Ambient Search - just-in-time document recommendations for speech fragments

Ambient Search - Empfehlung von Dokumenten in Echtzeit auf Basis von Sprachfragmenten
Bachelor-Thesis von Jonas Wacker
Tag der Einreichung:

1. Gutachten: Benjamin Milde
2. Gutachten: Prof. Dr. Chris Biemann

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Informatik
Language Technology

Ambient Search - just-in-time document recommendations for speech fragments
Ambient Search - Empfehlung von Dokumenten in Echtzeit auf Basis von Sprachfragmenten

Vorgelegte Bachelor-Thesis von Jonas Wacker

1. Gutachten: Benjamin Milde
2. Gutachten: Prof. Dr. Chris Biemann

Tag der Einreichung:

Erklärung zur Bachelor-Thesis

Hiermit versichere ich, die vorliegende Bachelor-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 3.3.2016

(Jonas Wacker)

Abstract

When taking part in a conversation or listening to a lecture, people may feel the need to look up helpful background information. Usually this requires the affected person to stop following the flow of speech and to manually look up the corresponding information. The goal of this thesis was to develop a system referred to as *Ambient Search* that proactively provides useful document recommendations to its users based on their potential information needs. We attempt to determine these needs by transcribing the speech input and extracting relevant keyphrases from it that are then used to formulate a search query against a database of text documents. The best-scoring search results are displayed to the user in a non-intrusive manner such that they can take a look at them whenever they feel the need for it.

Our focus is put on the automatic extraction of relevant keyphrases from speech transcripts. We introduce novel algorithmic methods that score potential keyphrases according to their topical relevance as well as their specificity. In particular, we employ *Word2Vec* instead of a traditional topic model for this purpose, which leads to a rather experimental approach since we have not seen any comparable methods yet. Nonetheless, our results seem very promising and leave plenty of space for future research.

Zusammenfassung

Teilnehmern einer Unterhaltung sowie Zuhörern eines Vortrags fehlt oft wichtiges Hintergrundwissen, das benötigt wird, um die dargebotenen Inhalte in vollem Umfang zu verstehen. Um potentielle Informationslücken zu schließen, müssen die Betroffenen entweder den Sprachfluss unterbrechen oder aufhören ihm zu folgen. Nur so können sie relevante Informationen nachschlagen oder erfragen. Das Ziel dieser Arbeit war die Entwicklung eines Systems, welches automatisch sinnvolle Dokumente oder Artikel vorschlägt, die potentielle Informationslücken der Nutzer bedienen. Wir nennen dieses System *Ambient Search* und gehen dabei wie folgt vor:

Zunächst werden die gesprochenen Inhalte transkribiert, sodass im darauffolgenden Schritt relevante Wörter aus den Transkripten extrahiert werden können. Die extrahierten Wörter werden sodann zu einer Suchanfrage kombiniert, mithilfe welcher passende Dokumente und Artikel in einer Datenbank gefunden werden. Die Suchergebnisse werden den Nutzern so präsentiert, dass sie sie sich zu jedem beliebigen Zeitpunkt anschauen können.

Der Fokus unserer Arbeit liegt in der automatischen Extraktion von Schlüsselwörtern aus Sprachtranskripten, die deren Inhalte sowohl thematisch als auch präzise erfassen sollen. Dazu haben wir neue Algorithmen entwickelt, die *Word2Vec* benutzen, um die thematische Relevanz von Wörtern zu erfassen. In vergleichbaren Arbeiten wurden hierfür vorwiegend *Topic Models* verwendet, weshalb wir unseren Ansatz als experimentell betrachten. Dennoch liefert unsere Methode vielversprechende Ergebnisse und lässt dabei noch viel Raum für zukünftige Forschung.

Contents

1 Introduction

Recent advancements in Automatic Speech Recognition (ASR) have led to very prominent commercial adoptions of voice-based personal assistants such as Apple Siri[1] or Microsoft Cortana[2]. The goal of these systems is to simplify the interaction between humans and computer devices in order to integrate them more easily into everyday life. However, all these systems have in common that they need to be actively used in order to carry out commands.

In this thesis we make an attempt of designing a just-in-time document recommendation system that actively listens to speech input in order to automatically determine a user's potential information needs. The document recommendations serve to satisfy those needs in the best possible way. A document in this regard can be any kind of textual information. In the implementation related to this thesis documents are represented by Wikipedia articles. There are multiple scenarios in which we imagine such a system to be useful. For instance, it could automatically provide additional information about technical terms when listening to lectures or presentations. This way the user would not be interrupted from following the lecture in order to manually look up corresponding information. The system could also suggest relevant documents to participants in business meetings without interrupting the conversation flow as it has been suggested in [19].

The Language Technology department at Technische Universität Darmstadt and the Sibis Institute in Berlin carried out a first empirical study regarding the potential applicability of such a system. A great part of the respondents (16/20) could imagine to use it in their everyday life. The system was perceived more useful for lectures, meetings and tv shows than for free conversations.

We call the process that transforms surrounding speech input into document recommendations *Ambient Search*. For our proposed system the process consists of three steps:

- Automatic Speech Recognition and Transcription
- Extraction of relevant Keyphrases
- Formulation of Search Queries and Presentation of Search Results

At first, the speech signal is transcribed by an ASR system. The textual transcript is then processed by an algorithm that serves to extract its most relevant information (keyphrases). The keyphrases are combined to form a search query for an information retrieval (IR) system which searches relevant documents that are displayed to the user as a result.

This process is reflected in the structure of this thesis. In the beginning, we will provide extensive background information that is needed to understand the implementation of all the Ambient Search components. We then give a brief overview over related work that deals with similar systems. Finally, we present our implementation of an Ambient Search system that will be evaluated afterwards.

Our main focus in this work has been put on the development of novel algorithmic methods used for keyphrase extraction from ASR transcripts. Our new methods will be integrated into an already existing version of an Ambient Search system[3] to improve on its performance. We will compare our approach to a state-of-the-art method that is used in a comparable system. Furthermore, we provide very detailed background information in Section 2.2 that is needed in order to understand both implementations.

Finally, we will draw scientific as well as practical conclusions with respect to our Ambient Search implementation.

[1] http://www.apple.com/de/ios/siri/
[2] http://www.windowsphone.com/de-de/how-to/wp8/cortana/meet-cortana
[3] https://github.com/bmilde/ambientsearch

2 Background

2.1 Automatic Speech Recognition

Automatic speech recognition (ASR) is the process of transforming an acoustic signal that contains human speech into a sequence of words. This process is also known as automatic speech-to-text transcription. Even though it may seem obvious to humans which words are used by people when they talk to one another, the automatic detection of these words is a very challenging task [50].

This is due to numerous reasons such as the ambiguity of homophones used in spoken language. The sound of the English phrase "ice cream" is equal to the sound of the word sequence "I scream" assuming that all words are stressed in a similar manner. Therefore, an ASR system has to be able to disambiguate homophones inside words or even sentences in order to find the correct text transcriptions. This disambiguation requires additional knowledge about a language that is not contained in the utterances themselves. In particular, solving this problem involves the detection of word boundaries inside the speech signal since modern systems are expected to process continuous speech instead of isolated words. Should the first word boundary be set after the sound of the word "I" or the word "ice"? Additional ASR challenges include dealing with different voices and speaking styles of people as well as distracting background noise.

Because of all these challenges and because ASR systems are developed for many languages that are subject to constant changes, statistical machine learning frameworks have emerged as the predominantly used tool in this domain. These frameworks provide the possibility to train ASR models on large datasets of aligned text and speech.

James Baker set the foundation for today's statistical frameworks by applying Hidden Markov Models (HMM) to speech recognition in the 1970s [3]. Acoustic models are generally based on HMMs that try to determine a sequence of subsegments (HMM states) of a phone in speech. These HMM states can be hierarchically combined to form phonemes and finally words that, according to the model, are most likely to have generated the speech signal [62].

Before the speech signal can be transformed into a sequence of HMM states by the acoustic model, it needs to be converted into a computation-friendly format. Therefore, acoustic features are extracted from the signal such that they contain discriminative information with respect to phonetic alignment. One of the commonly used feature representations in this regard are Mel Frequency Cepstral Coefficients (MFCC) [10].

The HMM can then be used to infer the most likely sequence of states that may have generated the underlying acoustic feature vectors. In this context, the HMM relies on conditional probabilities that determine the feature vectors' probabilistic distribution. Deep neural networks (DNN) have most recently been employed to (indirectly) obtain these probabilities leading to a substantial gain in performance of ASR systems [23]. This is also due to the advancements in computational technology that have made the training of such networks feasible on large amounts of data. An introduction to neural networks is given in Section 9.1. A deep neural network is a neural network with many hidden layers.

The acoustic model described until now is generally combined with a language model trained on large datasets of textual data. The language model is used to complement the acoustic model by indicating the probabilities of given word sequences. Therefore, the word sequence "ice cream" can be disambiguated from the word sequence "I scream" if the words that precede the given sequence are provided. In the example used above, the language model would yield a higher probability for the sequence "ice cream" if the preceding words were "I eat".

Figure 2.1 shows the architecture of modern ASR systems. The speech signal is processed by the acoustic model which generates potential HMM state sequences along with their probabilities of having generated the signal (AM score). A language model is used to complement these probabilistic scores with scores that describe how likely a word sequence is to occur in a given language (LM score). A

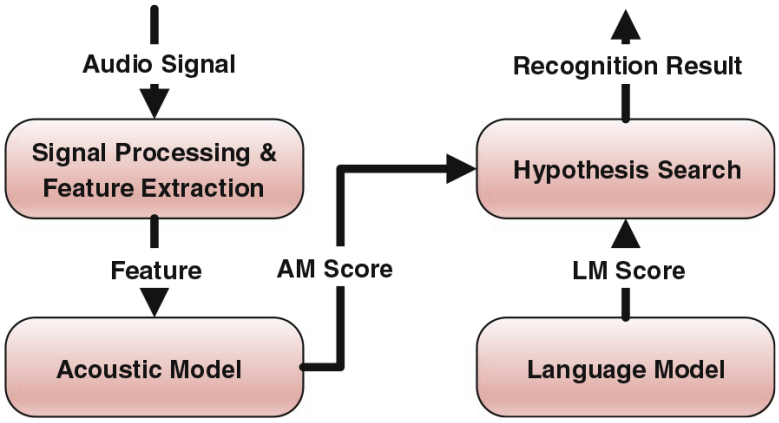

Figure 2.1: Architecture of modern ASR systems. Taken from [62].

hypothesis search combines both scores in order to return the most likely word sequence as the ASR system's recognition result.

2.2 Automatic Keyphrase Extraction

In order for the Ambient Search system to make useful document suggestions that match a transcribed fragment of speech, it needs to extract the transcript's most important features first. Since the ASR transcripts are text documents, this task can be defined as extracting their most important *keyphrases*. We intentionally use the term *keyphrase* instead of the term *keyword* because a keyphrase can consist of one or more keywords that altogether imply a semantic meaning. For instance, connecting the words *machine* and *learning* creates a new term with a semantic meaning that is not covered by its isolated components. Moreover, it is important to distinguish keyphrase extraction from information extraction which extracts task-dependend information and is therefore more restricted to specialised contexts. Keyphrase extraction is meant to extract relevant phrases from any text without taking any specifically asked question into account.

The automatic extraction of keyphrases has a wide range of applications. As text summarization carried out by humans is a very time consuming task, new methods were invented in order to deal with the summarization of the constantly growing amounts of textual data on the internet. Automatic keyphrase extraction has turned out to be a very useful tool in this domain. Additional applications include document indexing and formulating search queries [58]. We intend to use the latter for retrieving document suggestions.

There are two types of keyphrase extraction algorithms, supervised and unsupervised ones. The class of supervised keyphrase extraction algorithms was introduced by Turney in 1999 [58]. They rely on training data consisting of previously labelled keyphrases, which turns keyphrase extraction into a classification task, i.e. considering an observed phrase a keyphrase or not based on the training data. Supervised learning methods focusing on lexical frequency measures had been considered the state-of-the-art in keyphrase extraction for a long time. Especially the *term-frequency inverse-document-frequency (tf-idf)* weighting scheme occurred in a substantial part of these supervised methods (see Frank et al. [15] and Hulth [27]). The *tf-idf* measure [55] for a term w and a document d_i from the document collection $C = (d_1, d_2, \ldots, d_n)$ is calculated in the following way.

The term frequency $tf_{w,i}$ measures the number of times that the term appears in the document. Let $\#(w, d_i)$ be the number of times that the term w appears in document d_i.

$$tf_{w,i} = \#(w, d_i) \tag{2.1}$$

The inverse document frequency $idf_{w,C}$ measures the uniqueness of a term in the entire collection of documents. $\#(C,w)$ is the number of documents in C that contain w.

$$idf_{w,C} = \log \frac{|C|}{\#(C,w)} \tag{2.2}$$

The resulting *tf-idf* measure looks as follows.

$$tf\text{-}idf_{w,d_i} = tf_{w,d_i} \times idf_{w,C} \tag{2.3}$$

The logarithm in Equation 2.2 is used in order to avoid rating extremely rare terms too high. According to this measure, terms that occur frequently in the current document but are rare in the rest of the document collection obtain a high score. This seems quite intuitive as those terms are a discriminating factor for the document. However, this metric relies solely on term frequency features, which results in some drawbacks that will be discussed in the course of this paper.

That is why Hulth also included syntactical information in the scoring of relevant terms, which improved results compared to the previous methods. Turney, Frank et al. and Hulth evaluated their methods by comparing them to manually extracted keyphrases by humans. Hulth achieved a maximum precision of $\simeq$ 30% (proportion of the automatically extracted keywords that are correct, i.e. also extracted manually by humans), and a recall of $\simeq$ 66% (proportion of the manually assigned keyphrases that the algorithm could find). However, labelling training data by hand for supervised algorithms is a time consuming task that is not flexible enough when applying keyphrase extraction methods to multiple and constantly changing domains. Since Ambient Search should be able to adapt quickly to arbitrary conversation fragments, we will focus on the investigation of unsupervised keyphrase extraction methods.

In 2004, Matsuo and Ishizuka introduced an unsupervised keyphrase extraction method that was based on word co-occurrences [37]. This approach did not require any previous training whatsoever. Instead it was applied to a single document evaluating the frequency of words and their occurrences with other words in the same sentence. This way less frequent, yet important words in the document could benefit from more frequent words for their score if they co-occurred often enough. Despite the simplicity of this approach, according to Matsuo and Ishizuka the results were comparable to the previously examined supervised methods based on *tf-idf*.

In the same year Mihalcea and Tarau published their work related to a graph-based ranking method called *TextRank* [40]. This unsupervised method was inspired by Google's PageRank algorithm [8] that serves to compute the importance of a vertex in a graph-based network using the entire network's information. In contrast to PageRank, TextRank was not applied to the link structure of the world wide web but on textual data. TextRank is not restricted to the task of keyphrase extraction but has proven to work reasonably well in this area. In fact, it had been considered the state-of-the-art unsupervised keyphrase extraction approach in the domain of article summarization for a long time [35].

When TextRank was applied to keyphrase extraction, single words were usually considered to be the graph's vertices and the edges of the graph were the co-occurrence relations of those words. Unlike in the co-occurrence definition of Matsuo and Ishizuka, the words did not need to be part of the same sentence but of a given word window containing a set of N words. The extracted words could be restricted to certain types of terms such as nouns and adjectives as it was done in Mihalcea and Tarau's work. After calculating the scores for each word of the underlying text, adjacent words were connected to form phrases. It is important to note that this approach did not rely on any previous training data at all. Therefore, it was as flexible as Matsuo and Ishizuka's algorithm, but delivered slightly better results than any previous method.

However, one has to bear in mind that all the mentioned keyphrase extraction methods until now were applied to scientific articles only. These articles constitute well written and coherent texts that

include substantial amounts of sophisticated terms. Figure 2.2 shows a sample undirected graph from a short abstract of a scientific article generated by the TextRank method. It can be seen that words are recursively connected with each other when traversing several connected vertices. Thus, TextRank is a recursive algorithm that, unlike Matsuo and Ishizuka's algorithm, considers transitive co-occurrence relations between words. The relations among words are weighted according to their recursive importance, meaning that if a highly rated word (one that is linked to very often) links to another word, this link becomes a highly-rated recommendation.

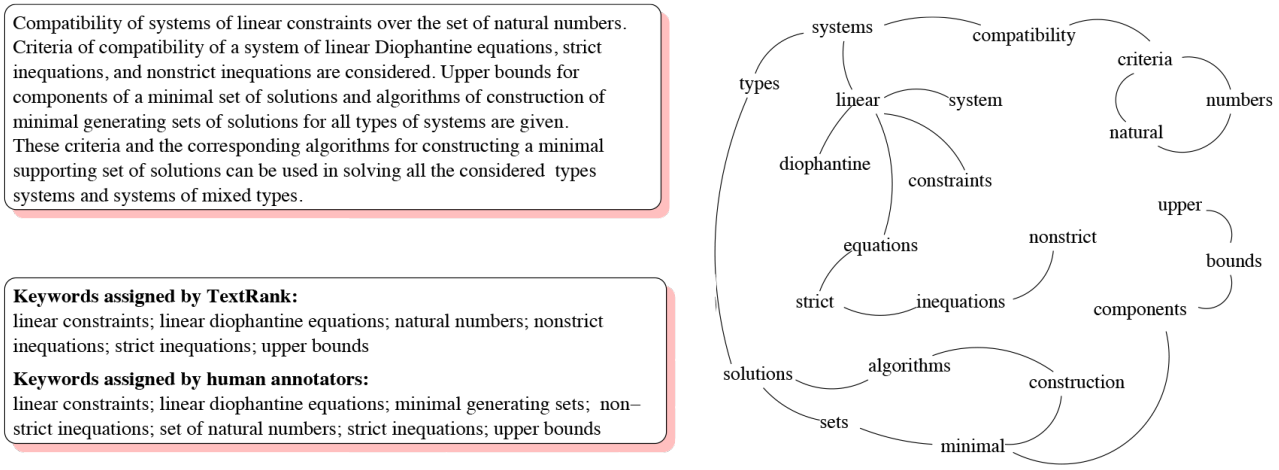

Figure 2.2: Sample graph generated from an abstract of a scientific article. Taken and adapted from [40].

Research conducted by Plas et al. [59] focused on another type of text corpora, namely conversation transcripts. Their method of keyphrase extraction was evaluated on conversation transcripts with an average of six participants talking about a single topical domain. Besides measures based on term-frequency, they also included lexical resources such as WordNet [13] into their approach. The lexical resources enabled them to assign words to semantic concepts which were then grouped into clusters. Keyphrases were extracted according to a score that was computed for each concept and each cluster based on their semantic connectivity and the associated term-frequency of their members in the text. This approach improved especially on the precision of extracted keyphrases compared to other methods, but it required larger amounts of human-labelled data reducing its flexibility. However, it has to be noted that considering a word's semantic meaning has great advantages over using solely frequency-based metrics. This can be illustrated with a simple example taken from [19]. Supposing a document contained the following words, each at a low frequency: *car*, *wheel*, *seat* and *passenger*. Frequency-based measures would assign each of the words a low score although taken together they can form a highly relevant set when a document deals with the semantic topic of *cars*. Using synonyms in a document would lower each of the related words' frequencies and thus their scores.

F. Liu et al. (2009) followed up on the research related to keyphrase extraction from conversational transcripts [34]. They were also dealing with audio transcripts from multi-party conversations. According to Liu et al., the spoken language in group meetings is different from written texts and other speech data. Discussions with multiple participants tend to be unorganized and contain spontaneous speech that sometimes does not constitute clear sentences. Therefore, the application of existing unsupervised keyphrase extraction methods to this domain was examined using the *ICSI meeting corpus*. This corpus contains transcripts of science-related discussions.

The research conducted by Liu et al. brought up valuable insights for the above mentioned domain. It was established that even human annotators had difficulties when manually extracting the most relevant keyphrases from the transcripts. The differences of extracted keyphrases among annotators depended on what kind of topic the transcripts were drawn from. Some topics were found to be more difficult than

others. This implies that the precision and recall-based evaluation methods for keyphrase extraction algorithms may not be representative for every domain. Therefore, two new evaluation methods were incorporated. The *pyramid method* [44], originally designed for the evaluation of automatic text summarization methods, includes all the human annotations and assigns weights to them that are based on the annotators' agreement. The other evaluation method used human subjective ratings. Its novelty lay in the assumption that annotated keywords by humans were not considered perfect. Instead, their error rates were assessed by other humans just as it would be done for automatically extracted keyphrases.

Unsupervised *tf-idf*-based methods were compared to graph-based methods for the extraction of single keywords only. In this study the *tf-idf*-based algorithms performed slightly better than the graph-based ones when incorporating sentence salience scores into the *tf-idf* measures. The scores in general changed drastically when applying the methods directly to ASR output which does not recognize all the words correctly. Yet even then, *tf-idf*-based algorithms outperformed graph-based algorithms, this time with the aid of word clustering methods or syntactical information respectively. It was also shown that adding syntactical information to either of the methods improved scores regardless of the textual input. It was concluded that the graph-based methods' loss in performance compared to other domains was due to the missing structure in human conversations. These studies show that this domain requires different approaches in keyphrase extraction from the ones utilised for scientific articles. Moreover, it can be seen that the simplicity of an algorithm such as *tf-idf* does not necessarily contradict its effectiveness.

In 2010 Z. Liu et al. presented their work considering *Topical Page Rank (TPR)* [35]. This graph-based keyphrase extraction method incorporates topical information, assuming that documents as well as words can be represented as a mixture of semantic topics. The topics were inferred based on a large collection of training documents using Latent Dirichlet Allocation (LDA). An introduction to probabilistic topic models and LDA will be given in the next section. According to Liu et al., previous graph-based methods had two major drawbacks. They scored words depending on their connectivity with other words only, without considering the words' contribution to a document's topics. Moreover, they did not take into account that a document could have multiple semantic topics of which the most important ones needed to be covered. Thus, Liu et al. used a separate PageRank algorithm for each of the document's topics to rate each word's relevance towards that particular topic. The algorithm was evaluated on two corpora consisting of research papers and news articles respectively. The evaluation yielded better results compared to TextRank and *tf-idf*-based approaches which proves that incorporating topic modelling techniques can improve the performance of existing keyphrase extraction algorithms. However, it was also shown that algorithms based only on LDA achieved worse results than the mixture of LDA and the graph-based approach. Therefore, it was concluded that successful keyphrase extraction in the underlying domain needed both, topical and structural-/frequency-based information of texts.

Habibi and Popescu-Belis followed up on the idea of topical coverage for keyphrase extraction using LDA [19]. However, their developed algorithm favoured topical diversity among extracted keyphrases as well. The idea to combine topical coverage with the topical diversity of keyphrases was inspired by Lin and Bilmes [33] who defined a diversity reward function for the domain of automatic text summarization. This function was based on the notion of increasing the extracted information's relevance (coverage) and decreasing its redundancy (ensuring a high diversity). Furthermore, they found out that a substantial part of well-performing text summarization algorithms shared one common feature in that they were solvers for submodular optimization problems. A *monotone non-decreasing submodular function* is characterized by having diminishing returns, i.e. its incremental value decreases with the increase of its input. While doing so, the incremental value always stays positive. When transferring this idea to the task of text summarization, we could formulate such a function to determine the coverage of a topic given the already extracted set of sentences. The more a topic has been covered by the extracted sentences, the less an additional sentence related to the same topic would add to the overall coverage score. This assumption seems suitable for the area of keyphrase and sentence extraction as redundant information should be avoided. According to Lin and Bilmes, the great advantage of this well-suited mathematical property is that associated computational problems can be solved using a greedy algo-

rithm that achieves a well-approximated solution regardless of the size of the problem. Without the greedy algorithm, solving this issue tends to be computationally expensive and in many cases intractable to compute.

The implementation by Habibi and Popescu-Belis proves that the above mentioned idea can be transferred to the domain of keyphrase extraction and that it also yields decent results when being applied to conversational transcripts. The evaluation of this method was done using binary comparisons between multiple algorithms. It was especially confirmed that enforcing topical diversity in the domain of conversational transcripts improves results compared to focusing solely on topical coverage. A coherent description of their implementation is presented in Section 3. As this method is applied to the domain of extracting keyphrases from conversation fragments, we consider their work highly relevant to our research. One restriction about their algorithm is that it only extracts single keywords. Habibi and Popescu-Belis also mentioned the possibility to further investigate on word clustering methods to be used for keyphrase extraction. However, it was stated that novel word clustering techniques that are based on distributional semantics such as Word2Vec [42] would not explicitly reward topical diversity. We will further investigate on this issue and try to find out whether we will be able to compensate on this drawback.

2.2.1 An Introduction to Topic Modeling and Latent Dirichlet Allocation

Keyphrase extraction algorithms can be improved by including a document's topical information when calculating the scores for relevant keyphrases. According to Liu et al. [35], this way one can ensure that the extracted information covers the major topics of the document and that a keyphrase is relevant to at least one of its major topics. This leads to noise reduction among the extracted keyphrases and provides a topical coverage that often better reflects a document's content [19]. This is a very helpful feature in order to filter out noise generated by ASR systems. Furthermore, unlike topical coherent articles, conversational speech tends to stray off topic more often. Minor topics of a speech fragment can be filtered out with the aid of the transcript's topical information [22]. Since we are dealing with conversational speech in our work as well, the idea of topical coverage is very important to us. Therefore, we will give an introduction to topic models in this section.

Topic models are algorithms that serve to discover and annotate thematic information in an unstructured collection of documents. These algorithms solely operate on the given documents and therefore do not need any prior annotations in order to uncover the documents' thematic structure [6].

A *document* in this context is defined as "a sequence of N words denoted by $\boldsymbol{w} = (w_1, w_2, \ldots, w_N)$ where w_n is the nth word in the sequence". These words do not necessarily need to be words in a text but can also relate to more abstract entities such as parts of an image [7]. In this thesis however, we only apply topic models to collections of text documents. The collection of text documents will from now on be called the *corpus*.

The most prominent topic models such as Probabilistic Latent Semantic Analysis (pLSA) [26] and Latent Dirichlet Allocation (LDA) [7] are built upon several assumptions about the documents as well as the process in which they were created. Documents are typically regarded as a probabilistic mixture of topics where each topic is a distribution over words. Furthermore, it is assumed that each document is created following a *generative process*. In this process, the creation of a new document starts with choosing a distribution over topics for this document. For each word that will be added to it, a random topic needs to be drawn from that distribution. Then, the word can again be drawn from the topic's distribution. The goal of topic models is thus to reverse the assumed process of document creation and to infer the topical structure for each document given only the words it contains [57].

Figure 2.3 illustrates a number of documents as a result of the generative process on the left and the problem of statistical inference given the documents and their words on the right. In this example, the generative process creates three documents of which only the second is a distribution over two topics. The other two documents draw words from a single topic. Topic assignments for each word are denoted

by the numbers in the superscript annotations attached to them. For instance, we can see that document 2 is a distribution over two topics where each topic holds a probability of 0.5. The words have therefore been generated by one of the two topics. Note that the generative process in this example generates the words only according to probabilistic distributions that are drawn for each document as well as for each topic. Therefore, the words and the number of times they appear in a document fit a document's thematic structure, but the documents do not represent an ordered sequence of words that are typically produced by natural language. The simplifying assumption of taking only the words and their counts into account is called the *bag-of-words* assumption which is central to topic models such as LDA which we will discuss in more detail in the next section. As it can be seen on the right hand side of Figure 2.3, the documents' distributions over topics as well as the topical distributions over words are latent (i.e. hidden) at first. The task of the topic model is thus to infer those distributions and to annotate each word with the topic they have been generated from.

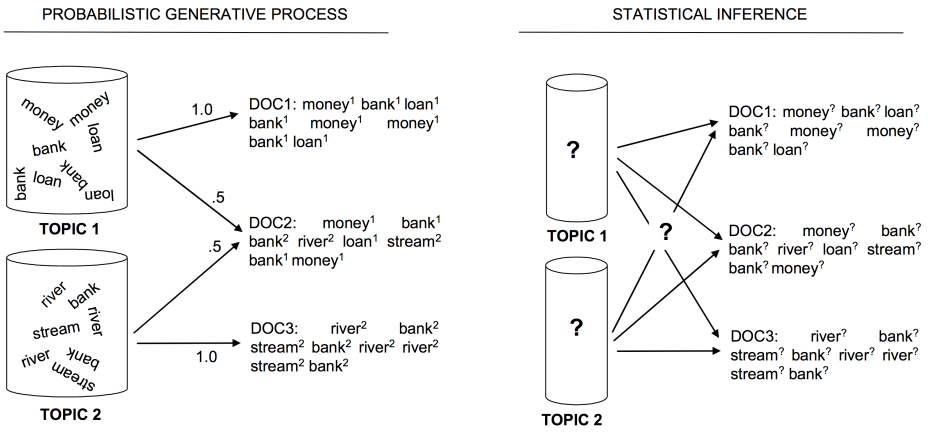

Figure 2.3: Illustration of the generative process and the problem of statistical inference of documents' thematic structure. Taken from [57].

Latent Dirichlet Allocation

Latent Dirichlet Allocation (Blei et al. [7]) is a probabilistic topic model that was developed in order to "fix an issue with [the] previously developed topic model *probabilistic latent semantic analysis*" [6]. Probabilistic latent semantic analysis (Hofmann [26]) does not make any assumptions on how a document's distribution over topics is generated within the generative process [57]. According to Blei et al., this leads to two major problems. Since in pLSA, this distribution is directly taken from the corpus that the model is trained on, there is a high risk of overfitting the model on the one hand and it is not clear how to assign topical distributions to unseen documents on the other [7].

For this purpose, Blei et al. introduced the random variable Θ_d which denotes the distribution over topics given a document d (i.e. the topic proportions of a document). This variable is drawn from a *dirichlet distribution* each time a new document is created. Without going into detail about dirichlet distributions, we will simply regard them as a parameterized distribution over distributions. We will also assume a symmetric dirichlet distribution (which is more convenient and often used in LDA implementations) [57]. Then, the parameter α is a K-dimensional vector with $\alpha_1 = \alpha_2 = \ldots = \alpha_K$. In this case, α is used to determine whether we tend to have similar topic distributions among documents that also assign similar weights to the topics (higher α), or more distinct topic distributions for each document where topic weights differ more strongly (lower α). In practice, the latter is often favoured.

Now we can define the probabilistic generative process of LDA for each document in a corpus according to [7]:

1. Choose $N \sim Poisson(\xi)$.
   [This distribution is not critical for the rest of the model. It simply serves to define the number of words for the document.]
2. Choose $\Theta \sim Dir(\alpha)$.
   [Θ is the topic mixture for the document.]
3. For each of the N words w_n:
   a) Choose a topic $z_n \sim Multinomial(\Theta)$.
      [z_n is the topic index to be chosen for word n.]
   b) Choose a word w_n from $p(w_n \mid z_n, \beta)$, a multinomial probability conditioned on the topic z_n.
      [β are the topics of which the z_nth index is taken. Each topic is a distribution over words from which we can draw the described probability.]

Note that in this model the topic-word-probability matrix β as well as α are assumed to be sampled once for the generation of the entire corpus. Furthermore, the total number of topics needs to be predefined.

In the following we will give a short real world example having trained an LDA model on a subset of the English Wikipedia of about 110,000 text documents. The number of topics was set to 100, α_n to 0.5 for each $n = 1, \ldots, 100$ and η to 0.01. In the generative process above, η is used to initialise β with $\beta \sim Dir(\eta)$. Thus, the distribution of topics for the entire corpus is drawn from a dirichlet distribution for which the same parametric conditions are met as for Θ. The only difference is that this time the parameter influences each topic's distribution over words and not the document's distribution over topics.

In our example the inferred β looks as follows:

- Topic 1: series - 5.92%, episode - 5.06%, television - 2.77%, tv - 2.74%, ...
- ...
- Topic 5: film - 12.10%, films - 2.49%, movie - 1.84%, directed - 1.53%, ...
- Topic 16: game - 7.65%, player - 2.47%, games - 2.32%, released - 1.15%, ...
- ...
- Topic 25: michael - 2.00%, paul - 1.55%, tom - 1.27%, mike - 1.25%, ...
- ...

For each topic the words are ordered by their probability within that topic.

Now we will infer Θ for the following short document:
"This is a wonderful action packed movie with Steven Seagal. The special effects are awesome."

This will result in $\Theta = [(1, 1.82\%), ..., (5, 2.59\%), ..., (16, 2.59\%), ..., (25, 2.20\%)]$. Each of the Θ-values is the topic number with their assigned probability within the document. Only the most likely topics are shown here.

We can see that the document is mainly about topic 5 and 16 followed by topic 25 and 1. Topic 5 and 16 can be described as topics about movies, tv shows and television. Topic 16 seems to be about games and topic 25 about names. Even though we have provided a very short document, the inferred topic distribution seems to represent a good thematic fit to the document. However, the low probabilities for each topic as well as the less suitable topic of games show that it is difficult to infer topic distributions from short text documents and that the model's inference process is only an approximation of reversing the generative process of document creation.

The remaining question is how the latent variables (i.e. Θ_d for every document d, β_k for every topic k, $z_{d,n}$ for every word n within document d) are inferred from the collection of documents where only the words for each document are observed. In other words, we want to answer the question of how the probabilities in the above stated example were computed and therefore how the assumed generative process of document creation is reversed.

It turns out that the conditional distribution of the topic structure given the observed documents is intractable to compute. This is because one would need to calculate the probability of observing the given documents under any combination of the hidden variables which is exponentially large. Thus, the goal is to approximate it. There are two types of approximation algorithms: Sampling-based algorithms and variational algorithms [6].

In the example above, the Mallet Toolkit [38] was used which is an implementation of the *Gibbs sampling* approach. Therefore, we will give a brief introduction to this sample-based algorithm.

For this algorithm, the topic-word-distributions β and the topic distributions Θ for each document are marginalized out at first. They are approximated once the word-topic assignments $z_{d,n}$ for every document d and every word n have been allocated. The goal of this allocation is to assign each word to a single topic, i.e. the topic that the word is assumed to be drawn from in the generative process. The entire corpus is sequentially processed for a fixed number of iterations. In the first iteration every word is assigned to a random topic from the predefined number of topics. In each of the following iterations, the probability of assigning the current word to each topic is calculated taking the topic assignments of every other word into account. Then, a topic assignment is drawn from this probability distribution.

The probability of a topic assignment z_i of a word instance i to a topic t within a document d is calculated in the following way:

$$p(z_i = t \mid \boldsymbol{z}_{-i}, w_i, d_i, *) = p(w \mid t) \times p(t \mid d) \tag{2.4}$$

The asterisk in the conditional probability on the left represents all the other given information (documents, words, parameters). Note that in this equation all other topic assignments $\boldsymbol{z}_{-i}$ are assumed to be correct. As mentioned before, β_t and Θ_d are not yet determined. $p(w \mid t)$ is the proportion of assignments to topic t that come from w. For $p(t \mid d)$ we take the proportion of words in document d that are currently assigned to topic t. In fact, once the sampling is done and all the word-topic-assignments are fixed, the final β and Θ distributions are computed in the same way. The conditional probabilities $p(w \mid t)$ and $p(t \mid d)$ are smoothed using the Dirichlet parameters α and η from the generative process that need to be predefined. The parameters have the same effect as stated earlier for the Dirichlet distributions and they especially serve to avoid zero-probability-assignments to topics or words respectively. This is because in the LDA model it is assumed that every word has a probability greater than 0 of appearing in a topic which is also true for every topic in a document.

The equation above leads to topic assignments that depend on the dominance of a topic within a document and on the likelihood of drawing a word from that topic [57]. As topics are reassigned for every word during each iteration, they become more coherent over time and documents become more focused on certain topics. The algorithm stops after a fixed number of iterations that have been run on the corpus. One should ensure that the number is high enough for the topic assignments to converge to a final state.

2.2.2 Word2Vec and Semantic Clustering

In this section, we will investigate on the usage of semantic clustering methods for keyphrase extraction. Unlike topic models from the previous section, semantic clustering methods are no *probabilistic mixed membership models*. This means that a phrase does not belong to several semantic topics at a certain probability. Instead, they are mapped to a fixed location in a semantic space so that clusters can be formed around them. Semantic clustering has been employed in the task of keyphrase extraction in order to improve on the accuracy of frequency-based extraction methods. Plas et al. [59] used manually built lexical resources such as WordNet, which improved results, yet reduced the method's flexibility since the lexical resources had to fit the task at hand. F. Liu et al. [34] showed that unsupervised methods for semantic clustering could improve results produced by *tf-idf*-based algorithms when being applied to direct ASR output from human conversations. However, the clustering was less effective than expected which, according to Liu et al., was probably due to the mediocre clustering accuracy. Z. Liu

et al. (2010) followed up on this research, however applying clustering techniques to scientific articles. They used these methods in order to ensure that the extracted keyphrases had a decent topical coverage of the document. The topics here did not relate to word-probability distributions generated by a topic model but to the semantic clusters of the document. Liu et al. compared different clustering techniques and similarity metrics that were mainly based on *tf-idf* values and word co-occurrence counts.

Their work brought up a few insights that are relevant to the research that will be conducted with respect to this thesis. In Liu et al.'s experiments, semantic similarity metrics that were trained on a large dataset (Wikipedia) did not yield much better results than similarity metrics derived from a single document's word co-occurrences. Liu et al. concluded that this was due to the fact that previously trained models did not capture the new document's semantic information well enough. Yet, it has to be noted that their experiments dealt with scientific articles only. These documents were very domain-specific in that they tended to introduce new terms or put existing terms in a new context. Therefore, it remains to be shown how previously trained semantic models perform when being applied to transcripts of conventional human speech. Moreover, a novel model called Word2Vec was introduced lately [42]. This model captures semantic as well as syntactic relations among words and phrases. It has gained a lot of traction in the research community since it is able to compute state-of-the-art word embeddings in a semantic vector space [16]. This is why we will investigate on the applicability of Word2Vec for semantic word clustering and keyphrase extraction.

Word2Vec

Word2Vec was introduced by Mikolov et al. in 2013 [41, 42]. It is based on the idea that words can be represented as dense numerical vectors embedded in a low-dimensional space. The transformations which produce these mappings are called *word embeddings*. These word embeddings have several advantages over a textual representation of words as they can be used for computational calculations and comparisons. To capture the idea behind such word representations, we will provide a simple example in which we see that we can also use LDA in order to obtain word embeddings.

Supposing that we trained an LDA model with 100 topics on a corpus of text documents. The *Gibbs Sampling* algorithm would assign one topic to every single word appearing in any document of the corpus. This way each word could be characterized by its topical distribution, i.e. its proportion of topic assignments to one particular topic. Now we could embed the word in a 100-dimensional vector space where each coordinate axis represents one of the topics. This would be a semantic representation of a word that depends on the probabilistic topic distribution of the entire corpus. Describing a word using its characteristics (here its topic proportions) is called a *distributed representation* of the word [25].

A word's characteristics in turn can be derived from the context in which the word appears. There is a famous quote by the British linguist John Rupert Firth that states: "You shall know a word by the company it keeps" [14]. This statement implies that words in similar contexts have similar meanings, which is commonly known as the distributional hypothesis [20]. In LDA for example the context is defined as a document that is assumed to consist of latent topics (concepts). Therefore, words that appear in similar documents (contexts) tend to be assigned to similar topics and hence have a similar semantic representation.

In fact, we can define a word's context in many different ways depending on the task at hand. If we are not interested in topical proportions of documents but solely in the semantic or syntactic relations among single words or phrases, we can narrow a word's context down by only looking at the words that directly surround the given word. This way we could analyse a text corpus by counting the word's co-occurrences with other words if they appear inside the same *context window* of limited number of words. Semantic similarities among words could then be derived by comparing the co-occurrence counts of different words. This is indeed the basic idea that many modern semantic word similarity models are based on. The semantic clustering methods using word co-occurrence counts that were discussed in section 2.2.2 also follow this idea. If we were to analyse a large text corpus however, we would notice

that a word only co-occurs with a very limited number of other words. A vector containing a word's co-occurrence counts would therefore be very sparse, i.e. contain a lot of zeros. Hence, the question arises whether this co-occurrence vector can be represented in a denser way while still maintaining most of its relevant information for semantic comparison. A widely used technique to achieve this goal is called *Singular Value Decomposition (SVD)* that became very popular in the research community when Deerwester et al. used it in their Latent Semantic Analysis (LSA) algorithm [11]. Therefore, LSA and even LDA can be considered a dimensionality reduction method given the fact that both methods reduce a high dimensional vector space (word co-occurrence matrix or word-document matrix) into a low dimensional representation of latent concepts. The "low" dimensionality of a word vector in this context can be defined by the algorithms' parameters, i.e. the number of latent concepts. These low dimensional vectors are very dense as opposed to sparse and their advantage over a high dimensional representation is that a similarity comparison among them becomes more computationally convenient [30]. One major drawback of SVD (and therefore LSA) is that the word co-occurrence matrix grows quadratically with the size of the vocabulary of the corpus. The factorisation of this matrix that is required for SVD can therefore become computationally expensive when being applied to very large corpora [29].

Word2Vec follows a slightly different approach in learning low dimensional distributed representations of words. Instead of (explicitly) decomposing a word co-occurrence matrix, it learns the dense word vectors directly while processing the text corpus. A word's context in Word2Vec is still defined as a window of a limited number of words that surround the given word. However, this method does not simply count the word co-occurrences. Instead, it trains a neural network's weights so as to maximize the likelihood of predicting a word's surrounding words, which results in the so-called *skip-gram* approach [16]. Learning a word's distributed representation using neural networks was introduced by Bengio et al. [4], which is why we consider Word2Vec a further development of this line of research. An introduction to neural networks is given in Section 9.1.

The dense vectors trained with Word2Vec have proven to capture semantic and syntactic similarities among words quite well. Mikolov et al. achieved a particularly high precision on word analogy tasks, which can be illustrated by applying algebraic operations to the word vectors.
For instance, $vector(\text{"}King\text{"}) - vector(\text{"}Man\text{"}) + vector(\text{"}Woman\text{"})$ results in a vector that is closest to $vector(\text{"}Queen\text{"})$ provided that the model was trained on sufficient data [42]. From this example we can infer that the similarity between *King* and *Man* must be similar to the similarity of *Queen* and *Woman*.

At first sight, the neural network based approach seems to be very different from the traditional SVD-based methods. Yet, it was shown that the very successful skip-gram model along with its *negative sampling* algorithm (SGNS) implicitly factorises a reweighted version of a word co-occurrence matrix that is shifted by a constant. Moreover, it was proven that similar performing results can be achieved with improved traditional methods. A great deal of the accuracy that Word2Vec achieves in word similarity tasks can be attributed to its set of hyperparameters (performance tweaks that are part of the algorithm but are not explicitly labelled as parameters) [31, 32]. Nonetheless, Word2Vec has definitely set new performance standards in word similarity and analogy tasks. Furthermore, the SGNS model can be efficiently trained on very large datasets, which makes it a state-of-the-art technique for word embeddings [30]. Its popularity may also be due to the fact that pretrained word vectors have been made publicly available. Therefore, it is very easy to try out this model.

The major question concerning Word2Vec is how the word vectors used for semantic and syntactic comparisons are learned, i.e. where the word embeddings come from.

Mikolov et al. presented two approaches for learning word vectors of which one is known to yield the best results. Therefore, we will only introduce the *continuous skip-gram-model* which outperformed the *continuous bag-of-words model* with respect to semantic precision in [42].

For each word inside the vocabulary of size V the skip-gram model considers a frame of C surrounding words that may accompany the word at a given position. This frame will change throughout the word's occurrences within the corpus but one underlying assumption of the skip-gram model is that there are

certain recurring patterns in the surrounding words. The patterns determine a word's semantic as well as syntactic meaning as it is stated by the distributional hypothesis. The center word is used as the network's input value and the probabilities of each word inside the vocabulary to appear in the input word's context window are the output values. The given input word is one-hot encoded meaning that the word is represented by a vector of dimensionality V containing only zeros except a value of one at the word's index inside the vocabulary. An illustration of the neural network projecting an input word as a V-dimensional one-hot encoded vector to its context probabilities is shown in Figure 2.4. We can see that the network contains only one hidden layer with N nodes. This layer is considered a projection because its dimensionality N determines the dimensionality of the learned word vectors by Word2Vec. As it is the case for backpropagation nets, our goal is to find the network's optimal set of weights which in our case are denoted as W_{VxN} and W'_{NxV}. In fact, these weight matrices already contain N-dimensional vector representations of the entire vocabulary that are initially set to random values. For the time being we have two different of such representations for each word. W_{VxN} contains the *input representations* as its rows and W'_{NxV} contains the *output representations* as its columns. The output representations serve as the word vectors when words are part of the context, i.e. surround the input word. In order to obtain the final word vectors we have several options such as taking the sum of the matrices W and W' or concatenating each input and output vector, which would result in a $Nx2$ dimensional representation for every word. In practice, the input weights between the input and hidden layer are often used as final word vectors once the model is trained. It is important to note however, that during the training process these final weights strongly depend on W'_{NxW}, which is why both matrices have to be trained [53].

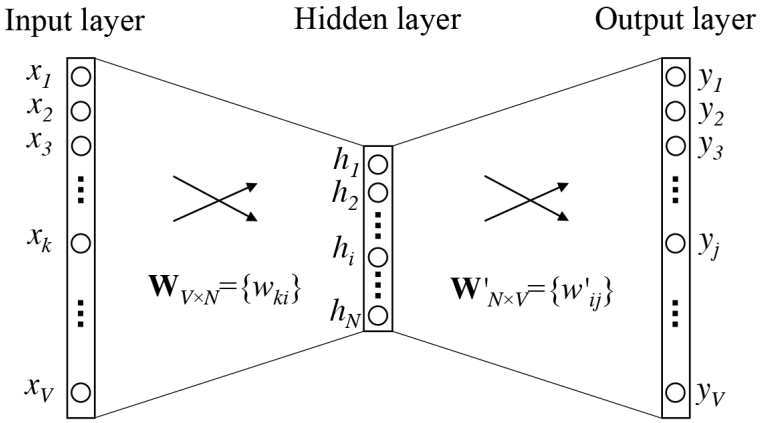

Figure 2.4: The neural network that is used to train the word vectors (weight matrices) inside Word2Vec. Taken from [53].

Since the input vector is one-hot encoded, the input word's weights inside the W_{VxN} matrix are projected to the neurons in the hidden layer. Therefore, the hidden layer simply becomes the input word's row inside the W_{VxN} matrix. Unlike in conventional backpropagation nets, the hidden layer does not apply a logistic activation function to the weighted sum of its inputs. Instead, the values are passed on to the output layer without any modification, which is why the hidden layer is called *linear* in this case. In fact, the hidden layer and therefore the input word's weights are broadcast to each node in the output layer. This essentially means that each neuron j in the output layer receives the scalar product of the input word's weight vector v_{w_I} and the neuron's corresponding output vector v'_{w_j}, which is the column of the jth (context) word in the W'_{NxV} matrix. Therefore, the input vector of the output layer consists of the scalar products of the input word vector and each output (potential context) vector. Since similar words in Word2Vec should have similar vectors, the goal of the objective function is to maximize the scalar product of v_{w_I} and v'_{w_j} in case the corresponding words are actually observed to occur in the same context. This way we can use the cosine similarity measure that relies on the scalar product to compare

the similarity among word vectors. At the same time, dissimilar words should have dissimilar word vectors. We can restate our objective by converting the scalar products among words into probabilities using the *softmax* function.

$$p(w_j \mid w_I) = \frac{\exp(v'^{\intercal}_{w_j} v_{w_I})}{\sum_{i=1}^{V} \exp(v'^{\intercal}_{i} v_{w_I})} \tag{2.5}$$

Given an input word w_I we can now compute the probability of a word w_j to appear in the same context. v'_i is the ith column of W' and V is the size of the entire vocabulary. The probabilities for all potential context words and a given input word sum up to one. Equation 2.5 computes a single probability of the underlying distribution. Thus, the trained neural network would produce the probability distribution over context words for a given input word. A difference between the neural network used by Word2Vec and conventional backpropagation nets is that we try to maximize the probability of the actual context words to appear around the input word opposed to minimizing the difference between a desired and an actual output value. Therefore, the weights are adjusted in order to increase the probability of observing the actual context words compared to the rest of the words in the vocabulary.

According to [41], the formal notation of the objective function looks as follows. Let $w_1, w_2, \ldots, w_T$ be the training words of the corpus.

$$\underset{W,W'}{\operatorname{argmax}} \frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j} \mid w_t) \tag{2.6}$$

$2c$ is equal to the number of context words surrounding the current word w_t. A larger context leads to more precise results at a higher computational cost. The goal is thus to find the set of optimal weights W and W' by maximizing the equation using gradient ascent. Since the gradient of this term will still include the softmax function, its computation becomes expensive when dealing with a large vocabulary. Word2Vec therefore uses several optimizations in order to overcome this issue such as *Hierarchical Softmax*. There is also the *Negative Sampling* approach that solves the problem by optimizing a different objective function. However, the explanation of these optimizations is beyond the scope of this introductory description of Word2Vec.

Mikolov et al. have also shown that it is possible to produce vector representations for word phrases using Word2Vec, which will help us in the extraction of keyphrases. These phrases are trained just as normal word vectors by recognizing word combinations that form phrases beforehand and treating them as single tokens. The distributed representations of word phrases can be compared to other phrases and single words without any restrictions.

In order to measure the semantic similarity between two word vectors the cosine similarity measure has proven to to be very effective. However, if we normalize the vectors to a unit length we can also apply different measures such as the euclidean distance. This enables us to directly find word clusters in the vector space using methods like K-Means clustering. Which of these methods is the most accurate one depends on the task at hand and has to be explored for each application [43].

Since Word2Vec was shown to be very accurate in word similarity tasks, we are keen to find out whether we can use it for word clusterings to improve our keyphrase extraction method. In the past, clustering approaches were not considered very successful in keyphrase extraction as discussed in the beginning of Section 2.2.2, which is why we want to see if Word2Vec can produce better clustering results. Moreover, we are interested in how comparable our keyphrase extraction method will be with LDA-based methods.

2.3 Information Retrieval

The last step of our Ambient Search process is to retrieve document recommendations based on a previously extracted set of keyphrases. We will use the keyphrases to automatically formulate a search query against a collection of Wikipedia articles. Whenever speaking of finding unstructured material from large collections of data that satisfies potential information needs, we are in the domain of *Information Retrieval (IR)*. The term *unstructured* typically refers to the information's readability for computers [36]. For instance, the information that is contained in a full-text Wikipedia article is rather difficult for a computer to interpret.

The most prominent examples of IR systems are search engines for the world wide web like Google[4] or Bing[5]. Those search engines operate on a very large scale of documents (websites). But there are also institutional IR systems that operate on a smaller and more domain-specific set of documents. For the Ambient Search implementation we restricted our search to Wikipedia articles and will therefore use an IR system that is suitable for this domain.

The users of an IR system are usually expected to express their information needs in a search query consisting of a small set of search terms. The IR system then tries to find the subset of documents that best fits the search query and hopefully also satisfies a user's information needs. There are numerous difficulties that an IR system needs to deal with when attempting to solve this problem. Human language can be very ambiguous, especially when information needs are expressed in a short search query. If a user searches for the term *Jaguar* it is very difficult for the IR system to find out whether the user wants to find documents related to animals or cars since the search query does not contain any disambiguating information. Moreover, the IR system needs to decide on the relevance of each document inside the entire collection with respect to a search query in order to determine a document's position inside the search results. Therefore, the notion of a document's *relevance* is the central aspect of information retrieval [2].

In Ambient Search, search queries are not expressed by the user. Instead, they are constructed based on an automatically extracted set of keyphrases. Therefore, we speak of *implicit queries* that highly rely on the performance of prior algorithms that we applied to our speech fragments. Thus, our system tries to predict a user's potential information needs when listening to human speech based on the assumption that they can be expressed as keyphrases inside a speech fragment. Of course, this assumption does not always hold for any user, but the goal of Ambient Search is to provide a set of documents that is likely to cover at least some of a user's potential information needs.

In a full-text search, a document's relevance to a given search query is usually determined by the matching scores of the document with respect to each term inside the query. A very popular measure for computing document-term matching scores is *tf-idf*. We have already introduced this measure in the domain of keyphrase extraction in Section 2.2. According to this measure, a document that contains a rare search term at a high frequency, receives a high matching score for this term. In fact, we can even treat the search query as a short document and compute its *tf-idf* scores in order to retrieve documents with similar scores. The similarity among documents including search queries can be obtained using the cosine similarity metric. The more similar a document is to a search query the higher will be its position in the search results [36].

Before any comparisons can be made however, each document needs to be indexed by the IR system in order to provide a faster and more convenient computation of matching scores. An index is a matrix that shows the relation between documents and the terms they may contain. These relations can be expressed as simple counts, the position at which the terms appear inside the document or a metric such as *tf-idf*. Moreover, an index provides the possibility to quickly filter documents for specific terms and logical combinations among them [2].

[4] https://www.google.com/
[5] https://www.bing.com/

3 Related Work

Before presenting our Ambient Search implementation, we want to give a brief overview over the related research that has been conducted in this area. Most of it uses the term *just-in-time information retrieval* that was introduced in 2000 by Rhodes and Maes [51].

However, the first approaches in this direction took place even before that time. For instance, Hart and Graham presented a prototype application of a query-free search system called *Fixit* in 1997 [21]. This system was meant to support service technicians troubleshoot complex copier issues by automatically providing them with instructional documents. This approach was strongly restricted to a certain domain as the technicians had to choose symptoms for their problems from a predefined list and were provided with a list of potential causes for the problem along with their probabilities. The system obtained these results by automatically building queries that were matched against a database of technical documents. Despite the simplicity of the system and the restrictions about the domain, Fixit included some of the ideas of Ambient Search since the database queries were constructed and run automatically in the background.

Rhodes and Maes define the term *just-in-time information retrieval agents* as "a class of software agents that proactively present information based on a person's context in an easily accessible and non-intrusive manner" [51]. The context is a very broad term in this regard that could relate to any information that is part of the person's environment such as e-mails or documents that the person is writing or reading. One of the most essential features of the agent is that it proactively requests information for the user without requiring any interaction with them. This definition comes very close to what we have defined as an Ambient Search system, although our system is restricted to an auditive context only. Rhodes and Maes also presented three implementations of the previously described agents. They differed in terms of the context that they were built for. The most interesting implementation for us is called *Jimminy* since it was supposed to use sensor devices from a computer to capture surrounding information related to the user. At that time, automatic speech recognition systems had not been as sophisticated as they are today but one could already imagine connecting Jimminy to them in order to recommend relevant documents based on audio transcripts. Extended *tf-idf* scores were used for the comparison of documents. The keyword extraction was supported by pattern recognition.

In 2004, Dumais et al. presented their idea of an *implicit query (IQ) system* as part of Microsoft's ongoing research activity [12]. This system was intended to run background search queries to support the user with additional information when writing an e-mail. The implicit queries were run online, i.e. while the user was typing, which is a challenging task since in such a scenario the queries have to adapt to a constantly changing context. This is also the case for our Ambient Search system as it runs implicit queries during speech recognition. Dumais et al. used *tf-idf* scores for the keyword extraction as well.

Just-in-Time Information Retrieval was applied to broader physical contexts including visual and audio monitoring a while later. The *FAME interactive space* developed by Metze et al. was a multi-modal system that served to interact with humans in multiple communication modes in order to suggest additional information to them [39]. Although FAME already supported speech recognition and voice commands, it only listened to conversations for a longer period of time when it came to guessing the conversation's topic. Unlike Ambient Search, documents were only suggested upon command. The topic detection was strongly supervised as keywords were matched to a predefined set of topics. However, the online detection of topics in a constant stream of audio data is an idea that is also relevant to Ambient Search.

With the improvement of speech recognition systems, a lot of research was carried out regarding the extraction of information from audio transcripts. This has been particularly relevant in the field of robotics that deals with human-computer-interaction. According to Kraft et al. for instance, domestic robots were not only expected to execute human commands but also to understand surrounding human-human conversations and to learn from them [28]. Kraft et al. developed a method to classify conversational transcripts according to a supervised collection of concepts by transforming the transcripts

into a *tf-idf* representation. This way robots could automatically decide whether or not a conversation was relevant to them, which is also an important privacy issue in this domain.

In 2011, Chella et al. published a work on how robots could use document classification algorithms in order to respond emotionally towards human statements [9]. These statements were ASR transcripts that were classified into emotional categories by generating a score for several basic emotional states using *Latent Semantic Analsysis (LSA)*.

Consequently, Harwath and Hazen employed *Probabilistic Latent Semantic Analysis (PLSA)* in summarization tasks of audio transcripts to show that topic models were able to boost existing algorithms' performance [22]. Topic models could especially ensure that off-topic conversation fragments were neglected for the summarization. Since summarization is a means to extract relevant information from text, these insights are of great importance to Ambient Search as the extracted information can be passed on to following document retrieval tasks.

A state-of-the-art just-in-time information retrieval system

In the following, we will examine a state-of-the-art just-in-time information retrieval system developed by Habibi and Popescu-Belis that is built on the previously described insights (for convenience reasons we will refer to it as JITIRS-HP[6]) [19]. Since this system will also be used as a benchmark when evaluating our own implementation, we will describe it in a more detailed manner than the previously mentioned systems.

JITIRS-HP includes everything that is related to our definition of Ambient Search. It extracts relevant keywords from ASR transcripts in order to use them for information retrieval. Thus, when providing an ASR transcript, the system returns a set of relevant documents that contain additional background information about the contents of the transcript. For instance, this information can be retrieved from Wikipedia and users can investigate on them further if they are interested. For obtaining the documents, the extracted keywords are clustered into topic-related subsets so that for each subset a separate implicit query is built. According to Habibi and Popescu-Belis, this leads to more diverse queries that are less prone to include ASR errors and cover the conversational transcript's topics better. Hence, there should be a higher chance of returning relevant information to the user in at least one of the queries. JITIRS-HP is an extension of the already developed Automatic Content Linking Device (ACLD) [46], a software that monitors meeting conversations in order to return relevant documents from local repositories or from the internet. For this purpose, the ACLD uses a speech recognition software and extracts relevant keywords from the transcripts in regular time intervals. The extracted keywords are then used for either a keyword-based or a semantic search. The semantic search compares the extracted information to documents that may be relevant using a semantic similarity metric and a random-walk-algorithm, which results in a high matching accuracy but also in high computational costs. These high computational costs cause great problems when searching large collections of documents which has motivated Habibi and Popescu-Belis to improve on the keyword-extraction scheme to model users' information needs better and to also develop a novel technique for building search queries for document suggestions. These novel methods are all implemented in JITIRS-HP and will be presented in the following.

JITIRS-HP uses a diverse keyword extraction algorithm that first converts the speech fragment into a topical representation using LDA. This way topical weights for the fragment can be derived from the words and their probability to appear in one of the previously trained topics. The topical weights represent each topic's importance for the fragment. The goal of the keyword extraction algorithm is to extract a fixed number of keywords with maximal topical coverage as well as diversity. For this purpose, a potential subset of extracted keywords is scored using a *reward function*. This function is inspired by a class of submodular functions for document summarization [33] that have proven to be very effective for this task. The reward function is characterized by finding the best trade-off between topical coverage (relevance) and diversity (low redundancy) of extracted keyphrases. For the actual implementation a

[6] Just-in-Time Information Retrieval System by Habibi and Popescu-Belis

greedy algorithm is used to approximate a solution to this problem with an acceptable computational cost. In order to maintain the topical diversity of the extracted keywords for constructing search queries, the keywords are assigned to clusters based on their topical proximity.

Habibi and Popescu-Belis have shown in another publication that running separate topically diverse search queries and merging their returned lists of documents outperforms running a single topically mixed query [18]. In this research paper, they also present their method of merging the returned lists of documents produced by each of the queries. The goal of the merged list is once again to maximize the coverage of topics presented in the transcribed speech fragment.

At first, each query's importance is weighted according to its topical similarity compared to the entire keyword set which is a good representation of the transcript's topics. As for the keyword extraction problem, a submodular reward function is defined that favours diversity among suggested documents. This function is used to rate a subset of documents from all the documents that are returned. The reward function is based on the documents' topical similarity to the entire set of extracted keywords, i.e. the topical approximation of the conversation fragment. In the actual implementation a greedy algorithm is used that keeps on adding new documents to the extracted subset of documents as long as the subset's score improves. The score depends on the reward function as well as on the query weights and the topical diversity of included documents. The returned set of documents by the algorithm is supposed to be the best fit between relevance to the transcript and topical diversity.

All of the provided algorithms in JITIRS-HP are evaluated using different corpora of conversational transcripts and it is shown that they outperform comparable methods without a diversity constraint. This is why we consider JITIRS-HP a state-of-the-art method in the area of just-in-time information retrieval tasks.

4 Speech and Text Corpora used in our Implementation

The following text and speech corpora were used in our Ambient Search implementation. The details of the actual implementation will be provided in the next section.

TED-LIUM Corpus

The TED-LIUM corpus was published by the university of Le Mans in France and contains 118 hours of speech from TED talks[7] along with their aligned transcripts [54]. The data was directly extracted from the freely available video talks that can be watched on the TED website. In order to improve the corpus' suitability for the training of acoustic models, the data went through an extensive filtering process. The speech was collected from 774 talks given by 666 unique speakers. Taken together these talks contain 2.56 million words.

TED talks are public speeches that are mostly held in English and cover a wide range of topics. The talks usually contain well articulated English language although they may not always be delivered by native speakers.

Switchboard Corpus

We use a transcribed version of the Switchboard corpus that is part of the American National Corpus (ANC)[8]. The original Switchboard corpus contains audio recordings from spontaneous telephone conversations. Our textual version consists of transcripts from 2320 of these conversations with an average length of 6 minutes. Combining these transcripts leads to a text corpus with about 3 million words of conversational English language.

Subset of the standard English Wikipedia

We use a subset of the English Wikipedia as our major textual training corpus since it contains a large number of words that cover a very diverse vocabulary. It was obtained from the Wikimedia website [9] and originally consisted of around 290,000 articles. After pruning articles with less than 50 words, we are left with around 110,000 articles that altogether contain around 80 million words with about one million unique text tokens.

Simple English Wikipedia

For information retrieval, we do not use the subset of the standard English Wikipedia mentioned above because that subset contains a very limited range of documents that does not cover all the major topics contained in the original English Wikipedia. Instead, we use a dump of the *Simple English Wikipedia*[10], a simplified version of the original English Wikipedia. This version requires articles to be written in a simplified English language. However, the major reason why we use it is not its language requirement, but simply its smaller size compared to the original Wikipedia. This version contains only around 120,000 articles compared to the original English Wikipedia with around 5 million articles. Therefore, this dump can be indexed at a lower computational cost while still covering the most relevant general articles.

[7] https://www.ted.com/
[8] http://www.anc.org/data/oanc/contents/#switchboard
[9] https://dumps.wikimedia.org/enwiki/
[10] https://simple.wikipedia.org

5 Implementation

In the following, we will present our implementation of an *Ambient Search* system. The system processes any kind of speech input and automatically provides additional background information by recommending relevant documents to the system's users. These document recommendations are found using implicit search queries that are based on the information needs that the user of the system or a participant of a conversation may have. There are multiple scenarios in which we imagine such a system to be helpful. For instance, it could be used to provide additional background information when watching a talk show or the news on television. Furthermore, it could be used for lectures and presentations. In fact, we will evaluate our system in the next section using fragments of TED talks.

Our main goal was to optimize an already existing Ambient Search system[11] with respect to keyphrase extraction and information retrieval. Therefore, our approaches are embedded into the current system's web application that displays relevant Wikipedia articles to users while processing their speech input. It is important to note that although our system will be evaluated on previously recorded speech fragments, the actual implementation also supports online, i.e. continuous, and even just-in-time processing of speech input.

The underlying Ambient Search process consists of three steps:

1. Automatic speech recognition and generation of transcription hypothesis
2. Extraction of relevant keyphrases
3. Formulation of search queries and filtering search results

The implementation details of each of these steps will be explained in the following subsections. The focus of this thesis was to develop novel methods for keyphrase extraction and to evaluate their performance compared to an existing state-of-the-art approach.

5.1 Automatic Speech Recognition

We use a distributed client-server based speech-to-text system that is based on the open-source Kaldi speech recognition toolkit [47]. The system supports real-time speech recognition and gives constant feedback to its users by displaying partially recognized utterances while processing their speech. The system was introduced in [1] and its implementation is freely available[12].

The distributed client-server based approach provides the possibility to separate the tasks of speech recording (client) and speech decoding (server) such that they may even take place on different machines. The client can therefore be implemented in a variety of programming languages and does not require bigger amounts of computational resources. In our setup, the client is implemented in Python and communicates with the speech decoding server over a websocket connection. The (partial) utterances that are returned by the decoding server at any given moment are displayed to the user in a web-based GUI. It is not only possible to send direct speech input to the server but also audio files can be sent over the websocket connection. In fact, for the evaluation of the system we used mp3-files of TED talks that were streamed to the decoding server via the client.

The server carries out the actual decoding of speech and therefore transcribes the streamed audio signal. It can handle multiple concurrent client connections as long as there is at least one decoding process available for each incoming audio signal. There can be an arbitrary number of decoding processes that are run in parallel on any number of (remote) machines as long as they do not exceed the respective server's resources. A decoding process in our case handles the decoding using the *KaldiNNet2OnlineDecoder* plugin[13].

[11] The system is publicly available on Github: https://github.com/bmilde/ambientsearch
[12] http://github.com/alumae/kaldi-gstreamer-server
[13] https://github.com/alumae/gst-kaldi-nnet2-online

This plugin uses pretrained acoustic models that were built using a combination of Hidden Markov Models (HMM) and Deep Neural Networks (DNN)[14]. Therefore, this approach adheres to the most recently introduced paradigm shift that favours DNNs over the formerly used Gaussian Mixture Models (GMM) when it comes to defining the probabilistic interface between the HMM and the feature vectors extracted from the speech signal.

For our experiments, we use acoustic models that were trained on the TED-LIUM corpus consisting of mp3-files and transcripts of TED talks. Therefore, the models are highly suitable for processing such kind of talks, which will be done in our evaluation. For the evaluation we used very recent talks and therefore took care that they were not part of the training corpus.

The pretrained acoustic models came bundled with a generic English language model provided by speechmatics[15] (formerly known as Cantab Research). Taken together, the language and the acoustic model provide the database that the speech decoding plugin works with.

5.2 Extraction of relevant keyphrases

In order for our Ambient Search system to suggest relevant documents to a user, it must find relevant words and phrases inside the speech transcripts first. We call these relevant phrases *keyphrases* because they contain characteristic information about the underlying transcript and they should also cover a user's information needs. If for instance, we were to listen to a speech or a news broadcast about climate change, we would expect relevant keyphrases to include words such as *energy, economy, green house gases, fossil fuels, temperature* or *glabal warming*. These phrases are labels for large amounts of background knowledge that a user may or may not know about. Therefore, a user may feel the *need* to learn more about these terms. Thus, when speaking about keyphrase extraction, our goal is to find computational techniques that are able to automatically determine such phrases without human intervention. Moreover, we would like to train our algorithms in an unsupervised way, i.e. without any prelabelled training data. This makes the training process much more cost-efficient and allows us to flexibly employ the algorithms in multiple domains. When addressing this problem, one encounters several major challenges.

We need to find a measure that the algorithm can use in order to determine whether a phrase is relevant for a speech fragment or not. This measure can be described as a *topical score* that is high for phrases matching the major topics of the fragment and low for off-topic phrases. This measure is particularly important in the domain of keyphrase extraction from ASR transcripts as these may contain numerous errors that should not be considered relevant. Moreover, spoken language differs from written text in that it tends to contain more speech disfluencies and more informal words that do not describe a topic as precisely as a technical term. Therefore, another problem that we need to address is that we need to find terms that are precise as well as informative. This is even more important when our task is to extract a limited number of keyphrases. In that case we would not want our keyphrases to contain words such as *the, and, do* or *take* because they are far too general.

We have developed our keyphrase extraction technique based on several insights that were brought up in related work (discussed in Section 2.2). We will always refer to these when explaining our algorithms. A major novelty regarding our approach is the incorporation of Word2Vec that serves to measure how related a keyphrase is to major ideas of the speech fragment. Furthermore, it is used to divide the fragment into several topically related clusters.

In the following, we will explain our method with respect to its underlying processing stages.

[14] http://kaldi-asr.org/doc/dnn2.html
[15] https://www.speechmatics.com

Text Preprocessing

Human language contains numerous words that serve to connect essential information following a language's grammatical features. However, we are not interested in producing syntactically correct language nor do we intend to find relevant information by analysing the syntactic style of speech. Instead, we treat the uttered terms as an independent set of text tokens, which is a simplifying assumption but it reduces the problem's complexity tremendously. Therefore, we need to filter out words that are considered irrelevant in this context. This task is generally described as *stopwords removal*. A stopword is a word that is not related to any text in particular. Instead, it is a very general word that could appear in any kind of document. Hence, we intend to filter out words such as *the, I, we, many* or *because*. This can be done by comparing a text to a predefined list of stopwords. In our case we employ a tf-idf model that is preferably trained on a corpus that is characteristic of the target domain. This enables us to define an idf-threshold that determines whether words are too common or not. We have trained the filtering tf-idf model on the *Switchboard conversational corpus* using the gensim toolkit [49] since spoken language has a different distribution of common words from written texts. The advantage over a list of stopwords is that such a model does not need to be manually maintained, which allows us to flexibly switch domains by retraining the model on different corpora.

Moreover, we use a pretrained parts-of-speech tagging model provided by the NLTK toolkit [5] that enables us to automatically find words that match certain parts of speech. Following the example of Liu et al. [35] and several others [59, 34, 27], we only focus on the extraction of nouns and adjectives, which has proven to improve results. However, we do not incorporate pattern matching in this regard since we have noticed that the parts-of-speech tagging model has a much higher error rate when being applied to ASR transcripts which is why pattern matching for noun phrases did not improve results in our tests.

However, we connect noun phrases using a pretrained DRUID dictionary [52]. DRUID finds multiword expressions by combining a phrase's *uniqueness* measure with a measure for its *incompleteness*. Uniqueness in this context is based on the assumption that multiword expressions can often be substituted by a single word without changing a sentence's meaning. Therefore, the score represents the proportion of a phrase's similar terms that consist of a single word and appear in the same context. The incompleteness measure serves to punish incomplete terms in that it counts the number of times that the same words appear next to a phrase. If a single word has a high proportion of these counts, the word is considered a likely candidate to complete the multiword phrase. Therefore, the incompleteness score for the corresponding phrase would be high. When combining both measures, we obtain the DRUID score which is a state-of-the-art unsupervised method for finding multiword expressions in text corpora. Using DRUID we can find multiword expressions among our filtered nouns and adjectives if they occur in the pretrained dictionary. In our experiments we obtained the best results when comparing potential noun phrases to the longest matching multiword expression in the dictionary above a threshold score. The dictionary itself was trained on a dump of the English Wikipedia.

Lastly, we apply the WordNet[16] lemmatizer from the NLTK toolkit to all the extracted words and phrases, which reduces inflected forms of terms such as *dogs, doing, took* to their lemmas *dog, do, take*. Without lemmatization, our algorithms would treat inflected forms of a term as different words, which could lead to a set of extracted keyphrases with a lot of redundant terms. Lemmatization decreases the likelihood of this scenario by teaching the algorithm that those words are derived from the same original word (lemma). Decreasing redundancies among words could also be achieved by word stemming which reduces words to their word stems. In fact, we will apply a word stemmer later on since it is computationally faster than a lemmatizer. The advantage of a lemmatizer however, is that it produces word forms that are easier to interpret for human beings and can therefore better be matched to words in written documents. Thus, lemmatized words are a better input for search queries in information retrieval tasks

[16] https://wordnet.princeton.edu/

than stemmed words and they may even be better than the original words from the processed speech transcripts in this regard.

Figure 5.1 depicts the preprocessing stage for an ASR transcript sample. It can be seen that multiword expressions are connected using underscores.

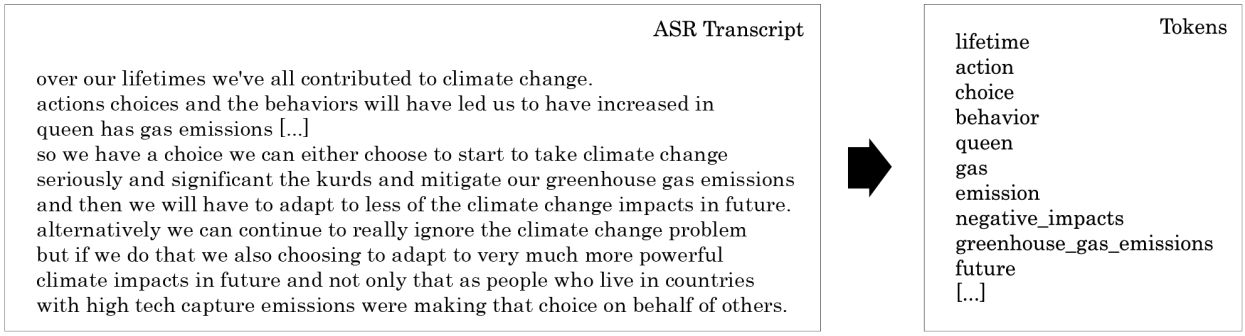

Figure 5.1: Illustration of the text preprocessing of an ASR transcript sample. The tokens on the right are the filtered output of the transcript on the left. Only a few tokens are shown here.

Semantic Word Clustering

At the current stage, we have obtained a set of text tokens containing single and multiword phrases. All we know about them for now is that they exceed an idf-threshold value, which means that these phrases should not be too common and therefore may contain valuable information related to our ASR transcript. However, we do not know yet how these phrases are semantically related to each other or to the transcript in general.

For this purpose, we employ Word2Vec in order to embed the extracted tokens into a semantic vector space. We have trained our Word2Vec model using gensim which has a very fast Python implementation of Word2Vec that makes use of parallelization techniques and precompiled C-code. For the training we used a subset of the English Wikipedia (see Section 4 for more details). In order to teach our model about word phrases, we used the same DRUID method as in the preprocessing stage for an input transcript. In general, Word2Vec is capable of modelling the semantic as well as the syntactic relations between words and phrases. However, we are only interested in the semantic relations, which is why we incorporated a word stemmer during the training process. We used the NLTK implementation of the Porter Stemmer [17]. The stemmer reduces words to their word stems by removing their suffixes. Therefore, inflections of words such as plural forms will be transformed into the same vector representations as other inflected forms as long as their word stems are the same. This improves the semantic precision of our trained model since there is more training data available for a word stem than for single inflections of each word.

Transforming our text tokens into word embeddings not only enables us to model their semantic relations but we can also find semantic clusters among words and phrases respectively. These clusters will help us to get a grasp of the major topics when processing an underlying speech transcript. In general, semantic similarity among words can be measured by computing the cosine similarity metrics of the corresponding vector representations [43]. However, Qian et al. have shown that the euclidean distance measure produces very similar results in high dimensional vector spaces when being applied to nearest neighbour searches [48]. The measures yield even more similar outputs when the vectors are normalized to a unit length which is done in gensim. Therefore, we can employ vector clustering techniques that are based on euclidean distance measures without any major drawbacks. In the original Word2Vec im-

[17] http://tartarus.org/ martin/PorterStemmer/def.txt

plementation [18] K-Means is suggested as a clustering algorithm that produces quite interpretable results. We have tried numerous other methods including Spherical K-Means, K-Means++, Affinity Propagation and DBSCAN which are all part of or adapted from the scikit-learn Python toolkit [45]. Our experiments have shown that center-based clustering algorithms such as K-Means and Affinity Propagation yield better results than density-based methods. This notion is quite intuitive as we measure semantic similarity of word vectors using euclidean distances. A density-based clustering algorithm would also assign words to the same cluster if the distance between them is large as long as they are connected to the same density region of data points. This means that even semantically dissimilar terms could be assigned to the same topical cluster. Spherical K-Means which is based on cosine similarity metrics produces very similar results to the standard K-Means algorithm that is based on euclidean distances. Therefore, our experiments confirm that the assertions made by Qian et al. also hold for our word vector space.

Affinity Propagation is a center-based clustering algorithm that yields results very similar to the ones produced by K-Means. Yet, its great advantage is that we do not need to specify a fixed number of clusters beforehand. Instead, the algorithm automatically determines the number of clusters based on the given dataset, which has worked very well in our tests. Moreover, the algorithm chooses representatives from the data set to act as the clusters' center points. This is very helpful when analysing the meaning and the interior relations of each cluster. The major drawback over K-Means is that the algorithm has a higher computational complexity and should therefore only be applied to smaller datasets. However, this does not pose a problem in our case as we apply the clustering algorithms to fragments of ASR transcripts that usually contain less than 100 prefiltered tokens. Thus, we have decided to choose Affinity Propagation based on euclidean distance metrics as our word clustering method.

Figure 5.2 shows a projection of our high dimensional vector space to two dimensions using Principal Component Analysis (PCA). It is important to note that a dimensionality reduction is always accompanied by a loss of information about the original high dimensional data points. However, the goal of PCA is to minimize this loss of information by fitting the data to a lower dimensional space so as to preserve the greatest possible variance within the dataset. Thus, in our case the original 100-dimensional word vectors are projected onto a two-dimensional plane that minimizes the squared projection errors between the word vectors and the plane. We can see in the Figure that the euclidean distances and therefore the semantic relations among words are mostly preserved in our two-dimensional projection. Semantically related words tend to be closer together and they also tend to be assigned to the same clusters which are denoted by the nets of different colours. The bigger dots indicate the clusters' center points. We can also see that the words *queen* and *climb* are assigned to independent clusters, which suggests that these words share little semantic similarity with the rest of the words contained in the original ASR transcript. In fact, these words constitute ASR errors corresponding to the originally intended words *green* as part of *green house gases* and *climate* respectively. Hence, we see that Word2Vec clusters can help us to determine off-topic terms as well as central topics of speech fragments.

The next problem that we need to address is how we can measure the importance of a semantic cluster. As we have already concluded, very small clusters tend to be off topic. However, we need to be aware that the clusters only show distinct words, which is why we also need to take each word's occurrence frequency within the transcript into account. This way even a smaller cluster can obtain a high weighting if the corresponding words appear often enough. Moreover, clusters containing specific terms should also be rewarded. In the example depicted in Figure 5.2, we would consider the red cluster containing terms related to climate change more important than the turquoise cluster that contains very general words that unfortunately were not captured by the idf prefiltering. Therefore, our first measure for a cluster's importance is its accumulated tf-idf score. Let CL denote the cluster which is a set of word phrases $\{p_1, p_2, \ldots, p_{|CL|}\}$. T is the speech transcript from which the clusters are derived.

[18] https://code.google.com/p/word2vec/

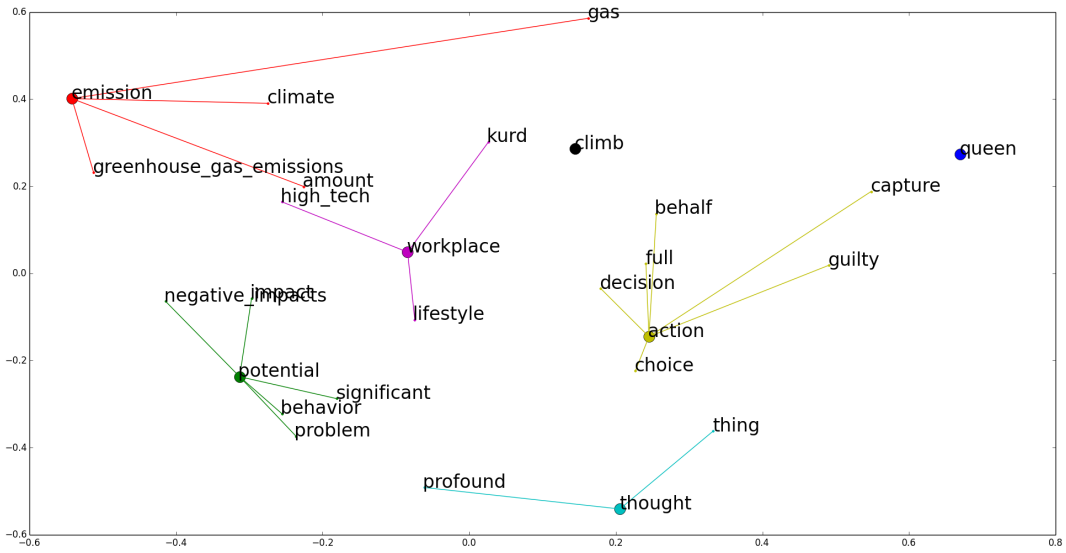

Figure 5.2: PCA projection of high dimensional word vector representations to two dimensions.

$$tf\text{-}idf_{CL,T} = \sum_{p \in CL} tf_{p,T} \times idf_p \tag{5.1}$$

Our second weighting metric is a cluster's connectivity score. The idea to include this score in our method was inspired by Plas et al. who employed this metric when scoring semantic WordNet clusters [59]. The connectivity metric measures the density of a cluster by computing the average euclidean distance of a cluster's words to its center word. Since a lower average distance indicates denser clusters, we take the inverse average distance computed as follows.

$$connectivity_{CL} = \begin{cases} \frac{|CL|}{\sum_{p \in CL} d(p,c)} & \text{if } |CL| > 1 \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

c is the cluster's center word and $d(p,c)$ denotes the euclidean distance between the corresponding word vectors. The euclidean distance for an n-dimensional space is

$$d(a,b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \ldots + (a_n - b_n)^2} \tag{5.3}$$

The underlying assumption for this measure is that clusters that are denser are more topically coherent. Even though this assumption does not seem to hold in Figure 5.2 we have obtained improved results when taking this measure into account. At the same time this measure leads to a cluster score of 0 for single-worded clusters assuming that these are produced by ASR errors or constitute minor details inside the transcript. The drawback of this assumption is that it highly relies on the word clusters produced by the Affinity Propagation algorithm. Yet, a low cluster score does not necessarily mean that the words inside these clusters will not be part of the final set of extracted keyphrases. We will explain the reason for this later on when looking at the actual procedure of keyphrase extraction.

Now we can determine the final weights β for each cluster CL.

$$\beta_{CL} = \textit{tf-idf}_{CL,T} \times \textit{connectivity}_{CL} \tag{5.4}$$

We have tried out numerous weighting schemes to investigate on the impact of the *tf-idf* and the *connectivity* scores respectively. However, we have come to the conclusion that a simple multiplication of both metrics yields the best results. Figure 5.3 visualises the word clusters for the speech transcript about climate change that has also been used in the previous examples. Higher cluster scores are reflected in bigger fontsizes that maintain the score proportions. We can quickly see that the transcript seems to deal with climate change and its associated problems. The off-topic words *queen* and *climb* as well as the very unspecific cluster {*profound, thought, thing*} have diminishing cluster scores.

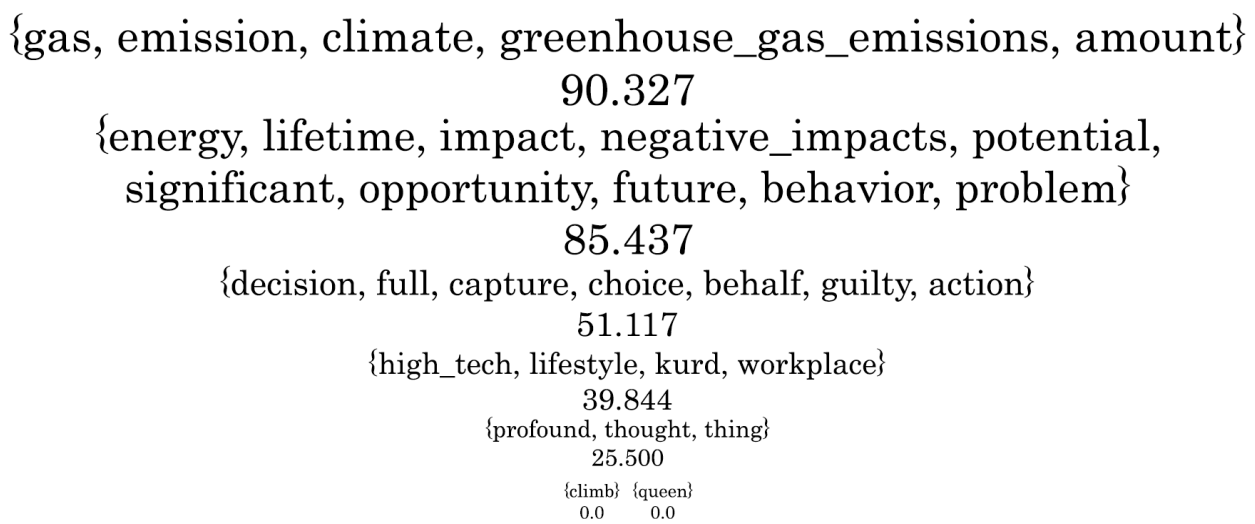

Figure 5.3: Word clusters produced by Affinity Propagation along with their corresponding scores. The score proportions are visualised through the respective font sizes.

Keyphrase Ranking

The cluster scores have given us a good grasp on which clusters and terms contribute to the major topics of the speech transcript. The final problem that needs to be solved is to rank each phrase with respect to its relevance for the transcript. This way we can automatically extract an arbitrary number of most important keyphrases. This will also help us to evaluate our algorithms with the LDA-based method by Habibi and Popescu-Belis.

We introduce a *centrality measure* that is inspired by the topical coverage measures used in [19, 22]. The centrality measure will be combined with a word's *tf-idf score* in order to rank terms according to their relevance for the transcript's major topics and their specificity. In spite of its simplicity, *tf-idf* has proven very effective in the domain of automatic keyphrase extraction from ASR transcripts [34] and should therefore not be neglected in our scoring function. We have also tried to incorporate a topical diversity constraint into our approach as it has been successfully done by Habibi and Popescu-Belis [19]. However, in our experiments that do not deal with spontaneous speech, a diversity constraint did not improve results, which can be seen in the evaluation section. Therefore, we have neglected this constraint and focus solely on the topical relevance as well as a term's specificity.

For each phrase, we define the centrality measure as follows. Let d_{max} be the maximum distance that any phrase can have towards any cluster center c of the corresponding cluster *CL*. We start off by computing a proximity score $\textit{proximity}_{p,CL}$ for each phrase p towards each cluster.

$$proximity_{p,CL} = 1 - \frac{d(p,c)}{d_{max}} \tag{5.5}$$

$d(p,c)$ is the euclidean distance between the phrase vector and the respective cluster center. According to this measure, a smaller distance towards a cluster will lead to a higher score for the corresponding phrase. We do not use the phrase's inverse distance as it was used in the connectivity measure for the cluster scores because we do not look at an accumulated distance. Thus, a very small distance of one vector to another would lead to an exploding proximity score. For instance, this problem would arise when computing the proximity of a cluster's center word towards itself given that we ignore the division-by-zero error. The advantage of our proximity score lies in its linearity with respect to the distances since d_{max} is a constant. Another advantage of this metric is that we could treat the proximity scores as probabilistic values that represent the likelihood of a phrase to appear in a cluster as the proximity values will always be between 0 and 1. In fact, Sridhar used the distributed representations produced by Word2Vec to train a topic model based on Gaussian mixture models that are built upon a similar idea [56]. It would be interesting to see how this model performs compared to LDA in the area of keyphrase extraction, which leaves space for further research.

Our next step is to compute the total weighted proximity score for each keyphrase. We call this score the *centrality* of a phrase vector since phrases that are located closer to important topics tend to obtain a higher score and are therefore more central ideas inside the underlying speech transcript. This idea is inspired by the topical reward function used in [19]. The centrality score sums up the weighted proximity scores for each phrase with respect to all n clusters. The weights are the cluster scores β_{CL}.

$$centrality_p = \sum_{i=1}^{n} \beta_{CL_i} \times proximity_{p,CL_i} \tag{5.6}$$

Finally, we combine the *centrality* score with the word's *tf-idf* score, which has shown to improve results significantly. We call our resulting measure *Central Relevance Score (CRS)*.

$$CRS_{p,t} = centrality_p \times tf_{p,t} \times idf_p \tag{5.7}$$

Thus, our keyphrase extraction problem is reduced to selecting the k phrases that have the highest Central Relevance Scores. It is quite simple to develop an algorithm that solves this problem since we do not need to take care of any limiting constraints that would require approximative methods. Algorithm 1 shows a pseudo code implementation for all the related computations that were mentioned in this section. The algorithm receives a sequence of preprocessed text tokens as its input and returns a set of k keyphrases. Figure 5.4 shows the algorithm's output for $k = 10$ meaning that the 10 most important keyphrases along with their scores are displayed in this Figure.

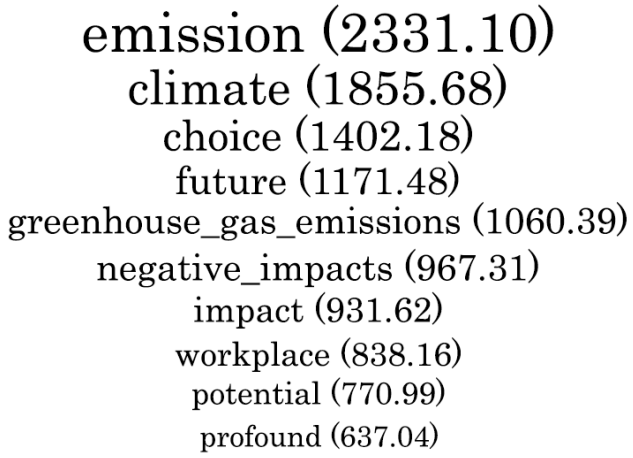

Figure 5.4: Top 10 keyphrases along with their corresponding scores produced by our keyphrase extraction algorithm. The score proportions are visualised through the respective font sizes.

We can see that most of these keyphrases are appropriate labels for the underlying transcript, which is a good indicator that the algorithm delivers a decent result. Yet, the algorithm also returns less relevant words such as *profound*. Therefore, it is important to quantitatively evaluate our algorithm based on a bigger dataset, which will be done in Section 6.2.

Data: a sequence of preprocessed text tokens $T = (t_1, \ldots, t_n)$, the number of keyphrases k
Result: a set of keyphrases S
Initialisation of word vectors and clusters:
phrase_vector_matrix[][] ← Word2Vec(T)
cluster_phrase_matrix[][], cluster_centers[] ← AffinityPropagation(phrase_vector_matrix, T)
distance_matrix[][] ← pairwise_euclidean_distances(cluster_centers, phrase_vector_matrix)
max_distance = max(distance_matrix)
# Computation of cluster scores β:
forall the *clusters* c_i **do**
  $\beta_{c_i} \leftarrow 0$
  if *size*$(c_i) > 1$ **then**
    $dinstances_{c_i} \leftarrow 0$
    $tf\text{-}idf_{c_i} \leftarrow 0$
    forall the *phrases* p_j *in* c_i **do**
      $distances_{c_i} = distances_{c_i}$ + distance_matrix[cluster_centers[c_i]][p_i]
      $tf\text{-}idf_{c_i} = tf\text{-}idf_{c_i} + tf\text{-}idf(p_j, T)$
    end
    $\beta_{c_i} = tf\text{-}idf_{c_i} *$ size(c_i) / $distances_{c_i}$
  end
end
Computation of keyphrase scores:
forall the *phrases* p_i **do**
  $centrality_{p_i} \leftarrow 0$
  forall the *clusters* c_j **do**
    $proximity_{p_i,c_j} = 1 -$ distance_matrix[cluster_centers[c_j]][p_i] / max_distance
    $centrality_{p_i} = centrality_{p_i} + \beta_{c_j} * proximity_{p_i,c_j}$
  end
  $CRS_{p_i} = centrality_{p_i} * tf\text{-}idf(p_i, T)$
end
Collection of most important keyphrases:
$S \leftarrow [\,]$
for *i=1 to k* **do**
  $S[i] = \text{argmax}_{p_i \in T \setminus S}\, CRS_{p_i}$
end
return S

Algorithm 1: Pseudocode of the entire algorithm used for automatic keyphrase extraction including the computation of cluster scores.

5.3 Formulation of search queries and filtering search results

Now that we are able to obtain a set of keyphrases from an ASR transcript, we can use these in order to automatically construct an (implicit) search query against a database of text documents. These search queries could generally be run on any kind of text corpus or search engine.

We have decided to index a dump of the Simple English Wikipedia (see Section 4) with the *elasticsearch*[19] open-source software. elasticsearch is built upon the *Apache Lucene*[20] Java library that allows very fast and scalable full-text information retrieval on a given text corpus.

The advantage regarding elasticsearch for our project is that it provides a very convenient application interface for a large variety of programming languages. Therefore, we can connect our Ambient Search system to the elasticsearch server and run our search queries directly inside our application code.

Whether elasticsearch considers a document (Wikipedia Article) of our text corpus relevant towards a given search query in a full-text search is typically determined by *tf-idf* scores[21]. These scores are calculated for each term inside the search query with respect to each article. More precisely, elasticsearch computes *tf-idf* scores for each term inside the search query in combination with each field inside a document that is required to be taken into account by the query.

For instance, we could construct a query that searches for the words *machine* and *learning* to appear inside the *title* or *text* field of any article from our Wikipedia dump. Elasticsearch would then compute a relevance score for each article inside our dump. This relevance score is a combination of the *tf-idf* scores with respect to the *title* and the *text* field. If the words *machine* and *learning* appear very often in the *title* field of a given article but very rarely in the *title* field of *any* article, then the given article receives a high score with respect to its *title* field. This score is even higher if this field is short, i.e. it contains very few other words. The latter optimization is called *field-length normalization*. The same is done for the *text* field resulting in a second score. Both scores are then combined to yield the given article's relevance score.

Of course, there can be more constraints that are formulated inside the search query such as requiring both terms *machine* and *learning* to be contained in a particular field or defining one field to be more important than another. There are numerous other adjustments that can be defined inside the search query in order to improve on the search results for a given text corpus. Mentioning all of them is far beyond the scope of this brief introduction to elasticsearch. Instead, we will shortly present the search query that we use to retrieve document suggestions inside Ambient Search and explain its components.

The query is shown in Figure 5.5. The entire query is JSON-formatted, which ensures a structured readability for the query parser. In fact, elasticsearch even offers different types of query parsers of which we chose the one that lets us formulate our intended query in a very convenient way. This parser is called *Query String* parser (see line 4 in Figure 5.5) and lets us define a set of fields (line 5) within every article that should be matched against our query. We want to match our query against the articles' titles, their text contents (texts) and their Wikipedia categories. The query itself (line 6) contains single words as well as phrases consisting of several words (denoted by the surrounding \¨ signs). We will use our extracted set of keyphrases as these terms. The numbers attached to them together with a ˆ sign indicate a *boost* factor. The higher the boost factor the more emphasis is put on the search term's *tf-idf* scores. We will simply take the *Central Relevance Scores* discussed in Section 5.2 as our boost factors.

The *minimum_should_match* attribute (line 7) lets us define a percentage of the terms inside the query that need to be matched in at least one of the given fields for an article to appear in the search results. In the example above 30% of the terms inside the query need to be found in either the *text*, *title* or *category* field of a given article. The percentage will be rounded to an actual number of terms. If *minimum_should_match* was set to zero, any article could be returned. Their order however, would be

[19] https://www.elastic.co/products/elasticsearch
[20] https://lucene.apache.org/core/
[21] https://www.elastic.co/guide/en/elasticsearch/guide/current/relevance-intro.html

```
1  POST en-simple-wiki/page/_search
2  {
3      "query": {
4          "query_string": {
5              "fields": ["text", "title", "category"],
6              "query": "term_one^4 term_two^3 \"phrase_one\"^2 term_three^1",
7              "minimum_should_match": "30%",
8              "use_dis_max": "false",
9              "phrase_slop": "0"
10         }
11     }
12 }
```

Figure 5.5: JSON-formatted elasticsearch query used to retrieve document recommendations inside Ambient Search.

determined by their relevance scores. The more matches and the higher the corresponding *tf-idf* scores are the higher is the article's relevance score. Setting *minimum_should_match* to 30% yielded the best results in our experiments as it ensures a decent term coverage of the resulting articles. Otherwise, an article that only contains a single term from the query could be returned in case the corresponding *tf-idf* scores are high enough.

The *use_dis_max* attribute (line 8) is set to *false* meaning that if a term inside the query is found within more than one of the provided fields, the fields' *tf-idf* scores will be added together. If we set this attribute to *true*, only the best scoring field would be taken into consideration. Once again setting this value to *false* gave us the best results in our experiments, which is quite intuitive since we want to match as many terms of our query on as many of the defined fields as possible.

Finally, the *phrase_slop* attribute in line 9 tells the query parser how far phrase terms are allowed to be apart while still considering a field inside the given article a match. If we set this value to 1 and were searching for the term *machine learning* in the *title* field, an article with the title *The machine is learning* would still be considered a match for this particular phrase because the words *machine* and *learning* are one word apart from each other. Setting this value to 0 gave us the best results.

The results of a query on the Simple English Wikipedia according to Figure 5.5 and with the extracted set of keyphrases from the example transcript used in Section 5.2 is shown on the next page. The underlying TED speech fragment is shown in the Appendix of this thesis. We have listed the top 10 search results that are represented by the titles of the corresponding articles. We can see that the results seem quite promising. In particular, it can be observed that the articles do not necessarily need to contain the exact terms that were part of the search query (and therefore part of the automatically extracted keyphrases) in their title field. In general, we obtain articles that contain a great part of the search query at a high frequency in any of the provided fields. This also shows us that the automatically extracted set of keyphrases does not need to be perfect in order to be used by a search query. It simply needs to capture a considerable deal of a speech's topical terminology. If there are erroneous words among the set of automatically extracted phrases, they will only contribute to some extent to the final search results since the scores assigned to each term are added together meaning that we use an *OR* search.

Yet, there are also less suitable articles contained in our search results: *sun angle*, *insolation* and *orbital forcing*. These articles contain the high scoring term *climate* in their few categories as well as in their short texts. The *field-length norm* therefore boosts the scores of these fields with respect to the term *climate* substantially. We try to compensate on this drawback of the field-length norm with

the *minimum_should_match* attribute. However, this approach does not succeed if the extracted set of keyphrases and therefore the query also contain rather general terms. Then, the *minimum_should_match* requirement can be fulfilled more easily (terms such as *profound* and *potential* are contained in a lot of articles). Nonetheless, we do not want to set the *minimum_should_match* value higher or deactivate the field-length norm as this would take away the positive impacts of these attributes and has shown to worsen our results.

Automatically extracted keyphrases:

emission (2331.10), climate (1855.68), choice (1402.18), future (1171.48), greenhouse_gas_emissions (1060.39), negative_impacts (967.31), impact (931.62), workplace (838.16), potential (770.99), profound (637.04)

Search results:

- Global warming
  - Categories: Energy, Climate change, Air pollution
  - Score: 1.1187227
- Climate commitment studies
  - Categories: Environment, Climate
  - Score: 1.0436358
- Climate of New York
  - Categories: Climate, New York
  - Score: 0.8518894
- Intergovernmental Panel on Climate Change
  - Categories: 1988 establishments, International environmental organizations, United Nations specialized agencies, Climate change
  - Score: 0.66155
- The Weather Makers
  - Categories: Non-fiction books, Climate change
  - Score: 0.63201964
- Sun angle
  - Categories: Climate, Seasons
  - Score: 0.5327941
- Insolation
  - Categories: Energy, Climate
  - Score: 0.53213936
- Orbital forcing
  - Categories: Climate, Earth
  - Score: 0.5240777
- 100,000-year problem
  - Categories: Climate change
  - Score: 0.5000886
- An Inconvenient Truth
  - Categories: 2006 movies, Climate change, Documentary movies, Energy, Green politics
  - Score: 0.4616956

6 Evaluation

We will evaluate our Ambient Search system using fragments of TED talks. We have selected 10 talks of which each deals with a different topic. From each of these talks we have selected three fragments of 1.5 to 2.5 minutes length. Thus, our methods are evaluated on a dataset of 30 speech fragments altogether.

6.1 Automatic Speech Recognition

When evaluating our proposed algorithms for Ambient Search, it is important to investigate on the quality of input data at each processing stage. The first component of our Ambient Search system is the ASR unit that processes a continuous audio signal in order to produce a speech transcript. As a first step of our evaluation, we want to examine the quality of the ASR output with respect to the original audio input. This way we can judge on the quality of ASR transcriptions that serve as the input for the keyphrase extraction methods in the next step.

We have transcribed each of the 10 TED talks of our dataset by directly passing on the audio stream of the corresponding mp3-files to the ASR unit. The mp3-files were obtained directly from the TED website[22]. In fact, the TED-LIUM corpus that our ASR system was trained on, was built using a different subset of the same type of audio files. In general, the audio signal contains very clear speech without any audible background noise (except for very occasional laughter that may occur between the speaker's utterances).

In order to evaluate the quality of the ASR transcripts, we compare them with manual transcriptions created by humans. These transcriptions can be obtained for each talk from the TED website as well. For the comparison we neglect any kind of punctuation and case sensitivity as these depend on the style of writing inside the reference transcript as well as on the previously transcribed words. Moreover, the keyphrase extraction algorithms presented in this thesis ignore any punctuation and case sensitivity as well.

As a quality measure we incorporate the Word Error Rate (*WER*) that is typically used to evaluate ASR systems [50]. The WER for an ASR transcript with respect to the reference transcript is calculated in the following way.

$$WER = 100 * \frac{S + D + I}{N}\% \tag{6.1}$$

This metric counts the minimum number of substitutions (S), deletions (D) and insertions (I) that are needed in order to transform the ASR transcript into the reference transcript that consists of N words.

Let us assume the reference transcript contained the word sequence "what a bright day" and our ASR system would produce the output sequence "what a light day". In this case we would need one substitution (exchange the word "light" with the word "bright") inside the ASR transcript to obtain the reference transcript. Therefore, the *WER* value would be $100 * 1/4 = 25\%$ since the reference transcript consists of four words. Thus, it can be seen that the *WER* value relies completely on the similarity of the ASR transcript to a manually created reference transcript. Therefore, it is important to choose very accurate reference transcriptions.

When transcribing the 30 speech fragments inside our 10 TED talks, we obtained *WER* values ranging from 5% to 41% leading to an average *WER* value of 15.65%. The standard deviation in our sample is 7.23%. Therefore, the fluctuations of *WER* values are not as strong as their range may suggest. The three highest error rates (28.62%, 30.77%, 40.91%) were all produced during the transcription of three fragments inside one particular talk. This talk contains rather scientific vocabulary with terms such as "topographical map" and "nuclear power plant". Moreover, the speaker has a French accent for which the acoustic model that was used for speech transcription may not have been sufficiently trained. Thus, we assume that the high error rates related to this talk come from a lack of training data for specific

[22] https://www.ted.com

scientific vocabulary pronounced in a French accent. The lowest error rates were observed for TED talks by speakers that have a US American accent. This can once again be attributed to the acoustic model's training data that contains a great proportion of US American speech.

We consider our *WER* values quite decent and the ASR output was mostly easy to understand in our experiments. Large enterprises like Google claimed to achieve *WER* values of around 8% in their state-of-the-art voice recognition systems[23] in May 2015. Yet, we have to take into account that their models are usually trained on datasets that are far beyond the size of the ones that were used for our models. Moreover, having influential ASR errors in our transcripts will help us to further evaluate how the subsequent keyphrase extraction algorithms deal with these errors.

6.2 Keyphrase Extraction

In this section, we will evaluate our proposed algorithm for keyphrase extraction based on Word2Vec. So far, we have not come across any comparable method, which is why we consider our approach experimental. Therefore, it will be very important to see what kind of impact Word2Vec has on the quality of extracted keyphrases. Furthermore, we will compare our results to the results produced by the keyphrase extraction method from Habibi and Popescu-Belis. It is important to note however, that we will not test our algorithms in the domain of spontaneous meeting conversations as in [19]. Instead, we will evaluate them using the previously described fragments of TED talks. For each of these fragments, we generated ASR transcripts that served as the algorithms' inputs. The algorithms' goal was to select the 9 most relevant keywords for each transcript.

In order to compare our results, we manually selected 9 words from each ASR transcript that should cover its main ideas as precisely as possible. These words describe an optimal set of extracted keywords and therefore serve as a gold standard in our evaluation. We set the number of extracted keywords to a fixed size because our proposed method as well as the method by Habibi and Popescu-Belis extract a predefined number of keywords. Since it may even be difficult for a human to decide on the best possible set of a limited number of keywords, we created an additional set of words for each transcript which has an arbitrary size. This set contains the rest of the words that are connected to the transcript's main ideas but were not considered as important as the top 9 words. We selected the optimal set of keywords from the ASR transcriptions and not from the original ones because we wanted to evaluate the keyphrase extraction methods in isolation. This means that they should have a chance of predicting a 100% correct result even if there are many ASR errors inside the transcript.

Given the two manually extracted sets, we introduce two evaluation metrics. The first one measures the fraction of manually extracted keywords in the first set that is extracted by the algorithm. This measure is typically called *recall* and has been employed by numerous others when evaluating extracted keywords [58, 27, 34, 35]. We do not calculate the recall with respect to both manually extracted sets because the size of the second set changes throughout the test samples. This would lead to a more difficult comparison of recall measures among the 30 transcripts. Moreover, a recall of 100% could not be achieved if both manual sets contain more words than the algorithms are supposed to extract. In the context of our limited sets, we define the recall formula as follows.

Let M_1 be the first set of manually extracted keywords. A is the set of automatically extracted keywords. Note that $|A| = |M_1| = 9$.

$$recall_{M_1}(A) = \frac{|A \cap M_1|}{|M_1|} = \frac{|A \cap M_1|}{|A|} \tag{6.2}$$

Usually this measure is accompanied by the *precision* metric that measures the proportion of automatically extracted keywords that are correct. However, when taking only the first manually extracted set of

[23] http://venturebeat.com/2015/05/28/google-says-its-speech-recognition-technology-now-has-only-an-8-word-error-rate/ - date of browsing the website was the 9th of January 2016

keywords into account, the precision and recall measure will be equivalent since the automatically and the manually extracted set are of equal size. Therefore, we have decided against this option.

Instead, we include a Human Rejection Rate (HRR) which has been introduced in [34]. The Human Rejection Rate measures the proportion of automatically extracted keywords that are definitely no candidates for optimal keywords. In our evaluation procedure we compute this metric by taking the proportion of automatically extracted keywords that neither match any word in the first nor in the second set of manually extracted keywords. The higher the Human Rejection Rate the more "useless" information is extracted by the algorithm. Therefore, this measure is a great complement to the recall which in our case is rather based on the similarity to a gold standard that was defined by a human. The formula for the Human Rejection Rate is

$$HRR_{M_1,M_2}(A) = \frac{|A \setminus (M_1 \cup M_2)|}{|A|} \tag{6.3}$$

where M_2 is the second (less prioritized) set of manually extracted keywords. The Human Rejection Rate is closely related to the precision measure. In fact, in our context it is the complement of taking the precision with respect to both manually extracted sets of keywords. Therefore, we could obtain such a precision value simply by computing $1-HRR$. However, this way we would assume that both manual sets of keywords are considered equally relevant to the speech fragment. In contrast, the Human Rejection Rate does not need to make such an assumption, which is why we consider it more appropriate for our evaluation.

In order to receive a final score that lets us rank our results, we simply subtract the HRR from the recall measure. We will simply call this difference *Keyphrase Relevance*.

In the following we provide a brief overview over the methods that we evaluated using the above mentioned metrics. The comparison of these methods and our own one will not only help us to assess our algorithm's relative performance, but it will also help us to find parts of the algorithm that can still be improved.

Method 1: Word2Vec-based method discussed in Section 5.2

We will evaluate the Word2Vec-based method that we introduced and discussed extensively in this thesis (see Section 5.2) with a full preprocessing pipeline as well as a reduced version in which the preprocessing is limited to stopwords removal only.

Method 2: *tf-idf* baseline

As a baseline algorithm, we incorporate a simple pretrained *tf-idf* model that ranks words and phrases according to their *tf-idf* scores. The model was trained on the same subset of the English Wikipedia as our Word2Vec algorithm and goes through the same preprocessing pipeline. This baseline will also help us to evaluate the contribution of the *centrality* score within our newly introduced Word2Vec-based method since that method combines both metrics.

Method 3: LDA keyphrase extraction favouring topical diversity

We compare our proposed Word2Vec-based method to a state-of-the-art method for keyword extraction that is based on LDA [19]. A more detailed description of this approach is given in Section 2.2 and 3. This method follows a topical diversity constraint that has been shown to improve results in the domain of spontaneous meeting speech. We will investigate on how this method performs compared to our own approaches when being applied to the domain of well structured speech from TED talks. We will also examine whether the positive influence of the topical diversity constraint used in this method still holds for our dataset. We trained the method's LDA model using the Gibbs Sampling implementation inside the Mallet toolkit [38] which was also chosen by Habibi and Popescu-Belis. We also trained the model on the same subset of the English Wikipedia that was used by our own method. This subset contains

about 110,000 articles with more than 50 words and is therefore comparable to the subset used in [19] that contained about 120,000 articles. Moreover, we set the number of topics for the model to 100 as well. Since the underlying algorithm only extracts single words instead of keyphrases, we will count a multiword expression consisting of n words as n distinct words when evaluating the Word2Vec and the *tf-idf*-based methods. Since the LDA-based method uses no preprocessing other than stopwords removal, we will also show how our method performs when the preprocessing pipeline is reduced to stopwords removal in order to present a fairer comparison of both methods.

Method 4: Word2Vec Implementation of the LDA-based method proposed by Habibi and Popescu-Belis

We tried to translate the above mentioned LDA-based approach that was introduced by Habibi and Popescu-Belis into a Word2Vec-based approach. The LDA-based approach makes use of word vectors, too. However, each field of such a vector is the value of $p(Topic|Word)$, i.e. the likelihood of the word to be drawn from a particular topic (see Section 2.2.2 for a more detailed explanation of this idea). Therefore, these word vectors are 100-dimensional (there are 100 topics) just as the Word2Vec vectors used for our own method. Indeed, we applied the exact same method as Habibi and Popescu-Belis to our Word2Vec vectors and therefore replaced the LDA model completely. The idea of this approach was to find out how much impact the different vector representations, i.e. Word2Vec or LDA, have on the evaluation results. One major difference between the representations is that LDA-based vectors contain only probabilities, i.e. positive values between one and zero that sum up to one. The Word2Vec vectors may contain arbitrary (also negative) values. Apart from that, LDA is actually intended for modelling documents instead of single words. Nevertheless, the vectors turn out quite useful when extracting keywords. For our Word2Vec version of the LDA-based method, we omitted the diversity constraint ($\lambda = 1.0$). The reason for this is explained in the discussion of the results. Moreover, we created one version with a reduced preprocessing pipeline (in order to mimic the LDA-based method more precisely) and a full version in which we also added *tf-idf* scores by simply multiplying them with the original scores for every word.

Algorithm 2 shows a pseudo-code implementation of this method. We added a *tf-idf* complement, preprocessing and set $\lambda = 1.0$ (not visible in the code since it serves as an exponent). The topical relevance of a keyphrase is not determined by its proximity towards a set of clusters. Instead, all vectors are summed up and normalized in order to form an average word vector (that does not belong to any actual word). Then, topical scores are computed by taking the dot product of each phrase vector with the average vector. Since our word vectors are normalized to have a vector length of one, the dot product is equal to the cosine similarity measure. A higher similarity to the average vector leads to a higher dot product between them and thus to a higher topical score for the given phrase. A multiplication of a phrase's topical score with its *tf-idf* score leads to the phrase's final score that is used for the extraction of n most relevant, i.e. highest scoring keyphrases.

Data: a sequence of preprocessed text tokens $T = (t_1, \ldots, t_n)$, the number of keyphrases k
Result: a set of keyphrases S
Initialisation of word vectors:
phrase_vector_matrix[][] ← Word2Vec(T)
# Computation of topical weights:
aggregated_phrase_vector = matrix_column_sum(phrase_vector_matrix)
average_vector = aggregated_phrase_vector / size(T)
# Computation of topical scores for each phrase vector:
topical_score_vector = dot_product(phrase_vector_matrix, average_vector)
# Computation of *tf-idf* complement:
tf_idf_vector = *tf-idf*(T)
final_score_vector = field_wise_multiplication(topical_score_vector, tf_idf_vector)
# Collection of most important keyphrases:
$S \leftarrow [\,]$
for *i=1 to k* **do**
  $S[i] = \text{argmax}_{t_i \in T \setminus S}\, score_{t_i}$
end
return S

Algorithm 2: Pseudocode of the Word2Vec translation of the LDA-based method (4).

Results

Method	Avg. Recall (Std. Dev. in %)	Avg. HRR (Std. Dev. in %)	Keyphrase Relevance
(4) with added *tf-idf* scores and preprocessing, $\lambda = 1.0$	**41.48%** (11.45%)	24.78% (16.4%)	**16.70%**
(1) with stopwords removal only	39.26% (11.01%)	**24.08%** (15.47%)	15.18%
(1) with a full preprocessing pipeline	39.63% (11.72%)	27.78% (18.76%)	11.85%
(3) with $\lambda = 1.0$	38.88% (16.42%)	32.59% (16.47%)	6.29%
(2) with a full preprocessing pipeline	37,78% (12.38%)	34.08% (16.48%)	3.70%
(4) with stopwords removal only, no *tf-idf*, $\lambda = 1.0$	34.44% (15.8%)	30.78% (13.05%)	3.66%
(3) with $\lambda = 0.75$	38.14% (15.62%)	34.81% (15.64%)	3.33%

Table 1: Evaluation of all previously discussed methods using a dataset of 30 ASR fragments from 10 different TED talks.

In Table 1 we can see the results from our evaluation of the previously described methods. The λ parameter in method (3) and (4) describes the diversity constraint. The lower λ is the more diversity is enforced within the set of automatically extracted keywords. $\lambda = 1.0$ implies no diversity enforcement whereas $\lambda = 0.75$ was found to be the best value in the domain of meeting transcripts in [19].

We can see that method (1) yields better scores than method (3) that was proposed by Habibi and Popescu-Belis and the *tf-idf*-baseline (2) with respect to the metrics that we introduced. It also yields the lowest sample standard deviation related to the recall score, which means that the recall was quite consistent among the sample fragments. Interestingly, the version with a reduced preprocessing pipeline performs better than the full version, which is due to a relative decrease in its HRR value. During our evaluation we have observed that the reduced version returns more keywords that are similar to each other such as inflected forms, which is not surprising since there was no lemmatization involved in this

Method	Keyphrases
Gold Standard M_1	data, graphical, map, sediment, ocean, nuclear, power, plant, fishermen
Tolerated Phrases M_2	project, seabed, sprinkle, pigments, strategy, river, system, expedition
(4) with added *tf-idf* scores and preprocessing, $\lambda = 1.0$	**sediment**, rough, seabed, **fisherman**, **plant**, estuary, pigment, cinnamon, coal
(1) with stopwords removal only	sprinkled, cinnamon, cheese, seabed, **fishermen**, **plant**, plants, rough, **sediment**
(1) with a full preprocessing pipeline	pigment, cheese, estuary, **plant**, **data**, seabed, **fishermen**, rough, **sediment**
(3) with $\lambda = 1.0$	estuaries, ***fishing***, **sediment**, seabed, river, sea, **fishermen**, cheese, cinnamon
(2) with a full preprocessing pipeline	cinnamon, bank, **data**, sushi, seabed, cetera, rough, **fishermen**, **sediment**
(4) with stopwords removal only, no *tf-idf*, $\lambda = 1.0$	**sediment**, seabed, textbftextitfishing, **plant**, plants, spread, rough, **ocean**, reproduce
(3) with $\lambda = 0.75$	estuaries, produce, **sediment**, ***fishing***, builds, seabed, cinnamon, closest, leaking

Table 2: Extracted set of keywords from a sample ASR fragment for each of the evaluated methods. The sample fragment contains a significant amount of ASR errors (WER = 41%)

method. Therefore, the reduced version returns less diverse keywords and therefore has a lower chance of returning many different irrelevant words, which is also reflected in its lower HRR standard deviation. However, this may be a problem when trying to retrieve a large variety of documents that match the ASR transcript.

We have observed that the extraction of keyphrases instead of keywords can be beneficial as well as disadvantageous. On the one hand, they improve the method's recall when adjacent words in a text are included in the set of manually extracted keywords. On the other hand, in terms of the Human Rejection Rate we had to count all words inside a multiword expression as errors if they did not appear in the set of manually extracted words. Therefore, an irrelevant or an incorrect multiword expression that was extracted increased the HRR value significantly, which may also explain the relatively high HRR standard deviation.

Despite its drawbacks we consider the extraction of phrases instead of single keywords a good idea since phrases are easier to interpret than disconnected words. Moreover, we expect phrases to be a more precise input for information retrieval.

Compared to the *tf-idf*-baseline (2), our proposed method (1) produces a significant decrease in its HRR-value. We explain the drop of around 7-10% with the fact that the centrality measure in our method favours words and phrases that contribute to the main ideas of the transcript. Therefore, we have successfully incorporated Word2Vec in order to create a topical relevance metric. For instance, the word *cetera* was extracted by the *tf-idf* method because of its high *idf* score even though it does not constitute any relevant information inside the transcript. Our method (1) successfully ignored this word. However, it can be observed that in spite of its high HRR value, the *tf-idf* method has a high recall. Thus, our results confirm on the previously conducted research related to keyphrase extraction from speech in which *tf-idf* was considered quite successful.

This insight becomes even more obvious when looking at the Word2Vec translation of the LDA-based method (4). Without a *tf-idf* complement this method scores relatively low in our test set with a recall of 34.44% and a HRR value of 30.78%. However, if we complement this method with *tf-idf* scores and some

preprocessing, we obtain a higher recall (+7%) and a lower HRR value (-6%). Interestingly, this way we even create a method that scores the highest among all of our evaluated methods (Keyphrase Relevance = 16.7%). At the same time we see once more that a topical coverage score improves a method's results over pure *tf-idf*. All methods that incorporate a topical coverage score in addition to a *tf-idf* score rank higher than the *tf-idf* baseline (2). This confirms once again the previously conducted research with respect to keyphrase extraction.

That method (3) achieved lower scores than our proposed methods (1,4) is probably due to the neglection of *tf-idf* scores and the reduced preprocessing pipeline. The LDA model and the topical coverage algorithm proposed by them performs relatively well. By translating the LDA-based method to Word2Vec (method (4)) we have even shown that the LDA-based version (3) performs slightly better. Therefore, it would be interesting to see how much the method proposed by Habibi and Popescu-Belis can be improved when complementing it with *tf-idf* values and more preprocessing. For our translated version of the method (4), this led to a substantial performance boost. Even more surprising is that we obtained the best scoring method within our test set as a by-product of our evaluation. The advantage of the method by Habibi and Popescu-Belis and our coincidentally created method is their simplicity. Both methods do not require any clustering and can be implemented in very few lines of code.

The diversity constraint within method (3) did not improve results in our evaluation. The reason for this might be the difference between spontaneous meeting speech and well-structured presentations. We consider a TED talk more topically focused since it is meant to deliver a clear message to the audience that deals with a certain issue. Therefore, the LDA-based method favouring topical diversity returned more irrelevant words rather than a set of words that covers a diversity of important ideas.

Table 2 shows the top 9 automatically extracted keywords for each method from a sample ASR fragment. The chosen fragment is the one that yielded the highest Word Error Rate (41%) during the evaluation of the ASR system. The underlying speech fragment deals with the topic of collecting radioactive sediment from the sea in the region around the nuclear power plant of Fukushima. It can be seen that the pure *tf-idf* method (2) extracts a lot of irrelevant words such as *cinnamon, bank, sushi, cetera* and *rough* of which some can be attributed to ASR errors. The Word2Vec-based method (1) stays more topically focused although also extracting the words *cheese* (ASR error) and *rough* which are not important for the fragment. We can also see that method (3) ($\lambda = 0.75$) extracts more diverse words than the one at $\lambda = 1.0$. However, not all of these diverse words contribute to the passage's major ideas (*produce, builds, cinnamon, closest, leaking*). Therefore, the diversity constraint rather leads to less specific words in our example. Yet, we have to say that all the examined methods extract some relevant keywords of the ASR transcript in spite of the transcript's high WER value. The speech fragment as well as its ASR transcription are shown in the Appendix of this thesis.

In summary, it can be stated that all of our evaluated methods lead to a similar average recall value, which indicates that all of them return relevant information for the underlying ASR transcripts. This proves the fact that unsupervised keyphrase extraction methods can achieve decent results in our test domain. Their major differences can be observed in the average Human Rejection Rate that contains more variation than the average recall. Our research confirms the insights that were previously brought up in related work, the only difference being that a topical diversity constraint does not seem to improve results when being applied to TED talks. Moreover, we have successfully made a first attempt in using Word2Vec for keyphrase extraction. In this regard we have proposed a novel method (1) as well as a Word2Vec translation (4) of the LDA-based method proposed by Habibi and Popescu-Belis (3). Both methods seem to perform quite well in our evaluation, which can be attributed to the combination of topical relevance and *tf-idf* scores. We cannot say yet whether LDA or Word2Vec is more suitable in order to model word vectors used for keyphrase extraction as the LDA-based method (3) is limited to topical relevance scores only. Since that method performs slightly better than its Word2Vec translation (4) without *tf-idf* and preprocessing, it would be very interesting to see in future research how much the LDA-based method can be improved with complementing scores. In general, we also think that the Word2Vec-based method (1) can still be improved since we only proposed a very experimental implementation

that was based on the insights of previously conducted research with respect to other models used for keyphrase extraction. For now, we would recommend the Word2Vec translation (4) of the LDA-based method (with added *tf-idf and preprocessing*) for practical use instead of method (1). The reason for this is that method (4) is very easy to implement, very fast, and achieves similar scores compared to the more complex clustering-based method (1). Thus, we have coincidentally found a well-performing Word2Vec-based method as a by-product of our evaluation that we will be using in the actual implementation of our Ambient Search system.

The remaining question is why our clustering-based approach (1) yields only similar (or even slightly worse) results in terms of topical relevance computation than the simple comparison of a given word vector towards an average word vector of the underlying speech fragment as it is done in method (4). This observation urges for an improvement of the topical clustering as well as the cluster score computation inside method (1). We will leave the investigation of this issue as well as a potential optimization of our method open to future research.

6.3 Document Recommendations

Lastly, we will evaluate the document recommendations given by our Ambient Search system. The evaluation will still be based on the 30 TED speech fragments from the previous evaluations. Moreover, we will not evaluate the IR system's performance in isolation. Instead, we take the entire Ambient Search process into account. Each speech fragment is processed by the ASR system leading to a speech fragment from which 9 keywords or phrases will be automatically extracted by the Word2Vec-based algorithm proposed in this thesis (see Section 5.2). The extracted keyphrases are then used to formulate a search query (see Section 5.3) that is run against our dump of the Simple English Wikipedia using elasticsearch.

There are numerous measures that can help us to evaluate the document recommendations given by an IR system. However, we have to bear in mind that we are not evaluating the system with respect to an explicit search query. In particular it is rather difficult to decide on an exact set of documents that we expect to be suggested by the system when listening to a fragment of a TED talk. Moreover, this would require us to know all the relevant documents in the Simple English Wikipedia that consists of around 120,000 articles. Therefore, a recall metric as it was used in Section 6.2 is rather difficult to obtain.

Hence, we will use a method that solely focuses on the quality of the top-ranked document recommendations for a given speech fragment. We will incorporate the *Normalized Discounted Cumulative Gain (NDCG)* measure that has gained increasing popularity for the evaluation of web search engines and related algorithms that return ranked search results [60, 36]. NDCG allows us to subjectively assign a *relevance* score to the top k document recommendations given by our system.

In this regard, two assumptions are made. The first one is that documents which appear earlier in the list of search results are considered more useful than equally relevant documents that appear at a lower rank. For web search engines this assumption is quite intuitive as users typically expect to obtain relevant results on the first page of the search results. The search results are considered less useful if the user has to browse many pages in order to find a decent result. But also our Ambient Search system ranks the recommended documents according to their relevance of which only the top results are predominantly displayed to the user. Therefore, for Ambient Search the top k returned documents need to be perceived useful by the user.

The second assumption made by NDCG is that we can assess a single document's relevance with respect to a user's information needs. NDCG would even allow us to grade each returned document such that its relevance would influence the search result's score at a fine-grained level. It is rather difficult for us to assign detailed relevance scores to every returned document because the information needs that emerge for different users when listening to a speech fragment may differ immensely. This is why we simplify the NDCG measure by assigning only binary relevance scores to documents. A document is therefore either considered relevant or non-relevant to a speech fragment. During our evaluation, we considered

a document relevant if it contained additional information that was topically related to the main ideas of the speech fragment.

The resulting NDCG measure that rates the top k document recommendations given by the Ambient Search system with respect to a speech fragment looks as follows.

$$NDCG_k = \frac{1}{DCG_{k,opt}}(rel_1 + \sum_{i=2}^{k} \frac{rel_i}{\log_2 i}) \tag{6.4}$$

The right factor in equation 6.4 is the *Discounted Cumulative Gain (DCG)*. rel_i refers to the relevance score of document i inside the top k results. We subjectively assign this value to each document recommendation. This value is either 1 - relevant, or 0 - non relevant. The relevance of each document is penalized by the logarithm of its rank i. A document that appears later in the list of recommended documents therefore receives a greater penalty on its relevance score. The logarithm is used to moderate this penalization. The left factor in equation 6.4 normalizes the DCG value on the right. $DCG_{k,opt}$ refers to the DCG value of an optimal sequence of k document recommendations. For our evaluation we assumed that there were k relevant documents for every speech transcript meaning that rel_i was set to one for each of the top k results when computing $DCG_{k,opt}$. The NDCG always returns values between 1 (highest score) and 0 (lowest score).

The drawback of our version of the NDCG measure is that it does not measure to what extent a user's potential information needs are covered. We neither defined a set of optimal documents to be returned for every speech fragment (which would allow us to compute a recall measure), nor did we rate the information gain of recommended documents in a detailed manner. They are either considered relevant or irrelevant. Therefore, in our context the NDCG measure is rather an indicator for how much information is contained in the recommended documents of which *some* might be considered useful by a certain user.

For our evaluation we set $k = 5$ meaning that we only took the top 5 of the recommended documents for each of the 30 TED fragments into account. We were particularly interested in the quality of the top 5 results because in the prototype implementation of Ambient Search, only the top scoring documents are displayed to the user at a given point in time. Moreover, we made the assumption that a user would not manually look up more than 5 articles in order to cover their information needs when listening to a speech fragment of 1.5 to 2.5 minutes length. This is of course a simplifying assumption since a user's information needs depend on the particular fragment as well as on the user's prior background knowledge about it.

When evaluating our 30 TED speech fragments we obtained an average NDCG score of 0.544 with a standard deviation of 0.314. This score indicates that our system seems capable of finding relevant articles for a given speech fragment even though a part of the document recommendations may run off-topic. We perceived the document recommendations more useful for fragments in which the speaker used many scientific terms. An example for a list of returned documents in such a case is shown in Section 5.3. A less specific vocabulary tended to return less specific or topically unsuitable documents. One reason for this might be that our keyphrase extraction algorithm favours the extraction of specific terms (high idf values), which may also lead to a better topical clustering as it is easier to assign those terms to a certain topic. Moreover, we presume that specific terms are more suitable to be part of a search query in information retrieval. In general, our entire approach is centred around words or phrases to appear inside a speech fragment. However, a speech fragment's major idea is often implied when combining simple words. The specific combination of words, especially the word order, is neglected by our approach.

Nevertheless, we are still interested in how much of the high variations with respect to the NDCG value can be explained by the performance of our algorithms that were applied prior to the incorporation of elasticsearch. More specifically, we want to examine the correlation between the NDCG value for a given

speech fragment and the Recall/HRR scores produced by our keyphrase extraction algorithm. We will do the same for the NDCG value with respect to the word error rate of each transcribed speech fragment in order to measure the ASR system's impact on the quality of recommended documents.

Figure 6.1 shows the NDCG values as well as the difference between recall and HRR (Keyphrase Relevance) across all 30 evaluated fragments. Both values appear to be strongly correlated, which can be confirmed by their correlation coefficient of 0.56. Therefore, a high Keyphrase Relevance value tends to be accompanied by a high NDCG value. The same is true for the opposite direction. Yet, this relation does not hold for every speech fragment. Speech fragment 29 for instance has an NDCG value of zero but an above average Keyphrase Relevance score. For the NDCG-recall and the NDCG-HRR correlation we obtain the coefficients 0.392 and -0.573, respectively. This means that NDCG is positively correlated with the recall scores of our keyphrases as well as negatively correlated with their HRR scores. The strong negative HRR correlation is quite intuitive since an unsuitable set of automatically extracted keywords tends to be a bad input for the formulation of a search query that would then return bad document recommendations. Therefore, our observations indicate that the quality of the document recommendations delivered by our Ambient Search system is strongly influenced by the performance of the implemented keyphrase extraction algorithms.

Figure 6.2 shows the NDCG values along with the word error rates for a given speech fragment. *ASR NDCG* refers to NDCG values that were obtained when our keyphrase extraction algorithm was applied to ASR transcripts, *Manual NDCG* refers to manually created transcripts (WER is 0). The NDCG and the WER values do not appear to be very correlated. An increase of the WER value sometimes leads to an increase of the NDCG value (see 13 or 23), but sometimes it also leads to a decrease (see fragment 21). The negative correlation coefficient of -0.128 with respect to the ASR NDCG values confirms the low correlation. According to this coefficient, an increase in the word error rate leads to a slight decrease in the NDCG value. The dashed line representing the *Manual NDCG* values is very similar to the *ASR NDCG* line. This shows that an overall decrease of the word error rate to 0 only leads to a small improvement of NDCG values (the average NDCG is raised from 0.544 to 0.622). We have two presumptions that may explain the low correlation.

One is that our keyphrase extraction method successfully ignores ASR errors to some extent, which might be due to a decent topical coverage of extracted keyphrases as it has been shown in related research. The very low WER-Keyphrase Relevance coefficient of -0.010 supports this notion (0 is the lowest possible correlation value). Yet, we have to be aware that we obtained the recall and HRR values with respect to a gold standard of keywords that was directly taken from the ASR transcripts and not from the original ones. Therefore, we expect the correlation to be higher if the gold standard is defined by manual transcripts. Moreover, a great deal of the ASR errors that we observed were contained in words that we would have considered as stopwords anyway. Therefore, it does not necessarily need to be only the topical coverage that resists these errors. The stopwords removal has a great impact on this result as well. In order to further investigate on this issue, additional research can to be carried out. For instance, one could compare the NDCG-WER correlation with respect to a pure *tf-idf* and a topical coverage method.

Another explanation for the low NDCG-WER correlation is that erroneous keyphrases or ASR errors do not substantially worsen the search results in IR as long as they only constitute a minor part of the search query. This way they do not contribute much to the relevance scores of recommended documents.

However, we need to be aware that the observed correlation values are only estimates obtained from a very limited dataset. In fact, the highest WER value that we observed was accompanied by an *ASR NDCG* value of 0. The NDCG obtained from the manually transcribed version of this fragment was much higher (0.719). Hence, for larger WER values we expect a stronger correlation than the one that we referred to in this evaluation. This intuition becomes quite clear when looking at the following example:

Supposing, we had a speech transcript that does not match a single word of the original utterance. This transcript would have a word error rate of 100% and since it does not contain a single matching word,

only erroneous words could be extracted from it. As we have also shown that keyphrase extraction performance and the quality of recommended documents are strongly correlated, a highly erroneous transcript would ultimately lead to poor document recommendations. This basic idea also illustrates the transitive correlation between WER, Recall, HRR and NDCG values. Thus, we have to be very careful when interpreting their connections.

In summary we can say that as long as the WER values stay inside a limited (low) range as it is provided for TED talks by our ASR system, the quality of document recommendations for a given transcript mostly depends on the performance of our keyphrase extraction algorithms. In fact, the high correlation between the NDCG and the Keyphrase Relevance scores proves that we have found a well-matching combination of a keyphrase and an IR evaluation metric. Since it is likely that a relatively high Keyphrase Relevance leads to a relatively high NDCG value, we can to some extent predict the average quality of recommended documents according to the NDCG measure by only looking at the Keyphrase Relevance scores provided that the IR system is configured properly. In general, the IR system has a great influence on the quality of recommended documents, too. However, the focus of this thesis was to develop and evaluate keyphrase extraction methods, which is why we have not investigated further on the optimization of IR systems. All in all, we have shown that an average WER of 15.65% with a low standard deviation is sufficient in our domain to restrict further optimization issues for Ambient Search to keyphrase extraction and information retrieval. We have also shown that our system is able to deliver reasonable results in the domain of TED talks.

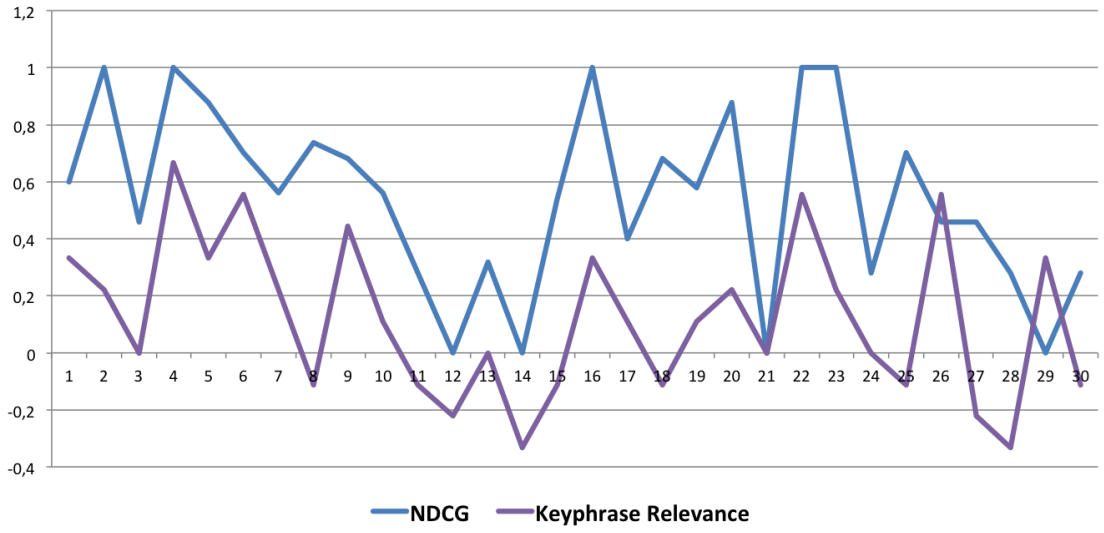

Figure 6.1: NDCG and Keyphrase Relevance values for each of the 30 transcribed speech fragments. A high Keyphrase Relevance tends to lead to a higher NDCG value (Correlation coefficient: 0.564).

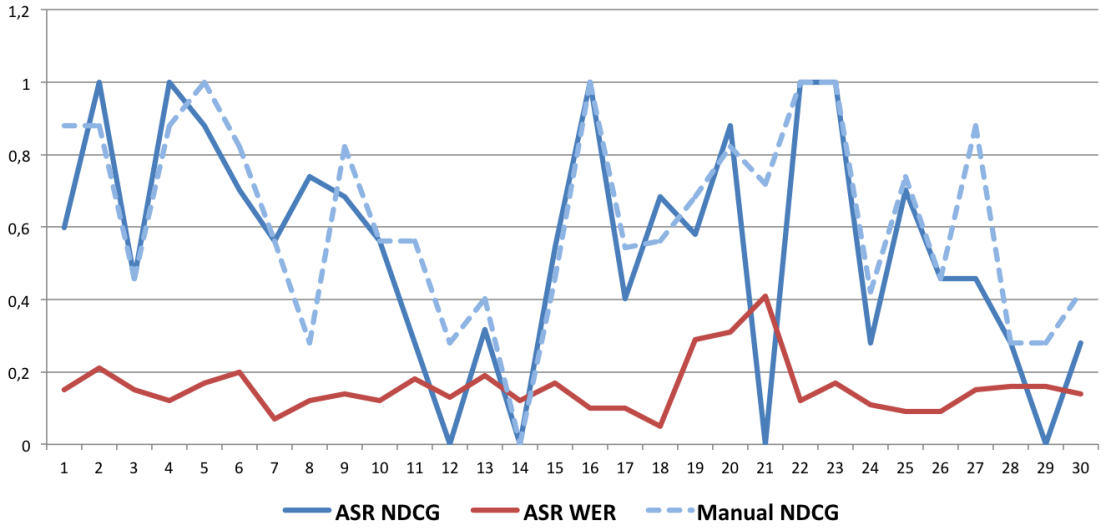

Figure 6.2: NDCG and WER values for each of the 30 transcribed speech fragments. The word error rate has little influence on the ASR NDCG score (Correlation coefficient: -0.128). The NDCG curve with respect to the ASR transcribed fragment is very similar to the one obtained from a manual transcript (WER=0).

7 Conclusion

In this thesis, we have made a proposal for the implementation of an Ambient Search system that has been shown capable of suggesting relevant information for speech fragments of TED talks.

We have developed a novel keyphrase extraction method that uses Word2Vec in order to determine a phrase's topical relevance with respect to a given ASR fragment. This method achieves reasonable results, which proves that Word2Vec is a suitable model for determining a term's topical relevance despite not being an actual topic model like LDA. This is not surprising since both, Word2Vec and LDA are based on the distributional hypothesis that determines a term's semantic meaning. The major difference between them in this regard is that LDA uses the notion of documents as a word's context whereas Word2Vec defines a word's context as a small frame of surrounding words. Furthermore, our evaluation results confirm previous research in the domain of keyphrase extraction from ASR transcripts claiming that both, *tf-idf* as well as topical relevance scores, positively influence an algorithm's performance. A topical diversity constraint has not proven useful for TED speech fragments in our evaluation.

We believe that there is still a lot of potential for future research regarding the optimization of keyphrase extraction methods that serve to produce the input for subsequent information retrieval. In particular, our topical clustering method as well as the computation of cluster scores have not produced significantly different results compared to a simple average vector comparison (see Section 6.2). The optimization of the keyphrase extraction methods is particularly important because their performance is highly correlated with the quality of recommended documents as it has been shown in Section 6.3. In fact, this correlation also shows that we have found a well-matching pair of metrics for the evaluation of extracted keyphrases and document recommendations, respectively. A relatively high Keyphrase Relevance tends to lead to a relatively high NDCG value. Therefore, we can to some extent predict the system's overall NDCG performance by only looking at the Keyphrase Relevance Scores provided that the IR system is properly configured and the rest of the Ambient Search components remain unchanged. This finding can be very helpful if it is more difficult to compute the NDCG values than the Keyphrase Relevance scores.

In contrast, the word error rate did not have a great influence in our evaluation. However, this observation does not necessarily hold for any WER variation. Nevertheless, we have shown that a reduction of WER values for our dataset does not lead to a substantial improvement in the quality of recommended documents. Therefore, the focus of future optimizations should be put on keyphrase extraction and IR methods instead of better ASR performance.

Lastly, we have proposed a prototype query that is run against a dump of the Simple English Wikipedia using elasticsearch in order to retrieve relevant articles. A detailed evaluation that measures the isolated performance of this query with respect to its parameters has not been carried out. Therefore, there most likely is much potential in order to optimize this query as well as the entire setup of the elasticsearch index. For now, documents are mainly matched against a query based on *tf-idf* scores. However, elasticsearch also provides the possibility to modify such scoring metrics. Moreover, we could run several queries against the database of documents and merge their resulting document lists afterwards as it has been done in [18]. Therefore, we consider our IR approach a very basic prototype although being quite effective already.

In terms of a practical or even a commercial adoption of the system, we think that the quality of recommended documents needs to be more consistent across speech fragments. Moreover, the system needs to be evaluated for different domains and by different groups of people in order to finally determine its major use cases as well as its social acceptance. Moreover, this issue would open up new dimensions for optimization. The system's usability including its installation would most likely play a major role in its acceptance. For now, it is rather difficult to set up all the required software and to train related statistical models. Furthermore, the ASR system that achieves decent WER values is computationally very expensive. Therefore, we recommend running it on a remote server, which may in turn raise privacy concerns that might have a negative impact on the system's social and legal acceptance since it actively listens to

surrounding speech input. Nonetheless, a first empirical study conducted by the Language Technology Department at Technische Universität Darmstadt and the Sibis Institute in Berlin has shown that an already running set up of an Ambient Search system was awarded with a good grade on a System Usability Scale. The respondents' major concerns in this regard were the overwhelming amount of information that the system showed to the user as well as the multitasking capabilities that are required in order to follow the document suggestions and the flow of speech at the same time. A habituation might improve the usability of Ambient Search over time. Lastly, the system should be configurable to better match a user's individual information needs. Future research should be carried out in order to investigate if these information needs can be automatically determined and therefore be learned automatically by the system.

8 Acknowledgements

I would first like to thank my thesis advisor Benjamin Milde whom I met on a regular basis during the course of writing this thesis. He took a lot of time reviewing my work and steered me in the right direction whenever it was needed. Furthermore, he and Prof. Dr. Chris Biemann gave me a lot of freedom in my research. In particular, the discussions with them about state-of-the-art research in Natural Language Processing have sparked my passion for this field of research. It was the first time that I extensively studied machine learning techniques and I will definitely continue to do so in the future.

Moreover, I would like to thank Maryam Habibi for sharing her code with me and answering my questions about it. Her work has been a great influence on my research.

Lastly, I want to acknowledge my family as well as Slava Kravchenko and Fedor Kosoy for proofreading my thesis.

References

[1] T. Alumäe. Full-duplex Speech-to-text System for Estonian. *Human Language Technologies - The Baltic Perspective*, pages 3–10, 2014.

[2] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*, volume 44. ACM Press, 1999.

[3] J. Baker. The DRAGON system - An overview. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 23(1):24–29, 1975.

[4] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A Neural Probabilistic Language Model. *The Journal of Machine Learning Research*, 3:1137–1155, 2003.

[5] S. Bird, E. Loper, and E. Klein. *Natural Language Processing with Python*. O'Reilly Media Inc., 2009.

[6] D. Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012.

[7] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3(4-5):993–1022, 2003.

[8] S. Brin and L. Page. The anatomy of a large-scale hyper- textual Web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.

[9] A. Chella, R. Sorbello, G. Pilato, G. Vassallo, G. Balistreri, and M. Giardina. An architecture with a mobile phone interface for the interaction of a human with a humanoid robot expressing emotions and personality. In *AI*IA 2011: Artificial Intelligence Around Man and Beyond*, pages 117–126. Springer-Verlag Berlin Heidelberg, 2011.

[10] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4):357–366, 1980.

[11] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[12] S. T. Dumais, E. Cutrell, R. Sarin, and E. Horvitz. Implicit queries (IQ) for contextualized search. *Research and development in information retrieval (SIGIR)*, page 594s, 2004.

[13] C. Fellbaum. WordNet. *The Encyclopedia of Applied Linguistics*, 2012.

[14] J. R. Firth. A synopsis of linguistic theory. *Studies in Linguistic Analysis*, pages 1–32, 1957.

[15] E. Frank, G. W. Paynter, I. H. Witten, C. Gutwin, and C. G. Nevill-Manning. Domain-specific keyphrase extraction. *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 668–673, 1999.

[16] Y. Goldberg and O. Levy. word2vec Explained: Deriving Mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, pages 1–5, 2014.

[17] I. Guyon. Neural networks and applications tutorial. *Physics Reports*, 207(3-5):215–259, 1991.

[18] M. Habibi and A. Popescu-belis. Enforcing Topic Diversity in a Document Recommender for Conversations. *International Conference on Computational Linguistics (Coling)*, pages 588–599, 2014.

[19] M. Habibi and A. Popescu-Belis. Keyword Extraction and Clustering for Document Recommendation in Conversations. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 9290(c):1–1, 2015.

[20] Z. Harris. Distributional structure. *Word*, 10:146–162, 1954.

[21] P. E. Hart and J. Graham. Query-free Information Retrieval. *IEEE Expert/Intelligent Systems & Their Applications*, 12(5):32–37, 1997.

[22] D. Harwath and T. J. Hazen. Topic identification based extrinsic evaluation of summarization techniques applied to conversational speech. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5073–5076, 2012.

[23] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. Deep Neural Networks for Acoustic Modeling in Speech Recogni-

tion. *IEEE Signal Processing Magazine*, pages 82–97, 2012.

[24] G. E. Hinton. Learning distributed representations of concepts, 1986.

[25] G. E. Hinton, J. L. McClelland, and D. E. Rumelhart. Distributed representations. *Parallel distributed processing: explorations in the microstructure of cognition*, 1:77–109, 1986.

[26] T. Hofmann. Probabilistic latent semantic analysis. *Uncertainty in artificial intelligence (UAI)*, pages 289–296, 1999.

[27] A. Hulth. Improved Automatic Keyword Extraction Given More Linguistic Knowledge. *Empirical Methods in Natural Language Processing (EMNLP)*, pages 216–223, 2003.

[28] F. Kraft, K. Kilgour, R. Saam, S. Stuker, M. Wolfel, T. Asfour, and A. Waibel. Towards social integration of humanoid robots by conversational concept learning. *Humanoids*, pages 352–357, 2010.

[29] T. K. Landauer, P. W. Foltz, and D. Laham. An Introduction to Latent Semantic Analysis. *Discourse Processes*, 25:259–284, 1998.

[30] O. Levy and Y. Goldberg. Dependency-Based Word Embeddings. *Association for Computational Linguistics (ACL)*, pages 302–308, 2014.

[31] O. Levy and Y. Goldberg. Neural Word Embedding as Implicit Matrix Factorization. *Advances in Neural Information Processing Systems (NIPS)*, 27:1–9, 2014.

[32] O. Levy, Y. Goldberg, and I. Dagan. Improving Distributional Similarity with Lessons Learned from Word Embeddings. *Association for Computational Linguistics (ACL)*, 3:211–225, 2015.

[33] H. Lin and J. Bilmes. A Class of Submodular Functions for Document Summarization. *Computational Linguistics*, 1:510–520, 2011.

[34] F. Liu, D. Pennell, and Y. Liu. Unsupervised approaches for automatic keyword extraction using meeting transcripts. *North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT)*, pages 620–628, 2009.

[35] Z. Liu, W. Huang, Y. Zheng, and M. Sun. Automatic Keyphrase Extraction via Topic Decomposition. *Computational Linguistics*, pages 366–376, 2010.

[36] C. D. Manning, P. Raghavan, and H. Schütze. *An Introduction to Information Retrieval*. Cambridge University Press, 2009.

[37] Y. Matsuo and M. Ishizuka. Keyword Extraction From a Single Document Using Word Co-Occurrence Statistical Information. *International Journal on Artificial Intelligence Tools*, 13(01):157–169, 2004.

[38] A. K. McCallum. MALLET: A Machine Learning for Language Toolkit, 2002.

[39] F. Metze, P. Gieselmann, H. Holzapfel, T. Kluge, I. Rogina, and A. Waibel. The "FAME" Interactive Space. *Machine Learning for Multimodal Interaction*, 2:126–137, 2006.

[40] R. Mihalcea and P. Tarau. TextRank: Bringing order into texts. *Empirical Methods in Natural Language Processing (EMNLP)*, 4(4):404–411, 2004.

[41] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Distributed Representations of Words and Phrases and their Compositionality. *Advances in Neural Information Processing Systems (NIPS)*, 26:1–9, 2013.

[42] T. Mikolov, G. Corrado, K. Chen, and J. Dean. Efficient Estimation of Word Representations in Vector Space. *International Conference on Learning Representations (ICLR)*, pages 1–12, 2013.

[43] T. Mikolov, W.-t. Yih, and G. Zweig. Linguistic regularities in continuous space word representations. *North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT)*, pages 746–751, 2013.

[44] A. Nenkova and R. Passonneau. Evaluating content selection in summarization: The pyramid method. *North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT)*, pages 145–152, 2004.

[45] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2012.

[46] A. Popescu-Belis, M. Yazdani, A. Nanchen, and P. N. Garner. A Speech-based Just-in-Time Retrieval System using Semantic Search. *Association for Computational Linguistics (ACL)*, pages 80–85, 2011.

[47] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely. The kaldi speech recognition toolkit. *IEEE Signal Processing Society*, pages 1–4, 2011.

[48] G. Qian, S. Sural, Y. Gu, and S. Pramanik. Similarity between Euclidean and cosine angle distance for nearest neighbor queries. *ACM symposium on Applied computing (SAC)*, pages 1232–1237, 2004.

[49] R. Rehurek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. *Language Resources and Evaluation (LREC)*, pages 45–50, 2010.

[50] S. Renals and T. Hain. *The Handbook of Computational Linguistics and Natural Language Processing*. John Wiley & Sons, Inc., 2010.

[51] B. J. Rhodes and P. Maes. Just-in-time information retrieval agents. *IBM Systems Journal*, 39(3):686, 2000.

[52] M. Riedl and C. Biemann. A Single Word is not Enough: Ranking Multiword Expressions Using Distributional Semantics. *Empirical Methods in Natural Language Processing (EMNLP)*, pages 2430–2440, 2015.

[53] X. Rong. word2vec Parameter Learning Explained. *arXiv:1411.2738v1*, pages 1–20, 2014.

[54] A. Rousseau, P. Deléglise, and Y. Estève. TED-LIUM: an Automatic Speech Recognition dedicated corpus. *Language Resources and Evaluation (LREC)*, pages 125–129, 2012.

[55] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988.

[56] V. K. R. Sridhar. Unsupervised Topic Modeling for Short Texts Using Distributed Representations of Words. *North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT)*, pages 192–200, 2015.

[57] M. Steyvers and T. Griffiths. Probabilistic topic models. *Handbook of Latent Semantic Analysis.*, 2007.

[58] P. D. Turney. Learning to Extract Keyphrases from Text. *Technical report, National Research Council, Institute for Information Technology*, page 45, 1999.

[59] L. van der Plas, V. Pallotta, M. Rajman, and H. Ghorbel. Automatic Keyword Extraction from Spoken Text. A Comparison of two Lexical Resources: the EDR and WordNet. *Language Resources and Evaluation (LREC)*, page 4, 2004.

[60] Y. Wang, L. Wang, Y. Li, D. He, W. Chen, and T.-Y. Liu. A Theoretical Analysis of NDCG Ranking Measures. *Learning Theory*, 26:1–30, 2013.

[61] B. J. Wythoff. Backpropagation neural networks: A tutorial. *Chemometrics and Intelligent Laboratory System*, 18:115–155, 1993.

[62] D. Yu. *Automatic Speech Recognition: A Deep Learning Approach*. Springer London, 2015.

9 Appendix

9.1 Artificial Neural Networks

The following is a very brief introduction to *artificial neural networks* derived from [17] and [24].

Computers tend to perform well at solving numerical problems but compared to humans it is difficult for them to solve cognitive tasks such as understanding human language. This is why researchers have investigated on how human cognition differs from the way computers solve problems.

An attempt of transferring human cognition to machines is the construction of artificial neural networks. These networks are a model of the brain's neural network which in reality is a very complex biological structure. Hence, artificial neural networks reduce this complexity a lot, which results in a mathematical model. This simplification is far from how an actual brain works. But it has been shown that artificial neural networks perform well at solving cognitive tasks such as speech recognition for instance.

The basic element of an artificial neural network is the formal neuron. The formal neuron is modelled as a mathematical function that takes a weighted sum of binary values as its input and maps it to a single binary value depending on whether a certain threshold has been reached or not. This model reduces a real neuron to only a few of its characteristics. However, it enables us to combine several neurons to build a neural network where neurons are connected to each other by using neurons' outputs as other neurons' inputs. The threshold value serves to mimic an observation of real neurons which emit a nervous influx once they have accumulated enough charge transmitted by other neurons attached to them.

A common choice for an artificial neuron's function is the logistic function. Let $\boldsymbol{x} = (x_1, \ldots, x_K)$ be the binary inputs and $\boldsymbol{w} = (w_1, \ldots, w_K)$ their weights. The logistic function f transforming the inputs into an output value y looks as follows.

$$y := f\left(\sum_{k=0}^{K} x_k w_k\right) = f(\boldsymbol{w}^\intercal \boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{w}^\intercal \boldsymbol{x}}} \tag{9.1}$$

We added an input value $x_0 = 1$ along with its weight w_0 as an additional offset for the function. The offset can be initialised with an arbitrary value which moves the graph of the function along the x-axis. Therefore, the threshold for the function which is needed to return the value of one can be shifted to an arbitrary value. This is done for convenience reasons, which will help in the construction of neural networks and adapting them to certain problems. The resulting graph of the activation function f is an s-shaped curve that can be seen in Figure 9.1.

We can see from this plot that the artificial neuron does not only produce a binary value depending on its threshold. Instead, there is a space around the threshold in which the function produces values between zero and one. This once again is done for mathematical convenience since the derivative of the function is simply $f(1-f)$.

In a neural network, the artificial neurons are organized in layers. The first layer only stores the initial input values which are broadcast to the first (hidden) processing layer. This process continues for each layer that receives the outputs from the preceding layer as inputs and produces new outputs for the following layer. This computational sequence terminates once the (final) output layer is reached resulting in the network's output values. The layers between the input and the output layer are called *hidden layers* because they do not take any inputs from the outside world and their produced outputs stay inside the network. Figure 9.2 shows an example of a neural network architecture.

When combining many artificial neurons to a neural network, we can formulate the problem of *back-propagation nets*. Given an n-dimensional input vector where n is the number of input values for the network, we want to find a mapping to produce a given m-dimensional output vector. This implies that the input as well as the output values are predefined and the goal is to find the correct network structure

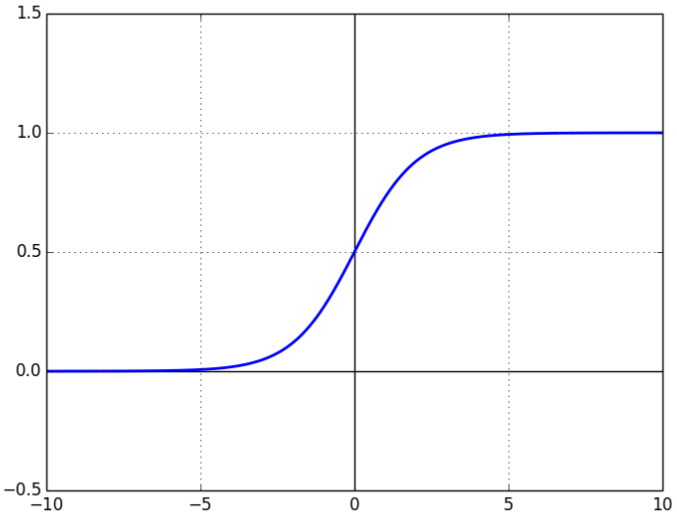

Figure 9.1: Illustration of the logistic function f with the neuron's accumulated input on the x-axis and the function's output value on the y-axis. The output is always between zero and one.

to produce this mapping. This is accomplished by approximating a big complex function that consists of many logistic functions represented by the neurons in the network. The major problem in this regard is to adjust the input weights for the logistic functions correctly. This approximation is done by minimizing the difference (i.e. the error) between the actual and the desired output vectors for the network. The error E is defined as:

$$E = \frac{1}{2}\sum_{c}\sum_{j}(y_j - d_j)^2 \tag{9.2}$$

j is an index over the neurons in the output layer. y_j is the actual output value of the jth neuron and d_j is its desired output value. c is the index over all input-output cases that the network should be trained on. As our goal is to minimize this error function with respect to the network's weights, we need to find the function's minimum first. This is done by using gradient descent where the current gradient, i.e. the slope of the function, indicates the direction in which the weights need to be adjusted in order to minimize the error step by step. In general, this method would only find a local minimum of the graph. Since we defined our error function to be quadratic however, its only local minimum value is global as well, which allows us to use gradient descent to find the best possible solution.

The process of minimizing the squared errors of the network's output values can be compared to a least squares error criterion as it is used in classical regression analysis. The difference however, is that the entire network's function does not have a linear form and the approximation is done in iterations in contrast to a direct matrix inversion [61]. Thus, our problem can be regarded as fitting the curve of a desired function.

The underlying process in our case is called *backpropagation* because we take the partial derivative of the error function with respect to the weights of one layer at a time starting with the input weights of the output layer and working ourselves through the hidden layers, one by one. The derivatives of the error with respect to the weights of each preceding layer will always depend on the resulting weights of its succeeding layer. Therefore, the error is backpropagated until the input weights of the first hidden layer are determined. It is important to note that backpropagation is not the only way to determine the network's set of weights. For instance, one could also randomly guess them until the error reaches a low

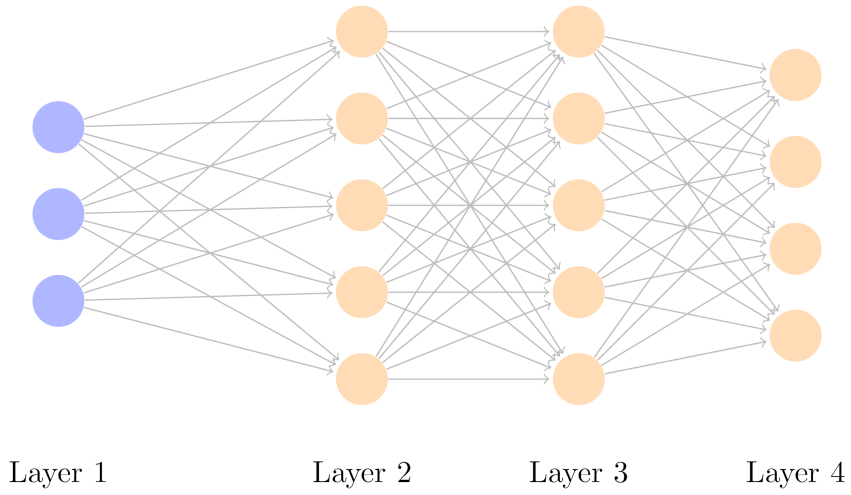

Figure 9.2: Illustration of an artificial neural network with four layers where layer one is the input layer and layer four the output layer. Layer two and three are called hidden layers. Taken from http://jay-mtl.github.io.

value. However, backpropagation is known to be a much more efficient method for this task in that it finds a very good solution in a short amount of time. This is why the algorithm was considered a serious discovery when it was introduced by Geoffrey Hinton in 1986 [24].

Of course, when solving a problem using neural networks one does not only need to find the correct set of weights for the inputs of each neuron but one also needs to define the entire network's structure. This structure, i.e. how many layers and neurons are used, highly depends on the task at hand and is usually predefined as it is the case for the input and output values for the network.

One example application for a problem that can be solved using neural networks is the automatic recognition of handwritten digits. For this problem, a set of sample handwritten digits are provided as the network's input values along with the related numbers as desired output values. The network's weights can then be trained to build a function that maps the provided handwritten digits to the correct numbers. Once this supervised learning procedure is done, the network can be applied to the recognition of handwritten digits that were not part of the training data. The reliability of the system depends on the amount of training data that was provided beforehand.

It turns out that the automatic recognition of handwritten digits belongs to the tasks that backpropagation nets perform very well at [17]. Yet, it is important to note that neural networks are merely a particular way of fitting an unknown function's curve. Whether this tool is the right choice for a specific problem or not strongly depends on the task, which leaves a lot of space for experimental research.

9.2 Samples from our Test Set of 30 TED Talk Fragments

9.2.1 Alice Bows-Larkin - Climate change is happening. Here's how we adapt

The following sample fragment was used as an example in Sections 5.2 and 5.3.

Original

Over our lifetimes, we've all contributed to climate change. Actions, choices and behaviors will have led to an increase in greenhouse gas emissions. And I think that that's quite a powerful thought. But it does have the potential to make us feel guilty when we think about decisions we might have made around where to travel to, how often and how, about the energy that we choose to use in our homes or in our workplaces, or quite simply the lifestyles that we lead and enjoy. But we can also turn that thought on its head, and think that if we've had such a profound but a negative impact on our climate already, then we have an opportunity to influence the amount of future climate change that we will need to adapt to. So we have a choice. We can either choose to start to take climate change seriously, and significantly cut and mitigate our greenhouse gas emissions, and then we will have to adapt to less of the climate change impacts in future. Alternatively, we can continue to really ignore the climate change problem. But if we do that, we are also choosing to adapt to very much more powerful climate impacts in future. And not only that. As people who live in countries with high per capita emissions, we're making that choice on behalf of others as well. But the choice that we don't have is a no climate change future.

ASR Transcript

over our lifetimes we've all contributed to climate change. actions choices and the behaviors will have led us to have increased in queen has gas emissions and i think that's quite a powerful full but it does have the potential to make us feel guilty when we think about decisions we might have made around where to travel to. how often and how about the energy that we choose to use in our homes or in our workplaces go quite simply the lifestyles that we lead and enjoy. but we can also turn our thoughts on his head and things if we had such a profound a negative impacts on our climb already them we have an opportunity to influence the amounts of future climate change that we will need to adapt to. so we have a choice we can either choose to start to take climate change seriously and significant the kurds and mitigate our greenhouse gas emissions and then we will have to adapt to less of the climate change impacts in future. alternatively we can continue to really ignore the climate change problem but if we do that we also choosing to adapt to very much more powerful climate impacts in future and not only that as people who live in countries with high tech capture emissions were making that choice on behalf of. others as well. the choice that we don't have is a no climate change future.

9.2.2 Cesar Harada - How I teach kids to love science

The extracted keywords from the following sample fragment are shown during the evaluation of our tested keyphrase extraction methods (see Section 6.2. The ASR transcript has the highest error rate among our samples (41%).

Original

But every night we would report to "Mission Control" – different masks they're wearing. It could look like they didn't take the work seriously, but they really did because they're going to have to live with radioactivity their whole life. And so what we did with them is that we'd discuss the data we collected that day, and talk about where we should be going next – strategy, itinerary, etc... And to do this, we built a very rough topographical map of the region around the nuclear power plant. And so we built the elevation map, we sprinkled pigments to represent real-time data for radioactivity, and we sprayed water to simulate the rainfall. And with this we could see that the radioactive dust was washing from the top of the mountain into the river system, and leaking into the ocean. So it was a rough estimate. But with this in mind, we organized this expedition, which was the closest civilians have been to the nuclear power plant. We are sailing 1.5 kilometers away from the nuclear power plant, and with the help of the local fisherman, we are collecting sediment from the seabed with a custom sediment sampler we've invented and built. We pack the sediment into small bags, we then dispatch them to hundreds of small bags that we send to different universities, and we produce the map of the seabed radioactivity, especially in estuaries where the fish will reproduce, and I will hope that we will have improved the safety of the local fishermen and of your favorite sushi.

ASR Transcript

but every night to report to mission control a different mask staring it could look like they didn't take their work seriously but they really did because they don't have. to live with what you actually t. there the whole life and so what to do with them is that we're discussing the data have collected that day and talk about what we should be going next strategy it's an area cetera. and to do this with bills are very rough couple graphical map of the region around in a coal power plant and so builds the elevation up we sprinkled pigments to represent real time data for activity and we spread what their to assimilate the rent for l.a. this who could see that the project. the dust is watching from the top of the mounting into the river system and leaking into the ocean so as a rough estimate. but with this in mind when organize this expedition which was the closest city and have been to the clean nuclear power plants were sailing one point five kilometers away from the nickel problems and with the help of the local fisherman we are collecting sediment from the sea bed with a cost on a cinnamon simpler we have invented in and build packed the sediment. two small banks. we then dispatched him to hundreds of small banks he was sent to different invested cheese and produce them up of the seabed really activity use fishing estuaries will officially reproduce and i would hope that you have improved the safety of the local fisherman and on your favorite sushi.