [13] or the co-occurrence statistics of the terms inside the phrases [19].

3 PHRASES AS A FUZZY SET

In linguistics, a *phrase* is a group of words acting as a single part of speech. Phrases usually contain a keyword, through which their type and category can be identified. This word is known as the *head* of the phrase. Phrases where their head is a noun are called *noun phrases* [20].

For summarizing news articles through keyphrases, understandable and syntactically correct phrases are needed to be extracted. Identifying the (candidate) phrases in a document is not the main focus of this work and can be fulfilled in various ways. The candidate phrases can either be selected using automatic methods (see section 2) or with the assistance of human annotators. In this work we assume that the candidate phrases are annotated manually by humans. This has the advantage that the quality of the ranking algorithm can be measured independent of the quality of the candidate selection step (assuming that the manually selected phrases are a gold standard and almost perfect).

The selected phrases, form now our set of phrases $\mathcal{K}$ which we define as a fuzzy set:

$$\mathcal{K} = \left\{ \left(x, \mu_{\mathcal{K}}(x)\right) \middle| x \in K,\ \ \mu_{\mathcal{K}}(x) \in [0,1] \right\}. \tag{1}$$

In Equation 1, K is a classical set that contains all the phrases in the document, and $\mu_{\mathcal{K}}(x)$ specifies the grade or degree to which any element $x \in K$ belongs to the fuzzy set $\mathcal{K}$. Larger values of $\mu_{\mathcal{K}}(x)$ indicate higher degrees of membership [21].

Intuitively, the membership value of each phrase will represent its significance in the document. Particularly the difference between K and $\mathcal{K}$ has to be emphasized at this point. The set K is a classical set that contains all the phrases of the document under analysis, independent of the definition of the term *phrase*. This means depending on the definition of the term *phrase*, the set K can contain *noun phrases*, *verb phrases*, a combination of them or even sequences of word intuitively identified by a human annotator as candidate phrases. In contrast to K, the fuzzy set $\mathcal{K}$ has an associated membership value for each phrase inside K. A phrase x with a membership value $\mu_{\mathcal{K}}(x) = 0$ represents a phrase inside the document with the least possible significance and a phrase x' with a membership value $\mu_{\mathcal{K}}(x') = 1$ represents a phrase with the maximum possible significance inside the document. It should be noted that both sets K and $\mathcal{K}$ has the same cardinality.

In order to find the keyphrases more easily using a threshold and also to make the membership values among different documents comparable, we use the normalized membership values of the phrases. For this we compute the *height* of

Proceedings of Second International Conference on Digital Information Processing, Data Mining, and Wireless Communications (DIPDMWC2015), Dubai, UAE, 2015

the fuzzy set $\mathcal{K}$ [21]:

$$h(\mathcal{K}) = \sup_{x \in K} \mu_{\mathcal{K}}(x), \quad (2)$$

and divide $\mu_{\mathcal{K}}(x)$ for all $x \in K$ by height of the fuzzy set $\mathcal{K}$. In this way we obtain the normalized fuzzy set with $h(\mathcal{K}) = 1$.

We define the set of keyphrases of a document to be the α-*cut* of the fuzzy set $\mathcal{K}$, defined as follows:

$$[\mathcal{K}]^{\alpha} = \begin{cases} \{x \in X \mid \mu_{\mathcal{K}}(x) \geq \alpha\} & 0 < \alpha \leq 1 \\ cl(supp(\mathcal{K})) & \alpha = 0 \end{cases}, \quad (3)$$

where $cl(supp(\mathcal{K}))$ is the closure of the support of $\mathcal{K}$ (We will not go into details of this definition as the case $\alpha = 0$ represents the membership value of candidates with the least possible significance inside the document).

Using the above definitions, the problem of finding the keyphrases of a document will be reduced to computing the membership values for the elements of the set K and determining the value of α.

4 MEMBERSHIP FUNCTIONS

In our work we analyze 3 features of the phrases in order to determine their significance within the documents. Length of keyphrases, their token frequency and skip-bigram co-occurrence frequency. Based on these features a final membership value (which is a weighted sum of membership values) will be assigned to each phrase. At the end, phrases with higher membership values will be selected as the keyphrases of the document.

4.1 Token Length of Keyphrases

During our experiments we have noticed that human annotators usually tend to select phrases of token length 2 or 3 as keyphrases. There are of course also cases where keyphrases of token length one or greater than 3 are selected.

In order to force the system to select keyphrases of a specific token length, a size threshold can be hard-coded into the algorithm [22]. However, this will have the disadvantage that some keyphrases will be completely ignored (although they might be significant). As an example consider the case of hard-coding the values 2 and 3 into the algorithm and forcing it to ignore keyphrases of length greater than 3. Given a phrase like *Fifa secretary general Jerome Valcke* with length of 5 tokens, this phrase will not be considered as keyphrase, but probably its consituents like *Fifa* or *Jerome Valcke* will be selected as potential keyphrases. This should be acceptable in an application that uses keyphrases for indexing, but in the case of news summarization, informativeness of keyphrases plays an important role.

In order to assure the before mentioned length property of keyphrases, we use a *generalized bell membership function* which is defined as follows [23]:

$$f_1(x) = bell(x; a, b, c) = \frac{1}{1 + |\frac{x-c}{a}|^{2b}} . \quad (4)$$

In Equation 4, c and a represent the center and width of the membership function respectively. Moreover b is a positive number that controls the slope at the crossover point of the function. To understand the reason behind this function choice, the graph of the function represented in Figure 1 will be helpful. The solid curve represents the bell function $bell(x; 2, 2, 2.5)$. The dotted curves represent the impact of the change of parameter a, and the dashed curves show the impact of the change of the parameter b.

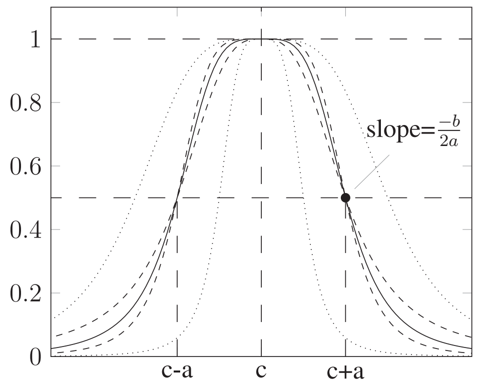


Figure 1. Graph of $bell(x, 2, 2, 2.5)$

By selecting $c = \frac{2+3}{2} = 2.5$ which is the average of the desired length, phrases with length 2

Proceedings of Second International Conference on Digital Information Processing, Data Mining, and Wireless Communications (DIPDMWC2015), Dubai, UAE, 2015

or 3 will be assigned higher membership values than the other phrases. Note that bigger values of b will increase the slope at the crossover point of the function and will cause in greater differences between the membership values of phrases with length 2 or 3, and phrases of other lengths. Due to this property, we recommend small values for b such as $b = 1$ or $b = 1.5$, which will reduce the amount of punishment for very small or very long phrases.

The above mentioned values for the parameters of the bell function are inexact predictions made intuitively based on desired properties of the algorithm. A more accurate approach is needed at this place to compute the parameter values of the bell function. For this we use the method of least squares. We construct a simple data set $D = \{(1, 0.75), (2, 1), (3, 1), (4, 0.75), (5, 0.5)\}$ consisting of (x_i, y_i) instances where x_i represents the token length of a phrase and y_i is its associated membership values. The data set is constructed in a way that phrases with lenghth 2 and 3 get a higher membership value than the other ones. Note that it is assumed that no phrase of length greater than 5 is selected by human annotators (or automatic candidate phrase selectors).

To estimate the parameters of the bell function, the square of residuals has to be minimized.

$$\|r\|^2 = \sum_{i=1}^{5} r_i^2 = \sum_{i=1}^{5} \left[y_i - \frac{1}{1 + |\frac{x_i - c}{a}|^{2b}}\right]^2 \quad (5)$$

The Equation 5 can also be written as:

$$\|r\|^2 = \sum_{i=1}^{5} y_i^2 + \frac{1}{(1 + |\frac{x_i - c}{a}|^{2b})^4} - \frac{2y_i}{(1 + |\frac{x_i - c}{a}|^{2b})^2} \quad (6)$$

In the next step, we compute the partial derivatives of $\|r\|^2$ with respect to the coefficients a, b and c. Equation 7 shows the partial derivative of the function with respect to the parameter a.

$$\frac{\partial \|r\|^2}{\partial a} = \sum_{i=1}^{5} \frac{8ba^{-2b-1}(x_i - c)^2 |x_i - c|^{2b-2}}{(a^{-2b}|x_i - c|^{2b} + 1)^5} \sum_{i=1}^{5} - \frac{8by_i a^{-2b-1}(x_i - c)^2 |x_i - c|^{2b-2}}{(a^{-2b}|x_i - c|^{2b} + 1)^3} \quad (7)$$

The partial derivatives $\frac{\partial \|r\|^2}{\partial b}$ and $\frac{\partial \|r\|^2}{\partial c}$ can be computed analogous to Equation 7. Finally by setting the partial derivatives to zero and solving the resulted linear equations, the values of coefficients will be $a = 2.4494$, $b = 1.1719$ and $c = 2.5111$. As we can see the estimated values are also consistent with our initial predictions. Notice that the parameters a, b and c are not dependent of the document under analysis and have to be set only once.

4.2 Token Frequency

Another property of phrases that plays a role in considering them as a keyphrase is their frequency. Phrases that contain tokens which occur frequently in the document are more likely to be keyphrases.
Let $p = t_1 \dots t_m$ be a phrase consisting of m distinct tokens. We define the frequency of a phrase inside a document as the sum of the frequencies of its constituting tokens. Formally this will be written as:

$$n_p = \sum_{i=1}^{m} n_{t_i}. \quad (8)$$

Note that n_{t_i} is actually the frequency of token t_i inside all available phrases in the document and not the document itself. In this way the effect of stop-words resulting in high frequencies for phrases will be avoided.
In order to model the property that phrases with frequent tokens are more likely than the ones with less frequent tokens to be keyphrases, we use a *sigmoid membership function* defined as follows [21]:

$$f_2(x) = sigmf(x; a', c') = \frac{1}{1 + e^{-a'(x - c')}}. \quad (9)$$

Proceedings of Second International Conference on Digital Information Processing, Data Mining, and Wireless Communications (DIPDMWC2015), Dubai, UAE, 2015

The parameter a' can be specified by the user ($a' = 1$ or $a' = 2$ is recommended). Depending on the sign of the parameter a', the function can be open right or left and thus will be appropriate for representing quantities such as *very frequent* or *very rare*. In order to determine the value of the parameter c', first the frequencies of all phrases (n_p) have to be computed. We compute then the value of c' as the p-Quantil of the calculated frequencies, with $p = 0.9$.
In Figure 2 the solid curve represents the sigmoid function $sigmf(x; 2, 15)$. The dotted curves represent the impact of the change of parameter c' and the dashed curves show the impact of the change of the parameter a'.

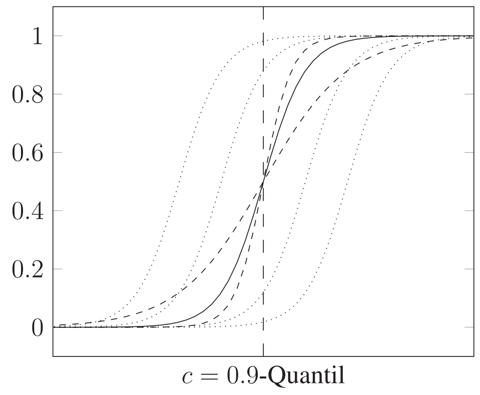


Figure 2. Graph of $sigmf(x, 2, 15)$

The parameter c' represents an absolute number and depending on the document under analysis can take various values. This makes the process of learning this parameter more difficult and document-dependent. In order to solve this problem we perform a zero-one normalization to transform the token frequency values into the range $[0, 1]$. Let $X = \{x_1, \ldots, x_n\}$ be the token frequencies of the n phrases in a document d. Each value x_i can be transformed as follows:

$$x_{i_{norm}} = \frac{x_i - min(X)}{max(X) - min(X)} \tag{10}$$

Similar to the previous section we construct a data set D to learn the parameters a' and c' (See Table 1).
The dataset represented in Table 1 is designed in a way that the normalized token frequencies

Table 1. Dataset to learn parameters of the sigmoid function

index	x	y
1	0.1	0.05
2	0.2	0.1
3	0.3	0.15
4	0.4	0.3
5	0.5	0.35
6	0.6	0.4
7	0.7	0.8
8	0.8	0.9
9	0.9	1.0
10	1.0	1.0

closer to 1 will be assigned very high membership values and the ones closer to zero will be mapped to very low membership values.
Using the least square method, we minimize the following function:

$$\|r\|^2 = \sum_{i=1}^{10} y_i - \frac{1}{1 + e^{-a'(x_{i_{norm}} - c')}} \tag{11}$$

By setting the partial derivatives $\frac{\partial \|r\|^2}{\partial a'}$ and $\frac{\partial \|r\|^2}{\partial c'}$ to zero and solving the resulted linear equations the values of coefficients will be $a = 7.6032$ and $c' = 0.5674$. Notice that the parameter values gained using the quantile approach are different from the values estimated using the least square method. This difference is expected and is because of the fact that in the case of quantile approach, absolute values of token frequencies are used and in the case of the least square method, normalized token frequencies are needed. Our experimental results show that both approaches return comparable results.
Notice that we did not consider the effect of length of phrases in the definition of the membership function. This will of course result in higher membership values for longer phrases. However this effect will be canceled by means of the generalized bell function which we already defined in section 4.1.

4.3 Skip-Bigram Co-Occurrence Frequency

Another feature from which we believe to play a significant role in identifying the keyphrases

of a document is the *skip-bigram co-occurrence frequency*. Assuming a phrase $p = t_1 \dots t_m$ consisting of m tokens, *skip-bigrams* are any pair of tokens inside the phrase, with arbitrary gaps [24]. As an example the phrase $p = t_1t_2t_3$ will consist of the following skip-bigrams: $(t_1, t_2), (t_1, t_3), (t_2, t_3)$. The total number of skip-bigrams inside a phrase of length m can be computed as the *2-combination* of the set of its tokens which is equal to $\binom{m}{2} = \frac{m!}{(m-2)!2!}$.
For each skip-bigram in the document we calculate the frequency of it among the set of skip-bigrams of available phrases. Finally, the skip-bigram co-occurrence frequency of a phrase will be computed as the sum of the frequencies of all its constituting skip-bigrams. This will be formally written as:

$$n_p^{skip} = \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} n_{t_i,t_j}, \qquad (12)$$

where n_{t_i,t_j} denotes the frequency of the skip-bigram (t_i, t_j). After computing the n_p^{skip} values for all phrases in the document, we again apply the sigmoid membership function to the computed values. As in the previous section, we select the parameter c'' to be the p-Quantil of the n_p^{skip} values with $p = 0.9$. We also recommend the parameter a'' to be equal to 1 or 2. The skip-bigram co-occurrence frequency of a phrase x will be denoted as $f_3(x)$.
Similar to the calculations performed by the token frequency feature, the parameters of the sigmoid membership function can be computed using the least square method. For this the n_p^{skip} values have to be normalized using the zero-one normalization method. We use the same data set as the one used for token frequency feature and calculate the parameter values as $a = 7.6032$ and $c = 0.5674$.

4.4 Final Membership Value Computation

As already mentioned in Section 3, our goal was to assign a membership value to each candidate phrase in the document. We have splitted this problem into 3 sub-problems and using 3 features, namely the phrase length f_1, token frequency f_2 and skip-bigrams co-occurrence frequency f_3, 3 membership functions have been defined. Finally, we define the membership value of a phrase x inside the set K to be the weighted sum of the 3 introduced membership functions:

$$\begin{aligned}\mu_K(x) = {} & \omega_1 \cdot bell(f_1(x); a, b, c) \\ & + \omega_2 \cdot sigmf(f_2(x); a', c') \\ & + \omega_3 \cdot sigmf(f_3(x); a'', c''), \end{aligned} \quad (13)$$

with $\sum_{i=1}^{3} w_i = 1$.
In our experiments we use $w_1 = 0.5$, $w_2 = 0.25$ and $w_3 = 0.25$. The keyphrases from the document can now be selected using the Equation 3. For larger values of α the algorithm will select the phrases which are most likely to be a keyphrase in a document. It should be also clear that by setting a higher value for α the number of extracted keyphrases from the document will be decreased and vice versa.

5 IMITATION GAME INSPIRED EVALUATION

In this section we introduce our evaluation method inspired by the imitation game suggested by Alan Turing in 1950 to decide whether a machine is intelligent or not [25].
Assuming a collection $D = \{d_1, \dots, d_n\}$ of n documents, our goal is to compare the automatically ranked keyphrases to the ones ranked by humans. For each document d_i with $1 \le i \le n$, let $K_i^{\mathcal{M}} = \{x_{i1}, \dots, x_{im}\}$ and $K_i^{\mathcal{A}} = \{x'_{i1}, \dots, x'_{im}\}$ be the set of manually and automatically ranked keyphrases respectively. Note that the set of automatically and manually ranked keyphrases, both have the same cardinality, namely m. The sets $K_i^{\mathcal{M}}$ and $K_i^{\mathcal{A}}$ posses the following properties:

1. Reflexivity
   $x_{ij} \le x_{ij}$ for all $x_{ij} \in K_i^{\mathcal{M}}$
   $x'_{ij} \le x'_{ij}$ for all $x'_{ij} \in K_i^{\mathcal{A}}$
2. Transivity
   $x_{ij} \le x_{ik}$ and $x_{ik} \le x_{is}$ implies $x_{ij} \le x_{is}$
   $x'_{ij} \le x'_{ik}$ and $x'_{ik} \le x'_{is}$ implies $x'_{ij} \le x_{is}$

The evaluation process consists of n iterations, and two human raters $\mathcal{R}_1$ and $\mathcal{R}_2$. In

each iteration a document d_i with its associated keyphrases $K_i^{\mathcal{M}}$ and $K_i^{\mathcal{A}}$ is shown to the raters (The raters are not told which of the keyphrases are human-ranked and which are machine-ranked). The raters are asked to classify the keyphrases either to the class *human-ranked (HR)* or to the class *machine-ranked (MR)*.
In this way the set of labels of the classification problem will be $C = \{\text{human-ranked}, \text{machine-ranked}\}$ and for each set of keyphrases K_i the raters have to determine its label $c(K_i)$. Note that the raters are not allowed to assign the same label for both sets of keyphrases. This means that assuming the sets $K_i^{\mathcal{A}}$ and $K_i^{\mathcal{M}}$, we will have $c(K_i^{\mathcal{M}}) \neq c(K_i^{\mathcal{A}})$. Finally, for each rater a confusion matrix will be constructed (see Table 2).

Table 2. Confusion Matrix of a Rater

		Predicted Labels		
		HR	MR	Total
Actual Labels	HR	A	B	$A+B$
	MR	C	D	$C+D$
	Total	$A+C$	$B+D$	$2n$

Notice that on the bottom right corner of the Table 2, the total number of classifications is written as $2n$ and not n. For each document d_i a rater will be encountered with two sets of keyphrases, namely $K_i^{\mathcal{M}}$ and $K_i^{\mathcal{A}}$. The rater will classify one of the sets as *machine-ranked* and the other one as *human-ranked*. This means that in each iteration a rater performs two classifications. As we have a total number of n documents, the total number of classifications at the end will be equal to $2n$.
In order to assess the agreement on the classification task, we use the *Cohen's Kappa Coefficient* [26] which is a mean to measure the inter-rater agreement and is defined as follows:

$$\kappa = \frac{Pr(a) - Pr(e)}{1 - Pr(e)}. \tag{14}$$

In Equation 14, $Pr(a)$ denotes the observed percentage of agreement and $Pr(e)$ is the expected percentage of agreement. We consider $\kappa \geq 0.60$ to be a good amount of agreement between the raters. For the case $\kappa < 0.60$ more documents have to be added to the collection, until the desired value is reached.
Finally, we compute the average accuracy of the raters $\tilde{\alpha}_{\mathcal{H}}$ as follows:

$$\tilde{\alpha}_{\mathcal{H}} = \frac{1}{4n} \sum_{i=1}^{2} A_i + D_i. \tag{15}$$

In Equation 15, A_i represents the number of times that rater $\mathcal{R}_i$ correctly classified a human-ranked set and D_i denotes the number of times that a machine-ranked set has been correctly classified by $\mathcal{R}_i$.
Based on the average accuracy of the raters, the accuracy of the keyphrase scoring algorithm can be determined. Intuitively this means that in cases where a rater classifies a set of keyphrases as human-ranked, but the set is actually machine-ranked, this would mean that the keyphrase extraction algorithm was successful in imitating the humans and has ranked keyphrases, even better than the human-ranked ones. Based on this idea we define the accuracy of the keyphrase scoring algorithm $\tilde{\alpha}_{\mathcal{M}}$ to be the *misclassification rate* of the human raters:

$$\tilde{\alpha}_M = 1 - \tilde{\alpha}_{\mathcal{H}}. \tag{16}$$

The lower the average accuracy of human raters, the higher the accuracy of the algorithm will be.

6 EVALUATION RESULTS

For our experiments we have collected 30 English news articles from *BBC News* and for each document inside our collection we have manually extracted the top 10 keyphrases from the documents. The extraction process has been performed by two annotators, and from the extracted keyphrases of each annotator, an intersection set of size 10 has been selected. In this way we assure the reliability of the extracted keyphrases. Additionally we used our algorithm to rank the top 10 manually extracted keyphrases from the documents.
For each document d_i, with $1 \leq i \leq 30$ we asked the raters to decide which element of the pair $(K_i^{\mathcal{H}}, K_i^{\mathcal{M}})$ is machine-ranked and which one is human-ranked (the actual labels of

the keyphrases are of course not shown to the raters).
The confusion matrices of the raters $\mathcal{R}_1$ and $\mathcal{R}_2$ can be seen in Table 3 and Table 4 respectively.

Table 3. Confusion Matrix $\mathcal{R}_1$

		Predicted Labels		
		HR	MR	Total
Actual Labels	HR	12	18	30
	MR	18	12	30
	Total	30	30	60

Table 4. Confusion Matrix of $\mathcal{R}_2$

		Predicted Labels		
		HR	MR	Total
Actual Labels	HR	47	53	30
	MR	53	47	30
	Total	30	30	60

To compute the *Cohen's Kappa Coefficient*, we collected the following information about the ratings:

1. Number of times that $\mathcal{R}_1$ and $\mathcal{R}_2$ classified the same set of keyphrases as human-ranked = 24
2. Number of time that $\mathcal{R}_1$ and $\mathcal{R}_2$ classified the same set of keyphrases as machine-ranked = 24
3. Number of times that $\mathcal{R}_1$ classified a set of keyphrases as human-ranked but $\mathcal{R}_2$ classified the same set as machine-ranked = 6
4. Number of times that $\mathcal{R}_2$ classified a set of keyphrases as human-ranked but $\mathcal{R}_1$ classified the same set as machine-ranked = 6

In total the number of observed agreements will be 48 (80% of the observations) and the number of expected agreements by chance is 30 (50% of the observations). Thus for *Cohen's Kappa Coefficient* we will have $\kappa = 0.6$ which is considered to be *good*.
Using the information contained in Table 3 and Table 4 we also compute the average accuracy of the raters which is $\tilde{\alpha}_{\mathcal{H}} = 0.4\overline{3}$. Finally, the accuracy of the algorithm will be $\tilde{\alpha}_{\mathcal{M}} = 1 - 0.4\overline{3} = 0.5\overline{6}$.
As an example for the output of our algorithm we extracted the top 10 keyphrase of a news article[1] from *Reuters*. A snippet of the news article can be seen in Figure 3. The top 10 automatically ranked keyphrases can be seen in Figure 4.

Act now on climate change or see costs soar, White House says
(Reuters) - Putting off expensive measures to curb climate change will only cost the United States more in the long run, the White House said on Tuesday in a report meant to bolster a series of actions President Barack Obama has proposed to address global warming.
"Each decade we delay acting results in an added cost of dealing with the problem of an extra 40 percent," said Jason Furman, chairman of Obama's Council of Economic Advisers.
"We know way more than enough to justify acting today," Furman told reporters.
The report drew its conclusions from 16 economic studies that modeled the costs of climate change. It was released as the U.S. Environmental Protection Agency holds public hearings on its plan to cut carbon emissions from power plants - the centerpiece of Obama's climate action plan.
Business groups have said the EPA's plan would hurt jobs in the coal sector and harm the U.S. economy.
The White House and environmental groups have pushed back against that argument.

Figure 3. A snippet of a news article

7 CONCLUSIONS

In this work we have introduced a fuzzy set approach to rank keyphrases from news articles. We defined several fuzzy membership functions to calculate the significance of candidate phrase and by means of a weighted membership func-

[1]http://www.reuters.com/article/2014/07/29/us-usa-climatechange-idUSKBN0FY0V820140729

energy and climate change, White House and environmental groups, natural gas transmission and distribution system, climate costs, White House, former New York City Mayor Michael Bloomberg, agriculture and food production, U.S. Environmental Protection Agency, Energy Secretary Ernie Moniz, methane emissions

Figure 4. Extracted keyphrases from the news article

tion, determined the final importance of the them. Finally we introduced a novel evaluation approach inspired by the Turing test.
Using the introduced approach we were able to partly imitate the humans in the way they rank keyphrases in documents. This led to the situation where in some cases our raters were unable to differentiate between the human-ranked and machine-ranked keyphrases. Also in many cases the machine-ranked keyphrases were wrongly classified by the judges as human-ranked keyphrases which is an indicator of the acceptable performance of our algorithm.
For future work we intend to perform an in-depth analysis of our evaluation method and generalize it to further types of summarization such as *sentence extraction* or even *abstractive summarization*.

REFERENCES

[1] Y. Zhang, E. Milios, and N. Zincir-heywood, "A comparative study on key phrase extraction methods in automatic web site summarization," *Journal of Digital Information Management, Special Issue on Web Information Retrieval*, 2007.

[2] L. Dolamic and C. Boyer, "Key-phrase based classification of public health web pages.," in *MedInfo* (C. U. Lehmann, E. Ammenwerth, and C. Nhr, eds.), vol. 192 of *Studies in Health Technology and Informatics*, p. 1133, IOS Press, 2013.

[3] N. Erbs, I. Gurevych, and M. Rittberger, "Bringing order to digital libraries: From keyphrase extraction to index term assignment.," *D-Lib Magazine*, vol. 19, no. 9/10, 2013.

[4] S. N. Kim, O. Medelyan, M.-Y. Kan, and T. Baldwin, "Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles," in *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, (Stroudsburg, PA, USA), pp. 21–26, Association for Computational Linguistics, 2010.

[5] M. Grineva, M. Grinev, and D. Lizorkin, "Extracting key terms from noisy and multitheme documents," in *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, (New York, NY, USA), pp. 661–670, ACM, 2009.

[6] X. Wan and J. Xiao, "Single document keyphrase extraction using neighborhood knowledge.," in *AAAI* (D. Fox and C. P. Gomes, eds.), pp. 855–860, AAAI Press, 2008.

[7] K. Barker and N. Cornacchia, "Using noun phrase heads to extract document keyphrases," in *Advances in Artificial Intelligence* (H. Hamilton, ed.), vol. 1822 of *Lecture Notes in Computer Science*, pp. 40–52, Springer Berlin Heidelberg, 2000.

[8] S. Rose, W. Cowley, V. Crow, and N. Cramer, "Rapid automatic keyword extraction for information retrieval and analysis," Mar. 6 2012. US Patent 8,131,735.

[9] F. Liu, D. Pennell, F. Liu, and Y. Liu, "Unsupervised approaches for automatic keyword extraction using meeting transcripts," in *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, (Stroudsburg, PA, USA), pp. 620–628, Association for Computational Linguistics, 2009.

[10] Z. Liu, P. Li, Y. Zheng, and M. Sun, "Clustering to find exemplar terms for

keyphrase extraction," in *In Proceedings of EMNLP*, pp. 257–266, 2009.

[11] R. Mihalcea and P. Tarau, "TextRank: Bringing order into texts," in *Proceedings of EMNLP-04and the 2004 Conference on Empirical Methods in Natural Language Processing*, July 2004.

[12] X. Wan and J. Xiao, "Collabrank: Towards a collaborative approach to single-document keyphrase extraction," in *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, (Manchester, UK), pp. 969–976, Coling 2008 Organizing Committee, August 2008.

[13] K. Barker and N. Cornacchia, "Using noun phrase heads to extract document keyphrases," in *Proceedings of the 13th Biennial Conference of the Canadian Society on Computational Studies of Intelligence: Advances in Artificial Intelligence*, AI '00, (London, UK, UK), pp. 40–52, Springer-Verlag, 2000.

[14] I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning, "Kea: Practical automatic keyphrase extraction," in *Proceedings of the Fourth ACM Conference on Digital Libraries*, DL '99, (New York, NY, USA), pp. 254–255, ACM, 1999.

[15] D. Kelleher and S. Luz, "Automatic hypertext keyphrase detection.," in *IJCAI* (L. P. Kaelbling and A. Saffiotti, eds.), pp. 1608–1609, Professional Book Center, 2005.

[16] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Inf. Process. Manage.*, vol. 24, pp. 513–523, Aug. 1988.

[17] T. D. Nguyen and M. yen Kan, "Keyphrase extraction in scientific publications," in *In Proc. of International Conference on Asian Digital Libraries (ICADL 07*, pp. 317–326, Springer, 2007.

[18] W.-t. Yih, J. Goodman, and V. R. Carvalho, "Finding advertising keywords on web pages," in *Proceedings of the 15th International Conference on World Wide Web*, WWW '06, (New York, NY, USA), pp. 213–222, ACM, 2006.

[19] Y. Matsuo and M. Ishizuka, "Keyword extraction from a single document using word co-occurrence statistical information," *International Journal on Artificial Intelligence Tools*, vol. 13, p. 2004, 2004.

[20] P. H. Matthews, *The concise Oxford dictionary of linguistics*. Oxford paperback reference, Oxford, England, New York: Oxford University Press, 2007.

[21] G. J. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1995.

[22] L. Wang and F. Li, "Sjtultlab: Chunk based method for keyphrase extraction," in *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, (Stroudsburg, PA, USA), pp. 158–161, Association for Computational Linguistics, 2010.

[23] G. Feng, *Analysis and Synthesis of Fuzzy Control Systems: A Model-Based Approach*. Boca Raton, FL, USA: CRC Press, Inc., 1st ed., 2010.

[24] S. Li, Y. Zhang, W. W. 0013, and C. Wang, "Using proximity in query focused multi-document extractive summarization.," in *ICCPOL* (W. Li and D. M. Aliod, eds.), vol. 5459 of *Lecture Notes in Computer Science*, pp. 179–188, Springer, 2009.

[25] A. M. Turing, "Computing machinery and intelligence," *Mind*, vol. 59, no. 236, pp. 433–460, 1950.

[26] J. Carletta, "Assessing agreement on classification tasks: The kappa statistic," *Comput. Linguist.*, vol. 22, pp. 249–254, June 1996.

5 Extractive Summarization

Extractive text summarization aims at extracting the most salient sentences from a textual document. A critical component in any text summarization system is the content selection component. The content selection process can be either generic (ignoring the needs of the user or, in other words, selecting the most important content from the author's perspective) or query-based (selecting the most relevant content regarding the queries of the user).

Although selecting the most relevant content with respect to the user queries (or even generic content extraction) has been proven to result in usable summaries, it cannot be ignored that every summarization process is a lossy transformation of the original document, and the resulting document contains a lower amount of information in comparison to the original document. This might be helpful in many scenarios, but at the same time, it increases the probability of missing critical information.

In this section, we tackle the problem of automatic text summarization from a different point of view. Instead of selecting the most salient information in a document, we introduce a component to eliminate the redundant information from the document. Especially in the context of multidocument summarization, such an approach can be utilized to reduce the amount of redundant information occurring in multiple documents and reduce the amount of text drastically. We accomplish this by incorporating an automatic paraphrase detection algorithm (See Section 5.1).

Moreover, any transformation applied to the text (deletion, paraphrasing, elimination) can lead to the decoherence of the text and make understating the document challenging. To tackle this problem, we introduce a sentence ordering algorithm (Section 5.2) to determine the order of two sentences automatically, and, in this way, assure the coherence of the sentences in the document.

5.1 HHU at SemEval-2016 Task 1: Multiple Approaches to Measuring Semantic Textual Similarity

Matthias Liebeck, Philipp Pollack, Pashutan Modaresi, and Stefan Conrad. HHU at SemEval-2016 Task 1: Multiple Approaches to Measuring Semantic Textual Similarity. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, pages 595–601, 2016.
Contributions: The author contributed with the design, development, and description of the the *Deep LDA* approach introduced in Section 3.3 of the publication. **Status**: Published.

In automatic paraphrase detection, it has to be decided whether two documents or sentences have the same semantic meaning. This task can either be modeled as a binary classification task (where the decision to be made is whether the sentences are semantically similar or not), or as a textual similarity measurement (where the degree of similarity between two sentences has to be measured). The second approach has the advantage that it can be easily transformed into a classification model.

Semantic textual similarity has been part of the SemEval[1] workshop series since 2014. In 2016, the participants of the semantic textual similarity task were given pairs of sentences and were asked to predict a semantic similarity score between zero and five for each pair of sentences. The performances of the systems were assessed using the Pearson correlations between the predicted scores and the gold-standard scores. The task offered two tracks: one called STS Core for English and one called Cross-lingual STS for Spanish-English bilingual sentence pairs.

In this work, we propose three different approaches to tackle the problem of paraphrase detection. First, we introduce an unsupervised approach called the *Overlap Method* that defines semantic similarity in terms of the amount of overlap between the tokens in the sentences. Our second approach utilizes a simple neural network with one hidden layer. Our third approach called DeepLDA computes the semantic similarity between two sentences from different aspects: surface-level similarity, context similarity, and topical similarity.

The proposed approaches can be employed in the context of extractive or abstractive text summarization to eliminate redundant information from the documents by detecting the sentences that are semantically very similar to each other. It has to be noticed that any inaccuracies in detecting paraphrases directly lead to higher error rates in the summarization systems.

[1]http://alt.qcri.org/semeval2016

HHU at SemEval-2016 Task 1: Multiple Approaches to Measuring Semantic Textual Similarity

Matthias Liebeck, Philipp Pollack, Pashutan Modaresi, Stefan Conrad
Institute of Computer Science
Heinrich Heine University Düsseldorf
D-40225 Düsseldorf, Germany
{liebeck,modaresi,conrad}@cs.uni-duesseldorf.de
philipp.pollack@hhu.de

Abstract

This paper describes our participation in the SemEval-2016 Task 1: *Semantic Textual Similarity* (STS). We developed three methods for the English subtask (STS Core). The first method is unsupervised and uses WordNet and word2vec to measure a token-based overlap. In our second approach, we train a neural network on two features. The third method uses word2vec and LDA with regression splines.

1 Introduction

Measuring semantic textual similarity (STS) is the task of determining the similarity between two different text passages. The task is important for various natural language processing tasks like topic detection or automated text summarization because languages are versatile and authors can express similar content or even the same content with different words. Predicting semantic textual similarity has been a recurring task in SemEval challenges (Agirre et al., 2015; Agirre et al., 2014; Agirre et al., 2013; Agirre et al., 2012). As in previous years, the purpose of the STS task is the development of systems that automatically predict the semantic similarity of two sentences in the continuous interval $[0, 5]$ where 0 represents a complete dissimilarity and 5 denotes a complete semantic equivalence between the sentences (Agirre et al., 2015).

The organizers provide sentence pairs whose semantic similarities have to be predicted by the contestants. The quality of a system is determined by calculating the Pearson correlation between the predicted values and a human gold standard that has been created by crowdsourcing. The data from previous STS tasks can be used for training supervised methods.

The test data consists of text content from different sources. In this year's shared task, the systems are tested on five different categories with different topics and varying textual characteristics like text length or spelling errors: *answer-answer*, *plagiarism*, *postediting*, *headlines*, and *question-question*

The remainder of the paper is structured as follows: Section 2 discusses related approaches to automatically determining semantic textual similarity. Section 3 describes our three methods in detail. We discuss their results in section 4. Finally, we conclude in chapter 5 and outline future work.

2 Related Work

In the last shared tasks, most of the teams used natural languages processing techniques like tokenization, part-of-speech tagging, lemmatization, named entity recognition and word embeddings. External resources like WordNet (Miller, 1995) and word2vec (Mikolov et al., 2013) are commonly used. In (Agirre et al., 2012) and (Agirre et al., 2013), the organizers provide a list and a comparison of the tools and resources used by the participants in the first two years, respectively.

In each year, the organizers provide a baseline value by calculating the cosine similarity of the binary bag-of-words vectors from both sentences in each sample. Since 2013, TakeLab (Šarić et al., 2012), the best ranked system in 2012, has also been used as another baseline value.

Most of the teams used machine learning in 2015

(Agirre et al., 2015). In 2014, the best two submitted runs were from unsupervised systems.

The work most closely related to our Overlap method is (Han et al., 2015), which uses a two-phased approach called *Align-and-Differentiate*. In the first phase, they compute an alignment score. Afterwards, they modify the alignment score in a differentiate phase by subtracting a penalty score for terms that can not be aligned. The idea behind the computation of our alignment scores is the same: For each sample, we average over the crosswise similarities between the sentences by aligning them, accumulating similarities between tokens and dividing by sentence lengths. The results of the alignment score in our Overlap method differ because (i) our alignment is different, (ii) we use another similarity function for tokens, and (iii) our preprocessing is different.

In (Vu et al., 2015), the similarity between LDA vectors calculated from documents is used together with syntactic and lexical similarity measures to compute the similarity between text fragments. This idea is also incorporated in our Deep LDA method. Moreover, both approaches use different flavors of regression analysis for the final model prediction. Regression analysis was also used in (Sultan et al., 2015), where the authors combine an unsupervised method with ridge regression analysis. Our approach differs in the sense that it introduces k-nearest neighbors as a lazy training layer before the regression analysis phase to decrease the effect of noisy data points.

3 Methods

In this section, we describe our three system runs. The ideas behind our methods are independent of the word order in a sentence. Our first method is unsupervised, whereas the other two methods are supervised. The first and second method share the same preprocessing.

3.1 Run 1: Overlap Method

Our first method is unsupervised. It measures the overlap between the tokens in sentence s_1 and the tokens in sentence s_2.

3.1.1 Preprocessing

For preprocessing the input text, we first process each sentence with Stanford CoreNLP (Manning et al., 2014). Afterwards, we use Hunspell[1] with the latest OpenOffice English dictionaries to suggest spelling corrections for tokens with at least two characters in length. For each token, we calculate the Levenshtein distance for all suggestions. If suggestions have the same lowest distance, we choose the longest word and replace the former misspelt word. Abbreviations are also replaced by their full forms. Afterwards, we process the corrected sentence with Stanford CoreNLP again. We use the WordnetStemmer from the *Java Wordnet Interface* (Finlayson, 2014) to look up lemmas with the help of WordNet (Miller, 1995). If the WordnetStemmer can not provide a lemma for a token, we use the predicted lemma from the Stanford CoreNLP.

Instead of accessing all tokens in a sentence, we start from the root token and recursively follow outgoing dependency edges and add all visited tokens to a list. This approach improves our results slightly because some tokens will be ignored. Furthermore, the tokens are filtered for stopwords[2].

3.1.2 Method

The Overlap method measures the token-based overlap between two sentences. Therefore, we need to define a similarity function for tokens: We first try to identify a textual similarity of 1 by comparing the lower case lemmas of both tokens or by checking if their most common WordNet synsets are the same. We assess their similarity as 0.5 if they share any synset. If this is not the case, we use word2vec (Mikolov et al., 2013) with the 300-dimensional *GoogleNews-vectors-negative300* model. We look up both words (or their lemmas if the words are not present in the model) and calculate the cosine similarity of their word embeddings. Otherwise, we return a default value.

This yields the following similarity function for two tokens:

[1]`http://hunspell.github.io/`

[2]`http://xpo6.com/list-of-english-stop-words/`

$$\text{sim}(t_1, t_2) := \begin{cases} 1 & \text{if } t_1.\text{lemma} == t_2.\text{lemma} \\ 1 & \text{if } t_1 \text{ and } t_2 \text{ have the same most common synset} \\ 0.5 & \text{if } t_1 \text{ and } t_2 \text{ share any synset} \\ d(t_1, t_2) & \text{if } t_1 \text{ and } t_2 \text{ have word embeddings} \\ \textit{default} & \text{otherwise} \end{cases}$$

where $d(t_1, t_2)$ denotes the cosine similarity between the two word embeddings of the tokens.

Given a token t from one sentence, we calculate its similarity to another sentence S by taking the maximum similarity between t and all tokens of S:

$$\text{msim}(t, S) := \max_{t_2 \in S} \text{sim}(t, t_2)$$

We define the similarity score between two sentences in $[0, 1]$ as follows:

$$\text{ssim}(s_1, s_2) := \frac{\sum_{t \in s_1} \text{msim}(t, s_2)}{2 \cdot |s_1|} + \frac{\sum_{t \in s_2} \text{msim}(t, s_1)}{2 \cdot |s_2|}$$

To predict the semantic similarity score in $[0, 5]$, we multiply ssim by 5, however, this does not change our evaluation results because the Pearson correlation is scale invariant:

$$\text{STS}(s_1, s_2) := 5 \cdot \text{ssim}(s_1, s_2)$$

We observed that some samples in the STS 2016 test data consist almost entirely of stopwords. For example, the STS 2016 evaluation data contained a sample with the sentences "*I think you should do both.*" and "*You should do both.*" before the final filtering. After filtering stop words, the first sentence would only contain the word "*think*" and the second sentence would be empty, which would result in a predicted score of zero. To avoid these extreme cases, we do not filter stop words if this would result in a sentence length of less than two tokens in both sentences.

3.2 Run 2: Same Word Neural Network Method

We train a neural network with 3 layers and a sigmoid activation function in Accord.NET (de Souza, 2014). Our network consists of 2 neurons in the input layer, 3 neurons in the hidden layer and 1 neuron in the output layer, as illustrated in Figure 1. The layer weights are initialized by the Nguyen-Widrow function (Nguyen and Widrow, 1990). We use the Levenberg-Marquardt algorithm (Levenberg, 1944; Marquardt, 1963) to train our network on the STS Core test data from 2015 and 2014.

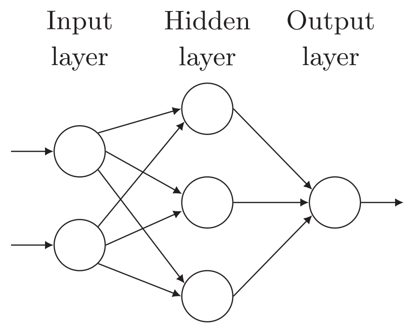


Figure 1: Architecture of our neural network

All samples are preprocessed as described in section 3.1.1. For each sample (s_1, s_2, gs) in the training set, we create a vocabulary list of the lowercase lemmas from both sentences. Lemmas that share a most common synset in WordNet are grouped together. Let n be the size of the vocabulary. We create two bag-of-words vectors bow_{s_1} and bow_{s_2}. For each lemma l, we calculate the minimum number of times l occurs in each sentence and the delta between the minimum and the maximum:

$$\text{min}_i := \min(\text{bow}_{s_1}[i], \text{bow}_{s_2}[i])$$
$$|\Delta_i| := |\text{bow}_{s_1}[i] - \text{bow}_{s_2}[i]|$$

As input vectors for the neural net, we build two sums per sample and use them as the two dimensional feature vector $(\text{sameWords}, \text{notSameWords})$ for the expected output gs:

$$\text{sameWords} := \sum_{i=1}^{n} \text{min}_i$$
$$\text{notSameWords} := \sum_{i=1}^{n} |\Delta_i|$$

Table 1 shows an example of the same word neural network method for the two input sentences "*Tim plays the guitar*" and "*Tim likes guitar songs*", which have the input vector $(2, 3)$.

i	Lemma	bow$_{s_1}$	bow$_{s_2}$	min$_i$	$\lvert\Delta_i\rvert$
1	tim	1	1	1	0
2	play	1	0	0	1
3	guitar	1	1	1	0
4	like	0	1	0	1
5	song	0	1	0	1
$\sum$				2	3

Table 1: An example for creating the two-dimensional feature vector for the *Same Word Neural Network* method

We trained the neural net until the error rate between two iterations did not change more than $\varepsilon = 10^{-5}$.

3.3 Run 3: Deep LDA Method

We represent the semantic similarity between two documents s_1 and s_2 by means of a vector $F = [f_1, f_2, f_3, f_4] \in \mathbb{R}^4$, where each component of F is responsible for modelling a different aspect of the semantic similarity, namely the *surface-level similarity*, *context similarity*, and the *topical similarity*.

Surface-level Similarity

The surface-level similarity can to some extent (although not entirely) capture the semantic similarity between documents. Let s_1 and s_2 be the reference and the candidate documents respectively. We compute the components $f_1, f_2 \in \mathbb{R}$ as follows:

$$f_1(s_1, s_2, N) = \frac{m_N}{l_N^{s_1}}$$
$$f_2(s_1, s_2, N) = \left(\prod_{n=1}^{N} \frac{m_N}{l_n^{s_2}}\right)^{\frac{1}{N}}$$

where m_N is the number of matched N-grams between s_1 and s_2, $l_N^{s_1}$ denotes the total number of N-grams in s_1 and $l_n^{s_2}$ is the total number of n-grams in s_2. f_1 is the common ROUGE (Lin, 2004) metric used in automatic text summarization and f_2 is a modified version of the BLEU (Papineni et al., 2002) metric (standard machine translation metric) where the brevity penalty is eliminated. Note that f_1 can be interpreted as the recall-oriented surface similarity and f_2 as the precision-oriented one.

Context Similarity

In order to model the context similarity between documents, we use word embeddings that learn semantically meaningful representations for words from local co-occurrences in sentences. More specifically we use *word2vec* (Mikolov et al., 2013) which seems to be a reasonable choice to model context similarity as the word vectors are trained to maximize the log probability of context words. We denote the context similarity of two documents s_1 and s_2 by $f_3 \in \mathbb{R}$ and compute it as follows:

$$\begin{aligned} f_3(s_1, s_2) &= \cos(\tilde{v}_{s_1}, \tilde{v}_{s_2}) \\ &= \cos\left(\frac{\sum_{v \in s_1} v}{|s_1|}, \frac{\sum_{v' \in s_2} v'}{|s_2|}\right) \end{aligned}$$

where v is the dense vector representation of a token and $\tilde{v}$ represents the centroid of the word vectors in a document.

Topical Similarity

To model the topical similarity between two documents, we use Latent Dirichlet Allocation (LDA) (Blei et al., 2003) to train models on the English Wikipedia. For both documents s_1 and s_2, we compute the topic distributions θ_1 and θ_2 and use the Hellinger distance to measure the similarity between the documents. This can be formally written as

$$f_4(s_1, s_2) = 1 - \frac{1}{\sqrt{2}} \sqrt{\sum_{i=1}^{k} \left(\sqrt{\theta_{1_i}} - \sqrt{\theta_{2_i}}\right)^2}$$

where k represents the number of learned LDA topics.

Similarity Prediction

In order to predict the semantic similarity between two documents, we use a combination of k-NN and Multivariate Adaptive Regression Splines (MARS) (Friedman, 1991).

Let $T = \{(s_1, s_1', gs_1), \ldots, (s_m, s_m', gs_m)\}$ be the training set consisting of m document pairs together with their corresponding gold standard semantic similarity and $(s_i, s_i') \notin T$ be a document pair for which the semantic similarity has to be computed. We construct a set $\mathcal{F} = \{(F_1, gs_1), \ldots, (F_m, gs_m)\}$ where each F_j is the four-dimensional vector representation of the semantic similarity between s_j and s_j'. Moreover, we

Sentence 1	Sentence 2	gs	STS
Unfortunately the answer to your question is we simply do not know.	Sorry, I don't know the answer to your question.	4	4.05800
You should do it.	You can do it, too.	1	4.39817
Unfortunately the answer to your question is we simply do not know.	My answer to your question is "Probably Not".	1	3.70982
$P(A\|B)$ is the conditional probability of A, given B.	$P(B\|A)$ is the conditional probability of B given A.	3	4.32017

Table 2: Examples for the results of the Overlap method with the corresponding gold standards

compute the vector F_i. Next, we construct a set $\mathcal{F}_k$ containing the k-nearest neighbors to the vector F_i. In order to calculate the distances between the vectors, we use the Euclidean distance. Finally, we construct a vector gs_k containing the gold standard similarity values of the k-nearest neighbors and feed it into a MARS model to predict the semantic similarity of the pair (s_i, s_i'). The choice of MARS is due to its capability to automatically model non-linearities between variables.

4 Results

We report the results of our three approaches for the STS Core test from 2016 and 2015.

4.1 STS 2016 Results

In this years shared task, 117 runs were submitted. We achieved weighted mean Pearson correlations of 0.71134, 0.67502 and 0.62078. In this year's run, our best result was the Overlap method, followed by the Same Word Neural Network method and the Deep LDA approach. Table 2 shows examples of good and bad results of our Overlap method on the 2016 data. Detailed results of our runs are given in Table 3 per test set. Our three approaches achieved different results.

From a semantic point of view, the most obvious value for the default value in our Overlap method is 0. However, we have discovered that a default value 0.15 returned better results on the STS Core test data from 2015 and also chose this default value for our submission.

In the Deep LDA approach, we set the parameter $N = 2$, although the use of unigrams did not show any significant statistical difference in the results. We choose the number of topics in the LDA model to be 300. In the prediction phase of the algorithm, we select $k = 100$ nearest neighbors from the data sets provided from 2012 to 2015.

Data set	Run 1	Run 2	Run 3
answer-answer	0.50435	0.42673	0.47211
headlines	0.77406	0.75536	0.58821
plagiarism	0.83049	0.79964	0.62503
postediting	0.83846	0.84514	0.84743
question-question	0.60867	0.54533	0.57099
Weighted Mean	0.71134	0.67502	0.62078

Table 3: Pearson correlation of the 2016 test data

Data set	Run 1	Run 2	Run 3
answers-forums	0.74163	0.70387	0.79987
answers-students	0.73685	0.76658	0.76733
belief	0.74046	0.73319	0.78242
images	0.82032	0.80813	0.84747
headlines	0.75358	0.74363	0.76076
Weighted Mean	0.76295	0.75922	0.79168

Table 4: Pearson correlation of the 2015 test data

4.2 STS 2015 Results

We list the results of our methods for the 2015 test data in Table 3 to discuss the effect of different evaluation sets. It is interesting to see that the Deep LDA method performed best out of our three systems on 2015. Its results on 2016 were surprisingly lower. We attribute this difference to the lack of domain specific training data for 2016. As an unsupervised approach, the Overlap method has fewer problems with the domain change.

It should be noted that the gold standard of the 2015 test data was available during the development of our methods. For the training phase, the Same Word Neural Network method used the STS Core test from 2014. The Deep LDA method was trained on the data from 2012 to 2014.

5 Conclusion and Future Work

We have presented three approaches to measure textual semantic similarity. This year, our unsupervised method achieved the best result. By comparing our result for 2016 and 2015, we showed that the approaches yielded different results in a different order.

In our future work, we will try to modify the Overlap method, by also using a penalty score and by applying certain similarity score shifters, for instance modifying the score by applying a date extraction with a specific distance function for dates. We tried to group words into phrases by using a sliding window approach with a shrinking window size and matching phrases in word2vec. In our initial attempt, this worsened the results for the Overlap method. We will adjust the similarity function to increase the weight of phrases in comparison to unigrams.

We aim to adapt the techniques for German and Spanish.

Acknowledgments

This work was partially funded by the PhD program *Online Participation*, supported by the North Rhine-Westphalian funding scheme *Fortschrittskollegs* and by the German Federal Ministry of Economics and Technology under the ZIM program (Grant No. KF2846504). The authors want to thank the anonymous reviewers for their suggestions and comments.

References

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 Task 6: A Pilot on Semantic Textual Similarity. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 385–393. Association for Computational Linguistics.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic Textual Similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43. Association for Computational Linguistics.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. SemEval-2014 Task 10: Multilingual Semantic Textual Similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 81–91. Association for Computational Linguistics and Dublin City University.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. SemEval-2015 Task 2: Semantic Textual Similarity, English, Spanish and Pilot on Interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 252–263. Association for Computational Linguistics.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022.

César Roberto de Souza. 2014. The Accord.NET Framework. `http://accord-framework.net`.

Mark Finlayson. 2014. Java Libraries for Accessing the Princeton Wordnet: Comparison and Evaluation. In *Proceedings of the Seventh Global Wordnet Conference*, pages 78–85.

Jerome H. Friedman. 1991. Multivariate Adaptive Regression Splines. *The Annals of Statistics*, 19(1):1–67.

Lushan Han, Justin Martineau, Doreen Cheng, and Christopher Thomas. 2015. Samsung: Align-and-Differentiate Approach to Semantic Textual Similarity. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 172–177. Association for Computational Linguistics.

Kenneth Levenberg. 1944. A Method for the Solution of Certain Non-Linear Problems in Least Squares. *Quarterly Journal of Applied Mathmatics*, II(2):164–168.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004)*, pages 74–81.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Donald W. Marquardt. 1963. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *SIAM Journal on Applied Mathematics*, 11(2):431–441.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *ICLR Workshop*.

George A. Miller. 1995. WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41.

Derrick Nguyen and Bernard Widrow. 1990. Improving the Learning Speed of 2-Layer Neural Networks by Choosing Initial Values of the Adaptive Weights. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume 3, pages 21–26. IEEE.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318. Association for Computational Linguistics.

Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2015. DLS@CU: Sentence Similarity from Word Alignment and Semantic Vector Composition. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 148–153. Association for Computational Linguistics.

Frane Šarić, Goran Glavaš, Mladen Karan, Jan Šnajder, and Bojana Dalbelo Bašić. 2012. TakeLab: Systems for Measuring Semantic Text Similarity. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 441–448. Association for Computational Linguistics.

Tu Thanh Vu, Quan Hung Tran, and Son Bao Pham. 2015. TATO: Leveraging on Multiple Strategies for Semantic Textual Similarity. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 190–195. Association for Computational Linguistics.

5.2 Neural Classification of Linguistic Coherence using Long Short-Term Memories

Pashutan Modaresi, Matthias Liebeck, and Stefan Conrad. Neural Classification of Linguistic Coherence Using Long Short-Term Memories. In *Proceedings of the 8th Annual Meeting of the Forum on Information Retrieval Evaluation*, FIRE '16, pages 28–31. ACM, 2016.
Contributions: The research and the preparation of this manuscript was done jointly by Pashutan Modaresi and Matthias Liebeck under the supervision of Prof. Conrad.
Status: Published.

A critical decision in automatic text summarization regards the order of the extracted or generated sentences that have to be presented to the user. In single document summarization, the answer to this question is almost trivial. In most of the existing extractive single document summarization approaches the natural order of the sentences in the document is taken to order the extracted sentences. In the case of extractive multidocument summarization, this problem is more challenging. The extracted sentences are from different documents, and they might contain redundancies and contradictions that might lead to a non-coherent text. Interestingly, it has also been shown that even in the case of extractive single document summarization, reordering the extracted sentences can improve the coherence and quality of the extracted summary [CQH16].

Given a set of sentences, sentence ordering aims to permute the sentences in a way such that the resulting text is coherent and readable for the user. A popular strategy to implement sentence ordering is to incorporate graph structures, where the sentences in the input documents are represented by a directed graph and the order of the sentences can be determined by analyzing the graph structure [BEM02]. One main problem with such approaches is that the semantic information in the sentences is completely ignored. Moreover, these models are mainly based on bag-of-words representations of the documents and do not capture the semantic information in the documents. Additionally, in the case of single document summarization, due to the low number of sentences, the graph structures do not seem to be useful [BEM02].

To tackle the problems mentioned above, we introduce a neural sentence ordering approach that takes the semantic information in the sentences into account. Our proposed method is not restricted to multidocument text summarization and can also be applied to single document text summarization. Our system achieved sound results for the task of sentence ordering in English and German.

Neural Classification of Linguistic Coherence using Long Short-Term Memories

Pashutan Modaresi
Institute of Computer Science
Heinrich Heine University of Düsseldorf
D-40225 Düsseldorf, Germany
modaresi@cs.uni-duesseldorf.de

Matthias Liebeck
Institute of Computer Science
Heinrich Heine University of Düsseldorf
D-40225 Düsseldorf, Germany
liebeck@cs.uni-duesseldorf.de

Stefan Conrad
Institute of Computer Science
Heinrich Heine University of Düsseldorf
D-40225 Düsseldorf, Germany
conrad@cs.uni-duesseldorf.de

ABSTRACT

Given a set of sentences, a sentence orderer permutes the sentences in a way that the final text is linguistically coherent and semantically understandable. In this work, we focus on the binary and ternary tasks of ordering a pair of sentences regarding their linguistic coherence. We propose a methodology to automatically collect and annotate sentence ordering corpora in the news domain for English and German documents. Furthermore, we introduce a data-driven end-to-end neural architecture to learn the order of a pair of sentences and also recognize the cases where no ordering can be determined due to missing context.

CCS Concepts

•Computing methodologies → Artificial intelligence; Natural language processing;

Keywords

Sentence ordering; long short-term memory; neural coherence classification

1. INTRODUCTION

The order of sentences in a document is what makes a text semantically meaningful. Assuming a single document $d = s_1, s_2, \dots, s_n$, consisting of n sentences, there are $n!$ possible permutations of the sentences to form a document. However, despite this huge search space, humans are extraordinarily good at determining the order of sentences.

On the other hand, machines require the ability to deal with linguistic concepts such as *discourse coherence*, *linguistic redundancy and contradiction*, and, in general, *pragmatics* [3] to order sentences into a meaningful and coherent text.

Various approaches have been introduced to solve the problem of sentence ordering. In [19], a similarity metric is used to group the sentences into clusters and sentences are selected from clusters in a way to maximize the similarity between adjacent sentences. Lin et. al [9] make the assumption that a coherent text implicitly favors certain types of discourse relation transitions. The closest to our approach are the works of Lin and Jurafsky [8] and Chen et. al [1]. Using a large corpus of academic texts, Chen et. al train an algorithm to learn the pairwise ordering of sentences using various neural architectures. In a different approach, Lin and Jurafsky concatenate the sentences and train a classifier to decide whether the resulted text is coherent or not. We refer the reader to [10] for a detailed overview of the literature.

Despite having applications in fields such as text planning [7] and question-answering [16], multi-document summarization [12] is considered as one of the main applications of sentence ordering.

In this work, we define sentence ordering as a classification task realized by a function $\Phi : \mathbb{R}^m \times \mathbb{R}^{m'} \rightarrow \mathbb{Z}$, where $\mathbb{R}^m$ and $\mathbb{R}^{m'}$ are the corresponding vector representations of the input sentences in an arbitrary semantic space. Without loss of generality, we set $m = m'$ and $\mathbb{Z} = \{0, 1\}$ (binary classification) or $\mathbb{Z} = \{-1, 0, 1\}$ (multiclass classification). Given a permutation $\sigma \in \Sigma$ of a list of sentences $s = [s_1, \dots, s_n]$, the optimal order σ^* can be computed as:

$$\sigma^* = \underset{\sigma \in \Sigma}{\operatorname{argmax}} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \Phi(s_{\sigma(i)}, s_{\sigma(j)}) \qquad (1)$$

Computing the optimal order using Equation 1 is computationally expensive. The focus of this work is to learn the function Φ and not to predict the optimal order σ^*. A possible strategy to compute σ^* is to use beam-search [1].

The open source implementation of our approach is hosted on Github[1].

2. ARCHITECTURE

As already stated, we treat the sentence ordering problem as a classification task. For this, we use a deep neural architecture to minimize the cross-entropy loss function [4].

$$\hat{p}(z|x_1; x_2) = \underset{p(z|x_1;x_2)}{\operatorname{argmin}} \Big\{ - \sum_{n \in N} \log p(z_n | x_1^{(n)}; x_2^{(n)}) \Big\} \qquad (2)$$

In Equation 2, N is the total number of training samples

[1]https://github.com/pasmod/reorderer

FIRE '16, December 08 - 10, 2016, Kolkata, India
© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM. ISBN 978-1-4503-4838-6/16/12…$15.00
DOI: http://dx.doi.org/10.1145/3015157.3015163

and $p(z_n|x_1^{(n)};x_2^{(n)})$ is the estimate of the class probability of the n-th ordered pair $(x_1^{(n)}, x_2^{(n)})$ returned by the neural network.

The architecture of the network is depicted in Figure 1. The network has two inputs corresponding to the ordered sentence pair (s_1, s_2). We use a one-hot encoding to map each sentence s_i into a list of token indices in the vocabulary V and obtain its corresponding vector representation $\vec{s_i} \in \mathbb{R}^{|V|}$. Furthermore, we pad each vector with a special padding symbol to the maximum length of sentences in the corpus. Additionally, using an embedding layer, the one-hot encoded inputs are projected into a low-dimensional space. The embedding layer is realized by a simple matrix multiplication $\vec{e_i} = E \cdot \vec{s_i}$ with $E \in \mathbb{R}^{d\times|V|}$. The number of rows in the embedding matrix is set to $d = 200$ and it has as many columns as the size of the vocabulary.

We also initialize the matrix E with weights from pre-trained embeddings. For English, we use 200-dimensional embeddings trained using the GloVe [15] algorithm and for German we use 200-dimensional embeddings trained on German Wikipedia[2] using the continuous bag-of-words approach introduced in [11]. Initializing the embeddings matrix with pre-trained embeddings is especially advantageous when the size of the training data is limited.

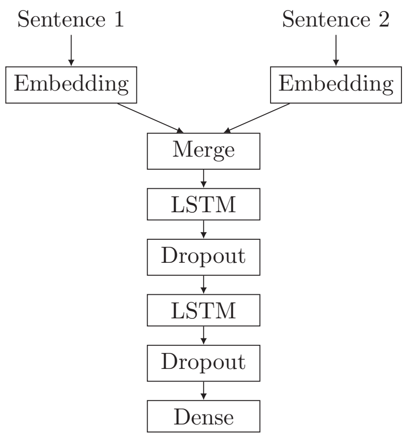


Figure 1: Deep learning architecture

The embeddings are then concatenated into a single vector $e = e_1 \oplus e_2$ with $e \in \mathbb{R}^{2d}$ that forms the input to a long short-term memory (LSTM) [6], which is a special kind of a recurrent neural network (RNN) addressing the difficulties in training RNNs [14].

In general, the hidden state of a vanilla RNN at time step t is updated as shown in Equation 3:

$$h_t = \sigma(W_{hh}h_{t-1} + W_{xh}x_t + b_h) \quad (3)$$

In Equation 3, W_{hh} represents the recurrent weights from the hidden layer to itself, W_{xh} denotes the weights from the inputs to the hidden layer, b_h is the bias vector, and σ represents a non-linear function.

In LSTMs, the problem of long dependencies is addressed by the introduction of cell states and gates that regulate if information can be added or removed from cell state. In total, LSTMs consist of 3 gates (input, forget, and output) and a cell state, where the forget gate controls what information shall be thrown away, the input gate decides what information shall be stored in the cell state, and the output gate determines what information will be returned by the hidden state.

We use dropout as regularization technique [5] to avoid overfitting. The dropout operator simply sets a random subset of its arguments to zero. Following the work of Zaremba et. al [17], we apply the dropout operator only to the non-recurrent units.

In our architecture, we use two LSTMs and finally, feed a softmax layer with the hidden state of the second LSTM to transform the hidden state representation into predictive probabilities. This can be formalized as follows:

$$p_t = \text{softmax}(W_{hx}h_t + bx) \quad (4)$$

We use Keras [2] to implement the proposed neural architecture. For the first LSTM we set the output dimension to 128 and for the second one to 64. Furthermore, we drop 50% of the non-recurrent units of the LSTMs. And finally to minimize the loss function (see Equation 2), we use an adaptive learning rate gradient descent method called Adadelta [18].

3. EXPERIMENTS

To train our neural architecture for the binary classification task, we used 30992 English and 22450 Germany samples and tested the final model with 7748 English and 5614 German unseen samples. For the ternary case, we used 50400 English and 36504 German samples for training and 12600 English and 9126 German samples for testing. Both train and test sets were balanced regarding the existing labels. In the following, the details of dataset construction and evaluation results are discussed in detail.

3.1 Dataset

To construct the underlying dataset for sentence ordering, we used the *Simurg*[3] [13] corpus which is an extendable multilingual collection of online news. In total, we used 9038 German and 12145 English news documents to train and validate our models.

To automatically label the dataset, we use a simple strategy. In the case of binary sentence ordering, for each ordered sentence pair (s_1, s_2), we define $\mathcal{L}(s_1, s_2) = 1$ if $s_1 \prec s_2$. Otherwise, we define $\mathcal{L}(s_1, s_2) = 0$. For this, we extract the first two sentences s_1 and s_2 of each news document, keep the natural order of the sentences and label them as positive. Additionally, we exchange the order of sentences and create a new ordered pair (s_2, s_1) with $\mathcal{L}(s_2, s_1) = 0$ to create negative examples.

The process for the ternary case is almost identical. We define $\mathcal{L}(s_1, s_2) = 1$ if $s_1 \prec s_2$ and $\mathcal{L}(s_1, s_2) = 0$ if $s_2 \prec s_1$. Furthermore, $\mathcal{L}(s_1, s_2) = -1$ if $s_1 \prec \ldots \prec s_i \prec \ldots \prec s_2$ or $s_2 \prec \ldots \prec s_i \prec \ldots \prec s_1$. For this, we extract the first and last sentence of each document and construct an ordered pair (s_1, s_2). This represents a situation where s_1 and s_2 are not adjacent and the ordered pair (s_1, s_2) is non-coherent due to missing context. In the remainder of this work we abbreviate this case with NC.

[2]https://de.wikipedia.org

[3]https://github.com/pasmod/simurg

3.2 Results

In total, we prepared an annotated dataset containing 63000 English and 42630 German samples for the ternary classification task and 38740 English and 28064 German samples for the binary classification problem. We randomly split the data into an 80% training set and a 20% validation set. We repeat the experiments three times and report the average number of false positives, false negatives, true positives, and true negatives together with their corresponding standard deviation.

Table 1: Confusion Matrix for English (Binary)

		Predicted		
		True	False	$\sum$
Actual	True	3703 ± 22	158 ± 22	3862
	False	223 ± 36	3662 ± 36	3886
	$\sum$	3927 ± 58	3820 ± 58	7748

The confusion matrices for the binary and ternary classification of English sentences are reported in Tables 1 and 2, respectively. As the experiments are conducted multiple times on various test sets, we report the mean values together with their corresponding standard deviation. Note that in the ternary case NC stands for "non-coherent".

Table 2: Confusion Matrix for English (Ternary)

		Predicted			
		True	NC	False	$\sum$
Actual	True	4132 ± 20	62 ± 18	5 ± 2	4200
	NC	77 ± 10	4153 ± 7	11 ± 3	4243
	False	45 ± 18	18 ± 14	4093 ± 13	4157
	$\sum$	4255 ± 44	4234 ± 28	4110 ± 18	12600

In both binary and ternary tasks, the test set is approximately balanced. It is also observable that the number of true positives and true negatives are very close to each other, which indicates that the classifier performs roughly equal for the existing labels.

Table 3: Confusion Matrix for German (Binary)

		Predicted		
		True	False	$\sum$
Actual	True	2654 ± 7	142 ± 7	2796
	False	151 ± 18	2667 ± 18	2818
	$\sum$	2805 ± 24	2809 ± 24	5614

The same properties also hold for the confusion matrices for the classification of German sentences. It is also observable that the standard deviations of diagonal elements are very low which indicates that the introduced neural architecture has a low variance. Also comparing the ratio of true negatives to the total number of samples yields that the classifier has a low bias. Notice that for readability we rounded all results in the confusion matrices to their next integers.

Additionally, we report the macro-averaged F_1 scores for both languages in Table 5. In the case of English sentences, the overall macro-averaged F_1 scores for the binary and the ternary classification tasks are 0.95 and 0.98, respectively. Interestingly, despite having an additional label in the ternary task, the F_1 score is higher than in the binary case.

Table 4: Confusion Matrix for German (Ternary)

		Predicted			
		True	NC	False	$\sum$
Actual	True	3065 ± 21	39 ± 16	12 ± 8	3117
	NC	43 ± 6	2974 ± 15	12 ± 10	3030
	False	38 ± 11	45 ± 15	2895 ± 5	2979
	$\sum$	3147 ± 38	3058 ± 46	2920 ± 9	9126

Table 5: Macro-Averaged F_1 Scores

	English		German	
	Binary	Ternary	Binary	Ternary
True	0.95	0.97	0.94	0.97
MC	—	0.98	—	0.97
False	0.95	0.98	0.94	0.98
Overall	0.95	0.98	0.94	0.97

The F_1 scores for the ternary classification of German and English sentences are almost identical and no significant difference could be observed. The same holds for the binary classification task.

Table 6: Macro-Averaged F_1 Scores (SVM)

	English		German	
	Binary	Ternary	Binary	Ternary
True	0.35	0.14	0.43	0.32
MC	—	0.14	—	0.12
False	0.12	0.20	0.07	0.04
Overall	0.24	0.16	0.25	0.16

We also compared the performance of our proposed algorithm to a baseline support vector machine (SVM) approach using the bag-of-words model. The results can be observed in Table 6 and can be compared to our results presented in Table 5. As expected, the SVM approach has much lower F_1 scores compared to our approach. Furthermore, no significant difference in F_1 scores for German and English can be observed. In general, support vector machines are not a suitable learning algorithm to model sequential data and thus have a poor performance on our data set.

4. CONCLUSIONS

We presented a neural architecture for classifying the linguistic coherence of a pair of sentences in German or English. In the binary case, we defined two sentences to be either adjacent or not and achieved an adequate F_1 score of 0.95 for English and 0.94 for German. In the ternary classification task, we defined a third label to represent the case when the sentences are non-coherent due to the missing context. For this task, we achieved an F_1 score of 0.98 for English and 0.97 for German sentences.

For future work, we plan to extend our corpora to assure the generalizability of our models and also increase the number of the inputs in the neural architecture to find the optimal order of multiple sentences without the use of search algorithms.

5. ACKNOWLEDGMENTS

This work was partially funded by the German Federal Ministry of Economics and Technology under the ZIM program (Grant No. KF2846504) and the PhD program *Online Participation*, supported by the North Rhine-Westphalian funding scheme *Fortschrittskollegs*. Computational support and infrastructure were provided by the “Centre for Information and Media Technology” (ZIM) at the University of Düsseldorf (Germany).

6. REFERENCES

[1] X. Chen, X. Qiu, and X. Huang. Neural Sentence Ordering. *ArXiv e-prints*, July 2016.

[2] F. Chollet. Keras. https://github.com/fchollet/keras, 2015.

[3] V. Fromkin, R. Rodman, and N. Hyams. *An Introduction to Language*. Cengage Learning, 2010.

[4] P. Golik, P. Doetsch, and H. Ney. Cross-Entropy vs. Squared Error Training: a Theoretical and Experimental Comparison. In *Interspeech*, pages 1756–1760, Aug. 2013.

[5] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[6] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780, Nov. 1997.

[7] E. H. Hovy. Planning Coherent Multisentential Text. In *Proceedings of the 26th Annual Meeting on Association for Computational Linguistics*, ACL ’88, pages 163–169. Association for Computational Linguistics, 1988.

[8] J. Li and D. Jurafsky. Neural Net Models for Open-Domain Discourse Coherence. *CoRR*, abs/1606.01545, 2016.

[9] Z. Lin, H. T. Ng, and M.-Y. Kan. Automatically Evaluating Text Coherence Using Discourse Relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT ’11, pages 997–1006. Association for Computational Linguistics, 2011.

[10] W. Liu, X. Luo, J. Xuan, Z. Xu, and D. Jiang. Cognitive Memory-inspired Sentence Ordering Model. *Knowledge-Based Systems*, 104:1 – 13, 2016.

[11] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space. *CoRR*, abs/1301.3781, 2013.

[12] P. Modaresi and S. Conrad. On Definition of Automatic Text Summarization. In *Proceedings of Second International Conference on Digital Information Processing, Data Mining, and Wireless Communications*, DIPDMWC2015, pages 33–40. SDIWC, 2015.

[13] P. Modaresi and S. Conrad. Simurg: An Extendable Multilingual Corpus for Abstractive Single Document Summarization. In *Proceedings of the 8th Forum for Information Retrieval Evaluation*, FIRE ’16. ACM, 2016.

[14] R. Pascanu, T. Mikolov, and Y. Bengio. On the Difficulty of Training Recurrent Neural Networks. *CoRR*, abs/1211.5063, 2012.

[15] J. Pennington, R. Socher, and C. D. Manning. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[16] S. Verberne, L. Boves, N. Oostdijk, and P.-A. Coppen. Evaluating Discourse-based Answer Extraction for Why-question Answering. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ’07, pages 735–736. ACM, 2007.

[17] W. Zaremba, I. Sutskever, and O. Vinyals. Recurrent Neural Network Regularization. *CoRR*, abs/1409.2329, 2014.

[18] M. D. Zeiler. ADADELTA: An Adaptive Learning Rate Method. *CoRR*, abs/1212.5701, 2012.

[19] R. Zhang. Sentence Ordering Driven by Local and Global Coherence for Summary Generation. In *Proceedings of the ACL 2011 Student Session*, HLT-SS ’11, pages 6–11. Association for Computational Linguistics, 2011.

6 Evaluating Summarization Systems

The evaluation methods of summarization systems can be categorized into two categories: *extrinsic* evaluation methods measure the qualities of the summaries in the context of a given task, such as classification or question answering. On the other hand, *intrinsic* evaluation methods directly measure the quality of the text summaries [SJ09]. In this work, we focus on the intrinsic methods.

Several automatic evaluation measures have been widely used in the literature to evaluate summarization systems. The ROUGE [Lin04] method is probably the most popular evaluation method for summarization systems. The idea behind the ROUGE method is to compare the n-grams in the original and the summary document. A high overlap results in a high ROUGE score and is an indicator of the good quality of the summary. The BLEU [PRWZ02] method has also been widely used to evaluate the quality of the summarization systems. Similar to the ROUGE method, the BLUE method also utilizes n-grams to compare the original document with the summary document.

In this chapter, we report the results of our studies for evaluating summarization systems. In Section 6.1, we propose a semi-automatic approach to evaluate obfuscation systems based on three criteria: safeness (if a forensic analysis does not reveal the author of the obfuscated texts), soundness (if the obfuscated texts are textually entailed with their originals) and sensibleness (if the obfuscated texts are inconspicuous). Moreover, in Section 6.2 we report the results of our study to evaluate summarization systems from a commercial point of view and claim that even naive summarization approaches can lead to a considerable amount of financial benefits.

6.1 Evaluating Safety, Soundness and Sensibleness of Obfuscation Systems

Matthias Liebeck, Pashutan Modaresi, and Stefan Conrad. Evaluating Safety, Soundness and Sensibleness of Obfuscation Systems. In *Working Notes of CLEF 2016 - Conference and Labs of the Evaluation Forum*, pages 920–928, 2016.
Contributions: The author contributed with the evaluation of the dimensions *safeness* and *sensibleness*. The manuscript was prepared jointly by Matthias Liebeck and Pashutan Modaresi. **Status**: Published.

Author masking is the task of paraphrasing a document so that its writing style no longer matches that of its original author [LMC16]. Although author masking and automatic text summarization may seem to have no commonalities, their evaluation techniques have many similarities. In both author masking and automatic summarization, the resulting documents should be evaluated regarding their coherence and grammaticality. Moreover, the resulting documents should reflect the semantic content of the original documents. A major difference is that in author masking the obfuscated documents should not preserve the writing style of the original documents, while in summarization, this can be an additional requirement.

Due to the similarities between the evaluation of summarization and obfuscation systems, and also available data for the evaluation of obfuscation systems, in this section, we propose manual and automatic evaluation methods for obfuscation systems and claim that the proposed methods can easily be adapted to summarization systems.

We evaluate an obfuscation system based on three criteria. An obfuscation system is called *safe* if a forensic analysis does not reveal the original author of the obfuscated texts. To measure the safeness of an obfuscation system, we propose the use of an automatic author verification method called GLAD [HWvdB+15]. We call an obfuscated text *sound* if it is textually entailed with the original document. To evaluate the soundness of the obfuscated texts, we used the automatic paraphrase detection approach proposed in Section 5.1. Additionally, we call an obfuscated text *sensible* if it is inconspicuous. Sensibleness can be interpreted as the language quality of the obfuscated texts. To evaluate the sensibleness of the obfuscated texts, we propose a manual evaluation method and report our results in terms of the integer scores in the range of $[0, 2]$.

The proposed approaches for the evaluation criteria introduced above (except safeness) can also be applied to evaluate the qualities of automatically created text summaries.

Evaluating Safety, Soundness and Sensibleness of Obfuscation Systems

Notebook for PAN at CLEF 2016

Matthias Liebeck, Pashutan Modaresi, and Stefan Conrad

Institute of Computer Science
Heinrich Heine University Düsseldorf
D-40225 Düsseldorf, Germany
{liebeck, modaresi, conrad}@cs.uni-duesseldorf.de

Abstract Author masking is the task of paraphrasing a document so that its writing style no longer matches that of its original author. This task was introduced as part of the 2016 PAN Lab on Digital Text Forensics, for which a total of three research teams submitted their results. This work describes our methodology to evaluate the submitted obfuscation systems based on their safety, soundness and sensibleness. For the first two dimensions, we introduce automatic evaluation measures and for sensibleness we report our manual evaluation results.

1 Introduction

Author masking is the task of paraphrasing a document so that its writing style no longer matches that of its original author. Due to the advances in fields such as *authorship attribution* and *author verification*, it is not clear whether authors (particularly in the age of the Internet and social media) can assure their anonymity anymore [20]. While in some scenarios, such as verifying the authorship of disputed novels or revealing the author of harassing messages in social media [19], author unmasking might be useful, there are situations where authors have the right to protect their privacy, among them the desire to avoid retribution from an employer or government agency [10].

The task of author masking was introduced as part of the 2016 PAN Lab on Digital Text Forensics [5], for which a total of three research teams, namely Mansourizade et al. [13], Keswani et al. [11] and Mihalvoya et al. [14] (called Team A, B and C respectively in the rest of this work) submitted their results. The evaluation was completely anonymous and the identities of the teams were revealed after the submission of our evaluation results.

Together with the task of author masking, *obfuscation evaluation* has been introduced as another task to evaluate the performance of the author masking submissions. Three dimensions have been defined by the task organizers for the performance evaluation of the obfuscation systems: *safety* to ensure that a forensic analysis does not reveal the original author of an obfuscated text; *soundness* to evaluate if the obfuscated texts are textually entailed with their originals; and *sensibleness* to ensure that the obfuscated texts are inconspicuous [18].

In this work, we describe our methodology to evaluate the performance of the submitted systems based on the aforementioned dimensions. In section 2 we define the problem of author masking more concretely and describe the provided training data. The evaluation results of the dimensions safety, soundness and sensibleness are reported in sections 3, 4 and 5, respectively. Finally, we conclude our work in section 6.

2 Problem Definition

Given a document, an author masking software has to paraphrase it so that its writing style no longer matches that of its original author. Although the organizers of the author masking task do not directly define this task as a supervised machine learning problem, a training set is provided so that the participant can evaluate their designed algorithms based on this dataset. The same dataset is also used as the test dataset for the final evaluation.

The provided dataset is a collection of 205 problems selected from author verification tasks from PAN2013 [9], PAN2014 [22] and PAN2015 [21]. Each problem is a collection of at most five known documents (written by the same author) and a questioned document. Normally in author verification problems, the author of the questioned document is unknown and the task of an author verifier is to figure out whether the questioned document has the same author as the known documents or not. But in the training dataset of the author masking task, all problems are selected from positive instances, meaning all questioned documents have the same author as the known documents. The language of all provided problems is English.

The participants were asked to develop a software that outputs a detailed list, how each piece of the original text has been paraphrased. For a detailed description of the desired system output, the reader is referred to the official task page[1].

3 Safety

An obfuscation software is called safe, if a forensic analysis does not reveal the original author of the obfuscated texts. We evaluate the safety of the obfuscation software using an automatic author verifier called GLAD [8]. The idea behind this automatic evaluation measure is that if an obfuscation system successfully masks the authors of the *questioned* documents in the training set (remember that all problems in the training set belong to the positive class), the author verifier will classify the problems as negative (meaning that the obfuscated document no longer has the same author as the other documents).

The GLAD algorithm was one of the top ranked systems at PAN2015 and treats the author verification problem as an intrinsic binary classification machine learning task. GLAD uses SVM [7] as a learning algorithm and makes use of simple feature classes such as N-Grams, tokens, sentences, visual, compression, entropy and syntactic features [8].

[1] http://pan.webis.de/clef16/pan16-web/author-obfuscation.html

To train the GLAD algorithm we used the English problems from the training datasets from PAN2013 to PAN2015. The statistics of the training dataset used are shown in Table 1.

Table 1. Statistics of the dataset used to train GLAD

	#Problems	#Documents	Avg. #Known Documents	Avg. #Tokens
PAN2015	100	200	1.0	366
PAN2014 (Essays)	200	725	2.6	848
PAN2014 (Novels)	100	200	1.0	3137.8
PAN2013	10	42	3.2	1037

Notice that the training dataset from PAN2014 consisted of novels and essays and we took both categories to train our model.

Using the trained model, we measure the performance of the GLAD algorithm once using the original documents from the training set of the author masking problem (labels are all considered to be true), and once on the submissions of each team (labels are all considered to be true). As the evaluation measure we use $c@1$ [17], as defined in Equation 1. The $c@1$ measure is a modified version of accuracy, where $c@1$ rewards approaches that maintain the same number of correct answers and decrease the number of incorrect answers by leaving some problems unanswered.

$$c@1 = \frac{1}{n} \cdot \left(n_c + \frac{n_u n_c}{n} \right) \tag{1}$$

In the definition of $c@1$ measure, n is the number of problems, n_c is the number of correct answers and n_u represents the number of unanswered problems.

Table 2 summarizes the evaluation results of the safety dimension. On the original dataset (the training dataset of author masking), the GLAD algorithm has a $c@1$ score of 0.656, meaning that the algorithm correctly classifies roughly 65% of the problems as positive. Notice that the gold standard labels are all set to be true and that teams having lower $c@1$ scores are more successful at the masking task than the other teams. It is also observable that none of the participants has a $c@1$ score below 0.5. This means that the portion of successful obfuscations for all submissions is below 50%. At the same time it can be seen that all submissions have a $c@1$ score below the baseline 0.656, meaning that all teams were partly successful at the author masking task.

Table 2. Evaluation results of the safety dimension

	Team A	Team B	Team C	Original
C@1	0.585	0.532	0.522	0.656

Although in previous PAN competitions, AUC (Area Under the Curve) [6] was also used to evaluate the automatic verifiers, the use of this measure was not possible in our scenario as the test datasets contain either only positive or only negative instances.

Another interesting analysis is to investigate the relation between true positives and false negatives. The idea behind this analysis is to figure out the portion of documents classified as positive before obfuscation, and the ones classified as negative after obfuscation. For this we select true positives from the original dataset and count the ones that have been classified as negative by the GLAD algorithm. Table 3 summarizes the results.

Table 3. Evaluation results of the safety dimension

	Team A	Team B	Team C
FN / TP	0.159	0.254	0.290

Notice that higher values in Table 3 are preferred. Team C has the highest score among the teams and has managed to obfuscate roughly 30% of the true positive problems to false negative ones. These results are consistent with the results shown in Table 2.

4 Soundness

We assume that the goal of author masking is to reword a text segment into a paraphrased one while retaining as much semantic similarity as possible. Therefore, we propose to quantify soundness by measuring the semantic textual similarity (STS) between the original text segment and its corresponding obfuscation.

The prediction of semantic textual similarity has been a recurring task in SemEval challenges since 2012 [1–4]. The aim of the STS task is to determine the semantic similarity of two sentences in the continuous interval $[0, 5]$ where 0 represents a complete dissimilarity and 5 denotes a complete semantic equivalence between the sentences. The task organizers provide sentence pairs with gold standards from different categories. The task is evaluated by calculating the Pearson correlation between the predicted values and a crowdsourced gold standard.

In this paper, we use the unsupervised semantic similarity approach called *Overlap* [12] to automatically determine the semantic similarity between the original segment and its paraphrase. There are two advantages of using an unsupervised approach: (i) human annotators can only annotate a subset of the paraphrases within a reasonable amount of time. An automatic approach can evaluate all original-paraphrase pairs and (ii) we do not need labeled training data as compared to a supervised approach.

The idea of the *Overlap* method is simple since it measures the overlap between the tokens in the original segment s_1 and the tokens in the paraphrase s_2 by aligning tokens to the best match in the other text segment. The authors first define a similarity function for two tokens which uses synsets from *WordNet* [16] and word embeddings from word2vec [15], as denoted in Equation 2.

$$\text{sim}(t_1, t_2) := \begin{cases} 1 & \text{if } t_1.\text{lemma} == t_2.\text{lemma} \\ 1 & \text{if } t_1 \text{ and } t_2 \text{ have the same most common synset} \\ 0.5 & \text{if } t_1 \text{ and } t_2 \text{ share any other synset} \\ \cos(t_1, t_2) & \text{if } t_1 \text{ and } t_2 \text{ have } \textit{word2vec} \text{ embeddings} \\ \textit{0.15} & \text{otherwise} \end{cases} \tag{2}$$

Afterwards, the similarity score between two text segments in $[0, 5]$ is defined as follows:

$$\text{STS}(s_1, s_2) := 5 \cdot \left(\frac{\sum_{t_1 \in s_1} \max_{t_2 \in s_2} \text{sim}(t_1, t_2)}{2 \cdot |s_1|} + \frac{\sum_{t_2 \in s_2} \max_{t_1 \in s_1} \text{sim}(t_2, t_1)}{2 \cdot |s_2|} \right) \tag{3}$$

Since we assume the obfuscations to be semantically as close as possible to the originals, the STS score between both segments should be 5. We predict the semantic similarity for all pairs for each team. Afterwards, we average over the predicted scores for each team. Table 4 summarizes the results for the soundness dimension.

Table 4. Evaluation results of the soundness dimension

	Team A	Team B	Team C
Mean STS	4.87	4.04	4.48

For the soundness dimension, the best semantic paraphrases were created by team A with an average STS score of 4.87. This is not surprising since team A only substituted a few words and often kept the original segment as a paraphrase. Therefore, the paraphrases are semantically very close or even identical to the original. Team C achieved a mean STS score of 4.48 and team B had the lowest score with 4.04. Since the *Overlap* approach from [12] is independent of the word order, the results of team B cannot be explained by changing the word order of the phrases. One factor that definitely influenced the semantic similarity is the appearance of German words in the paraphrases, which cannot be matched to the English tokens in the original texts.

5 Sensibleness

The dimension sensibleness describes the language quality of the obfuscations and whether it allows us to understand them. An author masking software might mask the author of a text at the cost of its comprehension. Therefore, it is also crucial to evaluate the quality of the produced obfuscations.

We observed that teams A and C used dictionaries to perform simple substitutions and team B usually changed the order of phrases. It is surprising to see that the paraphrases by team B sometimes contain random German words, as in the following example: "*it is difficult to across, Once the <u>Mitbürgers</u> <u>unschön</u> is faint, odor street, on the village so massed mold <u>Verfalls</u> and centuries.*"

Although there are approaches to automatically predict the grammatical quality of text, we chose to manually evaluate the sensibleness because portions of the text have a low language quality but still allow for a limited understanding of the content. For example, this can be compared to a non-native speaker who asks in an online forum a question that is poorly worded but still comprehensible.

After a manual inspection of a subset of the paraphrases from all three teams, we decided to annotate each pair with a score $s \in \{0, 1, 2\}$ to measure the language quality. We then drew a small sample and discussed annotation guidelines. Our three labels and their definitions are described in Table 5.

Table 5. Labels for the sensibleness dimension

Score	Name	Definition
2	*comprehensible*	The paraphrase can be understood immediately. Example: "*These things are deeply rooted in the Swedish people.*"
1	*partially comprehensible*	The paraphrase can be understood with some restrictions. It can contain smaller errors or some smaller parts that are incomprehensible. Example: "*they him. But ignored*"
0	*incomprehensible*	The language quality of the paraphrase is too low to allow any understanding of the content. Example: "*I a In certain years in a bookstore can help , than English , French English. French*"

In our evaluation, sensibleness is only evaluated by looking at the obfuscated text. This is due to the fact that only the paraphrased text after author masking is used in a real world scenario. Therefore, it is reasonable to only evaluate the output of the system. We ignore spacing and line breaks during the annotation process. Furthermore, we also ignore the substitutions of the words "*oof*" and "*tto*" from team C because they do not impact the understanding of the text.

We randomly drew a subset of 20 problems. For each team, we then drew three obfuscations per problem. All of these obfuscations were manually annotated by three annotators. In order to report a single value per team, we averaged all the scores from the annotators. Table 6 summarizes the results for the sensibleness dimension.

Table 6. Evaluation results of the sensibleness dimension

	Team A	Team B	Team C
Average score	1.94	0.57	1.20

Team A achieved the best results in the sensibleness dimension with an average score close to 2. The paraphrases from team B allow for the lowest understanding of all three teams with an average score of 0.57 which is between *partially comprehensible* and *incomprehensible*.

We should note that there are at least two problems for the evaluation of the sensibleness dimension: (i) it is difficult to formalize language quality and understanding and (ii) the sensibleness dimension is subjective. Although we observed a high agreement on the category *incomprehensible*, we had a lower agreement on whether a paraphrase is fully or partially comprehensible. This is plausible since one annotator might perfectly understand a text segment while another annotator may have some troubles with it.

6 Conclusion

In this work, we discussed our methodology to evaluate the performance of the obfuscation systems submitted to the PAN2016 Author Masking shared task. More concretely, submissions were evaluated based on their safety (Section 3), soundness (Section 4), and sensibleness (Section 5). The scripts for our evaluation are available on GitHub[2].

An automatic author verifier was used to measure the safety of the submissions. The ranking of the teams in terms of safety is as follows: team C, B, and A

We proposed to quantify soundness by automatically measuring the semantic text similarity between the original text fragments and their obfuscations. The best score was achieved by team A, followed by teams C and B.

Unlike the first two dimensions, the sensibleness of the submissions was evaluated manually. As sensibleness is subjective and difficult to formally define, we consider its measurement a nontrivial task. Regarding sensibleness, teams A, C and B were ranked first, second, and third, respectively.

Acknowledgments

This work was partially funded by the PhD program *Online Participation*, supported by the North Rhine-Westphalian funding scheme *Fortschrittskollegs* and by the German Federal Ministry of Economics and Technology under the ZIM program (Grant No. KF2846504). We would like to thank Daniel Braun for his help in the evaluation of the sensibleness dimension.

References

1. Agirre, E., Banea, C., Cardie, C., Cer, D., Diab, M., Gonzalez-Agirre, A., Guo, W., Lopez-Gazpio, I., Maritxalar, M., Mihalcea, R., Rigau, G., Uria, L., Wiebe, J.: SemEval-2015 Task 2: Semantic Textual Similarity, English, Spanish and Pilot on Interpretability. In: Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015). pp. 252–263. Association for Computational Linguistics (2015)
2. Agirre, E., Banea, C., Cardie, C., Cer, D., Diab, M., Gonzalez-Agirre, A., Guo, W., Mihalcea, R., Rigau, G., Wiebe, J.: SemEval-2014 Task 10: Multilingual Semantic Textual Similarity. In: Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014). pp. 81–91. Association for Computational Linguistics and Dublin City University (2014)

[2] https://github.com/pasmod/obfuscation

3. Agirre, E., Cer, D., Diab, M., Gonzalez-Agirre, A.: SemEval-2012 Task 6: A Pilot on Semantic Textual Similarity. In: *SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012). pp. 385–393. Association for Computational Linguistics (2012)
4. Agirre, E., Cer, D., Diab, M., Gonzalez-Agirre, A., Guo, W.: *SEM 2013 shared task: Semantic Textual Similarity. In: Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity. pp. 32–43. Association for Computational Linguistics (2013)
5. Balog, K., Cappellato, L., Ferro, N., Macdonald, C. (eds.): CLEF 2016 Evaluation Labs and Workshop – Working Notes Papers. CEUR Workshop Proceedings, CEUR-WS.org (2016)
6. Bradley, A.P.: The Use of the Area Under the ROC Curve in the Evaluation of Machine Learning Algorithms. Pattern Recogn. 30(7), 1145–1159 (Jul 1997)
7. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning 20(3), 273–297 (1995)
8. Hürlimann, M., Weck, B., van den Berg, E., Suster, S., Nissim, M.: GLAD: Groningen Lightweight Authorship Detection. In: Working Notes of CLEF 2015 - Conference and Labs of the Evaluation forum (2015)
9. Juola, P., Stamatatos, E.: Overview of the Author Identification Task at PAN 2013. In: CLEF 2013 Evaluation Labs and Workshop – Working Notes Papers (2013)
10. Kacmarcik, G., Gamon, M.: Obfuscating Document Stylometry to Preserve Author Anonymity. In: Proceedings of the COLING/ACL on Main Conference Poster Sessions. pp. 444–451. COLING-ACL '06, Association for Computational Linguistics (2006)
11. Keswani, Y., Trivedi, H., Mehta, P., Majumder, P.: Author Masking through Translation—Notebook for PAN at CLEF 2016. In: CLEF 2016 Evaluation Labs and Workshop – Working Notes Papers. CEUR-WS.org (2016)
12. Liebeck, M., Pollack, P., Modaresi, P., Conrad, S.: HHU at SemEval-2016 Task 1: Multiple Approaches to Measuring Semantic Textual Similarity. In: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016). pp. 607–613. Association for Computational Linguistics (2016)
13. Mansourizade, M., Rahgooy, T., Aminiyan, M., Eskandari, M.: Author Obfuscation using WordNet and Language Models—Notebook for PAN at CLEF 2016. In: CLEF 2016 Evaluation Labs and Workshop – Working Notes Papers. CEUR-WS.org (2016)
14. Mihaylova, T., Karadjov, G., Nakov, P., Kiprov, Y., Georgiev, G., Koychev, I.: SU@PAN'2016: Author Obfuscation—Notebook for PAN at CLEF 2016. In: CLEF 2016 Evaluation Labs and Workshop – Working Notes Papers. CEUR-WS.org (2016)
15. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient Estimation of Word Representations in Vector Space. ICLR Workshop (2013)
16. Miller, G.A.: WordNet: A Lexical Database for English. Communications of the ACM 38(11), 39–41 (1995)
17. Peñas, A., Rodrigo, A.: A Simple Measure to Assess Non-response. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1. pp. 1415–1424. HLT '11, Association for Computational Linguistics (2011)
18. Potthast, M., Hagen, M., Stein, B.: Author Obfuscation: Attacking the State of the Art in Authorship Verification. In: Working Notes Papers of the CLEF 2016 Evaluation Labs. CEUR Workshop Proceedings, CLEF and CEUR-WS.org (Sep 2016)
19. Rangel, F., Celli, F., Rosso, P., Potthast, M., Stein, B., Daelemans, W.: Overview of the 3rd Author Profiling Task at PAN 2015. In: CLEF 2015 Evaluation Labs and Workshop – Working Notes Papers. CEUR-WS.org (2015)
20. Rao, J.R., Rohatgi, P.: Can Pseudonymity Really Guarantee Privacy? In: Proceedings of the 9th USENIX Security Symposium. pp. 85–96. USENIX (2000)

21. Stamatatos, E., amd Ben Verhoeven, W.D., Juola, P., López-López, A., Potthast, M., Stein, B.: Overview of the Author Identification Task at PAN 2015. In: CLEF 2015 Evaluation Labs and Workshop – Working Notes Papers. CEUR-WS.org (2015)
22. Stamatatos, E., Daelemans, W., Verhoeven, B., Potthast, M., Stein, B., Juola, P., Sanchez-Perez, M., Barrón-Cedeño, A.: Overview of the Author Identification Task at PAN 2014. In: CLEF 2014 Evaluation Labs and Workshop – Working Notes Papers. CEUR-WS.org (Sep 2014)

6.2 On (Commercial) Benefits of Automatic Text Summarization Systems in the News Domain

Pashutan Modaresi, Philipp Gross, Siavash Sefidrodi, Mirja Eckhoff and Stefan Conrad. On (Commercial) Benefits of Automatic Text Summarization Systems in the News Domain: A Case of Media Monitoring and Media Response Analysis
Contributions: The author contributed with the conduction of the experiment and prepared the manuscript. **Status**: Submitted to EACL2017.

Summarization systems have been mainly evaluated in an intrinsic manner, meaning that the linguistic quality of the summaries has always been in the foreground. Not only the linguistic quality of the summaries, but also measures such as the coherence of the created summaries, their compression ratio and informativeness have been considered as critical criteria for the quality of the summaries.

Extrinsic evaluation has also been used widely to evaluate summarization systems. For example, in [HH12] the quality of the summaries were assessed based on their effectiveness in the task of automatic topic identification. In another extrinsic evaluation method, the qualities of the summaries were assessed in the context of question answering, where summaries of the documents were presented to the users instead of the original document, and they were asked to answer several questions only based on the summaries.

Although intrinsic and extrinsic evaluation methods are an essential part of any automatic summarization system, evaluating summarization systems from a commercial point of view have been mostly neglected in the research community. The central question to answer here is whether incorporating automatic summarization systems in companies will have financial benefits for them. A general answer to the question is, of course, not possible as the results will always depend on the activity domain of the companies. In this Section, we present the results of our attempt to answer this question in the field of media monitoring and media response analysis.

In our study, we investigate the performance of a group of eight media analysts, once when they accomplish their daily tasks using original documents and once when we provide them with summaries instead of the original documents. As a direct assessment of the reduction of the company costs by use of summaries is not a trivial task, we measure the decrease in the required time to accomplish the daily tasks by media analysts and claim that it is directly correlated to the reduction in costs in the company. Moreover, we also show that despite the decrease in the time, no significant reduction in the work quality of the media analysts could be observed.

On (Commercial) Benefits of Automatic Text Summarization Systems in the News Domain: A Case of Media Monitoring and Media Response Analysis

Pashutan Modaresi and **Philipp Gross** and **Siavash Sefidrodi** and **Mirja Eckhof**
pressrelations GmbH
D-40211 Düsseldorf, Germany
[first name].[last name]@pressrelations.de

Stefan Conrad
Institute of Computer Science
Heinrich Heine University of Düsseldorf
conrad@cs.uni-duesseldorf.de

Abstract

In this work, we present the results of a systematic study to investigate the (commercial) benefits of automatic text summarization systems in a real world scenario. More specifically, we define a use case in the context of media monitoring and media response analysis and claim that even using a simple query-based extractive approach can dramatically save the processing time of the employees without significantly reducing the quality of their work.

1 Introduction

Automatic text summarization has been an evolving field of research. Having started with the pioneering work of Luhn (Luhn, 1958), specifically in recent years, automatic text summarization has made remarkable signs of progress with the popularity of deep learning approaches (Rush et al., 2015; Chopra et al., 2016).

Providing a formal definition of automatic text summarization is rather a challenging task. This work pursues the following definition: Given a set Q of queries, automatic text summarization is a reductive transformation of a collection of documents D with $|D| > 0$ into a single or multiple target document(s), where the target document(s) are more readable than the documents in D and contain the relevant information of D according to Q (Modaresi and Conrad, 2015). This definition, comprises both extractive and abstractive approaches, where by extractive we mean methods that select the most salient sentences in a document and by abstractive we mean methods that incorporate language generation to reformulate a document in a reductive way.

Automatic text summarization has been applied to many domains, among which is the news domain the focus of this work. Despite many attempts to improve the performance of summarization systems (Ferreira et al., 2016; Wei and Gao, 2015), to the best knowledge of the authors, no systematic study was performed to investigate the (commercial) benefits of the summarization systems in a real world scenario.

We claim that using (even very) simple automatic summarization systems can dramatically improve the workflow of employees without affecting their quality of work. To investigate our claim we define a use case in the context of media monitoring and media response analysis (Section 2) and establish several criteria to measure the effectiveness of the summarization systems in our use case (Section 3). In Section 4 we discuss the design of our experiment and report the results in Section 5. Finally, in Section 6 we conclude our work.

2 Use Case Definition

We investigate the (commercial) benefits of integrating an automatic summarization system in the semi-automatic workflows of media analysts doing *media monitoring* and *media response analysis* at pressrelations GmbH[1]. In the following, we

[1]http://www.pressrelations.de/

shortly define the terms as mentioned above.

The goal of media monitoring is to gather all relevant information on specific topics, companies or organizations. To this end, search queries are defined, with which the massive amount of available information can be filtered automatically. Typically, in a post-processing step, the quality of the gathered information is increased using manual filtering by trained media analysts.

In media response analysis, the publications in the media (print media, radio, television, and online media) are evaluated according to various predefined criteria. As a result of this, it is possible to deduce whether and how journalists have recorded and processed the PR (Public Relations) messages. Possible questions to be answered in the context of media response analysis are: How are the publications distributed over time? How many listeners, viewers or readers were potentially reached? What are the tonality and opinion tendency of the publications? (Grupe, 2011)

Typically, analysis results are given to the clients in the form of extensive reports. In the case of textual media, the immense amount of time required to read texts and to write abstracts and reports is a high cost factor in the preparation of media resonance analysis reports.

We claim that the described process can be partially optimized by incorporating automatic summarization systems, leading to remarkable financial advantages for the companies.

3 Evaluation Criteria

From the commercial and academic point of view, the *quality* of the summaries plays an import role. Various automatic methods such as ROUGE (Lin, 2004), BLEU (Papineni et al., 2002), and Pyramid (Nenkova et al., 2007) have been used successfully to evaluate the quality of the summaries. Moreover, manual evaluation has also been incorporated for quality assessment of the summaries (Modaresi and Conrad, 2014). Another important criterion that is mostly neglected in academic publications is the *gain in time*, defined as the amount of saved time by a user through the usage of the summaries.

In our use case, the quality of a summary comprises of two aspects: *completeness* and *readability*. The term *completeness*, describes the requirement of a summary to contain all relevant information of an article. The relevance of information is determined based on a *query*. For instance, the query might be a named entity, and we expect that the summary contains all relevant information regarding the named entity.

The term *readability* refers to the coherence and the grammatical correctness of the summary. While the grammatical correctness is defined at the sentence level, the coherence of the summary is determined on the whole text. That means that the sentences of the summary should not only be grammatically correct in isolation, but also they must be coherent to make the summary readable.

Both *completeness* and *readability* are criteria that are difficult to evaluate and define formally, and it has been shown that they are both very subjective criteria, where their assessment varies from person to person (Torres-Moreno, 2014). In the case of completeness, it is unclear how to formalize the relevance of information, and in the case of readability the same holds for the concept of coherence.

Therefore, we define the quality of a summary from a practical and commercial point of view. For this, we define the quality of a summary in terms of a binary decision problem where the question to be asked is: *can the produced summary in its current form be delivered to a customer or not?*

Furthermore, in our use case, the *gain in time* is defined as the processing time that can be saved by media analysts, assisted by a summarization system. It should be clear that the reduction of the processing time could lead to the reduction of costs in a company.

In the following section, the design of our experiment with respect to the criteria mentioned above (quality and gain in time) will be explained.

4 Experiment Setup

To conduct our experiments we incorporated eight media analysts (specialists in writing summaries for customers) and divided them into two equisized groups. One group received only the news articles (Group A), and the other one received only the query-based extracted summaries without having access to the original articles. Given a query consisting of a single named entity, both groups were asked to write summaries with the following properties:

- The summary should be compact and consist of maximum two sentences.

- The summary should contain the main topic of the article and also the most relevant information regarding the query.

As previously stated, the summaries created by media analysts were evaluated based on two criteria: *quality* and *gain in time*. The *gain in time* was measured automatically using a web interface by tracking the processing time of the media analysis for creating the text summaries. We interpret the *gain in time* as the answer to the question: *In average, what percentage faster/slower is group A in compare to group B?*. Let $\tilde{t}_A$ and $\tilde{t}_B$ be the average processing times of the media analysts in group A and B respectively. We define *gain in time* as in Equation 1.

$$g(\tilde{t}_A, \tilde{t}_B) = \begin{cases} 100 \cdot \left|1 - \frac{\tilde{t}_A}{\tilde{t}_B}\right| & \text{if } \tilde{t}_A \leqslant \tilde{t}_B \\ 100 \cdot \left|1 - \frac{\tilde{t}_B}{\tilde{t}_A}\right| & \text{if } \tilde{t}_A > \tilde{t}_B \end{cases} \quad (1)$$

Notice that it holds $g(\tilde{t}_A, \tilde{t}_B) = g(\tilde{t}_B, \tilde{t}_A)$ and g reflects only the magnitude of the saved time and not its direction. The direction can be determined based on the values of $\tilde{t}_A$ and $\tilde{t}_B$.

On the other hand, the quality of the summaries was evaluated by a curator (an experienced media analyst in direct contact with customers). The curator was responsible for evaluating the summaries created by media analysts in both groups and scored them with a zero or a one. With *zero* meaning that the quality of the summary is not sufficient and the product cannot be delivered to the client and with *one* meaning the quality of the summary is sufficient enough to be delivered to the customer. Let the vector q of size m be a one-hot vector consisting of 0s and 1s, where the i-th element in q represents the evaluation of the curator for the i-th summary among the m available summaries. Given that, we compute the average summary quality of a set of summaries by computing the average of its corresponding evaluation vector q.

In total, ten news articles were provided to the media analysts. The articles for group A had an average word count of 1438 with the standard deviation being 497. Group B received only the summaries of the articles, created automatically with a heuristic-based approach. The automatically generated summaries had an average length of 81 words with the standard deviation being 23.

Algorithm 1 Query-based Summarization

```
1:  procedure SUMMARIZE(T, Q)
2:      $S \leftarrow \varnothing$
3:      $T' \leftarrow$ Segment($T$)
4:      $E \leftarrow$ EntityDistribution($T'$)
5:      $m \leftarrow$ Median($E$)
6:      $E' \leftarrow \varnothing$
7:      for e in E do
8:          if freq(e) > m then
9:              $E' \leftarrow E' \cup e$
10:     $S \leftarrow$ Lead($T'$)
11:     $S \leftarrow$ QueryMatch($T', Q$)
12:     $S \leftarrow$ CentralEntityMatch($T', E'$)
13:     return S
```

The pseudocode of the invoked query-based extractive summarizer is depicted in Algorithm 1.

In line 2 the summary S is initialized with an empty set. Given the input text T, the text is segmented into sentences and stored in the list T' (line 3). In line 4, the named entities of the text are recognized and stored in a dictionary where each key represents a named entity, and its corresponding value is the frequency of the named entity in the text. Lines 5-9 depict the procedure to select *central named entities*. Let m be the median of the named entities frequencies. A named entity e is called a *central named entity* if its frequency in the text is higher than twice the median. In line 10 we add the lead of the news article to the summary, as the lead usually can be interpreted as a compact summary of the whole article. Afterwards in line 11, the sentences that contain the query Q are added to the summary. Finally, we extend the list of summary sentences with sentences containing the *central named entities* and return the summary.

5 Results

In total, we collected 80 summaries created by the media analysts in both groups. For each summary, its processing time and its quality evaluated by a curator was recorded. Based on the collected data, we answered the following questions:

1. Intergroup processing time: Is there a significant difference between the processing times of individual media analysts in a group?

2. Intergroup quality: Is there a significant difference between the quality of the created summaries by the media analysts in a group?

3. Intragroup processing time: Is there a significant difference between the average processing times of media analysts in groups A and B? If so, which group has a faster processing time?

4. Intragroup quality: Is there a significant difference between the average qualities of created summaries by media analysts in groups A and B? If so, which group created more qualitative summaries?

The remaining of this section reports the answers to the above questions.

5.1 Intergroup Processing Time

The processing times of the media analysts in group A (A1-A4) and group B (B1-B4) are visualized using boxplots in Figures 1a and 1b respectively. In both groups, the differences among the average processing times are observable. Our goal is to investigate whether the differences between the processing times of media analysts is statistically significant.

To compare the means of processing times among the media analysts in a group we use the *one-way analysis of variance* (one-way ANOVA). The null hypothesis in the ANOVA test is that the mean processing times of the media analysts in a group are the same. To perform the ANOVA test we first examine if the requirements of the ANOVA test are satisfied (Miller, 1997).

The first requirement of the ANOVA test is that the processing times of the individual media analysts are normally distributed. For this, we use the Shapiro-Wilk test (Shapiro and Wilk, 1965) with the null hypothesis being that the processing times are normally distributed. Table 1 reports the results of the test.

Media Analyst	W	p-value
A1	0.90244	0.233
A2	0.91638	0.3277
A3	0.76592	0.0055
A4	0.73605	0.0024
B1	0.85143	0.0604
B2	0.95625	0.7425
B3	0.94609	0.6226
B4	0.93536	0.5026

Table 1: Shapiro-Wilk Test for Processing Times

In Table 1, W is the test statistic and we reject the null hypothesis if the p-value is less than the chosen significance level $\alpha = 0.05$. Thus the null hypothesis will be rejected for A3, A4, and B1, meaning that the processing times of them are not normally distributed. For other media analysts, the normality assumption holds. Although in several cases the normality requirement of the ANOVA test is violated, it is still possible to use the ANOVA test, as it was shown that the ANOVA test is relatively robust to the violation of the normality requirement (Kirk, 2012).

The second requirement to perform the ANOVA test is that the processing times of the media analysts have equal variances. For this, we use the Bartlett's test (Dalgaard, 2008) with the null hypothesis that the processing times of the media analysts have the same variance. The results of the Bartlett's test for groups A and B are reported in Table 2

Group	χ^2	p-value
A	4.3726	0.2239
B	6.9013	0.0751

Table 2: Bartlett's Test for Processing Times

In Table 2, χ^2 is the test statistic and we reject the null hypothesis if the p-value is less than the chosen significance level $\alpha = 0.05$. For both groups, the p-value is greater than the significance level, and thus there is no evidence that the variances of processing times of individual media analysts are different.

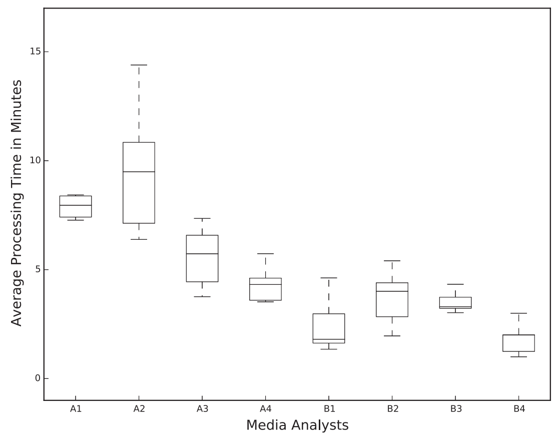
Having investigated the assumptions of the ANOVA test, we now report the results of the ANOVA test (See Table 3).

Group	F value	p-value
A	$1.413 \cdot 10^{33}$	$< 2 \cdot 10^{-16}$
B	$4.2 \cdot 10^{34}$	$< 2 \cdot 10^{-16}$

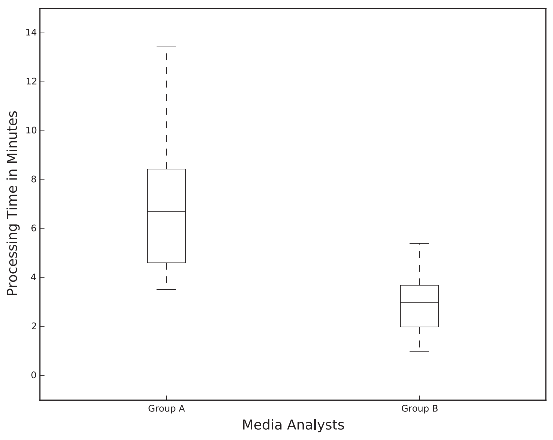
Table 3: ANOVA Test for Processing Times

In Table 3, the F value is the F test statistic and we reject the null hypothesis if the p-value is less than the chosen alpha level $\alpha = 0.05$. Thus, the mean processing times of media analysts in group A are not the same and there is a significant difference between them. The same hold for group B.

The results shown so far crystallize an important property of the summarization process. Given the same set of news articles and the same brief-



(a) Processing Times of Individual Media Analysts



(b) Processing Times of Groups A and B

Figure 1: Comparison of the Processing Times

ing to all media analysts, the average time required by the media analysts within a group to summarize the articles is significantly different from each other.

5.2 Intergroup Quality

The results of the manual evaluation of the summaries by the curator are represented in Table 4. In this section, our goal is to systematically investigate whether the qualities of the summaries produced by media analysts in a group are significantly different from each other.

Different from the previous section where we compared the processing times of the media analysts in a group using the ANOVA test, the comparison of the qualities among the media analysts cannot be performed using the ANOVA test (due to the huge violation of the normality assumption). Therefore, we interpret the evaluation results of each media analyst as a Binomial distribution $\mathrm{B}(n, p)$ with $n = 10$ (number of articles shown to each media analyst) and p being the numbers of times the curator was satisfied with the quality of the summaries created by the media analyst.

	Group A				Group B			
	A1	A2	A3	A4	B1	B2	B3	B4
Quality	0.9	1.0	0.8	0.6	0.3	0.9	1.0	0.7
Overall	0.82				0.72			

Table 4: Results of Manual Evaluation of Quality

To test whether the qualities of the produced summaries are significantly different from each other, we use the Fisher's Exact Test (Dalgaard, 2008) with the null hypothesis that the qualities are not different from each other. For group A we have a p-value of 0.1087, and for group B the p-value is 0.0022. Thus the null hypothesis can only be rejected for group B. The so far shown results lead us to the following conclusions: Given the news articles, no significant difference among the qualities of the produced summaries by the media analysts can be observed. Furthermore, given only the automatically created summaries, the media analysts produce summaries with significantly different qualities.

5.3 Intragroup Processing Time

So far, we only investigated the intergroup properties. In this section, we answer the question whether there exists a significant difference between the average processing times of group A and B?

In Figure 1b, the processing times of the groups A and B are compared using boxplots. Using the Equation 1, we compute the *gain in time* for group B, that is roughly 58%, meaning that as expected, the media analysts in group B required much less time to create the summaries in compare to the media analysts in group A. Similar to the Section 5.1, we use the ANOVA test to check the significance of this outcome. The results of the test are reported in Table 5.

In Table 5, F value is the F test statistic and we reject the null hypothesis if the p-value is less than the chosen alpha level $\alpha = 0.05$. Thus, the processing times of media analysts in group B are sig-

Group	F value	p-value
A vs. B	$2.14 \cdot 10^{33}$	$< 2 \cdot 2^{-16}$

Table 5: ANOVA Test for Intragroup Processing Times

nificantly lower than the processing times of media analysts in group A.

The results show that using a simple query-based extractive summarization system, the media analysts had a significant gain in time by the process of creating the text summaries.

5.4 Intragroup Quality

In the final step, we compare the quality of the produced summaries between both groups and answer the question whether there is a significant difference between the qualities? To answer this question we perform the Fisher's Exact Test and obtain the p-value of 0.4225. Thus the null hypothesis of the test cannot be rejected and we conclude that the qualities of the summaries among both groups are not significantly different.

Using the results above, we conclude that providing the media analysts with automatically created summaries does not have a negative impact on the quality of the summaries they generated and no significant difference in quality could be observed in compare to the media analysts that had access to the full new articles.

6 Conclusions

To investigate the (commercial) benefits of the summarization systems, we designed an experiment where two groups of media analysts were given the task to summarize news articles. Group A received the whole news articles and group B received only the automatically created text summaries. In summary, we showed that:

- The average time required by the media analysts within a group to summarize the articles is significantly different from each other.
- Given the news articles, no significant difference among the qualities of the produced summaries by the media analysts can be observed. Furthermore, given only the automatically created summaries, the media analysts produce summaries with significantly different qualities.
- The media analysts had a significant gain in time by the process of creating the text summaries (58%).
- Providing the media analysts with automatically created summaries does not have a negative impact on the quality of the summaries they generated

The results mentioned above indicate that incorporating even simple summarization systems can dramatically improve the workflow of the employees.

For future work we plan to repeat our experiment with more sophisticated summarization algorithms and compare the *gain in time* to our baseline setting. Furthermore, we plan to increase the number of media analysts to obtain more reliable results.

Acknowledgments

This work was funded by the German Federal Ministry of Economics and Technology under the ZIM program (Grant No. KF2846504).

References

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive Sentence Summarization with Attentive Recurrent Neural Networks. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 93–98.

P. Dalgaard. 2008. *Introductory Statistics with R*. Statistics and Computing. Springer New York.

Rodolfo Ferreira, Rafael Ferreira, Rafael Dueire Lins, Hilário Oliveira, Marcelo Riss, and Steven J. Simske. 2016. Applying Link Target Identification and Content Extraction to Improve Web News Summarization. In *Proceedings of the 2016 ACM Symposium on Document Engineering*, DocEng '16, pages 197–200. ACM.

S. Grupe. 2011. *Public Relations: Ein Wegweiser für die PR-Praxis*. Springer Berlin Heidelberg.

R.E. Kirk. 2012. *Experimental Design: Procedures for the Behavioral Sciences: Procedures for the Behavioral Sciences*. SAGE Publications.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81. Association for Computational Linguistics.

7 Digital Text Forensics

The fields of automatic text summarization and digital text forensics are closely related. Typical questions in the field of digital text forensics are: is a given text an original? Who is the author of a given text? Does a given text have a trusted source? [GPB$^+$13]

Automatic author profiling and verification and plagiarism detection have been used to answer the above questions. In author profiling, the goal is to predict the profile (age, sex or personality traits) of an author [MLC16a]. Given a set of documents with a known author, in author verification, we answer the question whether a questioned document and the set of given documents all have the same author [MG14]. In plagiarism detection and, more specifically, in text alignment, the goal is to identify all pairs of contiguous passages that are equal up to obfuscation [GM14].

In automatic text summarization and, specifically, for abstractive methods, it is a critical requirement that the writing styles of the original documents and the created summaries shall be almost identical. Preserving the writing style of the authors is a challenging problem. We claim that by using author profiling and author verification algorithms it can be evaluated whether the writing style of the original documents is preserved in the summaries. Author profiling can be used to test whether the author of the original document and the summary document both have the same profile. At the same time, author verification can be used to test whether the original and the summary document both have the same author.

Moreover, it is required that the original and summary document both transfer the same kind of information. To assure this requirement, we propose a text alignment method that can be used to examine whether both documents are the same. We claim that high plagiarism scores are a good indicator that both documents reflect the same kind of information.

7.1 A Language Independent Author Verifier Using Fuzzy C-Means Clustering

Pashutan Modaresi and Philipp Gross. A Language Independent Author Verifier Using Fuzzy C-Means Clustering. In *Working Notes for CLEF 2014 Conference*, pages 1084–1091, 2014.
Contributions: The author contributed with the design and development of the author verification algorithm and prepared the manuscript. **Status**: Published.

Authorship attribution is an important problem in the field of computational linguistics and has applications in areas such as law and journalism. The main question to answer in authorship attribution is: Given known sample documents from a small, finite set of candidate authors, which, if any, wrote a questioned document of unknown authorship? [SDV$^+$14]

A challenging sub-task of authorship attribution is the problem of automatic author verification. Given a set of documents written by a single person and a questioned document, the task is to determine whether the questioned document was written by the same person who wrote the known document set.

One of the popular approaches to tackle the problem of author verification is called the *unmasking* method [KSBD07]. The idea behind the unmasking method is to construct a classifier to discriminate between the set of documents with the known author and the questioned document. Having the classifier, the set of discriminating features can be identified. In the next step, the discriminating features will be removed and a new classifier will be constructed without the significant features. In the case that the questioned document has the same author as the known set, it is expected that the performance of the classifier decreases. The process of eliminating significant features will be repeated multiple times till we can make a meaningful statement about the authorship of the questioned document.

In the context of automatic text summarization, author verification can be used to determine whether the original and the summary document both have the same author or not. A positive classification result indicates that the summarization system was successful as preserving the writing style of the author in the summary document.

Our approach incorporated a fuzzy clustering algorithm and applies it on a large number of documents to construct clusters that are representative for different writing styles of the authors. The idea behind this approach is that two documents belonging to different clusters are probably not written by the same author.

A Language Independent Author Verifier Using Fuzzy C-Means Clustering

Notebook for PAN at CLEF 2014

Pashutan Modaresi[1,2] and Philipp Gross[1]

[1] pressrelations GmbH, Düsseldorf, Germany
{pashutan.modaresi, philipp.gross}@pressrelations.de
[2] Heinrich-Heine-University of Düsseldorf, Institute of Computer Science, Düsseldorf, Germany
modaresi@cs.uni-duesseldorf.de

Abstract In this work we describe our approach to solve the *author verification* problem introduced in the *PAN 2014 Author Identification* task. The *author verification* task presents participants with a set of problems where each problem consists of a set of documents written by the same author and a questioned document with an unknown author. The task is then to decide whether the questioned document has the same author as the other documents or not. Inspired by a psychological personality model, our approach uses basic lexical feature extraction and fuzzy clustering. Using the created fuzzy clusters, the membership values of documents to the clusters can be computed. The distribution of the cluster membership values will be used finally to solve the verification problem.

1 Introduction

Given a set of documents with known authors, *authorship attribution* is the task of identifying the author of an unseen document. Having a small number of candidate authors, this task can be easily solved using the state-of-the-art approaches[1]. A realistic and common scenario for *authorship attribution* is the *author verification* problem. Given a set of documents written by a single author, the task here is to determine whether a questioned document is written by the same author or not.

The *PAN 2014 Author Identification* task focuses on the *author verification* problem. To be more specific, in this task a multi-lingual corpus is provided which consists of several problems. Each problem contains a maximum of 5 documents written by a single author and a questioned document by an unknown author. The task in then to determine whether the questioned document is written by the same author or not.

The fact that an author may consciously or unconsciously vary his or her writing style, makes the task of *author verification* a hard problem[7]. In this paper we introduce a novel approach for solving the task of *author verification*. For this we extract language independent features from our training corpus and use a fuzzy clustering algorithm to construct our models. Finally using the membership distribution of documents over the clusters, we do solve the verification task.

In Section 2 we define the problem of *author verification* formally and introduce some notations. Section 3 addresses the process of feature extraction and normalization.

The process of clustering and model construction is discussed in Section 4. Section 5 covers the process of verification and scoring. An overview of the evaluation results can be seen in Section 6. Finally in Section 7 the work will be concluded.

2 Problem Statement

In this section we formally define the problem of *author verification* in the context of the *PAN 2014 Author Identification* task.

Let $P = \{D, d_u\}$ be a problem consisting of a set of documents $D = \{d_1, \ldots, d_n\}$ with $1 \leq n \leq 5$ written by a single author, and a questioned document d_u with an unknown author. The task in *author verification* is to determine whether the questioned document d_u is written by the same author or not. We denote the author of a document d_i by $\mathcal{A}(d_i)$. In other words an author verifier φ is a binary classification function of the following form:

$$\varphi(d_u, D) = \begin{cases} 1, & \text{if } \mathcal{A}(d_u) = \mathcal{A}(d_i) \ \ \forall d_i \in D \\ 0, & \text{if otherwise} \end{cases} \tag{1}$$

In the *PAN 2014 Author Identification* task, problems are from 4 different languages, namely Dutch, English, Greek and Spanish. The *author verification* algorithm has to be able to deal with documents from the specified languages. The performance of the *author verifier* will be evaluated according to the area under the ROC curve (AUC) of its probability scores and also based on the c@1 measure[8]. The evaluation process will be discussed in more details in Section 6.

In the following section, we start the description of our algorithm by discussing the feature extraction and normalization step.

3 Feature Extraction and Normalization

Feature extraction is considered as one of the important steps in *author verification*[9]. Different kinds of stylometric features like lexical, syntactic or semantic features have been used for solving the *author verification* task. In order to design an efficient *author verification* algorithm, which can deal with huge amounts of documents, we only consider a limited number of lexical features and construct our learning algorithm in a way that would result in an acceptable performance even with a small number of features. Lexical features have the advantage over the syntactic or semantic features, that this kind of features can be computed very efficiently and without the use of any external knowledge or training.

We represent documents as vectors in $\mathbb{R}^4$. Each component of these 4-dimensional vectors can be computed using the feature extraction functions. Independent of the document language we use the following functions to compute the feature vector components of documents:

Average Sentence Length *(f_{sl})* : Using a sentence detector, sentence boundaries of the document will be detected (In our case we use a regular expression based sentence detector for optimizing the performance). For each sentence s in the document, its length $l(s)$ will be computed. We denote the set of all sentences inside a document with S. Finally the average sentence length of the document can be computed as follows:

$$f_{sl}(d) = \frac{\sum_{s \in S} l(s)}{|S|} \tag{2}$$

Punctuation Marks Usage *(f_{pm})* : Using a predefined set of punctuation marks $T = \{ \; (\;) \; , \; : \; ; \; ! \; ? \; \}$ the frequency of the elements of the set T inside the document will be computed and finally normalized by the length of the document. With $f(t, d)$ we denote the frequency of the punctuation mark t in document d.

$$f_{pm}(d) = \frac{\sum_{t \in T} f(t, d)}{|d|} \tag{3}$$

Space After Comma *(f_{sac})* : Our experimental results show that whether a space is used after a comma or not, can be a good discriminating feature in the *author verification* task. Let α denote the number of times a comma is followed by a space and β be the number of times a comma is not followed by a space. In his way f_{sac} can be defined as follows:

$$f_{sac}(d) = \frac{\alpha - \beta}{|d|} \tag{4}$$

Analogue to f_{sac} we define f_{sbc} which is the *Space Before Comma* feature. Through this feature, authors that use a space before comma can be discriminated from the ones who do not use a space before comma.

As the extracted features may exhibit significant differences in their range and distribution, out learning algorithm could be more sensitive to features that are in a wider range (e.g. Average Sentence Length). In order to avoid this behavior we use feature normalization through which we can modify the mean and variance of the features using a transformation function. The transformation function that we use in this work is the *min-max* function. Given a feature f, the *min-max* transformation function which is defined as follows:

$$\boldsymbol{f'} = \frac{\boldsymbol{f} - min(\boldsymbol{f})}{max(\boldsymbol{f}) - min(\boldsymbol{f})} \tag{5}$$

In the above formula $\boldsymbol{f}$ denotes the feature vector and $\boldsymbol{f'}$ is the transformed feature vector.

4 Fuzzy Clustering and Model Construction

In this section we illustrate the main idea behind our learning algorithm. We believe that different personality dimensions have a close relationship with the writing style of authors. In psychology, the *Big Five Personality Traits* are 5 dimensions of personality that are used to describe the personality of humans[3]. Openness, Conscientiousness, Extraversion, Agreeableness and Neuroticism are the personality dimensions which are described as the factors of the *Big Five* model. Based on these dimensions, each persons personality can be described using a combination of the above dimensions. Inspired by the *Big Five* model, we construct c clusters, where each cluster represents a personality dimension. An author's personality can then be determined by computing his or her membership to these clusters. Finally two authors that have the same (or similar) membership distribution over the clusters would be considered as the same.

For this we collect all the documents in our training set from which we know that they are written by the same author and extract their features (See Section 3). This will result in a matrix $Z = [z_1^{tr}, z_2^{tr}, \ldots, z_N^{tr}] \in \mathbb{R}^{4\times N}$ where N is the number of collected documents and z_i^{tr} denotes the transpose of the vector z_i. As already mentioned the personality of an author can be determined using his or her membership values to the available clusters. Due to this consideration, we use the *Fuzzy C-Means*[2] clustering algorithm to construct fuzzy clusters. For constructing c clusters, we assign initial cluster membership values for each document in the collection (The collection of these values constructs the partition matrix $U = [\mu_{ik}] \in \mathbb{R}^{c\times N}$). The partition matrix will be updated after each iteration of the algorithm until no significant changes are observable. After initializing the partition matrix randomly, the *Fuzzy C-Means* algorithm can be summarized as follows:

Repeat for $l = 1, 2, \ldots$

Step 1: Compute the cluster centers with $m \in [1, \infty)$

$$v_i^{(l)} = \frac{\sum_{k=1}^{N} (\mu_{ik}^{(l-1)})^m z_k}{\sum_{k=1}^{N} (\mu_{ik}^{(l-1)})^m}, \quad 1 \le i \le c \tag{6}$$

Step 2: Compute the distances

$$D_{ik}^2 = \left\| z_k - v_i^{(l)} \right\|^2 = (z_k - v_i^{(l)})^T (z_k - v_i^{(l)}), \quad 1 \le i \le c, \quad 1 \le k \le N \tag{7}$$

Step 2: Update the partition matrix:

for $1 \le k \le N$

if $D_{ik} > 0$ for all $i = 1, 2, \ldots, c$

$$\mu_{ik}^{(l)} = \frac{1}{\sum_{j=1}^{c} (D_{ik}/D_{jk})^{2/(m-1)}} \tag{8}$$

otherwise

$$\mu_{ik}^{(l)} = 0 \text{ if } D_{ik} > 0, \text{ and } \mu_{ik}^{(l)} \in [0,1] \text{ with } \sum_{i=1}^{c} \mu_{ik}^{(l)} = 1 \tag{9}$$

Until $\left| U^{(l)} - U^{(l-1)} \right| < \epsilon$

We use the cluster information produced by the cluster algorithm, to verify whether two documents are written by the same author or not. The process of *author verification* will be discussed in the following section.

5 Verification and Scoring

In order to find an answer to an *author verification* problem P, we compute the cluster membership values for documents with known authors and documents with unknown authors. Then using the membership values we will decide if the documents have the same author or not.

Given a problem $P = \{D = \{d_1, \ldots, d_n\}, d_u\}$ and c cluster prototypes (centroids) $V = \{v_1, \ldots, v_c\}$ we compute the membership values of the documents with known authors to the constructed clusters. In this way, for each document d_i, $1 \leq i \leq n$ a cluster membership vector $\mu_i = \{\mu_{i1}, \ldots, \mu_{ic}\}$ will be computed where the j-th element in the vectors represents the membership value of the document d_i to the cluster j.

In the same way we compute the cluster membership values of the document with unknown author d_u. This would result in the membership vector μ_u. At this step the cluster membership values for all documents in the problem P are known. Notice that the documents $d_1, \ldots, d_n$ are assumed to be written by the same author. Theoretically we would expect that the cluster membership vectors of these documents look very similar to each other. Experimental results show that this is usually not the case, which relies on the fact the authors write in different psychological states.

In order to solve the above problem, for the documents with known authors, we compute a mean cluster membership vector. Through this vector a more stable estimation of membership to available personality dimensions can be made. The mean cluster membership vector of a set of documents $d_1, \ldots, d_n$ with known authors can be computed as follows:

$$\tilde{\mu} = \frac{\sum_{i=1}^{n} \mu_i}{n} \tag{10}$$

Now using the cosine similarity between the average cluster membership vector of documents with known authors and the questioned document, the similarity between these two vectors can be computed. The cosine similarity between these two vectors is defined as follows[6]:

$$S_{\tilde{\mu},\mu_u} = \frac{\tilde{\mu} \cdot \mu_u}{||\tilde{\mu}||\, ||\mu_u||} \tag{11}$$

Through the cosine similarity measure we compute the angle between the vectors. A cosine values of 0 means that the vectors are orthogonal to each other and a cosine value of 1 means that the vectors are identical. Through the cosine similarity measure we assigned a score to each problem. Additionally we need a transformation function which can return binary values for author verification problem. In Section 2 we defined the function $\varphi(d_u, D)$. Here we modify this definition and redefine the function:

$$\varphi(d_u, D) = \begin{cases} 1, & \text{if } S_{\tilde{\mu},\mu_u} \geq 0.5 \\ 0, & \text{if otherwise} \end{cases} \tag{12}$$

Using the above function definition, for each problem P it can be decided if the documents inside P belong to the same author or not. A value of 1 means that the documents inside P have the same author and a value of 0 means that the questioned document has a different author than the documents with a known author.

6 Evaluation Results

In order to evaluate our approach we used the training set provided by the *PAN 2014 Author Identification* task. The training set consists of documents belonging to 4 different languages, namely Dutch, English, Greek and Spanish. Dutch documents are divided into essays and reviews, and English documents into essays and novels. Greek and Spanish documents belong only to the genre Articles. In total we constructed 6 models, where each model corresponds to a specific language and a specific genre.

For constructing the clusters of language L and genre G, we randomly selected 20% of the available training data to create the clusters. The experiments have been repeated 1000 times and average $c@1$ measure of the iterations has been computed. The $c@1$ measure of a single iteration can be computed as follows[8]:

$$c@1 = (\frac{1}{n})(n_c + (n_u \frac{n_c}{n})) \tag{13}$$

where, n = number of problems, n_c = number of correct answers and n_u = number of unanswered problems. The results are summarized in Table 1.

Table 1: Evaluation results for the training set

Language	Genre	#Clusters	m	c@1	AUC	c@1 · AUC
Dutch	Essays	4	4	0.731	0.752	0.549
Dutch	Reviews	3	4	0.680	0.763	0.518
English	Essays	3	3	0.664	0.651	0.432
English	Novels	4	3	0.852	0.852	0.725
Greek	Articles	4	3	0.671	0.697	0.467
Spanish	Articles	3	5	0.684	0.712	0.487

In Table 1 the number of created clusters and the parameter m are also specified. These parameters are the ones that returned the best results during our experiments. As we can see the algorithm returns the best results for the English novels with an $c@1$ value of 0.852. The worst results are also for the English documents but the ones in the genre essays. Even though the $c@1$ values for all languages and genres are greater than 0.66.

Beside the above approach, we evaluate the performance of our algorithm according to the area under the ROC curve (AUC)[4] of its returned probability scores. Table 1 summarizes the results. As it can be seen in the table, the AUC values are consistent and comparable with $c@1$ values. The reason for this is that the verification algorithm outputs very high probability scores for the positive cases, and very low probability scores for the negative cases.

For ranking the performance of participants in the competition a test corpus has been provided. We have evaluated our algorithm using *Tira*[5] which is a service for running experiments in computer science. Table 2 represents the performance results and also the run-time of our algorithm on the test corpus.

From the performance results based on the test corpus it can be seen that our algorithm performs very well for English *Novels*, and *Essays* reaching a final score of 0.508 and 0.349 respectively. But for the other languages the results are not as satisfactory as expected. This difference between the results indicates that for languages other than English, a deeper feature engineering is needed.

Table 2: Evaluation results for the test set

Language	Genre	c@1	AUC	c@1 · AUC	Runtime (in seconds)
Dutch	Essays	0.635	0.594	0.377	4
Dutch	Reviews	0.500	0.493	0.246	6
English	Essays	0.580	0.602	0.349	6
English	Novels	0.715	0.711	0.508	7
Greek	Articles	0.540	0.543	0.293	4
Spanish	Articles	0.650	0.640	0.416	7

The run-time of our algorithm on different data sets also shows that the introduced algorithm can be efficiently used for large collections of *author verification* problems. This is due to the small number of features that we extract from documents. This has from one side the advantage that the *author verification* problems can be solved very efficiently, but from the other side, it will result in a lower performance for specific languages.

7 Conclusion

In this work we have described our approach to solve the author verification problem introduced in the *PAN 2014 Author Identification task*. Using the *fuzzy c-means* clustering algorithm, we partitioned the provided training set (Section 4) into several clusters. Given an *author verification* problem, we used the membership values of the documents inside the problem to verify whether two documents have the same author or not.

In order to design an efficient algorithm we only considered a limited number of features for each language. This resulted in very low run-times for our algorithm. Accordingly we acquired the 1st place among the participants regarding the run-time of algorithms.

Our introduced approach also revealed sound results for the English language achieving the 1st place for English Novels and the 5th place for English Essays among the 13 participating teams. For other languages we did not get the expected satisfactory results.

The reason for this lies in the small amount of training set that we use for constructing our fuzzy clusters. We also use the same set of features for all available languages which is probably the main reason for insufficient results for languages other than English.

References

1. Argamon, S.: Scalability issues in authorship attribution.kim luyckx. LLC 27(1), 95–97 (2012)
2. Bezdek, J., Ehrlich, R., Full, W.: FCM: The fuzzy c-means clustering algorithm. Computers & Geosciences 10(2-3), 191–203 (1984)
3. Digman, J.M.: Personality Structure: Emergence of the Five-Factor Model. Annual Review of Psychology 41(1), 417–440 (1990)
4. Fawcett, T.: Roc graphs: Notes and practical considerations for researchers. ReCALL 31(HPL-2003-4), 1–38 (2004)
5. Gollub, T., Potthast, M., Beyer, A., Busse, M., Rangel, F., Rosso, P., Stamatatos, E., Stein, B.: Recent trends in digital text forensics and its evaluation. In: Forner, P., Müller, H., Paredes, R., Rosso, P., Stein, B. (eds.) Information Access Evaluation. Multilinguality, Multimodality, and Visualization, Lecture Notes in Computer Science, vol. 8138, pp. 282–302. Springer Berlin Heidelberg (2013)
6. Han, J.: Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2005)
7. Koppel, M., Schler, J.: Authorship verification as a one-class classification problem. In: Proceedings of the Twenty-first International Conference on Machine Learning. pp. 62–. ICML '04, ACM, New York, NY, USA (2004)
8. Peñas, A., Rodrigo, A.: A simple measure to assess non-response. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1. pp. 1415–1424. HLT '11, Association for Computational Linguistics, Stroudsburg, PA, USA (2011)
9. Stamatatos, E.: A survey of modern authorship attribution methods. J. Am. Soc. Inf. Sci. Technol. 60(3), 538–556 (Mar 2009)

7.2 Exploring the Effects of Cross-Genre Machine Learning for Author Profiling in PAN 2016

Pashutan Modaresi, Matthias Liebeck, and Stefan Conrad. Exploring the Effects of Cross-Genre Machine Learning for Author Profiling in PAN 2016. In *Working Notes of CLEF 2016 - Conference and Labs of the Evaluation Forum*, pages 970–977, 2016.
Contributions: The author contributed with the design of the architecture, evaluated the developed system and prepared the manuscript jointly with Matthias Liebeck.
Status: Published.

The aim of author profiling is to predict various profile dimensions of an author such as her or his age or gender [MLC16a]. From a marketing point of view, author profiling can be used to determine the demographics of the customers of a company to support strategic decisions. From a forensic viewpoint, it can be used to gain insight and information about the author of a suspicious text. Similar to author verification, in the context of automatic text summarization, author profiling can be used to check whether the original and the summary document have consistent writing styles [PRV$^+$16].

The problem of author profiling can be tackled either as a regression or classification task. For instance, in the case of age prediction, a classifier can be used to classify an input document to various age classes such as 18-25 or 25-34 [MLC16a]. There also exist methods that approach the task as a prediction task where given an input document, the age of author can be predicted in a continuous range [NSR11].

As in other areas of machine learning, having access to a large annotated data set with age and gender information of the author plays a significant role in designing robust and accurate author profiling algorithms. To tackle this problem, in the PAN 2016 [1] challenge, the task of cross-genre author profiling was introduced. The idea behind the cross-genre author profiling task was to use data sets from a genre where collecting age and gender information was easier to train the models and apply the trained models to other genres. More specifically, tweets were used to train the models and the participants were asked to apply the trained models to other genres such as blogs or social media [PRV$^+$16].

In this work, we introduce a machine learning approach based on logistic regression to construct two separate models to predict the age and the gender of the authors based on their documents. Our approach uses manual feature engineering to represent the documents by stylistic and lexical features. We obtained sound results and report them in terms of accuracy.

[1]http://pan.webis.de/clef16/pan16-web/author-profiling.html

Exploring the Effects of Cross-Genre Machine Learning for Author Profiling in PAN 2016

Notebook for PAN at CLEF 2016

Pashutan Modaresi, Matthias Liebeck, and Stefan Conrad

Institute of Computer Science
Heinrich Heine University Düsseldorf
D-40225 Düsseldorf, Germany
{modaresi,liebeck,conrad}@cs.uni-duesseldorf.de

Abstract Author profiling deals with the study of various profile dimensions of an author such as age and gender. This work describes our methodology proposed for the task of cross-genre author profiling at PAN 2016. We address gender and age prediction as a classification task and approach this problem by extracting stylistic and lexical features for training a logistic regression model. Furthermore, we report the effects of our cross-genre machine learning approach for the author profiling task. With our approach, we achieved the first place for gender detection in English and tied for second place in terms of joint accuracy. For Spanish, we tied for first place.

1 Introduction

Author profiling deals with the study of various profile dimensions of an author [2]. The focus of this study is to gain an understanding of how authors of different classes (e.g., old men and young women) use different characteristics while writing text and which textual features might be characteristic for all people in the same class. For instance, younger people might make more spelling mistakes than older people.

Due to its applications in fields such as security, forensics and marketing, the study of various profile aspects of an author has gained more importance in recent years. This in turn has attracted the attention of the scientific community [1]. More specifically, the PAN (Uncovering Plagiarism, Authorship, and Social Software Misuse) competition has been focusing on the task of author profiling as a part of the CLEF conference since 2013.

Author profiling is useful in a context where missing information about authors is relevant for an organization. For instance, a company might want to know how old their target group in social media is in order to customize advertising campaigns. In other contexts, such as *political online participation* [8], where cities allow their citizens to participate in politics via internet, it is interesting to automatically estimate demographic distributions of the users without the need to directly ask them for personalized data. Even in fields such as abstractive text summarization, author profiling techniques can be used to differentiate between human-written and machine-generated summaries [9].

Two profile aspects, namely *age* and *gender*, have been the focus of the PAN author profiling competitions. The focus of the 2016 shared task [15] is on cross-genre age and gender identification. That means that the training documents are on one genre (Twitter) and the evaluation is on another (unknown to the participant at the time of the software submission) genre, such as blogs or social media. English, Spanish and Dutch are the languages that were addressed in this year's challenge.

2 Related Work

Author profiling has been a recurrent PAN task since 2013. Until today, the age and gender classification tasks have always been part of the author profiling challenge. The first challenge in 2013 [14] was on English and Spanish blog posts. The focus in 2014 [13] was on four domains (*blogs*, *Twitter*, *social media*, and *hotel reviews*), each of which was provided with an individual training and test set. The 2015 challenge [12] was on English, Spanish, Italian, and Dutch tweets and provided an additional classification task of identifying personality traits (extroversion, emotional stability, agreeableness, conscientiousness, and openness to experience). The challenge in 2016 also consisted of English, Spanish, and Dutch tweets as training data but had the additional difficulty of a cross-genre evaluation dataset.

Since there have been 53 participating teams in the last 3 years, various approaches to author profiling have been tested. The teams have used different preprocessing steps, features and classifiers. [12] provides an overview of the approaches of the participating teams in the 2015 challenge: For preprocessing, steps such as removing HTML code, removing hashtags and URLs, lowercasing text, and stop word filtering have been used. Character n-grams, word n-grams, POS n-grams, punctuation signs, topic modeling with Latent Semantic Analysis (LDA), and Twitter-specific features, such as links, hashtags, and mentions, have frequently been used as features. The most frequently trained classifier is the support vector machine.

In our approach, we need to keep in mind that the evaluation is cross-genre. This means that we cannot use features that are specific for Twitter, such as hashtags. Furthermore, we have to take into account that tweets are limited to 140 characters in length and our evaluation genre may be comprised of longer text. Features based on absolute length, like word counts, were, therefore, not relevant in this year's challenge. As a result, all of our features are normalized to account for the domain change.

Furthermore, there has also been research into author profiling outside of PAN, for instance, to predict demographic information [16], such as annual income, having children, religious beliefs, and education levels from Twitter users.

3 Methodology

This section describes our approach to this year's PAN Author Profiling challenge. First, we outline preprocessing steps that we used to clean the data. Then, we describe the features that we used in our machine learning approach. Afterwards, we briefly explain why we chose logistic regression as our classifier. The dockerized source code of our profiler is available on GitHub[1].

3.1 Preprocessing

As the genre of the training and test sets are not the same, we processed the documents in the training set in such a way that most of the genre-specific information was eliminated. In this way, the risk of overfitting on genres other than *Twitter* was reduced. This was accomplished by a composition of multiple preprocessors that each map an input document d (string of characters) to a modified document d'. The individual preprocessors are defined as follows:

- $p_1(d)$: Returns a string in which all case-based characters have been lowercased.
- $p_2(d)$: Filters all occurrences of URLs in the string. This is an important step toward creating genre-neutral documents.
- $p_3(d)$: A mention is a tweet that contains another user's @username anywhere in the body of the tweet and does not occur in other genres. This function eliminates all mentions in the document.
- $p_4(d)$: Hashtags are used to categorize tweets. Although hashtags may contain important information about the profile of an author, obtaining a meaningful representation of it is not always trivial (e.g., #timetoact). This function eliminates all hashtags from the document.
- $p_5(d)$: A retweet is a re-posting of someone else's tweet. As this feature is also tweet-specific and may not generalize to other domains, we eliminate all retweets.
- $p_6(d)$: Although the training set is claimed to be only consisting of English, Spanish, and Dutch tweets, it also contains tweets in other languages such as Arabic and Persian. We consider these tweets as noise by eliminating all non-latin characters from the input document.
- $p_7(d)$: Specific lexical features such as unigrams and bigrams result in better accuracies when accents are removed from the input document. This is accomplished by means of this function.
- $p_8(d)$: Eliminates all non-alphabetic characters from an input document. This function is applied when dealing with token-based features.
- $p_9(d)$: Eliminates all stop-words from the document using a language-specific predefined list.

The composition of preprocessors is feature-specific, meaning that for each feature a distinct set of functions is composed in order to preprocess the document (A detailed description of features is provided in Section 3.2). Table 1 lists the preprocessors per feature (category).

3.2 Features

After preprocessing the tweets, we need to extract features for a vector representation. The challenge in our particular task is the need for features that are genre-independent.

[1] https://github.com/pan-webis-de

Table 1. Features and their corresponding preprocessors

Feature	Preprocessors
Unigrams	$(p_9 \circ p_8 \circ p_7 \circ p_6 \circ p_5 \circ p_4 \circ p_3 \circ p_2 \circ p_1)(d)$
Bigrams	$(p_8 \circ p_7 \circ p_6 \circ p_5 \circ p_4 \circ p_3 \circ p_2 \circ p_1)(d)$
Average Spelling Error	—-
Character N-Grams	$(p_2 \circ p_3 \circ p_4 \circ p_1)(d)$
Punctuation Features	—-

Even though we use tweets from Twitter for training our classifier, we cannot use features that are specific for Twitter because the evaluation dataset is from another domain. Features that depend on absolute text length are not good candidates because tweets are limited to 140 characters, whereas the text length in the evaluation domain is probably unrestricted.

Since the tweets are in three different languages, we can either find language-specific features or language-independent features. Given that we are not familiar with all languages in this task, we decided to find language-independent features.

We experimented with multiple features in the course of our experiments. In the end, we decided to use the combination of features that worked best on the *Blog* dataset from 2014 as test set (with the dataset from 2016 as training set):

– **Word unigrams** that occur at least two times
– **Word bigrams**
– **Character 4-grams** within word boundaries
– We utilize Hunspell[2] with LibreOffice dictionaries for all three languages to measure an **average spelling error** by determining a relative value for correctly spelled words.
– In addition, we make use of four token-based **punctuation features**: average comma count, average dot count, average exclamation count, and average question mark count.

All these features were used for the *age* subtask. We omitted the punctuation features for the *gender* subtask.

We also experimented with punctuation n-grams and used *polyglot*[3] to retrieve L_2-normalized POS-Tag distributions of UTS tags [11]. Unfortunately, both attempts only worsened our results and we did not pursue them further.

3.3 Classification

We used logistic regression [10] to train our final models. Logistic regression belongs to a family of classifiers that have high bias and low variance. Although this classifier has a low variance (which could lead to underfitting), due to the cross-genre nature of the problem, this will not have negative implications as the test dataset does not consist of tweets. On the other hand, logistic regression has a high bias which can lead to overfitting. This could also be handled using regularization techniques. These

[2] http://hunspell.github.io/
[3] http://polyglot-nlp.com/

properties make the logistic regression classifier a suitable choice for our cross-genre classification problem.

As the classification task is a multiclass problem, we use the one-vs-rest scheme for logistic regression. Moreover, we set $C = 10^{-3}$ as the regularization strength.

Additionally, we also experimented with random forest [3] and gradient boosting [5]. Both of these techniques use randomization to build decision trees (or regression trees) to combat overfitting. The results obtained using logistic regression were superior to both above-mentioned classifiers.

4 Evaluation

As the focus of this year's competition is cross-genre author profiling, we only used the tweets dataset provided by the organizers to train our models. For *age*, the following classes are provided: 18-24, 25-34, 35-49, 50-64 and 65-xx. Moreover, *gender* consists of the two classes: *male* and *female*.

The provided dataset is in the three languages English, Spanish and Dutch. The corpus was annotated with the age and gender information of the authors, except for Dutch, which was only annotated with gender information. For each individual author, there exists an XML document consisting of several tweets. For English there are 436 documents, for Spanish 250 documents, and for Dutch 384 documents. In our approach, we concatenated all tweets of an author into a single document.

In order to evaluate our models, we used stratified k-fold cross validation $(k = 10)$ on the tweets dataset. For this we used the implementation provided by *scikit* [4]. Furthermore, we used the available training datasets from PAN2014 to evaluate our models on genres other than Twitter (blogs, social media, reviews). This was accomplished by using the TIRA experimentation platform, which provides a service to handle software submissions [6, 7].

4.1 Official PAN 2016 Benchmark

For each language (except Dutch, which contained only the age annotation), two distinct models were trained: one for gender and one for age. For both labels we used the same set of features, except for punctuation features that were only used for age.

For the final evaluation, two test datasets were provided by the task organizers, where the first dataset is a subset of the second one. The official results for the first and second test datasets are reported in Table 2 and Table 3 accordingly.

Table 2. Evaluation results in terms of accuracy for the first test dataset

Language	Joint	Gender	Age
English	0.1552	0.5029	0.3017
Spanish	0.2031	0.6406	0.2813
Dutch	0.5	0.5	—-

Table 3. Evaluation results in terms of accuracy for the second test dataset

Language	Joint	Gender	Age
English	0.3846	0.7564	0.5128
Spanish	0.4286	0.6964	0.5179
Dutch	0.5040	0.5040	—-

In general, higher accuracies are achieved on the second dataset. On the second test dataset, for both English and Spanish, the accuracies for age prediction are slightly above 0.5. The highest accuracy (0.7564) is achieved by gender classification for the English language. Outstanding is the joint accuracy of 0.4286 for the Spanish language. Also, the lowest accuracies are reported for Dutch. Unfortunately, at the time of authoring this work, no access to the test datasets was granted to explain this behavior. As all features used for gender classification are token centric, we assume that the out-of-vocabulary rate is too high in the prediction phase and this leads to the unsatisfiable results for Dutch.

4.2 Cross-Genre Effects

One of the main intentions behind using simple features in our approach is to avoid overfitting on genres other than Twitter. We also performed tests on the training datasets from PAN 2014 which are publicly available. The training dataset of PAN 2014 is also annotated with age and gender information and both labels have exactly the same categories as in PAN 2016. In comparison with PAN 2016, the PAN 2014 corpus only contains English and Spanish documents belonging to four different genres, namely blogs, Twitter, social media and hotel reviews. The accuracies of our model on blogs and Twitter are reported in Table 4 and Table 5 respectively.

Table 4. Evaluation results in terms of accuracy for PAN2014 training dataset (Blogs)

Language	Joint	Gender	Age
English	0.3878	0.8435	0.4830
Spanish	0.4091	0.7727	0.4773

Table 5. Evaluation results in terms of accuracy for PAN2014 training dataset (Twitter)

Language	Joint	Gender	Age
English	0.9510	0.9804	0.9542
Spanish	0.4270	0.7640	0.5281

Among all genres, the highest joint accuracies are achieved for blogs with 0.3878 for English and 0.4091 for Spanish. These values are even higher than the ones obtained during k-fold cross validation on PAN 2016 tweet dataset and signal that no overfitting occurred in case of blogs. It can also be observed that the accuracies for English tweets are extremely high with a score above 0.9. This is most probably due to the high overlap between the datasets from 2014 and 2016.

Table 6. Evaluation results in terms of accuracy for PAN2014 training dataset (Socialmedia)

Language	Joint	Gender	Age
English	0.2000	0.6000	0.2000
Spanish	0.1627	0.5951	0.2563

Table 7. Evaluation results in terms of accuracy for PAN2014 training dataset (Reviews)

Language	Joint	Gender	Age
English	0.1524	0.6067	0.2572
Spanish	—-	—-	—-

Unlike in the case of blogs, the accuracies for the genres social media and reviews are not satisfactory (see Table 6 and Table 7). The lengths of the documents in social media and reviews are much greater than the length of the documents in Twitter. This leads to a high out-of-vocabulary rate and consequently to unsatisfactory results.

Table 8. Confusion matrix for PAN2014 (English/Blogs/Age)

		Predicted					
		18-24	25-34	35-49	50-64	65-xx	$\sum$
Actual	18-24	0	3	3	0	0	6
	25-34	0	28	32	0	0	60
	35-49	0	9	45	0	0	54
	50-64	0	1	20	2	0	23
	65-xx	0	0	3	1	0	4
	$\sum$	0	41	103	3	0	147

As the measure of accuracy is not suitable to study the performance of our approach for each individual category, we also exemplarily provide the confusion matrix for a model trained on PAN 2016 tweets and tested on PAN 2014 English blogs on the category age (see Table 8). The confusion matrix shows that no instance from the categories 18-24 and 65-xx is correctly classified. The reason for this is the low support of these categories, which implies that the classifier has not enough data to learn from. Another interesting point is the high similarity between the categories 25-34 and 35-49. From 60 instances in the category 25-34, 32 instances are incorrectly classified to the class 35-49. This indicates that the features defined in our approach are not capable of discriminating between the aforementioned categories.

5 Conclusion and Future Work

We have presented our approach for the cross-genre PAN 2016 author profiling task. Our best results for the gender and age classification tasks in terms of accuracy are 0.7564 for English and 0.5179 for Spanish, respectively. Furthermore, we evaluated our approach on multiple genres to explore the effects of cross-genre machine learning.

The training set for age was imbalanced (see the confusion matrix in Table 8), which resulted in poor performance. We could use techniques such as sampling or SMOTE to tackle this problem.

In our experiments, we tested different feature combinations. It turned out to be difficult to find good genre- and language-independent features. For instance, the POS distribution turned out not to be a good genre-independent feature. In our future work, we will include more language-dependent features to better capture the characteristics of each language. Additionally, we will include lists of sentiment-bearing words in our features.

Acknowledgments

This work was partially funded by the PhD program *Online Participation*, supported by the North Rhine-Westphalian funding scheme *Fortschrittskollegs* and by the German

Federal Ministry of Economics and Technology under the ZIM program (Grant No. KF2846504).

References

1. Álvarez-Carmona, M.Á., López-Monroy, A.P., Montes-y-Gómez, M., Villaseñor-Pineda, L., Escalante, H.J.: INAOE's Participation at PAN'15: Author Profiling task. In: CLEF (Working Notes). CEUR Workshop Proceedings, vol. 1391. CEUR-WS.org (2015)
2. Argamon, S., Koppel, M., Pennebaker, J.W., Schler, J.: Automatically Profiling the Author of an Anonymous Text. Commun. ACM 52(2), 119–123 (2009)
3. Breiman, L.: Random Forests. Mach. Learn. 45(1), 5–32 (2001)
4. Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., Varoquaux, G.: API design for machine learning software: experiences from the scikit-learn project. In: ECML PKDD Workshop: Languages for Data Mining and Machine Learning. pp. 108–122 (2013)
5. Friedman, J.H.: Greedy Function Approximation: A Gradient Boosting Machine. Annals of Statistics 29, 1189–1232 (2000)
6. Gollub, T., Stein, B., Burrows, S.: Ousting Ivory Tower Research: Towards a Web Framework for Providing Experiments as a Service. In: 35th International ACM Conference on Research and Development in Information Retrieval (SIGIR 12). pp. 1125–1126. ACM (2012)
7. Gollub, T., Stein, B., Burrows, S., Hoppe, D.: TIRA: Configuring, Executing, and Disseminating Information Retrieval Experiments. In: 9th International Workshop on Text-based Information Retrieval (TIR 12) at DEXA. pp. 151–155. IEEE (2012)
8. Liebeck, M., Esau, K., Conrad, S.: What to Do with an Airport? Mining Arguments in the German Online Participation Project Tempelhofer Feld. In: Proceedings of the 3rd Workshop on Argument Mining. p. (in press). Association for Computational Linguistics (2016)
9. Modaresi, P., Conrad, S.: On definition of automatic text summarization. In: Proceedings of Second International Conference on Digital Information Processing, Data Mining, and Wireless Communications (DIPDMWC2015). pp. 33–40 (2015)
10. Nelder, J.A., Wedderburn, R.W.M.: Generalized linear models. Journal of the Royal Statistical Society, Series A, General 135, 370–384 (1972)
11. Petrov, S., Das, D., McDonald, R.: A universal part-of-speech tagset. In: Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12). European Language Resources Association (ELRA) (2012)
12. Rangel, F., Celli, F., Rosso, P., Potthast, M., Stein, B., Daelemans, W.: Overview of the 3rd Author Profiling Task at PAN 2015. In: Working Notes Papers of the CLEF 2015 Evaluation Labs. CLEF (2015)
13. Rangel, F., Rosso, P., Chugur, I., Potthast, M., Trenkmann, M., Stein, B., Verhoeven, B., Daelemans, W.: Overview of the 2nd Author Profiling Task at PAN 2014. In: Working Notes for CLEF 2014 Conference. pp. 898–927. CLEF (2014)
14. Rangel, F., Rosso, P., Koppel, M., Stamatatos, E., Inches, G.: Overview of the Author Profiling Task at PAN 2013. In: Working Notes for CLEF 2013 Conference. CLEF (2013)
15. Rangel, F., Rosso, P., Verhoeven, B., Daelemans, W., Potthast, M., Stein, B.: Overview of the 4th Author Profiling Task at PAN 2016: Cross-genre Evaluations. In: Working Notes Papers of the CLEF 2016 Evaluation Labs. CEUR Workshop Proceedings, CLEF and CEUR-WS.org (Sep 2016)
16. Volkova, S., Bachrach, Y.: On Predicting Sociodemographic Traits and Emotions from Communications in Social Networks and Their Implications to Online Self-Disclosure. Cyberpsychology, Behavior, and Social Networking 18(12), 726–736 (2015)

7.3 Plagiarism Alignment Detection by Merging Context Seeds

Philipp Gross and Pashutan Modaresi. Plagiarism Alignment Detection by Merging Context Seeds. In *Working Notes for CLEF 2014 Conference*, pages 966–972, 2014. **Contributions**: The author contributed with the evaluation and error analysis of the developed algorithm and prepared the manuscript. **Status**: Published.

Plagiarism detection can be categorized into intrinsic and extrinsic approaches. In extrinsic approaches, the suspicious document is compared to a collection of documents to check the existence of plagiarized passages in the suspicious document. In intrinsic approaches, the identification of plagiarized passages is solely made based on the variations in the writing styles of the suspicious document. In this work, we focus on extrinsic approaches and, more specifically, on the task of text alignment [SLP11].

Given a pair of documents, text alignment aims to identify all contiguous passages of reused text between them [PHB$^+$14]. Text alignment is a challenging task, as in many cases, the text reuse is not only limited to the lexical level. This means that the original and the plagiarized document have little lexical overlap, and techniques such as paraphrasing or translation might have been used to plagiarize.

In the context of automatic text summarization, text alignment can be used to measure the degree of contextual similarity between the original and summary document. In such a scenario, high plagiarism scores are an indicator that the original and summary document reflect the same kind of information.

One of the most popular approaches to tackle the text alignment problem is the *seeding* approach. Given a suspicious document and a source document, matches are identified between the documents using some heuristics [PHB$^+$14]. For instance, Alvi et al. [ASC14] use character 20-grams as heuristic to identify matches between suspicious and source documents. Sanchez-Perez et al. [SPSG14] compute a vector representation for the sentences using the *term frequency - inverse document frequency* weighting and find the matches based on the cosine similarities of the sentences.

In this work we introduce an automatic text alignment approach where we use skip word 2-grams with skips ranging from 1 to 4 and exact matching. Our approach consists of three steps. First, we generate the matches based on a similarity measure. Second, we merge matches that are close to each other to form longer plagiarism sequences, and finally, in a post-processing step, we filter outliers. We achieved sound results and report the performance of our system in terms of precision and recall measures.

Plagiarism Alignment Detection by Merging Context Seeds

Notebook for PAN at CLEF 2014

Philipp Gross[1] and Pashutan Modaresi[2,3]

[1] pressrelations GmbH, Düsseldorf, Germany
{philipp.gross, pashutan.modaresi}@pressrelations.de
[2] Heinrich-Heine-University of Düsseldorf, Institute of Computer Science, Düsseldorf, Germany
modaresi@cs.uni-duesseldorf.de

Abstract We describe our submitted algorithm to the text alignment sub-task of the plagiarism detection task in the PAN2014 challenge that achieved a plagdet score 0.855. By extracting contextual features for each document character and grouping those that are relevant for a given pair of documents, we generate seeds of atomic plagiarism cases. These are then merged by an agglomerative single-linkage strategy using a defined distance measure.

1 Introduction

Given a pair of text documents, the problem of text alignment is the task of identifying all pairs of contiguous passages that are equal up to obfuscation. The various strategies of the latter pose a challenge for this task. They reach from randomized simple operations like word or sentence permutations to more sophisticated transformations like semantic word variation or translation cycles, or even manual paraphrasing.

The problem of text alignment is a sub-task of the *PAN 2014 Plagiarism Detection* task. It also contains the sub-task *source retrieval* that concerns he retrieval of source documents when a suspicious document is given. Our submission only deals with the problem of text alignment. We follow the common strategy as explained in [2]:

1. *Seed generation*: Given a suspicious document X and a source document Y select a set of likely plagiarism cases, which are pairs of small passages in X and Y that are very similar by a defined measure.
2. *Merging*: Merge two plagiarism cases whose passages in X and Y are close to each other. Repeat this step until there are no adjacent plagiarism cases left.
3. *Extraction and filtering*: Postprocess the remaining plagiarism cases, filter outliers, and generate output plagiarism cases.

In the following we describe our submitted algorithm in detail and give the evaluation results for various corpora.

2 Problem Statement

In this section we formally define the problem of *text alignment* as part of *PAN 2014 Plagiarism Detection* task. The precise objective of this task is, for a given set of plagiarism cases S, to find a set of detections R such that the score $\mathrm{plagdet}(S, R)$ is high.

We follow a slightly different terminology compared to [3] and recall all definitions for the readers convenience.

A *document* of length n_X is a finite totally ordered set $X = \{x_i \colon i = 0, \dots, n_X\}$ of (positioned) characters $x_i = (c, i)$, $c \in \mathcal{C}$, where $\mathcal{C}$ denotes some finite set of symbols. A *passage* $P \subseteq X$ is a connected subset, thus either empty or of the form

$$P \;=\; \{\, x_i \;:\; 0 \;\le\; a \;\le\; i \;<\; b \;\le\; n\}. \tag{1}$$

For brevity we use the notion of closed intervals and define $[x_a, x_b] = P$ for non-empty passages.

Given a pair of documents (X, Y) we define a *passage reference* r as a rectangular subset in the Cartesian product set $r \subseteq X \times Y$. Every non-empty passage reference is always of the form

$$r = [x_a, x_b] \times [y_c, y_d] = \{(x_i, y_j) \colon 1 \le a \le i < b \le n_X, 1 \le c \le j < d \le n_Y\} \tag{2}$$

for passages $[x_a, x_b] \subseteq X$ and $[y_c, y_d] \subseteq Y$. Each pair $(x, y) \in X \times Y$ gives rise to a passage reference by taking the singleton set $\{(x, y)\}$, called *seed*. It is a minimal passage reference. Note that every non-empty passage reference is the linear span of finitely many seeds.

We say that a passage reference r *detects* another passage reference s if both belong to the same product space $X \times Y$ and have non-empty intersection $r \cap s$. The latter is also a passage reference. By embedding a document into the disjoint union of all documents, the definition $r \cap s$ extends naturally to passage references with different pairs of parent documents (namely, by the empty intersection).

We define the *perimeter* of a passage reference r as

$$\pi(r) = 2(b - a) + 2(d - c) \text{ if } r = [x_a, x_b] \times [y_c, y_d], \quad \text{and} \quad \pi(\emptyset) = 0. \tag{3}$$

The union of passage references is in general not a passage reference. But the perimeter extends in a natural way for such sets by taking the (one-dimensional) volume of the boundary. The upshot of the perimeter is that a passage reference r detects another passage reference s if and only if $\pi(r \cap s) > 0$.

A *set (or corpus) of plagiarism cases* S is just a set of passage references for varying document pairs (X, Y), $X, Y \in \mathcal{D}$ and some set of documents $\mathcal{D}$.

The quality of a set of detections R is evaluated by the numerical $\mathrm{plagdet}$ score. It is a composition of the micro precision, micro recall and granularity. For the sake of completeness, we recall their definitions. With $S \cdot R = \{s \cap r \mid s \in S, r \in R\}$ let

$$\mathrm{prec}(S, R) = \frac{\pi(S \cdot R)}{\pi(R)}, \qquad \mathrm{rec}(S, R) = \frac{\pi(S \cdot R)}{\pi(S)}. \tag{4}$$

Precision and recall give rise to the classical F_1-measure, i.e. the harmonic mean of precision and recall. In order to penalize fragmented passage references one weights the F_1 measure with the granularity,

$$\mathrm{gran}(S,R) = \frac{\sum_{s \in S_R} |R_s|}{|S_R|} \in [1, |R|], \tag{5}$$

where $S_R = \{s \mid s \in S, \exists r \in R \text{ detects } s\}$ and $R_s = \{r \mid r \in R, r \text{ detects } s\}$.
Then the $\mathrm{plagdet}$-score is defined as the weighted F_1-measure:

$$\mathrm{plagdet}(S,R) = \frac{F_1(\mathrm{prec}(S,R), \mathrm{rec}(S,R))}{\log_2(1 + \mathrm{gran}(S,R))} \in [0,1]. \tag{6}$$

3 Feature Extraction and Seed Generation

In this section we describe our approach to extract seeds of passages from $X \times Y$ for some pair of documents X and Y. We decided to apply feature extraction on a per document basis. Hence, this step can be done in a preprocessing phase before actually considering pairs of documents.

3.1 Extraction of contextual features for documents

Throughout the paper, F denotes the set of all features. As seen below, we use skip word ngrams as features, but for the sake of clarity we keep it general first.

Given a document $X = \{x_i\}$ we map each character to a finite set of binary features $x_i \mapsto \varphi(x_i) = \{f_{i1}, \ldots, f_{id}\}$, $d = d(i)$, $f_{ij} \in F$. Recall that the power set $\mathcal{P}(F)$ is the set of all subsets of F. Therefore, we defined a *feature map*

$$f_X \colon X \to \mathcal{P}(F). \tag{7}$$

In return, by mapping $F \ni f \mapsto \{x_i \mid f \in \varphi(x_i)\} \in \mathcal{P}(X)$ we get an *index map*

$$\iota_X \colon F \to \mathcal{P}(X). \tag{8}$$

It tells us at which character positions in the document a feature f is present.

In our approach we first tokenized the document X into a sequence of lowercased stemmed words $w_0, w_1, \ldots$ by omitting whitespaces, non-alphanumeric characters and stop words, and used skip-bigrams of length 1 to 4:

$$\varphi(x_i) = \begin{cases} \{w_\beta w_\alpha\}_{\beta = \alpha-4,\ldots,\alpha} & \text{if } x_i \text{ is the beginning character of a word } w_\alpha. \\ \emptyset & \text{otherwise.} \end{cases} \tag{9}$$

Table 1 illustrates the feature extraction for the example phrase *The quick brown fox jumps over the lazy dog*.

Table 1. Features extracted for the characters $x_0, \dots x_{43}$ of the example string *The quick brown fox jumps over the lazy dog*. Characters that were not at the beginning of a contiguous alphanumeric substring have no features and are omitted from the table. The symbol $\star$ is a placeholder for the empty word.

Offset	Token	f_1	f_2	f_3	f_4
4	quick	⋆_quick	⋆_quick	⋆_quick	⋆_quick
10	brown	quick_brown	⋆_brown	⋆_brown	⋆_brown
16	fox	brown_fox	quick_fox	⋆_fox	⋆_fox
20	jump	fox_jump	brown_jump	quick_jump	⋆_jump
35	lazy	jump_lazy	fox_lazy	brown_lazy	quick_lazy
40	dog	lazy_dog	jump_dog	fox_dog	brown_dog

3.2 Selection of relevant features

Clearly, features which are not present at all, or appear at almost every character are useless for the text alignment task. In order to reduce the number of generated features, we apply a simple feature selection strategy. If $\iota_X(f)$ is non-empty and has low cardinality, then we consider f as meaningful. We say that $f \in F$ is a *relevant feature (for X)*, if the cardinality satisfies the following estimation:

$$1 \leq |\iota_X(f)| \leq \varrho \tag{10}$$

for some given threshold parameter ϱ. The latter is also called *relevance threshold*. The subset of all relevant features is denoted as $F_X \subseteq F$.

In our approach we use a constant relevance threshold $\varrho = 4$. For time constraints we kept this simple approach because it already worked quite well.

3.3 Feature extraction of document pairs and seed generation

The feature extraction for documents carries over to feature extraction to pairs of documents. Given a suspicious document X and a source document Y their index maps give rise to a natural index map of $X \times Y$:

$$\iota_{XY} : F \to \mathcal{P}(X \times Y), \quad f \mapsto \{(x_i, y_j) \mid f \in \varphi(x_i) \text{ and } f \in \varphi(y_i)\}. \tag{11}$$

It maps a feature f to the set of all pairs (x_i, y_j) of characters such that f is simultaneously present at x_i and y_i. Now let $F_X \cap F_Y \subseteq F$ be the subset of features which are relevant for X and Y. The union

$$\sigma(X,Y) = \bigcup_{f \in F_X \cap F_Y} \iota_{XY}(f) \subseteq X \times Y \tag{12}$$

is the *seed set* of plagiarism cases between X and Y. These atomic plagiarism cases are the starting point of a merge process, which is explained in the next section.

We can deduce an estimation of the cardinality in terms of the relevance threshold. Namely, for each $f \in F_X \cap F_Y$ holds the estimation

$$1 \leq |\iota_{XY}(f)| = |\iota_X(f)| \cdot |\iota_Y(f)| \leq \varrho^2. \tag{13}$$

Hence, $|\sigma(X,Y)| \leq \varrho^2 |F_X||F_Y|$. Consequently, the number of seeds can be estimated just in terms of X and Y without having to inspect the pair (X,Y).

4 Merging

In this section we describe the process of merging passage references in the product space $X \times Y$ for two documents X and Y. The merge criterion will be given in terms of a distance function and the merge process is then the agglomerative single-linkage clustering with an additional termination condition.

4.1 Merge criteria

In order to define a distance between two passage references, let us first introduce further notation.

For two non-empty passages $P_1 = [x_{a_1}, x_{b_1}]$ and $P_2 = [x_{a_2}, x_{b_2}]$ in X let their *distance* be $\operatorname{dist}(P_1,P_2) = \min\{|u_1 - u_2| \colon a_1 \leq u_1 \leq b_1, a_2 \leq u_2 \leq b_2\}$. It is positive if and only if P_1 and P_2 are disjoint. Now, for two non-empty passage references $P_1 \times Q_1, P_2 \times Q_2 \subseteq X \times Y$ their *distance* is

$$\operatorname{dist}(P_1 \times Q_1, P_2 \times Q_2) = \frac{2\operatorname{dist}(P_1,P_2) + 2\operatorname{dist}(Q_1,Q_2)}{\sigma + \pi(P_1 \times Q_1) + \pi(P_2 \times Q_2)}, \tag{14}$$

where $\sigma > 0$ denotes a constant smoothing parameter that is defined empirically. The distance is zero if and only if $P_1 \cap P_2 \neq \emptyset$ and $Q_1 \cap Q_2 \neq \emptyset$, or equivalently $P_1 \times Q_1 \cap P_2 \times Q_2 \neq \emptyset$, and thus reflects the fact that two passages are directly adjacent. If the distance is positive, but lesser than a given threshold τ, the passages have empty intersection but are relatively close.

The *merge* of two passage references in $X \times Y$ is the smallest passage reference that contains both. It always contains their union but is in general strictly larger.

4.2 Agglomerative clustering

Having defined criteria for merging two passage references, we apply agglomerative single-linkage clustering. That is, in each step we merge a pair of passage references that have minimal distance. If there is no pair whose distance is lesser or equal than a given constant $\tau > 0$, the process terminates.

5 Filtering and passage extraction

At the end of the merge process we remove all passage references where the suspicious passage has less than 15 words. The remaining passage references are the detected plagiarism cases.

6 Evaluation Results

We evaluated the algorithm on the data sets of the previous competition PAN2013 [3] and the current competition PAN2014 using *Tira*[1]. For completeness, we also state the runtime, although it does not affect the ranking of the algorithm in the competition. See Table 2 for the development results and Table 3 for the end results.

We also ran smaller experiments restricted to a fixed obfuscation strategy (Table 4). To no surprise the algorithm underperforms for summary obfuscated plagiarism cases because we use no synonym dictionaries.

Table 2. Text alignment results with retrieval performance and runtime for the data sets of PAN2013. The experiments were executed on an 8-core system.

Corpus	Pairs	PlagDet	Precision	Recall	Granularity	Runtime
pan13-training-corpus	4007	**0.825**	0.935	0.743	1.00485	48s
pan13-test-corpus1	399	**0.837**	0.930	0.760	1.00000	5s
pan13-test-corpus2	4042	**0.831**	0.939	0.750	1.00421	41s

Table 3. Text alignment results with retrieval performance and runtime for the data sets of PAN2014. The experiments were executed on an 1-core system, provided by the host.

Corpus	Pairs	PlagDet	Precision	Recall	Granularity	Runtime
pan14-training-corpus	5164	**0.821**	0.928	0.763	1.02805	183s
pan14-test-corpus2	?	**0.826**	0.933	0.766	1.02514	180s
pan14-test-corpus3	?	**0.855**	0.925	0.818	1.02187	169s

Table 4. Text alignment *plagdet* scores with respect to obfuscation strategies

Corpus	None	Random	Cyclic translation	Summary
training-corpus-2013-01-21	0.903	0.812	0.824	0.299
test-corpus1-2013-03-08	0.901	0.791	0.854	0.412
test-corpus2-2013-01-21	0.913	0.811	0.835	0.316

7 Conclusion

The simple heuristics we used in our approach, already worked quite well and comparable to the state of the art text alignment algorithms for random and translation cycle obfuscations. In order to tackle the problem of summary obfuscation in the future, we intend to incorporate more semantic knowledge into the feature extraction stage.

References

1. Gollub, T., Potthast, M., Beyer, A., Busse, M., Rangel, F., Rosso, P., Stamatatos, E., Stein, B.: Recent trends in digital text forensics and its evaluation. In: Forner, P., MÃijller, H., Paredes, R., Rosso, P., Stein, B. (eds.) Information Access Evaluation. Multilinguality, Multimodality, and Visualization, Lecture Notes in Computer Science, vol. 8138, pp. 282–302. Springer Berlin Heidelberg (2013)
2. Potthast, M., Hagen, M., Gollub, T., Tippmann, M., Kiesel, J., Rosso, P., Stamatatos, E., Stein, B.: Overview of the 5th international competition on plagiarism detection. In: CLEF 2013 Evaluation Labs and Workshop - Working Notes Papers (2013)
3. Potthast, M., Stein, B., Barrón-Cedeño, A., Rosso, P.: An evaluation framework for plagiarism detection. In: Proceedings of the 23rd International Conference on Computational Linguistics: Posters. pp. 997–1005. COLING '10, Association for Computational Linguistics, Stroudsburg, PA, USA (2010), http://dl.acm.org/citation.cfm?id=1944566.1944681

8 Conclusion and Future Work

In this work, we focused on the problem of automatic single document summarization in the news domain. We followed a bottom-up approach to tackle the summarization problem and introduced the essential components or building blocks of an automatic summarization system.

We started our work with an overview of the existing approaches in the field of automatic single document summarization. We introduced classical approaches and mentioned modern deep learning methods that have been developed in recent years (Section 2.1). To gain a better understanding of the problem we reviewed the existing definitions of automatic text summarization and proposed our formal definition of the problem (Section 2.2).

We claimed that the automatic acquisition of data for training machine learning algorithms is a critical step in the development of summarization algorithms. To achieve this goal, in Chapter 3, we introduced two methods to create corpora for training machine learning algorithms automatically. We made the constructed corpora publicly available to the research community and used them to train our learning algorithms.

In Chapter 4, we focused on the problem of automatic keyphrase extraction from textual documents. Different from previous approaches, we defined the set of phrases in a document as a fuzzy set and determined the importance of the phrases using various membership functions. Moreover, we proposed an evaluation method inspired by the Turing test and showed that the evaluators could barely distinguish whether it was humans or the machines that extracted the keyphrases outputted by our algorithm.

Next, we focused on the problem of extractive text summarization and tackled this problem from a different point of view. Instead of extracting the most important content of a text we proposed to eliminate the least important (or redundant) content of the text. For this, we proposed a paraphrase detection algorithm to detect the sentences that semantically have the same meaning (Section 5.1). Moreover, we proposed a neural sentence ordering approach to determine the correct order of the extracted sentences (Section 5.2).

Evaluation of the summarization systems was the focus of Chapter 6. We proposed automatic and manual evaluation methods that assess the quality of the summaries. Moreover, we studied the commercial benefits of the summarization as part of an extensive study in the field of media monitoring and media response analysis. Our study showed that even simple summarization systems could reduce the costs of the company dramatically.

Finally, in Chapter 7, we studied the relationship between the fields of automatic text summarizaton and digital text forensics. We claimed that the summaries should have the following two additional properties: the summary and the original document should obey the same writing style, and the summary document should reflect the same information as in the original document. To assure the first property, we proposed an author verification and an author profiling approach, and for the second property, we proposed a plagiarism alignment detection method.

The followings recommendations are made for future research directions:

- Section 3.2: The Simurg corpus construction tool uses hard-coded CSS selector to select the news hyperlinks. As the layout of the websites changes constantly, the defined selectors have to be modified accordingly. This problem can be solved by the integration of a link detection algorithm.

- Section 4.1: The Turing test inspired evaluation method could be compared to the standard evaluation methods such as ROUGE or BLUE to determine their correlation.

- Section 5.2: The neural sentence ordering approach determines the order of only two sentences. Determining the correct order of more than two sentences requires a search algorithm such as the beam search [CQH16]. To avoid a search algorithm, the neural architecture could be extended to accept more than two input sentences.

- Section 6.2: The commercial benefits of the summarization systems were only studied for a simple summarization technique in the domain of media monitoring and analysis. An extensive study is still required to cover multiple summarization approaches and other domains.

In this work, we proposed several components for an automatic summarization system. For future work, we plan to integrate these components into a single summarization system.

9 Publications

2017

1. Pashutan Modaresi, Philipp Gross, Siavash Sefidrodi, Mirja Eckhoff and Stefan Conrad. On (Commercial) Benefits of Automatic Text Summarization Systems in the News Domain: A Case of Media Monitoring and Media Response Analysis. **Contributions**: The author contributed with the conduction of the experiment and prepared the manuscript. **Status**: Submitted to EACL2017.

2016

2. Pashutan Modaresi and Stefan Conrad. Simurg: An Extendable Multilingual Corpus for Abstractive Single Document Summarization. In *Proceedings of the 8th Annual Meeting of the Forum on Information Retrieval Evaluation*, FIRE '16, pages 24–27. ACM, 2016.
   Contributions: The research and the preparation of this manuscript was done entirely by the main author under the supervision of Prof. Conrad. **Status**: Published.

3. Pashutan Modaresi, Matthias Liebeck, and Stefan Conrad. Neural Classification of Linguistic Coherence Using Long Short-Term Memories. In *Proceedings of the 8th Annual Meeting of the Forum on Information Retrieval Evaluation*, FIRE '16, pages 28–31. ACM, 2016.
   Contributions: The research and the preparation of this manuscript was done jointly by Pashutan Modaresi and Matthias Liebeck under the supervision of Prof. Conrad. **Status**: Published.

4. Matthias Liebeck, Pashutan Modaresi, Alexander Askinadze, and Stefan Conrad. Pisco: A Computational Approach to Predict Personality Types from Java Source

Code. *Working notes of FIRE*, pages 7–10, 2016.
Contributions: The author and Matthias Liebeck applied pair programming to create the code. The manuscript was prepared jointly by the author, Matthias Liebeck, and Alexander Askinadze. **Status**: Published.

5. Pashutan Modaresi, Matthias Liebeck, and Stefan Conrad. Exploring the Effects of Cross-Genre Machine Learning for Author Profiling in PAN 2016. In *Working Notes of CLEF 2016 - Conference and Labs of the Evaluation Forum*, pages 970–977, 2016.
   Contributions: The author contributed with the design of the architecture, evaluated the developed system and prepared the manuscript jointly with Matthias Liebeck. **Status**: Published.

6. Matthias Liebeck, Pashutan Modaresi, and Stefan Conrad. Evaluating Safety, Soundness and Sensibleness of Obfuscation Systems. In *Working Notes of CLEF 2016 - Conference and Labs of the Evaluation Forum*, pages 920–928, 2016.
   Contributions: The author contributed with the evaluation of the dimensions *safeness* and *sensibleness*. The manuscript was prepared jointly by Matthias Liebeck and Pashutan Modaresi. **Status**: Published.

7. Matthias Liebeck, Philipp Pollack, Pashutan Modaresi, and Stefan Conrad. HHU at SemEval-2016 Task 1: Multiple Approaches to Measuring Semantic Textual Similarity. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, pages 595–601, 2016.
   Contributions: The author contributed with the design, development, and description of the the *Deep LDA* approach introduced in Section 3.3 of the publication. **Status**: Published.

2015

8. Pashutan Modaresi and Stefan Conrad. On Definition of Automatic Text Summarization. In *Proceedings of The Second International Conference on Digital Information Processing, Data Mining, and Wireless Communications (DIPDMWC2015)*, pages 33–40, 2015.
   Contributions: The research and the preparation of this manuscript was done entirely by the main author under the supervision of Prof. Conrad. **Status**: Published.

9. Pashutan Modaresi and Stefan Conrad. Identification and Evaluation of Keyphrases: Fuzzy Set based Scoring and Turing Test Inspired Evaluation. In *The Second International Conference on Digital Information Processing, Data Mining, and Wireless Communications (DIPDMWC2015)*, page 107, 2015.

Contributions: The research and the preparation of this manuscript was done entirely by the main author under the supervision of Prof. Conrad. **Status**: Published.

10. Pashutan Modaresi. Automatic Identification of Multipage News: A Machine Learning Approach. In *Proceedings of the LWA 2015 Workshops: KDML, FGWM, IR, and FGDB*, page 75, 2015.
    Contributions: The research and the preparation of this manuscript was done entirely by the main author under the supervision of Prof. Conrad. **Status**: Published.

2014

11. Pashutan Modaresi and Stefan Conrad. From Phrases to Keyphrases: An Unsupervised Fuzzy Set Approach to Summarize News Articles. In *Proceedings of the 12th International Conference on Advances in Mobile Computing and Multimedia*, MoMM '14, pages 336–341, 2014.
    Contributions: The research and the preparation of this manuscript was done entirely by the main author under the supervision of Prof. Conrad. **Status**: Published.

12. Philipp Gross and Pashutan Modaresi. Plagiarism Alignment Detection by Merging Context Seeds. In *Working Notes for CLEF 2014 Conference*, pages 966–972, 2014.
    Contributions: The author contributed with the evaluation and error analysis of the developed algorithm and prepared the manuscript. **Status**: Published.

13. Pashutan Modaresi and Philipp Gross. A Language Independent Author Verifier Using Fuzzy C-Means Clustering. In *Working Notes for CLEF 2014 Conference*, pages 1084–1091, 2014.
    Contributions: The author contributed with the design and development of the author verification algorithm and prepared the manuscript. **Status**: Published.

References

[ACD$^+$98] James Allan, Jaime Carbonell, George Doddington, Jonathan Yamron, Yiming Yang, James Allan Umass, Brian Archibald Cmu, Doug Beeferman Cmu, Adam Berger Cmu, Ralf Brown Cmu, Ira Carp Dragon, George Doddington Darpa, and Alex Hauptmann Cmu. Topic Detection and Tracking Pilot Study Final Report. In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, pages 194–218, 1998.

[AIAA15] Asad Abdi, Norisma Idris, Rasim M. Alguliyev, and Ramiz M. Aliguliyev. Query-based Multi-documents Summarization using Linguistic Knowledge and Content Word Expansion. *Soft Computing*, pages 1–17, 2015.

[AKS05] Stergos Afantenos, Vangelis Karkaletsis, and Panagiotis Stamatopoulos. Summarization from Medical Documents: A Survey. *Artificial intelligence in medicine*, 33(2):157–177, 2005.

[AOGL99] Chinatsu Aone, Mary E. Okurowski, James Gorlinsky, and Bjornar Larsen. *A Trainable Summarizer with Knowledge Acquired from Robust NLP Techniques*, pages 71–80. 1999.

[ASC14] Faisal Alvi, Mark Stevenson, and Paul Clough. Hashing and Merging Heuristics for Text Reuse Detection—Notebook for PAN at CLEF 2014. In Linda Cappellato, Nicola Ferro, Martin Halvey, and Wessel Kraaij, editors, *CLEF 2014 Evaluation Labs and Workshop – Working Notes Papers, 15-18 September, Sheffield, UK*. CEUR-WS.org, September 2014.

[Bax58] P. B. Baxendale. Machine-made Index for Technical Literature: An Experiment. *IBM J. Res. Dev.*, 2(4):354–361, October 1958.

[BDVJ03] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A Neural Probabilistic Language Model. *J. Mach. Learn. Res.*, 3:1137–1155, March 2003.

[BEM02] Regina Barzilay, Noemie Elhadad, and Kathleen R. McKeown. Inferring Strategies for Sentence Ordering in Multidocument News Summarization. *J. Artif. Int. Res.*, 17(1):35–55, August 2002.

[BP98] Sergey Brin and Lawrence Page. The Anatomy of a Large-scale Hypertextual Web Search Engine. In *Proceedings of the Seventh International Conference on World Wide Web 7*, pages 107–117. Elsevier Science Publishers B. V., 1998.

[CLLW16] Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. AttSum: Joint Learning of Focusing and Summarization with Neural Attention. *CoRR*, abs/1604.00125, 2016.

[CO01] John M. Conroy and Dianne P. O'leary. Text Summarization via Hidden Markov Models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 406–407. ACM, 2001.

[CQH16] Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. Neural Sentence Ordering. *CoRR*, abs/1607.06952, 2016.

[DBK16] Greg Durrett, Taylor Berg-Kirkpatrick, and Dan Klein. Learning-Based Single-Document Summarization with Compression and Anaphoricity Constraints. *CoRR*, abs/1603.08887, 2016.

[DWPP08] Mark Dredze, Hanna M. Wallach, Danny Puller, and Fernando Pereira. Generating Summary Keywords for Emails Using Topics. In *Proceedings of the 13th International Conference on Intelligent User Interfaces*, IUI '08, pages 199–206. ACM, 2008.

[Edm69] H. P. Edmundson. New Methods in Automatic Extracting. *J. ACM*, 16(2):264–285, April 1969.

[ER04] Günes Erkan and Dragomir R. Radev. LexRank: Graph-based Lexical Centrality As Salience in Text Summarization. *J. Artif. Int. Res.*, 22(1):457–479, December 2004.

[EW61] H. P. Edmundson and R. E. Wyllys. Automatic Abstracting and Indexing; Survey and Recommendations. *Commun. ACM*, 4(5):226–234, May 1961.

[GG16] Mahak Gambhir and Vishal Gupta. Recent Automatic Text Summarization Techniques: A Survey. *Artificial Intelligence Review*, pages 1–66, 2016.

[GGL09] Maria Grineva, Maxim Grinev, and Dmitry Lizorkin. Extracting Key Terms from Noisy and Multitheme Documents. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 661–670. ACM, 2009.

[GKG+05] Suhit Gupta, Gail E. Kaiser, Peter Grimm, Michael F. Chiang, and Justin Starren. Automating Content Extraction of HTML Documents. *World Wide Web*, 8(2):179–224, 2005.

[GL01] Yihong Gong and Xin Liu. Generic Text Summarization Using Relevance Measure and Latent Semantic Analysis. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 19–25. ACM, 2001.

[GL11] Pierre-Etienne Genest and Guy Lapalme. Framework for Abstractive Summarization Using Text-to-text Generation. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, MTTG '11, pages 64–73. Association for Computational Linguistics, 2011.

[GL12] Pierre-Etienne Genest and Guy Lapalme. Fully Abstractive Approach to Guided Summarization. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers*, ACL '12, pages 354–358. Association for Computational Linguistics, 2012.

[GM14] Philipp Gross and Pashutan Modaresi. Plagiarism Alignment Detection by Merging Context Seeds. In *Working Notes for CLEF 2014 Conference*, pages 966–972, 2014.

[GMCK00] Jade Goldstein, Vibhu Mittal, Jaime Carbonell, and Mark Kantrowitz. Multi-document Summarization by Sentence Extraction. In *Proceedings of the 2000 NAACL-ANLPWorkshop on Automatic Summarization*, NAACL-ANLP-AutoSum '00, pages 40–48. Association for Computational Linguistics, 2000.

[GPB+13] Tim Gollub, Martin Potthast, Anna Beyer, Matthias Busse, Francisco Rangel, Paolo Rosso, Efstathios Stamatatos, and Benno Stein. Recent Trends in Digital Text Forensics and Its Evaluation. In *Proceedings of the 4th International Conference on Information Access Evaluation. Multilinguality, Multimodality, and Visualization*, CLEF 2013, pages 282–302. Springer-Verlag New York, Inc., 2013.

[GR09] Albert Gatt and Ehud Reiter. SimpleNLG: A Realisation Engine for Practical Applications. In *Proceedings of the 12th European Workshop* *on Natural Language Generation*, ENLG ’09, pages 90–93. Association for Computational Linguistics, 2009.

[Gre11] Charles F. Greenbacker. Towards a Framework for Abstractive Summarization of Multimodal Documents. In *Proceedings of the ACL 2011 Student Session*, HLT-SS ’11, pages 75–80. Association for Computational Linguistics, 2011.

[HH12] David Harwath and Timothy J Hazen. Topic Identification based Extrinsic Evaluation of Summarization Techniques Applied to Conversational Speech. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5073–5076. IEEE, 2012.

[HL98] Eduard Hovy and Chin-Yew Lin. Automated Text Summarization and the SUMMARIST System. In *Proceedings of a Workshop on Held at Baltimore, Maryland: October 13-15, 1998*, TIPSTER ’98, pages 197–214. Association for Computational Linguistics, 1998.

[HN14] Kazi Saidul Hasan and Vincent Ng. Automatic Keyphrase Extraction: A Survey of the State of the Art. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1262–1273. Association for Computational Linguistics, June 2014.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[HSL07] Meishan Hu, Aixin Sun, and Ee-Peng Lim. Comments-oriented Blog Summarization by Sentence Extraction. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 901–904. ACM, 2007.

[Hul03] Anette Hulth. Improved Automatic Keyword Extraction Given More Linguistic Knowledge. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, EMNLP ’03, pages 216–223. Association for Computational Linguistics, 2003.

[HWvdB$^+$15] Manuela Hürlimann, Benno Weck, Esther van den Berg, Simon Suster, and Malvina Nissim. GLAD: Groningen Lightweight Authorship Detection. In *Working Notes of CLEF 2015 - Conference and Labs of the Evaluation forum*, 2015.

[IGC16] Yoshua Bengio Ian Goodfellow and Aaron Courville. Deep Learning. Book in preparation for MIT Press, 2016.

[KM02] Kevin Knight and Daniel Marcu. Summarization Beyond Sentence Extraction: A Probabilistic Approach to Sentence Compression. *Artif. Intell.*, 139(1):91–107, July 2002.

[KMTD14] Mikael Kågebäck, Olof Mogren, Nina Tahmasebi, and Devdatt Dubhashi. Extractive Summarization using Continuous Vector Space Models. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)@ EACL*, pages 31–39, 2014.

[KPC95] Julian Kupiec, Jan Pedersen, and Francine Chen. A Trainable Document Summarizer. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '95, pages 68–73. ACM, 1995.

[KSBD07] Moshe Koppel, Jonathan Schler, and Elisheva Bonchek-Dokow. Measuring Differentiability: Unmasking Pseudonymous Authors. *J. Mach. Learn. Res.*, 8:1261–1276, December 2007.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[Lin99] Chin-Yew Lin. Training a Selection Function for Extraction. In *Proceedings of the Eighth International Conference on Information and Knowledge Management*, CIKM '99, pages 55–62. ACM, 1999.

[Lin04] Chin-Yew Lin. ROUGE: A Package for Automatic Evaluation of Summaries. In Stan Szpakowicz Marie-Francine Moens, editor, *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81. Association for Computational Linguistics, July 2004.

[LJH05] Chang-Shing Lee, Zhi-Wei Jian, and Lin-Kai Huang. A Fuzzy Ontology and its Application to News Summarization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 35(5):859–880, 2005.

[LMAC16] Matthias Liebeck, Pashutan Modaresi, Alexander Askinadze, and Stefan Conrad. Pisco: A Computational Approach to Predict Personality Types from Java Source Code. *Working notes of FIRE*, pages 7–10, 2016.

[LMC16] Matthias Liebeck, Pashutan Modaresi, and Stefan Conrad. Evaluating Safety, Soundness and Sensibleness of Obfuscation Systems. In *Working Notes of CLEF 2016 - Conference and Labs of the Evaluation Forum*, pages 920–928, 2016.

[LPLL09] Feifan Liu, Deana Pennell, Fei Liu, and Yang Liu. Unsupervised Approaches for Automatic Keyword Extraction Using Meeting Transcripts. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 620–628. Association for Computational Linguistics, 2009.

[LPMC16] Matthias Liebeck, Philipp Pollack, Pashutan Modaresi, and Stefan Conrad. HHU at SemEval-2016 Task 1: Multiple Approaches to Measuring Semantic Textual Similarity. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, pages 595–601, 2016.

[Luh58] H. P. Luhn. The Automatic Creation of Literature Abstracts. *IBM J. Res. Dev.*, 2(2):159–165, April 1958.

[MC14] Pashutan Modaresi and Stefan Conrad. From Phrases to Keyphrases: An Unsupervised Fuzzy Set Approach to Summarize News Articles. In *Proceedings of the 12th International Conference on Advances in Mobile Computing and Multimedia*, MoMM '14, pages 336–341, 2014.

[MC15a] Pashutan Modaresi and Stefan Conrad. Identification and Evaluation of Keyphrases: Fuzzy Set based Scoring and Turing Test Inspired Evaluation. In *The Second International Conference on Digital Information Processing, Data Mining, and Wireless Communications (DIPDMWC2015)*, page 107, 2015.

[MC15b] Pashutan Modaresi and Stefan Conrad. On Definition of Automatic Text Summarization. In *Proceedings of The Second International Conference on Digital Information Processing, Data Mining, and Wireless Communications (DIPDMWC2015)*, pages 33–40, 2015.

[MC16] Pashutan Modaresi and Stefan Conrad. Simurg: An Extendable Multilingual Corpus for Abstractive Single Document Summarization. In *Proceedings of the 8th Annual Meeting of the Forum on Information Retrieval Evaluation*, FIRE '16, pages 24–27. ACM, 2016.

[MG14] Pashutan Modaresi and Philipp Gross. A Language Independent Author Verifier Using Fuzzy C-Means Clustering. In *Working Notes for CLEF 2014 Conference*, pages 1084–1091, 2014.

[MGSH12] Hadi Mohammadzadeh, Thomas Gottron, Franz Schweiggert, and Gerhard Heyer. TitleFinder: Extracting the Headline of News Web Pages Based on Cosine Similarity and Overlap Scoring Similarity. In *Proceedings of the Twelfth International Workshop on Web Information and Data Management*, WIDM '12, pages 65–72. ACM, 2012.

[Mih04] Rada Mihalcea. Graph-based Ranking Algorithms for Sentence Extraction, Applied to Text Summarization. In *Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions*, ACLdemo '04. Association for Computational Linguistics, 2004.

[MLC16a] Pashutan Modaresi, Matthias Liebeck, and Stefan Conrad. Exploring the Effects of Cross-Genre Machine Learning for Author Profiling in PAN 2016. In *Working Notes of CLEF 2016 - Conference and Labs of the Evaluation Forum*, pages 970–977, 2016.

[MLC16b] Pashutan Modaresi, Matthias Liebeck, and Stefan Conrad. Neural Classification of Linguistic Coherence Using Long Short-Term Memories. In *Proceedings of the 8th Annual Meeting of the Forum on Information Retrieval Evaluation*, FIRE '16, pages 28–31. ACM, 2016.

[Mod15] Pashutan Modaresi. Automatic Identification of Multipage News: A Machine Learning Approach. In *Proceedings of the LWA 2015 Workshops: KDML, FGWM, IR, and FGDB*, page 75, 2015.

[MT04a] R. Mihalcea and P. Tarau. TextRank: Bringing order into texts. In *Proceedings of EMNLP-04and the 2004 Conference on Empirical Methods in Natural Language Processing*, July 2004.

[MT04b] Rada Mihalcea and Paul Tarau. TextRank: Bringing Order into Texts. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 404–411. Association for Computational Linguistics, July 2004.

[NFK02] Joel Larocca Neto, Alex Alves Freitas, and Celso A. A. Kaestner. Automatic Text Summarization Using a Machine Learning Approach. In *Proceedings of the 16th Brazilian Symposium on Artificial Intelligence: Advances in Artificial Intelligence*, SBIA '02, pages 205–215. Springer-Verlag, 2002.

[NK07] Thuy Dung Nguyen and Min-Yen Kan. Keyphrase Extraction in Scientific Publications. In *Proceedings of the 10th International Conference on Asian Digital Libraries: Looking Back 10 Years and Forging New Frontiers*, ICADL'07, pages 317–326. Springer-Verlag, 2007.

[NSR11] Dong Nguyen, Noah A. Smith, and Carolyn P. Rosé. Author Age Prediction from Text Using Linear Regression. In *Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, LaTeCH '11, pages 115–123. Association for Computational Linguistics, 2011.

[NZZ16] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. SummaRuNNer: A Recurrent Neural Network based Sequence Model for Extractive Summarization of Documents. *CoRR*, abs/1611.04230, 2016.

[PHB$^+$14] Martin Potthast, Matthias Hagen, Anna Beyer, Matthias Busse, Martin Tippmann, Paolo Rosso, and Benno Stein. Overview of the 6th International Competition on Plagiarism Detection. In Linda Cappellato, Nicola Ferro, Martin Halvey, and Wessel Kraaij, editors, *Working Notes Papers of the CLEF 2014 Evaluation Labs*, CEUR Workshop Proceedings. CLEF and CEUR-WS.org, September 2014.

[PL13] Matthew E. Peters and Dan Lecocq. Content Extraction Using Diverse Feature Sets. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13 Companion, pages 89–90. ACM, 2013.

[Por97] M. F. Porter. Readings in Information Retrieval. chapter An Algorithm for Suffix Stripping, pages 313–316. Morgan Kaufmann Publishers Inc., 1997.

[PRV$^+$16] Francisco Manuel Rangel Pardo, Paolo Rosso, Ben Verhoeven, Walter Daelemans, Martin Potthast, and Benno Stein. Overview of the 4th Author Profiling Task at PAN 2016: Cross-Genre Evaluations. In *Working Notes of CLEF 2016 - Conference and Labs of the Evaluation Forum*, pages 750–784, 2016.

[PRWZ02] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318. Association for Computational Linguistics, 2002.

[Qui93] J. Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers Inc., 1993.

[RCW15] Alexander M. Rush, Sumit Chopra, and Jason Weston. A Neural Attention Model for Abstractive Sentence Summarization. *CoRR*, abs/1509.00685, 2015.

[Sar14] Kamal Sarkar. A Keyphrase-Based Approach to Text Summarization for English and Bengali Documents. *Int. J. Technol. Diffus.*, 5(2):28–38, April 2014.

[SB88] Gerard Salton and Christopher Buckley. Term-weighting Approaches in Automatic Text Retrieval. *Inf. Process. Manage.*, 24(5):513–523, August 1988.

[SDV$^+$14] Efstathios Stamatatos, Walter Daelemans, Ben Verhoeven, Martin Potthast, Benno Stein, Patrick Juola, Miguel A. Sanchez-Perez, and Alberto Barrón-Cedeño. Overview of the Author Identification Task at PAN 2014. In Linda Cappellato, Nicola Ferro, Martin Halvey, and Wessel Kraaij, editors, *CLEF 2014 Evaluation Labs and Workshop – Working Notes Papers.* CEUR-WS.org, September 2014.

[She11] Victor S. Sheng. *Fast Data Acquisition in Cost-Sensitive Learning*, pages 66–77. Springer Berlin Heidelberg, 2011.

[SJ04] Josef Steinberger and Karel Ježek. Using latent semantic analysis in text summarization and summary evaluation. In *In Proc. ISIM '04*, pages 93–100, 2004.

[SJ09] Josef Steinberger and Karel Jezek. Evaluation Measures for Text Summarization. *Computing and Informatics*, 28(2):251–275, 2009.

[SL02] Horacio Saggion and Guy Lapalme. Generating Indicative-informative Summaries with sumUM. *Comput. Linguist.*, 28(4):497–526, December 2002.

[SLP11] Benno Stein, Nedim Lipka, and Peter Prettenhofer. Intrinsic Plagiarism Analysis. *Lang. Resour. Eval.*, 45(1):63–82, March 2011.

[SM86] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, Inc., 1986.

[SPSG14] Miguel A. Sanchez-Perez, Grigori Sidorov, and Alexander Gelbukh. A Winning Approach to Text Alignment for Text Reuse Detection at PAN

2014—Notebook for PAN at CLEF 2014. In *CLEF 2014 Evaluation Labs and Workshop – Working Notes Papers*. CEUR-WS.org, September 2014.

[SSJ01] Tetsuya Sakai and Karen Sparck-Jones. Generic Summaries for Indexing in Information Retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 190–198. ACM, 2001.

[TM14] Juan-Manuel Torres-Moreno. *Automatic Text Summarization.* Wiley, 2014.

[Tur00] Peter D. Turney. Learning Algorithms for Keyphrase Extraction. *Inf. Retr.*, 2(4):303–336, May 2000.

[WC12] Lu Wang and Claire Cardie. Focused Meeting Summarization via Unsupervised Relation Extraction. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, SIGDIAL '12, pages 304–313. Association for Computational Linguistics, 2012.

[WHW$^+$09] Junfeng Wang, Xiaofei He, Can Wang, Jian Pei, Jiajun Bu, Chun Chen, Ziyu Guan, and Gang Lu. News Article Extraction with Template-independent Wrapper. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 1085–1086. ACM, 2009.

[WLF15] Shanchan Wu, Jerry Liu, and Jian Fan. Automatic Web Content Extraction by Combination of Learning and Grouping. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 1264–1274. International World Wide Web Conferences Steering Committee, 2015.

[WPF$^+$99] Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin, and Craig G. Nevill-Manning. KEA: Practical Automatic Keyphrase Extraction. In *Proceedings of the Fourth ACM Conference on Digital Libraries*, DL '99, pages 254–255. ACM, 1999.

[WX08] Xiaojun Wan and Jianguo Xiao. Single Document Keyphrase Extraction Using Neighborhood Knowledge. In *Proceedings of the 23rd National Conference on Artificial Intelligence*, AAAI'08, pages 855–860. AAAI Press, 2008.

[XGMR13] Wei Xu, Ralph Grishman, Adam Meyers, and Alan Ritter. A preliminary study of tweet summarization using information extraction. *NAACL 2013*, page 20, 2013.

[YKYM05] Jen-Yuan Yeh, Hao-Ren Ke, Wei-Pang Yang, and I-Heng Meng. Text Summarization Using a Trainable Summarizer and Latent Semantic Analysis. *Inf. Process. Manage.*, 41(1):75–95, January 2005.

[ZLFU16] Wenyuan Zeng, Wenjie Luo, Sanja Fidler, and Raquel Urtasun. Efficient summarization with read-again and copy mechanism. *CoRR*, abs/1611.03382, 2016.