Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

FAKULTÄT
FÜR MATHEMATIK, INFORMATIK
UND NATURWISSENSCHAFTEN

# Unsupervised Multi-Document Summarization

A thesis for the academic degree
'Master of Science'
of

**Michael Eisele**

7eisele@informatik.uni-hamburg.de
Field of study: Computer Science
Matriculation number: 7089760

| | |
|---|---|
| Examiners: | Prof. Dr. Chris Biemann |
| | Hans-Peter Zorn (inovex GmbH) |
| Advisor: | Jannis Bergbrede (inovex GmbH) |
| Submitted: | 23.10.2019 |

Hiermit versichere ich, Michael Eisele, an Eides statt, dass ich die vorliegende Arbeit im Masterstudiengang Informatik mit dem Titel: *"Unsupervised Multi-Document Summarization"* selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

**Hamburg, den 23.10.2019**

. . . . . .
Michael Eisele

# Abstract

Today, vast amounts of textual information are available at any given time. Automatic summaries can help deal with this and allow the reader to focus on the most important information. To automatically summarize a text, current systems often rely on supervised machine learning techniques trained on large datasets. The practical adaptation of these systems proves to be an issue, as many domains only have small datasets to work with. This thesis investigates how well an unsupervised summarization system can work on small multi-document datasets. The multi-document aspect is in contrast to other current systems, as they oftentimes focus on single-document summarization. Therefore, a system that combines extractive and abstractive techniques is proposed. At first, it extractively summarizes each document and then it generates an abstractive summary of all documents by fusing similar sentences. The system has been evaluated on multiple datasets using the quality aspects defined by Grusky et al. [29]. Based on the experiments, the findings suggest that unsupervised systems can compete with supervised systems in generating summaries and the abstractiveness of a system negatively correlates with the information contained in a summary. Given that the information content is the most important aspect of a summary, extractive systems outperformed the proposed system. The thesis is aimed at readers who are interested in current research in NLP and who want to deepen their knowledge in automated summarization.

# Contents

# 1 Introduction

## 1.1 Motivation

We live in an era where tremendous amounts of information are available at all times. Through the internet, the latest literature, current news and product reviews are never more than a few clicks away. Although openly available information is an inherently good thing, being confronted with it can be quite overwhelming. *Summarization* can help us deal with this by extracting and compressing valuable information from text, which saves time and allows us to focus on the important content. To automate this, computers need to be taught to make sense out of words and their implied meaning as well as how to compress text into a fluent and coherent summary. The research field of *Natural Language Processing* (NLP) deals with this challenging task.

Current research in automated summarization focuses mostly on summarizing single documents, which is in contrast to how information is oftentimes presented to us. Most of the time, we are presented with related information from multiple sources. Examples of this are customer reviews, newspapers from different authors about a shared topic or academic research. A summary of multiple documents should contain the relevant information from each document, without repeating redundant content. The work done in the context of this thesis aims to reduce this discrepancy by exploring ways to summarize multiple documents.

To automate the summarization process, systems oftentimes rely on statistical and machine learning approaches. Approaches in machine learning are generally divided into two categories: *supervised learning* and *unsupervised learning*. The former requires labeled data and tries to find a generalized transformation from the input data to predefined output. The latter in contrast does not require labeled data and tries to predict the output based on reoccurring patterns in the input data. Although NLP has made several breakthroughs in the last years due to effective text representations and an increasing amount of large training datasets, the practical adaptation to specific domains still proves to be very difficult. One major issue is that many domains only have small datasets to work with. Unsupervised learning, since it does not rely on domain-specific training data, could be one way to solve the adaptation problem.

There are two distinctive techniques to summarize text: *extractive* summarization, which involves identifying the most salient parts and choosing a subset of the original text and *abstractive* summarization, where the original text is paraphrased in a compressed way. Extractive approaches are generally less expensive compared to abstractive ones, as they do not require a comprehensive understanding of language. Although extractive systems did achieve good results in the past, current research focuses on abstractive systems in

hope of creating more natural summaries. Part of this thesis is to explore how well these two approaches can be combined.

With some of the earlier works dating back to 1958 and 1969 [51, 21], automatic summarization is not a new field of research. Still, many challenges remain unsolved when it comes to generating high-quality summaries that could rival those written by humans. This thesis investigates whether some of these challenges can be solved using unsupervised approaches, especially in domains with only small datasets available.

## 1.2 Research objectives

The thesis' main hypothesis is:

*Unsupervised systems can improve the generated summarizes in domains where only small datasets are available.*

To discuss this, the following research questions are asked:

1. Does combining extractive and abstractive approaches improve the generated summaries?
2. How can current research in NLP help unsupervised summarization systems?
3. How well can unsupervised summarization systems generalize to diverse input data and summary lengths?

## 1.3 Outline

This section briefly summarizes the outline of this thesis. After this introduction, Chapter 2 introduces the theoretical foundations necessary to understand the approach and briefly introduces recent advances in NLP, which made this thesis possible in the first place. Chapter 3 investigates the research landscape in multi-document summarization and introduces ways to evaluate summaries. The baseline approach is explained in detail in Chapter 4. Chapter 5 proposes an experimental setup and shows the results from these experiments. Chapter 6 analyses the results and proposes ways to improve upon the baseline. Chapter 7 concludes the findings of the previous chapters and provides an outlook to future research.

# 2 Theoretical foundations

This chapter provides an introduction to the relevant theoretical foundations and concepts of this thesis. The first section defines the task of summarization and provides an overview of different summarization approaches and categories. The subsequent section explains important aspects of a summary and how they can be used to evaluate the quality of it. Automating the process of summarization is one of the many tasks of Natural Language processing, which is introduced in Section 2.3. The last section presents *word embeddings*, a way of turning words into a machine-readable representation.

## 2.1 The summarization task

Summarization is the act of producing a condensed form of text while still retaining most of the important information from the source. The following paragraphs follow the classification of approaches to summarization from Gambhir and Gupta [25].

Approaches to summarization are usually divided into two categories: *extractive* and *abstractive*. Extractive approaches identify the important parts of a text and select a subset of the text's sentences to form the summary. This is similar to how one would use a text marker to highlight the most important parts. They tend to be easier to compute as they do not need a form of language generation but they are also limited since they can not generate new sentences. Abstractive approaches aim to summarize the text by generating a new text which conveys the information. This is akin to how humans would summarize something using their own words. Abstractive approaches tend to be more complicated, as text generation requires some form of language understanding. Table 2.1 shows a short source text, which is summarized in an extractive and in an abstractive way.

Based on the input text, summarization is either classified as *single-document* or *multi-document*. Single-document summarization, as the name implies, focuses on a single document and can therefore rely on sentence position and the document structure. Multi-document summarization deals with a *corpus* of text, which in this context is a set of at least partially related documents. Multi-document summarization is a task that often occurs in everyday life - news articles, blogs or customer reviews share a common topic but oftentimes have a different focus. The main challenge is to identify important bits of information while at the same time keeping redundancy to a minimum.

Summarization can also be categorized using the summary's objective. It can either be a *generic* summary or a *query-based* summary. Whereas the former tries to summarize all key aspects of the source text, the latter tries to answer a query or question and the

| | Source |
|---|---|
| True | Bilbo was very rich and very peculiar, and had been the wonder of the Shire for sixty years, ever since his remarkable disappearance and unexpected return. The riches he had brought back from his travels had now become a local legend, and it was popularly believed, whatever the old folk might say, that the Hill at Bag End was full of tunnels stuffed with treasure. And if that was not enough for fame, there was also his prolonged vigour to marvel at. Time wore on, but it seemed to have little effect on Mr. Baggins. At ninety he was much the same as at fifty. |

| Extractive | Abstractive |
|---|---|
| Bilbo was very rich and very peculiar, and had been the wonder of the Shire for sixty years, ever since his remarkable disappearance and unexpected return. Time wore on, but it seemed to have little effect on Mr. Baggins. | Bilbo was very wealthy and peculiar, the riches he brought back after his disappearance and unexpected return made him a local legend. He was also very vigorous for his age. |

**Table 2.1:** An excerpt from the fantasy novel *The Fellowship of the Ring* from Tolkien [88], alongside an exemplary extractive and abstractive summary.

produced summary only contains information relevant to the query. The main focus of this thesis lies on generic summarization.

## 2.2 Summary quality

To put the knowledge about summarization into practice and produce good summaries, a definition of what makes a summary "good" is needed, as well as ways to measure and compare the summary quality. In the context of this thesis, the definition of Grusky et al. [29] is used, which defines the following four key aspects of a good summary: *informativeness, relevance, fluency* and *coherence*. Together they can provide a good picture of a summary's quality and allow to compare the strengths and weaknesses of different summary systems. Dang [18] provide a similar definition with the additional aspects of *referential clarity* and *focus*. These aspects are not included in the evaluation, as they can only be manually evaluated as of today. *Abstractiveness*, although not an indicator of summary quality, is an important aspect when comparing automatically generated summaries.

**Informativeness** is the most obvious way to assess a summary. The main question that can be asked with an informativeness metric is: 'does it capture the important information?' To answer this question, *lexical-similarity-based* metrics are commonly used. These metrics rank summaries based on how similar the system-generated summary, known as a *candidate summary*, is to a human-written *reference summary*. *Content-based* metrics serve a similar purpose, but instead of relying on word-overlap in the candidate and reference summary, they aim to measure how many of the key-points from the source document are found in the candidate summary.

**Relevance** is an indicator of how consistent the content of a candidate summary is compared to the source material or a reference summary. This is especially important for affirmation and negation of facts, which might go unnoticed in content-based approaches. To measure it, lexical-similarity-based metrics are used. They commonly work on multiple *n-grams*. A n-gram is a contiguous sequence of $n$ items in a text (e.g. characters or words). *Bi-grams* (e.g. *the dog* or *dog barked*) and *four-grams* (e.g. *the dog barked loudly*), with the values of $N = 2$ and $N = 4$ respectively, are commonly used in relevance metrics.

**Fluency** is what makes humans perceive sentences as natural. It is based on the same intuition as *readability* or *grammaticality* and distinguishes a sentence from a list of unrelated words. Commonly, the perplexity of a *language model* (see Section 2.5), which indicates how well a model can predict a sample, is used to measure the fluency of a sentence. This is based on the intuition that words, which commonly occur after one another, are likely to be grammatically correct.

**Coherence** is what differentiates a well-written text from a set of unrelated sentences. It is often neglected in summarization systems, as current research focuses on the informativeness and relevance of the generated summaries. One way to assess this is by measuring how similar the follow-up sentences are or how similar the topics of subsequent sentences are.

**Abstractiveness** is not defined as a quality aspect by Grusky et al. [29], but it helps to set the other aspects in context. It measures how abstract the candidate summary is compared to a reference summary. Therefore, a lot of paraphrasing correlates with a high abstractiveness. A common metric to measure the abstractiveness is the *copy rate*, which counts the number of words that occur in both the candidate summary and the source documents. Extractive systems have a copy rate of 100% by definition. A summary that has a low copy rate is considered to be highly abstractive as it expresses the content of a text in its own words. High abstractiveness values in combination with content-based informativeness metrics can also compensate for low scores in lexical similarity metrics as they often rely on word overlap.

Automating the summarization process is one of the many tasks in the field of Natural Language Processing, which is introduced in the following section.

## 2.3 Natural Language Processing

The field of natural language processing (NLP), which is sometimes referred to as *computational linguistics*, is a research area that deals with computational models and processes to solve problems in understanding human languages [65]. Application areas for NLP besides summarization range from *part-of-speech* (POS) tagging to machine translation and automatically answering questions for example. To work on these tasks, a wide variety of rule-based, statistical and *machine learning* approaches is employed in NLP.

**Machine learning** is a subfield of *artificial intelligence* (AI), which involves algorithms that are able to learn from data. Mitchell [60] defines the term *learning* in this context as: "A computer program is said to learn from experience $E$ with respect to some class of tasks $T$ and performance measure $P$, if its performance at tasks in $T$, as measured by $P$, improves with experience $E$". This includes a wide variety of tasks and disciplines, which to explain would go beyond the scope of this thesis. Interested readers are referred to textbooks by Mitchell [60] and Bishop [8].
Approaches in machine learning are generally divided into two categories: *supervised learning* and *unsupervised learning*. The former requires labeled data and tries to find a generalized transformation from the input data to predefined output. In contrast, unsupervised learning does not require labeled data and tries to predict the output based on features or patterns in the input data [28].

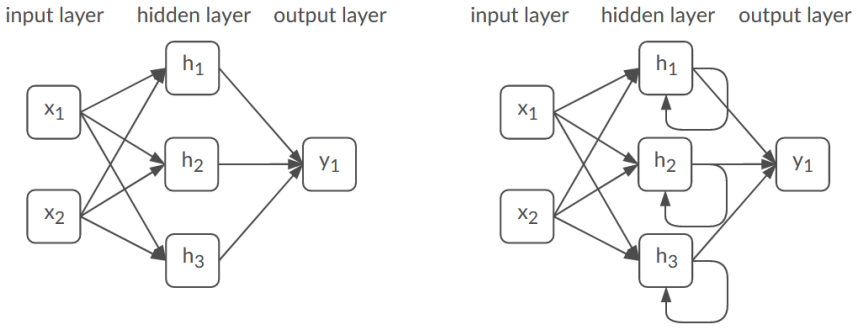
A fundamental problem in Natural Language Processing is finding a representation which allows machines to understand text in a similar way humans do. This can be formalized as a machine learning problem with a text $t$, which consists of $n$ words $w_1, ..., w_n$, as input and a numerical representation $r$ as the desired output. One of the most well-known approaches to language representation is *bag-of-words* (BOW). It is simply an unordered set of words alongside their respective frequency in a document, which can be used to classify documents based on their content. This approach is not the best way of representation, as a lot of information cannot be represented in terms of word frequencies.

To address this, earlier research in NLP focused on statistical approaches along with machine learning approaches. However, in the last years many machine learning disciplines, including NLP, shifted the research focus to *neural networks* (NNs), which enhanced or replaced many of these approaches [65].

## 2.4 Neural networks

A neural network, sometimes referred to as *artificial neural network* (ANN), is a widely used machine learning method inspired by the biological brain [28]. Its basic building block is the *neuron*, which takes a set of input values and performs a weighted sum computation over a non-linear activation function (e.g. softmax). The weights of each neuron are learned parameters, which get corrected using the error of the network's output. Multiple neurons are grouped on a *layer* and a neural network typically consists of an *input layer* and an *output layer*, as well as one or multiple *hidden layers* inbetween.

The most intuitive neural network architecture is the *feed-forward network* (FFN). It consists of multiple layers with connected neurons. The output of each neuron is passed to the neurons in the next layer, with no connection to pass values back to lower layers. Figure 2.1 shows the standard architecture of an FFN on the left side. The *recurrent neural network* (RNN) in contrast does allow values to be passed back to lower layers, which makes it very suitable for sequence tasks. The architecture of an RNN is shown on the right side of Figure 2.1. These architectures form the basis of the networks employed for this thesis and the current state of the art in Section 2.7.




**(a)** Feed-forward neural network **(b)** Recurrent neural network

**Figure 2.1:** Conceptual architecture of an FFN and an RNN, with two input neurons, a single hidden layer and one output neuron.

## 2.5 Language Modeling

A language model (LM) is a function that estimates the probabilistic distribution of words in a text. Common use-cases of LMs are next-word-prediction in assisted typing and grammaticality checks. Given a sequence of words $w_1, ..., w_n$, the goal is to learn the probability $P(w_1, ..., w_n)$. Using this probability, a LM can derive the probability $P(w_i|h)$ of a word $w_i$ given its history of previous words $h = w_1, ..., w_{i-1}$. Language models can be classified as either *count-based* or *continuous-space* LMs.

**N-gram language models,** which are count-based, are a simple approach to model language. Using a large training corpus, they count how often a sequence of words $h$ is followed by word $w$ to estimate the probability distribution. The problem is that natural language has an almost infinite variety of word sequences and some sequences might not be in the training corpus. To solve this, the *chain rule of probability* is applied:

$$P(w_1, ..., w_n) = \prod_{i=1}^{n} P(w_i|w_i, ..., w_{i-1}) \tag{2.1}$$

Due to the fact that temporally closer words are statistically more dependent [6] and word order is an important aspect of natural language, n-gram models simplify the computation by approximating the history using the last $n-1$ words:

$$P(w_1, ..., w_n) = \prod_{i=1}^{n} P(w_i|w_{i-n+1}, ..., w_{i-1}) \tag{2.2}$$

The assumption that the probability of a word depends only on the $n$ previous words is called $n$-th order *Markov assumption* [38]. Commonly, the n-gram probabilities are computed using *maximum likelihood estimation* (MLE). MLE estimates the probability of a word $w_i$ and its history $h = w_{i-n+1}, ..., w_{i-1}$ by counting how many times $w_i$ appears in $h$ and normalizing it by all occurrences of $h$:

$$P(w_i|w_{i-n+1},...,w_{i-1}) = \frac{count(w_{i-n+1},...,w_{i-1},w_i}{count(w_{i-n+1},...,w_{i-1})} \tag{2.3}$$

For example, given a *tri-gram* model and the history "After school she reads", the goal is to predict the probability of the word "books". To do this, the model would count how often the last $n-1$ words "she reads" is followed by "books" in the training corpus:

$$P(books|\text{she reads}) = \frac{count(\text{she reads books})}{count(\text{she reads})} \tag{2.4}$$

One problem of n-gram LMs is that known words might appear in a context that was not in the training data. To avoid the assignment of zero probability to these sequences, smoothing (e.g. *Laplace* or *Kneser-Ney*) is required [54]. This slightly reduces the probability of sequences that appear in the training data and adds a small amount to sequences that do not appear in it. One way to overcome this issue is to employ neural networks, as they can generalize well to sequences of similar words [38].

**Neural language models,** which are continuous-space LMs, do approximate the probability of a word based on the previous $n$ words, just like n-gram LMs do. What makes them generalize better, is the fact that the history is not represented by words but by *embeddings* instead. Embeddings are words embedded in a high-dimensional vector space, allowing them to capture semantic similarities [38]. Coming back to the previous example, if the training data does not contain the phrase "she reads newspapers", an n-gram LM could only predict "books" as a possible next word. A neural LM could also predict "newspapers", based on the fact that the embeddings of "books" and "newspapers" are likely to be similar. To train a neural network on a task, one needs some way to evaluate its performance, in order to correct the network's weights and thereby minimize its error. For an LM, this way of evaluation is the *perplexity*.

**Perplexity** is a probability-based metric to evaluate the performance of a language model. Achieving a low perplexity on a test dataset is the goal of a language model, as task-specific evaluation can be very costly. The perplexity of a sequence of words $s = w_1, ..., w_n$ is defined as the inverse probability normalized by the number of words:

$$perplexity(s) = b^{-\frac{1}{N}\sum_{i=1}^{N} log_b P(w_i)} \tag{2.5}$$

The perplexity is therefore lower for words, which are very likely to follow after another and a low perplexity of a sequence is a reasonable estimate of a good grammaticality.

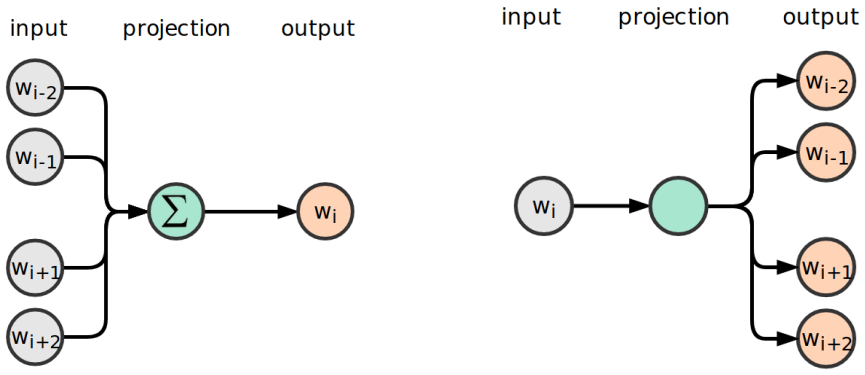
One of the first neural LM was proposed by Bengio et al. [6] and uses a feed-forward architecture which needs a pre-defined history length. This was overcome in 2010 when Mikolov et al. [59] introduced a language model based on a recurrent neural network architecture. RNN language models became the state-of-the-art and recent approaches are still using related architectures.

Embedding words to preserve some kind of semantic meaning is not only important for language modeling, but is also a core aspect of many NLP applications. Hence, the next section will introduce methods to learn word embeddings.

## 2.6 Word embeddings

Whereas the previous section introduced embeddings to preserve semantic relationships of words in the sole context of language modeling, this section summarizes some of the most influential work done to obtain stand-alone word representations. Traditionally, these systems are not obtained from language models, but recent approaches (see Section 2.7) achieve state-of-the-art results by extracting the weights of an LM as embeddings.

**word2vec** is a framework to obtain embeddings proposed by Mikolov et al. [58, 57], which is credited for the strong increase in popularity of using embeddings in various NLP tasks. This is due to two reasons. Firstly, it simplifies the embedding generation task by treating it as a linear classification by asking "is word $w_i$ likely to appear in the context words $w_{i\pm n}$" [39]. This is less complex than predicting the next word, as there is an almost infinite amount of word combinations in natural language. Secondly, it reduces the computational cost by approximating the softmax using the *hierarchical softmax* from Morin and Bengio [61]. Word2vec consists of two neural network architectures, the *continuous-bag-of-words* (CBOW) and the *skip-gram* architecture, which are illustrated in Figure 2.2.




**(a)** CBOW **(b)** Skip-gram

**Figure 2.2:** Architectures of the proposed models from Mikolov et al. [58, 57] with an exemplary context size of 5. The CBOW model is trained to predict the word $w_i$ based on the sum of the previous and the future words in its context. The Skip-gram works the other way around by trying to predict the context words based on the center word $w_i$.

**Continuous-bag-of-words** uses a similar architecture to a feed-forward neural language model without a non-linear hidden layer. The projection layer is shared for all inputs. It aims to predict the center word $w_i$ based on the sum of the context words $w_{i-c}, ..., w_{i-1}, w_{i+1}, ..., w_{i+c}$ in a window of size $2c$.

$$-\frac{1}{n}\sum_{i=1}^{n} logP(w_i|w_{i-c}, ..., w_{i-1}, w_{i+1}, ..., w_{i+c}) \tag{2.6}$$

**Skip-gram** works the other way around and aims to find word representations to predict the context words of word $w_i$ by maximizing the average log probability:

$$-\frac{1}{n}\sum_{i=1}^{n}\sum_{-c\leq j\leq c, j\neq 0} logP(w_i+j|w_i) \tag{2.7}$$

A skip-gram model trained on a large corpus as well as the source code can be obtained from the official word2vec site[1]. It uses the very common context size of four words before and after the center word.

**Related approaches** to word2vec are *Global Vectors* (GloVe) from Pennington et al. [69], which learns to construct a low-dimensional representation of a co-occurrence matrix (i.e how often a word occurs in a given context) and *fastText* from Bojanowski et al. [10], which extends word2vec by representing words as the sum of their characters' embeddings.

One major issue with these embedding techniques is that they are not *context-sensitive*. This means, a word has only one representation without including the surrounding context. This can lead to problems for words with multiple meanings e.g. the word *fair*, which can either mean equal treatment, a local festivity or be used to describe light hair color. Recent approaches like *Embeddings from Language Models* (ELMo) from Peters et al. [71] and *Bidirectional Encoder Representations from Transformers* (BERT) from Devlin et al. [20] address this by extracting context-sensitive embeddings from a language model. To understand how these approaches work, the following section provides a brief overview of *Sequence to Sequence processing* and the concept of *attention* in neural networks.

## 2.7 Sequence to Sequence processing

Many problems in NLP involve the transformation from one sequence to another. These tasks belong to the category of *Sequence to Sequence* (Seq2Seq) tasks and include for example machine translation, language modelling and text summarization. Traditionally, recurrent neural networks (see Section 2.4) are employed to handle sequences. Using RNNs for Seq2Seq tasks has two major drawbacks. Firstly, long-term dependencies cannot be handled due to the *vanishing gradient problem* and secondly, neural networks need a fixed-length representation for both input and output. The former problem is solved by employing a variation of the RNN architecture called *Long Short-Term Memory* (LSTM) network from Hochreiter and Schmidhuber [32]. The latter problem can be solved by employing an architecture design pattern called the *encoder-decoder* model, which allows variable lengths of input and output.

**Encoder-Decoder** The network is separated into two recurrent networks, the *encoder* and the *decoder*. The encoder, as the name implies, maps a sequence of tokens $(x_i, ..., x_n)$ to a fixed-length *hidden state* $h$. With $h$ as input, the decoder generates an output sequence $y = (y_0, ..., y_m)$. Starting from the first input token $x_1$, a transformation to the hidden state $h_1$ is calculated. All subsequent hidden states $h_2, ..., h_n$ take their respective input $x_i$, with $2 \leq i \leq n$, and the previous hidden state $h_{i-1}$ into account. The encoder's last

[1] `http://code.google.com/p/word2vec`

hidden state $h_n$ is referred to as *context vector* $c$ [2]. The context vector is used as input for $y_0$ and all hidden states of the decoder. Figure 2.3 illustrates the encoder-decoder architecture.

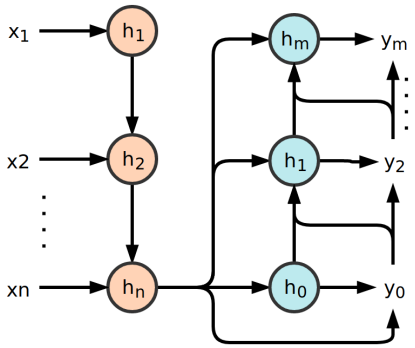



**Figure 2.3:** Encoder-decoder architecture, drawn after Kamath et al. [40]. The input tokens $xi, ..., x_n$ are mapped to the encoders fixed-length hidden state $h$ and the last hidden state $h_n$ is used as input for the first output token $y_0$ and the decoders hidden states $h$ Kamath et al. [40].

Given that the context vector has a fixed size, longer sequences become increasingly difficult to encode and long-term dependencies can get lost. To overcome this, Bahdanau et al. [2] propose a modified encoder-decoder architecture, which uses an *attention* mechanism to attend to different parts of the input sequence instead of encoding it into a context-vector.

### 2.7.1 Attention

Similar to how neural networks are inspired by the biological brain, the attention mechanism is loosely inspired by the visual attention of humans, where some regions are focused on and others are blurred out. This relieves the encoder from storing all information in the context-vector $c$ and instead computes $c_j$ as the weighted average of the hidden states $h_j$ of each element in the input sequence with respect to the current output hidden state $h_i$:

$$c_j = \sum_{i=1} \alpha_{ji} h_i \tag{2.8}$$

where a variable-length *alignment vector* $\alpha_{ji}$ is derived from the *attention score* of the current output hidden state $h_i$ with each input hidden state $h_j$ [2, 52]:

$$\alpha_{ji} = \frac{exp(score(h_i, h_j)}{\sum_j exp(score(h_i, h_j))} \tag{2.9}$$

Attention score functions can be categorized as either *additive attention* or as *multiplicative attention*. The former, as proposed by Bahdanau et al. [2] defines the score as:

$$score(h_i, h_j) = v_a^\top \tanh W_a[h_i, h_j] \tag{2.10}$$

where $v_a^\tau$ and $W_a$ are learned parameters of a single layer. Luong et al. [52] in contrast define the multiplicative attention as either the dot-product of $h_i$ and $h_j$ or a *general* score

with an additionally learned parameter $W_a$:

$$score(h_i, h_j) = \begin{cases} h_i^\top h_j & \text{dot-product} \\ h_i^\top W_a h_j & \text{general} \end{cases} \tag{2.11}$$

Figure 2.4 illustrates a conceptual architecture of a sequence to sequence model with an attention layer [95].

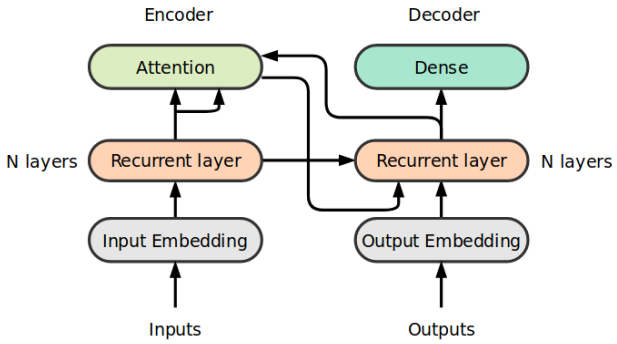



**Figure 2.4:** Sequence to sequence model with attention, draw after Zhang et al. [95].

Although the attention mechanism solves the problem of encoding longer sequences, the sequential nature of recurrent neural networks still proofed to be a bottleneck. To address this, Vaswani et al. [90] proposed the *Transformer* architecture, which is an encoder-decoder network that relies on an attention mechanism called *self-attention* and replaces the recurrent layers with simpler, fully connected layers.

### 2.7.2 The Transformer

The *Transformer* architecture, as proposed by Vaswani et al. [90], follows the general architecture of an encoder-decoder network, where all recurrent layers are replaced by a combination of a self-attention layer and a feed-forward layer. The self-attention layer uses *multi-head attention*, which differs from regular self-attention by using the concatenation of multiple scaled dot-product attention values.

Without recurrent layers, the model can easily be parallelized and therefore can use a lot more data during both training and inference. Several of these layer combinations are grouped into an encoder or decoder block. The Transformer consists of a stack of $N$ of these blocks, Vaswani et al. [90] recommend $N = 6$ but others use a similar architecture with dozens of blocks. To compensate for the lack of sequence information without recurrent layers, an additional positional encoding is used as input. This positional encoding maps the input tokens to a sequence of numbers, indicating the token position in a sentence. Figure 2.5 illustrates the architecture of the Transformer model.

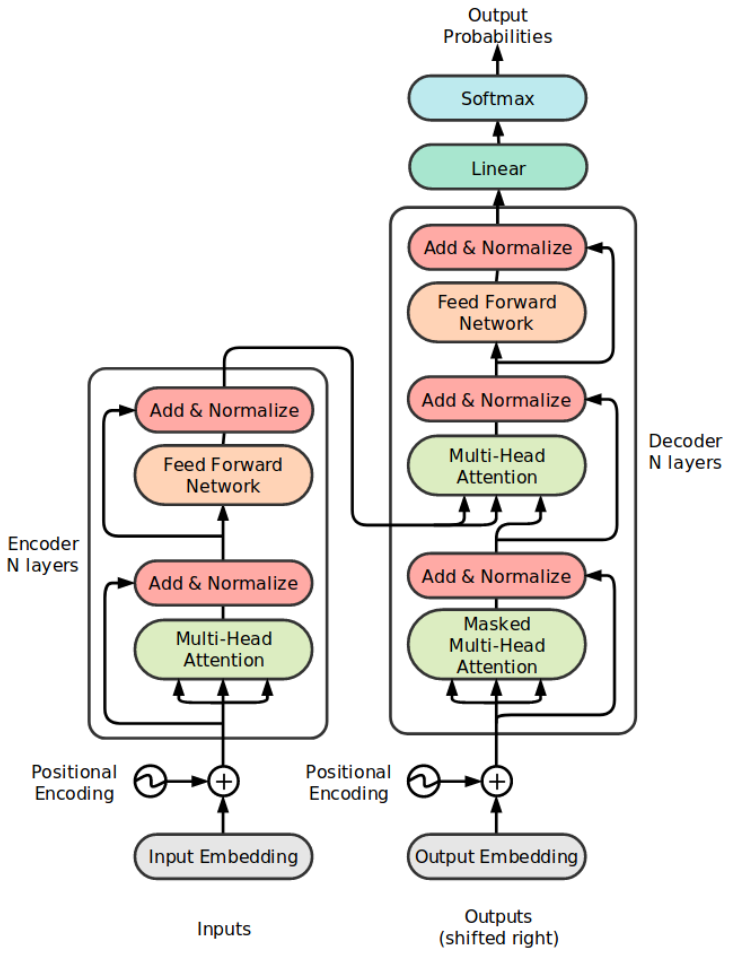



**Figure 2.5:** The encoder-decoder architecture of the *Transformer* model, drawn after Vaswani et al. [90]. Both the encoder on the left side and the decoder on the right side are composed of $N$ identical layers.

The Transformer architecture achieved state-of-the-art results in several NLP tasks [90] and variations of it are found in many of the most recent models. Some of these recent models are briefly introduced below.

**Generative Pre-Training 2 (GPT-2)** is a transformer-based unidirectional language model proposed by Radford and Salimans [73]. It uses a two-step training procedure of unsupervised pre-training and supervised fine-tuning. This separation of pre-training and fine-tuning has become very popular in various image processing tasks after the release of the *ResNet* model from [30]. ResNet is trained on a large amount of images and outperformed the previous state-of-the-art models in many tasks after only a few iterations of fine-tuning. Fine-tuning allows GPT-2 to train once and adapt to various tasks. It consists of 12 Transformer decoder-only blocks and is trained on the *BookCorpus* dataset [97].

**Bidirectional Encoder Representations from Transformers (BERT)** is a transformer-based encoder-only model from Devlin et al. [20], which learns text representations. These

representations can either be used as word-embeddings or can be fine-tuned to solve various NLP tasks. BERT is trained using two training objectives. Firstly, *masked language modeling*, where 15% of the words are replaced by a special token and the objective is to predict the masked word and secondly *next sentence prediction*, where the model is given two sentences $A$ and $B$ and the objective is to predict if $B$ actually follows $A$ [20]. BERT distinguishes itself from GPT-2 by reading the entire input sequence (up to a maximum length) at once, which results in a bidirectional representation.

**Recent transformer-based systems** include *Transformer XL* from Dai et al. [17], which focuses on learning longer dependencies and removes the fixed-length context, XLNet from Yang et al. [93], which combines ideas from BERT and Transformer XL, RoBERTa from Liu et al. [50], which optimizes the pre-training of BERT to an extend that it can match the results of the other models, and *DistillBert* from Tang et al. [86], which is a version of BERT optimized to be fast and more light-weight that uses only one fifth of the parameters BERT uses.

The following chapter builds upon these theoretical foundations and provides an overview of related work in multi-document summarization and summary evaluation.

# 3 Related work to summarization

This chapter introduces the field of text summarization and focuses primarily on multi-document summarization systems. The first section provides an overview of related work and identifies the most influential work in order to evaluate the performance of the proposed system in Chapter 5. The second section explains common metrics to assess the performance of summarization systems with the quality aspects defined in Section 2.2 in mind.

## 3.1 An overview of multi-document summarization techniques

Although early text summarization efforts date back to 1958 when Luhn [51] and Edmundson [21] first used statistical techniques on single documents based on word positions and cue words, multi-document summarization has only been widely applied in the last two decades. Compared to single documents, redundancy plays a major role in multi-document settings, since related documents often share facts and concepts. Goldstein et al. [27] were one of the first to provide summaries for multiple documents and solved the redundancy problem by applying the concept of *Maximal Marginal Relevance* (MMR). MMR is a greedy approach to select sentences that are both salient and non-redundant. Salient in this context means that these sentences are important for the overall meaning of the text and they contain valuable information.

**Graphs** Erkan and Radev [22] proposed the well-known LexRank algorithm, which selects important sentences based on the concept of *Eigenvector centrality*. Sentences form the nodes in a undirected graph structure and edges, which are weighted by the sentences similarity, connect all nodes with one another. Summaries are formed through a random walk on the graph on which similar sentences have a high transition probability. Mihalcea and Tarau [56] proposed TextRank, which is very similar to LexRank except that it was designed for summarizing single documents. Ganesan et al. [26] used a similar graph-based approach that was adapted to use word units instead of sentences and directed edges to represent sentence structure in their *Opinosis* system.

**Sentence compression & fusion** Barzilay and McKeown [5] employ a technique called *sentence fusion* to create summaries. Sentence fusion is about generating sentences, which aim to include all salient information in a cluster of similar sentences. Filippova [24] extend this approach with a graph-based approach, in which fused sentences are generated by finding a set of shortest paths in the content clusters. Boudin and Morin [11] improved Filippova's approach [24] by ranking the fused sentences based on keywords they contain.

Nayeem et al. [62] use a modified version of TextRank to select fused sentences from a word graph whose edges are weighted based on the similarity between sentence embeddings.

**Centroid based** Rossiello et al. [77] identify the document centroid based on *term frequency-inverse document frequency* (TF-IDF) weights and word embeddings. Salient sentences are selected by measuring the similarity to this centroid. Ma et al. [53] share the basic idea of measuring the similarity to a centroid but apply it on a paragraph level instead of on document level. Beam search is used to generate the summary from a set of candidate sentences.

**Integer Linear Programming (ILP)** Banerjee et al. [3] initially identify the most important document and turn each of its sentences into a separate cluster to which all other sentences are aligned. A word graph generates candidate sentences from which the top sentences are selected using integer linear programming. The optimization goal is to maximize information content, which is based on the sentences' TextRank score, and minimize redundancy. A similar approach is used by Bing et al. [7] where sentences are broken down into verb and noun phrases which are ranked by their redundancy. New valid sentences are then generated using ILP. Tuan et al. [89] improve the performance of this approach by including a syntax factor.

**Neural Networks** Some very recent systems use a neural-networks-based approach. Chu and Liu [14] employ a LM (see Section 2.5) trained on the summary dataset and two *auto-encoders* with tied encoder/decoder for selecting relevant content. Yasunaga et al. [94] use a *graph convolutional neural network* (GCNN) to estimate the salience of sentences and extract highly salient sentences to form the summary. Li et al. [46] propose the use of *variational auto-encoders* (VAEs) to estimate salient sentences and reconstruct abstractive summaries from the VAEs latent semantic space. Schumann [78] employs a very similar approach and focuses on producing very short summaries.

## 3.2 Summary evaluation

The preceding section introduced the past and recent work done in multi-document summarization. A well-performed evaluation is crucial to compare and improve upon these works. Therefore, this section explores the field of summary evaluation, which is the baseline for the decisions made in Chapter 5. Gambhir and Gupta [25] define a taxonomy to classify different evaluation techniques, which incorporates the summary quality aspects defined in Section 2.2.

As Figure 3.1 illustrates, evaluation measures are categorized into *extrinsic* and *intrinsic* metrics. The former is task-specific and determines a summary's quality by how well it performs on a given task. Steinberger and Ježek [84] define three important extrinsic tasks, namely question answering, information retrieval and text classification.
Intrinsic evaluation is task-agnostic and rates both text quality and how well the summary covers important aspects of its source. Quality measurements are commonly based on

linguistic features, e.g. grammaticality, non-redundancy and coherence. Informativeness is usually based on how well a generated candidate summary represents a human-written reference summary or the concepts it covers. The focus of this thesis lies on intrinsic evaluation measures as these features can evaluate a summary on their own.

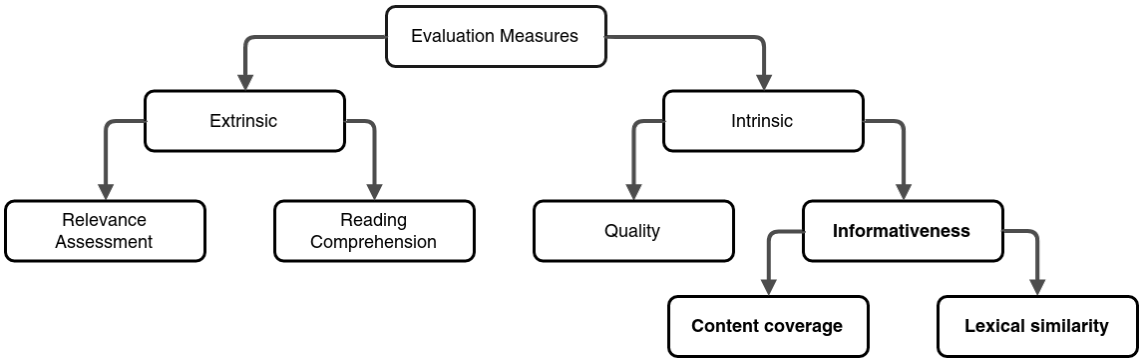



**Figure 3.1:** Evaluation measures taxonomy by Gambhir and Gupta [25]. Generally, evaluation measures are divided into extrinsic (task-specific) and intrinsic (task-agnostic) measurements. This thesis focuses on the latter, which includes the quality aspects defined in Section 2.2.

### 3.2.1 Informativeness

Informativeness metrics are divided into two categories: *lexical-similarity-based* and *content-coverage-based* evaluation. Relevance, although not included in Gambhir and Gupta's taxonomy, is often evaluated with lexical-similarity-based metrics and therefore no additional section is included. The following section summarizes common metrics, their limitations and current research in this area.

#### 3.2.1.1 Lexical similarity

Lexical-similarity-based metrics follow the idea that the closer a candidate summary is to a given human-written reference summary in terms of word-overlap, the more relevant information it contains. Following the foundations laid by Edmundson [21], earlier summarization systems adapted three very intuitive metrics from the field of *Information Retrieval* (IR): *precision*, *recall* and *f1-score*. The latter is the harmonic mean of recall and precision.

**Precision** denotes how many n-grams (see Section 2.2) in the candidate summary appear in the reference summary.

$$\text{precision} = \frac{\text{reference n-grams} \cap \text{candidate n-grams}}{\text{candidate n-grams}} \tag{3.1}$$

Given a reference (a) and a candidate sentence (b), which slightly differs in the chosen words but conveys the same information:

*(a) The dog walked around in the village*
*(b) The dog walked through the settlement*

The precision score would be $\frac{4}{6}$ for uni-grams ($n = 1$), as four out of six words occur in the reference summary and the candidate summary, divided by the total number of uni-grams in the candidate summary. For bi-grams ($n = 2$), the score would be $\frac{2}{5}$, as only the word pairs "The dog" and "dog walked" occur in both summaries.

While comparing to a reference summary can work well in some cases, using precision in text summarization can lead to high scores without the candidate summary containing much information of the reference summary, as the following short sentence (c) illustrates:

*(c) The dog the dog the dog the dog*

This sentence would result in a perfect precision score of $\frac{8}{8}$ for uni-grams and a still very high score of $\frac{4}{7}$ for bi-grams, as all n-grams in the candidate sentence exist in the reference sentences. To overcome this, Papineni et al. [67] developed the *BLEU* metric.

**Bilingual Evaluation Understudy (BLEU)** was originally developed to evaluate machine translation systems, but has been widely adapted in the summarization field. Its basic idea is to calculate the precision, but it improves upon earlier works by counting the maximum n-gram occurrence in any reference summary and introducing a *best match length*. BLEU is designed to work with a corpus of text, including multiple reference summaries, since information can be conveyed in multiple ways (e.g. by paraphrasing, the use of synonyms etc.) [63]. By design, precision penalizes only too long sentences. To also penalize too short candidate sentences, BLEU introduces a *brevity penalty* (BP) which is defined as:

$$\mathrm{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases} \tag{3.2}$$

where $c$ is the length of the candidate sentence and $r$ is the length of the reference sentence.

The final score is defined as:

$$\mathrm{BLEU} = \mathrm{BP} \times \exp\left(\sum_{n=1}^{N} w_n \log p_n\right), \quad \text{where} \quad w_n = \frac{1}{n} \tag{3.3}$$

where $p_n$ is the *modified n-gram precision*, in which the candidate n-grams are clipped to the maximum occurrence in the reference corpus and $N$ is the maximum sequence length

of n-grams considered (usually 4).
Returning to example (c), BLEU would penalize the redundancy by limiting the precision to the count of *the* = 2 and *dog* = 1 in the reference sentence, resulting in a uni-gram score of $\frac{3}{8}$ and bi-gram score of $\frac{1}{7}$.

**Recall** denotes how many n-grams in the reference summary exist in the candidate summary and is therefore the natural complement of precision.

$$\text{recall} = \frac{\text{reference n-grams} \cap \text{candidate n-grams}}{\text{reference n-grams}} \tag{3.4}$$

Recall usually favors longer candidate sentences, as it is normalized by the n-grams of the reference sentence. Example (d) scores a perfect score for both uni-gram and bi-gram recall, as the additional information does not affect it, which can lead to unfocused and unnecessarily long sentences.

*(d) The dog walked around in a hurry, because he could not find his owner in the village*

**Recall-Oriented Understudy for Gisting Evaluation (ROUGE)** is not a single metric but a set of metrics and it is the most widely accepted mean of comparing results from different systems. It was inspired by BLEU but focuses on the harmonic mean of recall and precision, the so called f1-score. Several variants exist with the most prominent being ROUGE-N, which is commonly calculated from multiple references:

$$\text{ROUGE-N} = \frac{\sum_{\text{s} \in \text{references}} \sum_{\text{n-gram} \in \text{s}} \text{n-gram} \cap \text{candidate n-grams}}{\sum_{\text{s} \in \text{references}} \sum_{\text{n-gram} \in \text{s}} \text{n-gram}} \tag{3.5}$$

It is commonly calculated for $N = 1$ (unigrams) and $N = 2$ (bigrams). Other ROUGE metrics include *ROUGE-L* where the *longest common subsequence* between a candidate sentence and a reference sentence is calculated and *ROUGE-SU4*, which calculates bigram-overlap with up to 4 n-gram gaps in between.

**Limitations of lexical similarity** ROUGE and BLEU are generally considered to correlate well with human judgements (>90%) according to Owczarzak et al. [66] when ranking different summarization systems. The bi-gram versions is performing better than uni-gram versions. Depending on the data and variant used, Steinberger and Ježek [84] and Gambhir and Gupta [25] argue that this can drop below 50% correlation. Despite being used very frequently, both metrics are criticized for not taking semantic similarity into account, making them less suitable for paraphrasing and highly abstractive systems, which try to imitate the way humans perform summaries. Grammaticality and sentence structure are also not evaluated, which are important factors for system acceptance.

**Current research** Ng and Abrecht [64] try to address the paraphrasing problem by weighting n-grams by their word embeddings (see Section 2.6). This allows for sentences without direct overlap to get high scores when n-grams are semantically related. Their new metric is called *ROUGE-WE*. A similar approach is used by ShafieiBavani et al. [80] in their work on *ROUGE-G*, which adds a graph-based semantic similarity ranking of n-grams to allow for paraphrasing and fairer comparison of extractive and abstractive summaries. Both are shown to correlate better with human judgments than regular ROUGE.

A very recent approach from Böhm et al. [9] learns to rank summaries by fine-tuning BERT (see Section 2.7) to imitate human evaluation. It shows a higher correlation with human judgements than common lexical-similarity (ROUGE, BLUE, METEOR) [9] and can be used without references summaries.

#### 3.2.1.2 Content coverage

While the preceding section covered metrics based on n-gram overlap on a lexical level, this section introduces several content-coverage based metrics. These metrics metrics try to overcome the "shallow" evaluation on lexical features by measuring how well the topics and facts of a summary represent its source. Earlier summary systems were evaluated by human annotators, who rated each candidate summary based on a set of criteria with a predefined scale (e.g. 1=worst to 5=best). Naturally, this led to very subjective evaluations, as often two annotators would not agree on a grade. Lin and Hovy [47] report that human annotators agreed in only 82% of the cases with their own prior judgement. The following sections cover methods that address these issues and promote more objective evaluations.

**Factoid Method** The Factoid Method proposed by Teufel and Van Halteren [87] is based on small information units called *factoids*. A factoid is a small textual element, which represents the meaning of a sentence and human annotators can assign each sentence one or multiple factoids. The following example from [87] illustrates how the sentence "The police have arrested a white Dutch man" is represented using factoids:

- A suspect was arrested
- The police did the arresting
- The suspect is white
- The suspect is Dutch
- The suspect is male

These elements are then weighted based on the information overlap they contain with other factoids. Summaries are evaluated by how many weighted factoids they contain.

**Basic Elements** (BE) Hovy et al. [36] use a very similar approach but they try to remove the need for human annotators by automatically selecting important elements of summaries and scores accordingly. Here, a sentence is segmented into small units of content, which can be either a single word or a relationship triplet. These triplets consist of

a *head*, which is the subject, a *modifier*, which is the object, and their relationship. For example, a content unit for the word sequence "student writes thesis" the corresponding triplet would be (student|thesis|writes). Summaries are scored based on the triplet overlap in candidate and reference summary. Similar approaches have been developed by Steinberger and Jezek [83] called *Latent Semantic Analysis evaluation* and from Cohan and Goharian [15] by the name of *Summarization Evaluation by Relevance Analysis* (SERA). Both approaches score summaries on how well the representation of each topic in the source matches the representation in the candidate summary.

**Pyramid Method** is a semi-automatic technique that has been widely used since its introduction in DUC 2006[1]. Nenkova et al. [63] follow the idea that a system can be evaluated based on how many *summary content units* (SCUs), considered important by human annotators, are contained in a candidate summary. These SCUs are created by identifying similar phrases and ranking them in a pyramid model according to how often they occur in the reference summaries. The phrases, which form an SCU can be as short as a single word or as long as a whole sentence, as the same information can be conveyed in different ways. For example, an SCU of weight $w = 4$ might consist of the following facts from four different documents:

- **SCU1 (w=4):** *The dog moves around the village*
- the dog took a stroll in the settlement
- Dog Buddy, ..., walked the village's streets
- ... where his dog walked around
- he ... walking through the village

A pyramid model has $n$ levels with $n$ being the number of reference summaries used in the evaluation. A weight is assigned to each SCU corresponding to the number of human assessments, which identify the same content. For example, an SCU that occurs in two out of four reference summaries would be placed on the second level from the bottom. Figure 3.2 shows a pyramid with four levels and a candidate summary is evaluated to contain all of the most important SCUs and two out of four from the third level. An optimal summary should first and foremost contain all SCUs from the highest level and then, if lengths limits permit it, descend to the lower levels.

[1]Document Understanding Conference - `https://duc.nist.gov/`

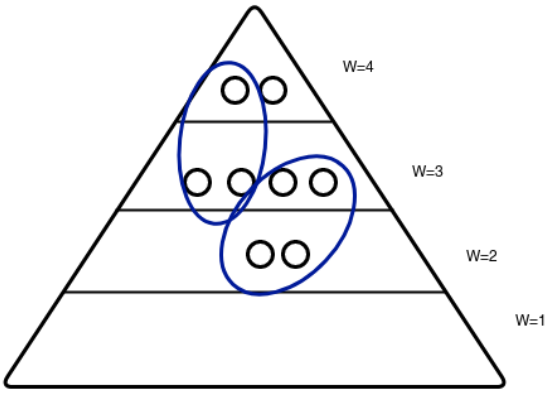



**Figure 3.2:** Example pyramid with matching SCUs selected adapted from Nenkova et al. [63]. Higher levels indicate more important SCUs, based on the fact that they are mentioned in multiple reference summaries.

The optimal content score for a summary with $X$ SCUs on a pyramid with $n$ levels $L_1, ..., L_n$ is calculated as:

$$\text{Max} = \sum_{i=j+1}^{n} i \times |L_i| + j \times \left( X - \sum_{i=j+1}^{n} |L_i| \right), \text{where j} = \max_i \left( \sum_{l=i}^{n} |L_l| \geq X \right) \tag{3.6}$$

The pyramid method is known to correlate well with human judgements, but it needs well trained human annotators to identify and match SCUs in both the candidate and the reference summaries. To address this, several attempts to automate the pyramid evaluation have been made.

Passonneau et al. [68] developed an algorithm to automate the process of scoring summaries with manually annotated SCUs. This is done using distributional semantics and scores are assigned if the low dimensional vectors of an SCU and an n-gram exceed a similarity threshold. Steinberger et al. [85] proposed a similar system to automate the evaluation of manually-created SCUs using an *abstract meaning representation graph*. Yang et al. [92] proposed the *Pyramid Evaluation Via Automatic Knowledge Extraction* (PEAK) framework, which does both automatically extract SCUs from a source corpus and score candidate summaries on how many SCUs they contain. Peyrard and Eckle-Kohler [72] proposed a genetic summarization system, which approximates pyramid scores. They noted that summaries, which achieve high pyramid scores from PEAK, receive low ROUGE scores and vice versa.

### 3.2.2 Quality

Quality metrics aim to measure two important aspects: *fluency* and *coherence*. Text quality assessment is not as often reported as informativeness score in summary evaluations, but is an important element that differentiates keyword lists from well-written texts.

#### 3.2.2.1 Fluency

Fluency, which also known as *readability* or *grammaticallity* is commonly evaluated using the perplexity of a language model (see Section 2.5). As the perplexity indicates how well a model can predict a sample, it can estimate how grammatically correct a sentence is, based on how likely one word follows another. Since this is based on follow-up word likeliness, it assigns high values (lower is better) to words, which are uncommon or simply rare in a given context. To correct this, Kann et al. [41] propose the *SLOR* metric, which normalizes the perplexity of a sentence by the unigram log-probability and sentence length. For a given sentence $S$, the SLOR score as computed by a language model is defined as:

$$SLOR(S) = \frac{ln(p_{LM}(S)) - ln(p_u(S))}{|S|} \tag{3.7}$$

where $p_{LM}(S)$ is the probability for a sentence and $p_u(S)$ the unigram probability which is the product of the probabilities for each token in a sentence. This allows sentences with uncommon words to gain good scores and therefore is used in the evaluation.

**Grammar parsers** are another way of evaluating the grammar of a given text. Commonly, the text is annotated with POS tags and the annotated text is checked against a manually created set of rules. Identifying these rules is expensive, as it requires in-depth knowledge of a certain language, but these systems excel at providing explanations as to why a certain sentence contains a grammatical error. With the availability of large documents for training, language models have become more popular [82].

#### 3.2.2.2 Coherence

Coherence metrics typically rank the structure of a text and how well information or concepts traverse from one sentence to the next.

**Follow-up sentence similarity** is an approach proposed by Lapata and Barzilay [44]. A text is considered to be coherent if the follow-up sentences are similar to each other. The coherence metric for a document $D$ consisting of $n$ sentences $S_1, ..., S_n$, is defined as the sum of the normalized sentence similarity function $sim$:

$$coherence(D) = \frac{\sum_{i=1}^{n-1} sim(S_i, S_{i+1})}{n-1} \tag{3.8}$$

This presupposes a high-quality similarity function. Lapata and Barzilay [44] used a word-overlap-based similarity function that is up to date with current research. Modern vector representations promise improvements in terms of correlation of this metric with the evaluation from human experts.

**Entity grids** are a form of text representation, where each column corresponds to an entity (e.g. a proper noun) and each row corresponds to a sentence [5]. The grid's cells

reflect the grammatical role of an entity in this sentence (e.g. subject, object etc.). Based on the assumption that coherent texts follow certain entity distributions, Barzilay and McKeown [5] define the coherence of a document $D$ with entities $e_1, ..., e_n$ as the joint probability distribution of entities across the documents sentences $S_1, ..., S_m$:

$$coherence(D) = \prod_{i=1}^{n} P(e_i; S_1, ..., S_m) \tag{3.9}$$

With this overview of summarization systems and evaluation metrics in mind, the next chapter presents the approach employed in this thesis.

# 4 Approach and implementation

This chapter covers the main approach and its implementation for this thesis. The approach acts as a baseline for the experiments conducted in Chapter 5 and improvements are proposed in Chapter 6. This chapter is structured as follows: First, the system's architecture will be introduced on a conceptual level and the motivation and intuition behind this design will be explained. Then, the following two sections will dive deeper into the inner workings of each component.

## 4.1 Architecture overview

The proposed system is based on the intuition that, by using the output of an extractive approach as input for an abstractive approach, the resulting system can take advantage of the strengths (see Section 2.1) of both approaches without suffering from the disadvantages.

Multi-document summarization begins with a set of documents with similar content as input. The summarization now works in two steps: in the first part, each of these documents is summarized extractively and stored temporarily. The summaries are then used as input for the second part, which produces a single abstractive summary from the set of temporary summaries. Figure 4.1 shows how the two main parts interact.

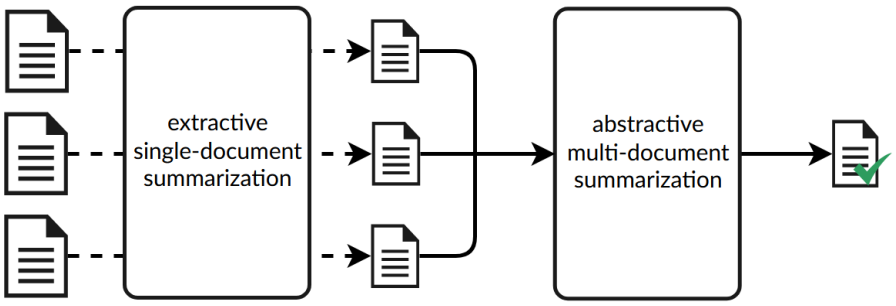



**Figure 4.1:** A high-level view of the system's architecture

To implement this, *python* version 3.7 and the machine learning framework *PyTorch*[1] are used.

[1] `https://pytorch.org/`

## 4.2 Extractive single-document summarization

The extractive part's architecture is based on the *SummCoder* model from Joshi et al. [37]. It is a single-document summarization system, so each input document will be processed separately. The model works by assigning a score to each sentence and selecting the top $N$ sentences to form a summary. The whole process will be briefly explained in the following paragraphs and the remainder of this section will explore each step in more detail. Figure 4.2 illustrates the architecture.

**Preprocessing:** first, the input text is "cleaned" by replacing non-alphabetic characters and accented characters with a normalized version and splitting the text into sentences.

**Encode sentences:** these clean sentences are then encoded into high-dimensional vectors called sentence embeddings (see Chapter 2.6). Using these embeddings, three metrics are calculated, which make up the sentence score.

**Position metric:** assigns high scores to sentences at the beginning or at the very end of the text, as those tend to contain important information.

**Relevance metric:** indicates how important a sentence is for the meaning of the whole document.

**Novelty metric:** indicates how novel the information in a sentence is to reduce redundancy and assigns high scores to sentences whose embeddings differ from those of the other sentences. An equally novel sentence is preferred if it contains relevant information.

**Combine scores:** each sentence is ranked based on a combination of the position, novelty and relevance metric. The top sentences are then selected to form the summary.

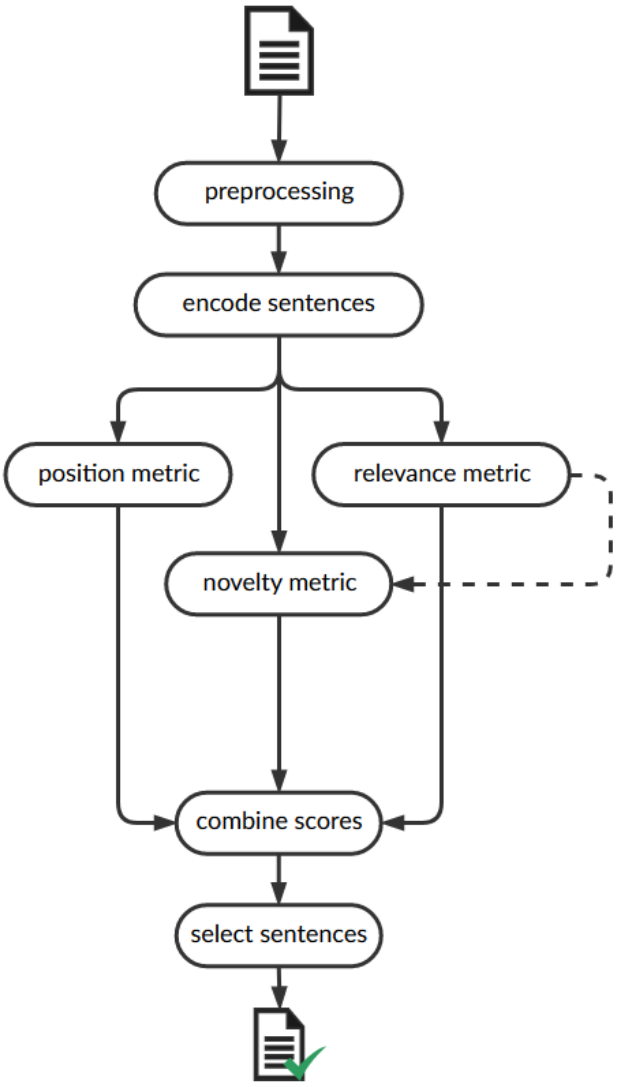



**Figure 4.2** Architecture of the extractive part, based on Joshi et al. [37].

### 4.2.1 Pre-processing

Pre-processing is an important step to improve the data quality and is implemented using *spaCy*[2]. SpaCy is an NLP library developed by Honnibal and Johnson [35], which focuses on speed and ease of development but lacks the flexibility of other NLP libraries like *NLTK*[3]. As a first step, non-alphanumeric characters (e.g. accented characters) are removed or replaced with an alphanumeric equivalent. The next step is to remove all special characters that are not used as punctuation marks. The last step is to convert the text to lower-case and split it into a list of sentences. The lower-casing is important for the sentence fusion word graph to work, as it allows it to merge words from the sentence start with words from within a sentence.

### 4.2.2 Sentence encoding

The clean sentences are encoded into a high-dimensional vector representation using the *skip-thoughts* model from Kiros et al. [42]. Skip-thoughts consists of a recurrent neural network, which takes a sentence as input and is trained to reconstruct both the previous and the next sentence from it. Figure 4.3 shows a triplet of sentences, which is used to learn the sentence representation.

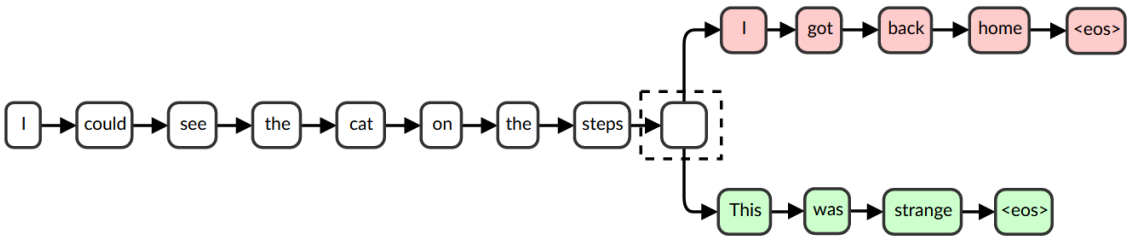



**Figure 4.3:** Example skip-thought sentence triplet from Kiros et al. [42]. *<eos>* represents the end of a sentence.

### 4.2.3 Sentence relevance metric

The content relevance metric aims to assign high values to sentences, which have a high impact on the meaning of the whole document. Low values are assigned to those sentences that could potentially be left out, without losing much information. To achieve this, a mechanism to compute the relevance of a sentence in a document and a document representation are needed. The basis of the document representation are the sentence embedding vectors, which are one-dimensional vectors with $M$ entries. To get to the document representation, two transformations need to be performed. First, all sentence embeddings are concatenated on the second axis, which results for $N$ sentences in a two-dimensional matrix with the shape $(N, M)$. The second step is to use this matrix as input for a deep *auto-encoder network* to compress the information from all sentences back into a

[2] `https://spacy.io/`
[3] `https://www.nltk.org/`

one-dimensional vector again. The intuition behind this is, that by compressing the data, only the most relevant information is encoded and irrelevant information is left out.

**Deep auto-encoder** is a neural network architecture proposed by Hinton and Salakhutdinov [31]. The term *deep* implies that it consists of more than two layers. It is used to convert high-dimensional data into a low-dimensional representation and can be thought of as a non-linear generalization of the *principal component analysis* (PCA). It consists of two symmetrical *feed-forward networks*, which share a central layer, storing the *latent representation*. The first part called the *encoder* is trained to learn a lower-dimensional representation of the input data and the second part, called *decoder* is trained to reconstruct the input data from the lower-dimensional representation.

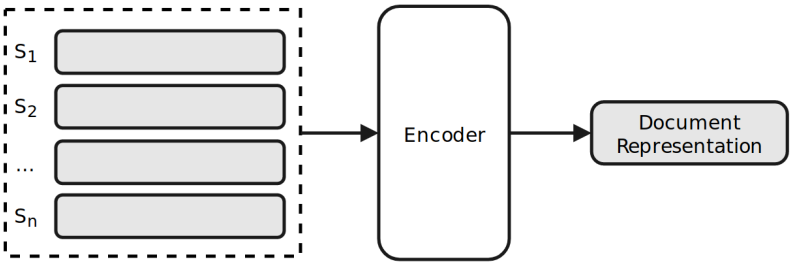



**Figure 4.4:** Conceptual processing pipeline of the document embedding creation.

The auto-encoder used for the document representation consists of four layers for the encoder and decoder and is first trained on a set of documents with the goal to minimize the reconstruction loss. After training, the decoder is discarded since the metric only depends on the latent representation. The document representation $\hat{D}$ for a given document $D$ is computed by feeding the two-dimensional embedding matrix of all sentences into the auto-encoder network.

To compute the relevance of a sentence $S_i$, the document representation of the whole document is compared with a modified document representation $mod\hat{D}_{S_i}$, which excludes sentence $S_i$:

$$mod\hat{D}_{S_i} = \hat{D} - \hat{S}_i \tag{4.1}$$

The original representation is now compared to the modified one to define the relevance score:

$$\text{relevance}(D, S_i) = 1 - \frac{\hat{D} \cdot mod\hat{D}_{S_i}}{||\hat{D}||||mod\hat{D}_{S_i}||} \tag{4.2}$$

### 4.2.4 Sentence position metric

The sentence position was one of the first metrics to indicate the relevance of a given sentence. For a sentence $S_i$ in a document $D$, Joshi et al. [37] define it as:

$$position(D, S_i) = max(0.5, exp(\frac{-P(S_i)}{\sqrt[3]{N}})) \tag{4.3}$$

The values of $position(D, S_i)$ are bound in range $[0.5, 1.0]$ and the function $P$ returns the relative position of a sentence in its containing document starting from 1.

### 4.2.5 Sentence novelty metric

The idea behind the novelty metric is to identify which sentences in a given text contain new information and which contain redundant information. This is obtained by measuring how alike the sentence embedding vectors are. Therefore, choosing a good sentence representation is critical for this step.

The novelty score is calculated the following way: first, the similarity between two sentences $S_i$ and $S_j$ is determined by the cosine similarity of their respective embeddings vectors $\vec{S_i}$ and $\vec{S_j}$:

$$\cos(S_i, S_j) = \frac{\vec{S_i} \cdot \vec{S_j}}{||\vec{S_i}|| \cdot ||\vec{S_j}||} \tag{4.4}$$

Using the similarity value, a sentence is considered novel when it subceeds a certain threshold $\tau$. The threshold is highly depended on the sentence embeddings, therefore a small experiment was conducted by Joshi et al. [37]. To start the experiment, 50 sentence pairs were chosen randomly and encoded as sentence embeddings. Out of these sentence pairs, half of them are classified as semantically similar. For each sentence embedding pair the cosine similarity is computed and the value of $\tau$ is set to the third quartile ($Q_3$) of the sentence similarity, which for the skip-thought embeddings is 0.85.

A sentence $S_i$, is considered to be novel if the *global similarity*, which is computed by determining the highest similarity with any of the other sentences in the input document $D$, is lower than $\tau$. If it is higher than $\tau$ it can still be considered novel, in case $S_i$ has a higher relevance score (see Section 4.2.3) than its most similar sentence. Otherwise, the highest cosine distance is its score.

$$\text{novelty}(D, S_i) = \begin{cases} 1, \text{ if } \max\left(\left\{cos\left(\vec{S_i}, \vec{S_j}\right)\right\}\right) < \tau, & 1 \leq j \leq N, i \neq j \\ 1, \text{ if } \max\left(\left\{cos\left(\vec{S_i}, \vec{S_j}\right)\right\}\right) > \tau, & \text{relevance}(D, S_i) > \text{relevance}(D, S_k) \\ & k = \text{argmax}(cos(\vec{S_i}, \vec{S_j})), 1 \leq j \leq N, i \neq j \\ 1 - cos(\vec{S_i}, \vec{S_j})) & \text{Otherwise} \end{cases} \tag{4.5}$$

### 4.2.6 Sentence selection

The sentence score of a sentence $S_i$ in a document $D$ is defined as the weighted sum of position, novelty and relevance:

$$score(D, S_i) = \alpha \cdot relevance(D, S_i) + \beta \cdot novelty(D, S_i) + \gamma \cdot position(D, S_i) \tag{4.6}$$

A grid search optimization yielded the values of $\alpha = 0.45$, $\beta = 0.35$ and $\gamma = 0.20$ for the authors. Joshi et al. [37] define a relative rank $rank(S_i)$ for a sentence in a document, which is the ordinal rank of the sentence score and is defined as follows:

$$\text{rank}(S_i) = 1 + \sum_{e=1}^{N} \gamma(e, i) \tag{4.7}$$

$$\gamma(e, i) = \begin{cases} 1 & \text{if score}(D, S_i) + \epsilon \cdot i > \text{score}(D, S_e) + \epsilon \cdot e \\ 0 & \text{otherwise} \end{cases} \tag{4.8}$$

where $1 \leq i, e \leq N$ for a document with $N$ sentences, $\epsilon$ is a tiny positive value, that is used to assign a different value when $\text{score}(D, S_i) = \text{score}(D, S_e)$ with a preference to the sentence position.

Generating the candidate summary is done by selecting the highest-scoring sentences from the document combining them into a single text. This text will then be used as input for the abstractive multi-document part, which will be explained in the succeeding section.

## 4.3 Abstractive multi-document summarization

The abstractive part is based on the architecture of the *ParaFuse_doc* model from Nayeem et al. [62]. It works by first clustering highly similar sentences and then generating new sentences, which combine the information in one cluster. These generated sentences are then paraphrased and ranked using the *TextRank* algorithm from Mihalcea and Tarau [56]. A subset of these generated sentences is then selected through an optimization step, trying to maximize the sentence score and the weighted keyphrases included in the summary. The architecture is illustrated in Figure 4.5 and each step is explained in the following paragraphs.

**Preprocessing:** is very similar to the preprocessing done by Joshi et al. [37], with the addition of *part-of-speech* (POS) tagging.

**Encode sentences:** using a *bi-directional gated recurrent unit* (Bi-GRU) the sentences are encoded into high-dimensional representations.

**Cluster sentences:** the sentences are grouped into clusters based on the cosine similarity of their respective embeddings. These clusters are very small and typically contain a handful of highly similar sentences.

**Fuse sentences:** for each cluster, a *word graph* is constructed that aims to generate candidate sentences which capture the content and information of the cluster.

**Paraphrase candidates:** these candidates are then paraphrased if a fitting lexical substitute is available.

**Rank candidates:** the candidates from the word graph and their substituted peers are ranked based on the information they contain and their grammaticality.

**Extract keyphrases:** from the original input documents, keyphrases are extracted and weighted on the importance of the contained information.

**Select sentences:** a linear optimization procedure is employed to select a subset of sentences, which a) includes highly relevant keyphrases and b) has high sentence ranks.

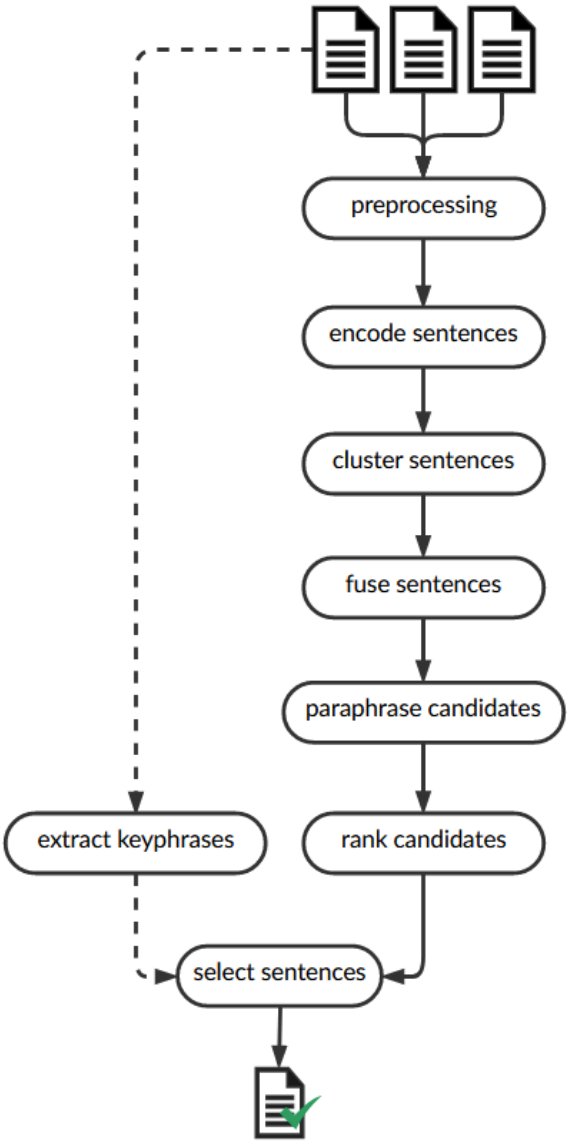



**Figure 4.5** Architecture of the abstractive part, based on Nayeem et al. [62].

### 4.3.1 Pre-processing

The pre-processing step shares its functionality with the pre-processing step for the extractive part with an additional part-of-speech (POS) tagging step. Using spaCy, the input sentences are transformed into lists of tuples consisting of the lower-cased word and its associated POS-tag.

**Part-of-speech tagging** is the process of assigning words to a specific category or marking them as a specific *part-of-speech*, for example *nouns, verbs, adjectives* etc. [54]. Table 4.1 shows a sentence annotated with both coarse and fine POS tags from spaCy.

| **Input** | The | dog | walked | around | in | the | village | . |
|---|---|---|---|---|---|---|---|---|
| **Coarse** | DET | NOUN | VERB | ADV | ADP | DET | NOUN | PUNCT |
| **Fine** | DT | NN | VBD | RB | IN | DT | NN | . |

**Table 4.1:** Exemplary sentence with coarse and fine POS tags.

### 4.3.2 Sentence encoding and clustering

Given a sentence $S$, which is a sequence of words $(w_1, w_2, ..., w_L)$ where $L$ is the number of words, a sentence is encoded into a high-dimensional embedding using a *bi-directional GRU* (Bi-GRU) architecture. As input for the Bi-GRU network, each word is embedded into pre-trained word embeddings from *Google news*. The networks architecture follows the general encoder-decoder architecture (see Section 2.7), but processes the input in both forward and backward direction. Figure 4.6 shows the the input sequence $(w_1, w_2, ..., w_L)$ being processed in both directions in the hidden states $(h_0, ..., h_L)$.

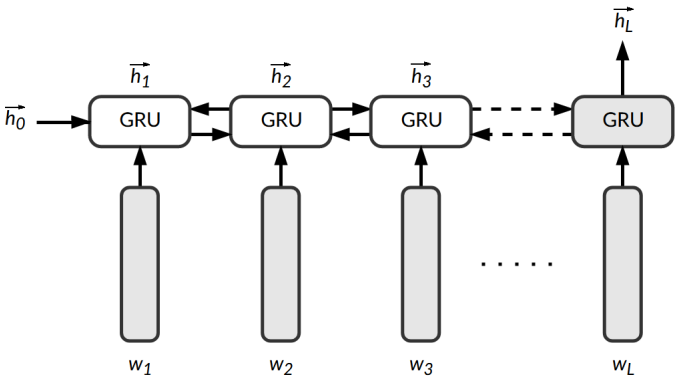



**Figure 4.6:** Bi-directional GRU for sentence encoding, drawn after Nayeem et al. [62].

Similar sentences are then clustered using a technique called *agglomerative clustering*. The clusters are formed based on the cosine similarity of the sentence embeddings. These clusters are needed for the next step, where new sentences are generated that represent the

information of a cluster. Therefore, finding a good sentence representation is an essential component to form clusters with highly similar sentences.

**Agglomerative clustering** is a hierarchical approach to form clusters by assigning each element its own cluster and iteratively merging close clusters. It employs a *complete-linkage criterion* to maximize the inter-cluster-distance during the merging procedure [75].

### 4.3.3 Sentence Fusion Word Graph

Given a cluster of highly similar sentences, the aim of this step is to summarize the cluster's content or topic in a single sentence, which preserves the most important information. This task is referred to as *multi-sentence compression* and is based on the work of Filippova [24].
This is done by constructing a *word graph*. The graph is created by iteratively adding the sentences in a cluster to it. The nodes of the graph are the words in a sentence and their respective POS-tag. Nodes are connected by forming directed edges for words in a sentence based on their position. A graph has an additional *start* and *end* node, to which all sentences are connected. Figure 4.7 shows a graph consisting of two example sentences about former Chilean president Augusto Pinochet.

1. *Pinochet left government in 1990.*
2. *He ruled from 1973 to 1990 but remained commander until March.*

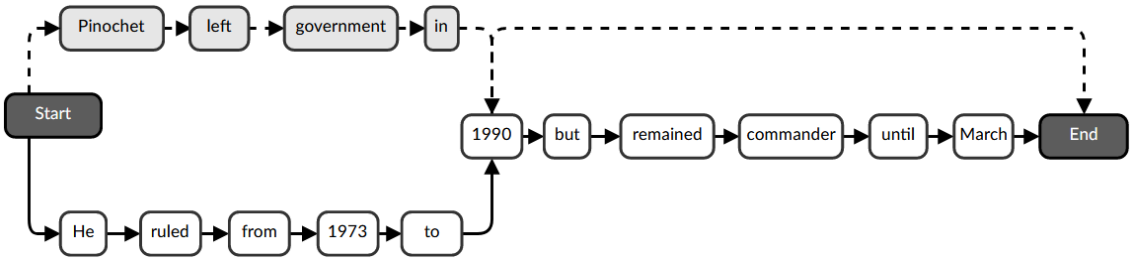



**Figure 4.7:** Example word graph based on real sentences from the DUC 2004 dataset. To simplify the figure, only two sentences are shown which connect on the *1990* node. POS-tags are omitted for clarity as well.

From this word graph with only one common word (*1990*) two new sentences can be constructed by generating the $K$-shortest paths from start node to end node:

1. *Pinochet left government in 1990 but remained commander until March.*
2. *He ruled from 1973 to 1990.*

The sentences for a word graph can differ in length and syntax, but they need at least one common word to generate new sentences. In practice, where word graphs are constructed from clusters with multiple sentences, a common word is almost always found, given that the sentence representation and clustering function work as intended. From a word graph, between 50 and 200 $K$-shortest paths are generated according to Nayeem et al. [62].

The biggest problem when using this approach is how to rank the generated sentences. Filippova's implementation [24] ranks the candidate sentences based on the sentence length, which does not fully correlate with how much information is contained in a sentence (although longer sentences tend to contain more information). Boudin and Morin [11] propose to rank candidate sentences based on the keywords they include. The authors of the *ParaFuse_doc* system decided to use a modified version of the *TextRank* algorithm to compute a score for each generated sentence. TextRank scores sentences based on their similarity to all other sentences in a document. In the original implementation, the number of overlapping words is used as a similarity measurement. Nayeem et al. [62] propose to use the cosine similarity of the sentence embeddings instead.

### 4.3.4 Lexical substitution

To increase the abstractiveness of the generated summary, the candidate sentences are paraphrased with lexical substitutes from the paraphrase database *PPDB 2.0*[4]. It is an open-source database with several million alternative phrases given a word and its context. As the paraphrasing should not change a sentence's meaning, only nouns (excluding *named entities*, also known as *proper nouns*) and verbs are used as substitution candidates. Multiword expressions are not considered as substitute or target.

To decide if a word substitution is fitting, the *appropriateness* score is calculated based on the word vectors of a possible substitute. This is done using the *context2vec* model from Melamud et al. [55], which aims to learn embeddings for sentential contexts around a target word. The appropriateness of a substitution $s$ for a target word $t$ and its context elements $C = \{c_1, ..., c_n\}$ is defined as:

$$appropriateness(s|t, C) = \frac{cos(s, t) + \sum_{c \in C} cos(s, c)}{|C| + 1} \tag{4.9}$$

A target word $t$ is considered suitable for substitution if the candidate $s$ has an *appropriateness*, which exceeds a certain threshold. Nayeem et al. [62] recommend a value of 0.7 for this, without specifying how they obtained this value. The candidate with the highest *appropriateness* value is selected and a new sentence is formed by replacing the target word with the candidate.

### 4.3.5 Keyphrase extraction

To distinguish sentences with a high information content from those with little information content, keyphrases are extracted from the original input text. Rose et al. [76] define a keyphrase as a sequence of one or multiple words, which represent a given document's content. The extraction is performed using *Rapid Automatic Keyword Extraction* (RAKE) from Rose et al. [76]. The intuition behind RAKE is that keyphrases rarely contain stop words (such as *the, of,* and *in*) or punctuation and therefore, sequences without these are likely to contain relevant information. The algorithm works as follows:

[4] `http://paraphrase.org`

Using a set of phrase and word delimiters, the input text is split into sequences of candidate keyphrases. For example, a keyphrase might start after a stop word and end right before a punctuation. These candidates are then scored by the ratio of the *word degree* and the *word frequency*. The degree of a word $w$ is defined as the total number of words that occur in candidate keyphrases, which contain $w$. The frequency of a word is defined as how often it occurs in the entire list of keyphrases. The weight of a candidate $c = (w_1, w_2, ..., w_n)$ with $n$ words is defined as:

$$\text{weight}(c) = \sum_{i=1}^{n} \frac{degree(w_i)}{frequency(w_i)} \tag{4.10}$$

For example, using the last two paragraphs as input text, RAKE would consider the following phrases to be the most important information:

- *Rapid Automatic Keyword Extraction (RAKE)* - 19.9
- *high information content* - 9.28
- *original input text* - 9.17

where the number after a keyphrase represents its RAKE score. The score a keyphrase with multiple words is based on sum of the word-degrees of each word, normalized by their frequencies.

### 4.3.6 Sentence selection

The summary is formed by selecting a subset from the candidate sentences using an *Integer Linear Programming* (ILP) framework. ILP is an optimization procedure that tries to select an optimal result under constraint of a set of linear equations and inequations. An optimal result is defined as having the maximum value of the sum of the weighted keyphrases included in the summary and the sum of the scores of the sentences included (see Equation 4.11). This is implemented using the linear programming modeler PuLP[5].

Given a set of candidate sentences, two binary variables are defined, which the ILP framework tries to solve: $s_j$ indicates if the candidate sentence $j$ is selected and $k_i$ indicates if a keyphrase $i$ is present in the selected sentences. Let $w_i$ be the weight of a keyphrase $i$ and $l_j$ the length of a sentence $j$. The variable $Occ_{ij}$ is set to 1 if the keyphrase $i$ occurs in sentence $j$, otherwise it is set to 0. The selection procedure for a summary of length $L$ is defined as:

$$max : (\sum_i w_i k_i + \sum_j (score(s_j) + \frac{l_j}{L} \cdot s_j) \tag{4.11}$$

$$\text{Subject to} : \sum_j l_j s_j \leq L \tag{4.12}$$

[5] `https://github.com/coin-or/pulp`

$$s_j Occ_{ij} \leq k_i, \forall i, j \tag{4.13}$$

$$\sum_j s_j Occ_{ij} \geq k_i, \forall i \tag{4.14}$$

$$\sum_{j \in g_c} s_j \leq 1, \forall g_c \tag{4.15}$$

$$k_i \in \{0, 1\} \forall i \tag{4.16}$$

$$s_j \in \{0, 1\} \forall j \tag{4.17}$$

To reduce redundancy, the constraints in Equations 4.13 and 4.14 restrict the selection of sentences based on the keyphrases they contain. From each sentence cluster, only one sentence can be selected to increase information diversity (Eq. 4.15). The constraints in Equations 4.16 and 4.17 restrict the values of $s$ and $k$ to binary values, in order to represent the inclusion of a sentence or keyphrase. Equation 4.12 restricts the summary length to the value of $L$. The selected sentences are then concatenated to a single body of text to form the final summary.

# 5 Evaluation

This chapter first proposes a strategy to evaluate how good the candidate summaries are. This strategy is based on insights gained in Section 3.2. The second section introduces the experimental setup and the last section shows the results obtained from the implemented system.

## 5.1 Evaluation strategy

Following the definition from Section 2.2, the evaluation strategy focuses on the key aspects of a good summary: *informativeness, relevance, fluency* and *coherence* as well as the supporting metric of *abstractiveness*. Together they can provide a good picture of a summary's quality and allow to compare the strengths and weaknesses of different summary systems.

**Informativeness** is commonly evaluated using lexical similarity metrics (e.g. ROUGE-1). Due to the drawbacks of the latter (see Section 3.2.1.1), a content-based approach is preferable. Originally this thesis used he *PEAK* framework (see Section 3.2.1.2) from Yang et al. [92] to generate and evaluate the pyramids. Peyrard and Eckle-Kohler [72] pointed out that high pyramid scores from PEAK result in low ROUGE scores and vice versa, which is why instead the *reward* function from Böhm et al. [9] is used to evaluate the informativeness of a summary. Böhm et al.'s system [9] is trained to approximate human summary evaluation and can be employed without reference summaries.

**Relevance** measures how consistent the content of a candidate summary is compared to the source material, which might go unnoticed in content-based approaches. To measure it, the ROUGE-2 metric is used. It is based on the bi-gram overlap of candidate and reference summary. When taking a summary's abstractiveness into account, this overlap measurement can help identify incorrect information in a candidate summary.

**Abstractiveness** is commonly measured using the copy rate, which is the number of unique words that occur in both the candidate summary and the source documents in relation to the total number of unique words in the candidate summary. Seeing that the copy rate cannot differentiate between a purely extractive system and a word fusion system (i.e. where new sentences are formed using the original vocabulary), the copy rate is not the most appropriate metric. Instead, the number of unique bi-grams, occurring

in both the candidate summary and the input documents is counted and divided by the total number of unique bigrams in the candidate summary.

**Fluency** is what makes humans perceive sentences as natural. Commonly, the perplexity of a language model (see Section 2.5) is used to measure the fluency of a sentence. Although this is a good indicator on how correct and natural a sentence feels, it assigns lower values to sentences with rare or uncommon words. To correct this, Kann et al. [41] propose the *SLOR* metric (see Section 3.2.2.1), which allows sentences with uncommon words to achieve good scores and therefore is used in the evaluation. The perplexity and uni-gram probabilities are obtained from the language model GPT-2 (see Section 2.7).

**Coherence** is what differentiates a well-written text from a set of unrelated sentences. Lapata and Barzilay [44] propose to measure the text coherence based on the follow-up sentence similarity. This approach is not as popular as the *entity-grid* method which was also proposed by Barzilay and Lapata [4], but got recent attention with the rise of large scale transfer-learning and the resulting improvements in sentence encoding. Therefore, for this thesis, the follow-up sentence similarity is used and the *Sentence BERT* model from Reimers and Gurevych [74] is used to calculate the sentence representations.

## 5.2 Datasets

Whereas the previous section explained how the evaluation was performed, this section introduces the datasets used in the experimental setup. For the experiments done in this thesis, three datasets have been chosen, which differ in the length of the reference summaries. The reason for this is to show that the system can perform well when generating both long and short summaries. An instance in these datasets consists of multiple documents with related and partly-redundant content. Each instance has one or multiple human-written reference summaries. The datasets will be listed in order of their reference summary length from shortest to longest.

### 5.2.1 Opinosis - Opinion Dataset

Opinosis is an aspect-separated user review dataset obtained from Amazon[1], Edmunson[2] and TripAdvisor[3]. It was published by Ganesan et al. [26] to evaluate their identically named system and can be obtained from the authors' personal homepage[4].

It consists of 51 instances, each with an average of 134 sentences and four human-composed reference summaries. Some of the sentences in this dataset are incomplete, adding an additional challenge. To deal with this, incomplete sentences that contain a conjunction

[1] `http://www.amazon.com`
[2] `http://www.edmundson-electrical.co.uk`
[3] `http://wwww.tripadvisor.com`
[4] `https://kavita-ganesan.com/opinosis-opinion-dataset/`

are cropped and punctuation is added. Incomplete sentences without a conjunction are excluded. This affects 124 out of 6846 sentences. The target word length for a reference summary is 15 words which makes the Opinosis dataset a good candidate to evaluate if the proposed system can generate very short summaries with a compression rate of 99.5%.

### 5.2.2 Document Understanding Conference (DUC) Dataset

DUC was an annual conference from 2000 to 2007 to further the progress in document understanding. In 2008 it was included into the Text Analysis Conference (TAC)[5] as the summarization track. As part of the conference, several multi-document datasets were released to the public[6]. This thesis uses the dataset from DUC2004, which contains articles from The Associated Press[7] and The New York Times[8]. It has 50 instances with 6945 words on average and four human-written reference summaries. The target word length for a reference summary is 100 words, which translates to a compression rate of 98.6%. The dataset is chosen to evaluate if the proposed system can generate medium sized summaries.

### 5.2.3 Multi-News Dataset

Fabbri et al. [23] released this very recent multi-document corpus consisting of 56,216 articles-summary pairs with 2-10 articles per topic and 1876 words on average. The summaries are obtained from the news curator website Newser[9]. The authors released the dataset to the public on GitHub[10]. The target word length for a reference summary is 250 words, which translates to a compression rate of 86.7%. This makes this dataset a good candidate to evaluate the generation of large summaries.

## 5.3 Results

Each instance of these three datasets is summarized with the baseline system and every summary is then evaluated on all quality aspects. To put the evaluation in context, several summarization systems from Chapter 3 are used for comparison. Table 5.1 shows the results of the proposed system and follows the order of the dataset presentation from shortest to longest. Lower values are better for abstractiveness and fluency. On each dataset, some of the highest scoring peer systems are used for comparison.

The results show that the proposed system performs quite differently on each dataset. It is to note that fluency should be lower (i.e. better) for extractive systems (i.e with an abstractiveness value of 100.0). The proposed system is able to generate more abstractive

[5] https://tac.nist.gov/
[6] https://www.nlpir.nist.gov/projects/duc/data.html
[7] https://www.ap.org
[8] https://www.nytimes.com/timeswire
[9] https://www.newser.com/
[10] https://github.com/Alex-Fabbri/Multi-News

text for longer summaries. An important aspect of the informativeness values is that they should not be interpreted as absolute values but are meant to rank one summary over another, which is why the upper and lower bounds differ greatly between different datasets. The coherence of the baseline system is rather low, as it currently concatenates the selected sentences and ignores the original sentence position. To improve the generated summaries, the next chapter discusses the results and suggests improvements.

| System | Inf. | Rel. | Abs.* | Flu.* | Coh. |
|---|---|---|---|---|---|
| **Opinosis** | | | | | |
| TextRank [56] | 14.03 | **20.30** | 100.0 | 135.28 | 66.87 |
| Opinosis [26] | **27.09** | 18.85 | 92.05 | 83.76 | **73.06** |
| Pointer-Generator-MMR [45] | 3.20 | 10.24 | 91.03 | 98.10 | 63.54 |
| Proposed system | 9.09 | 7.84 | **61.24** | **83.20** | 65.15 |
| **DUC 2004** | | | | | |
| Lead-100w | 34.83. | 9.42 | 100.0 | 53.74 | 73.21 |
| Submodular [48] | 35.82 | 13.68 | 100.0 | 47.81 | 73.59 |
| RegSum [34] | **44.25** | **14.66** | 100.0 | 44.74 | **81.34** |
| DDP [43] | 42.49 | 14.56 | 100.0 | **40.43** | 78.77 |
| Proposed system | 30.37 | 6.69 | **58.72** | 123.61 | 64.79 |
| **Multi-News** | | | | | |
| Transformer [79] | 8.79 | 8.69 | 36.86 | 100.84 | 58.22 |
| AllSummarizer [1] | **13.85** | **14.87** | 100.0 | **70.39** | 66.64 |
| Hi-Map [23] | 2.38 | 8.70 | **36.11** | 120.54 | 59.16 |
| Proposed system | 11.78 | 9.20 | 65.03 | 115.17 | **66.79** |

**Table 5.1:** Results for the proposed approach in comparison to several influential peer systems. (* lower is better)

# 6 Analysis and discussion

This chapter analyses and discusses the results of the baseline systems and proposes ways to improve the system. Each section identifies a problem in the baseline approach and proposes a solution to improve the system in this area. This chapter also reflects the iterative way in which the practical work for this thesis was performed.

The system follows a component-based architecture, making it possible to analyse and improve each component on its own. To ensure that the whole system improves as well, the summaries are generated for each dataset in every iteration. The order in which each component was analysed follows the order in which this thesis is structured - from extractive to abstractive components.

## 6.1 Generalizing the extractive selection

The first thing to notice about the results is that the summaries of the Opinosis dataset are relatively worse than those of the other datasets when compared to competing systems. Also, the *Lead-3* baseline is not as strong on this dataset. Joshi et al. [37] do report high ROUGE values from their evaluation on several different datasets. Unfortunately, all of the datasets the authors evaluated their system on are based on news articles. News articles oftentimes start by providing an overview of the content to come in the first few lines. The position score exploits this statistical feature but just as the *Lead* baseline, it only improves the summaries of news datasets.

To remove this bias towards certain document structures, only the first two of the three metrics are used for this thesis – namely *novelty* and *relevance*. This has an effect on the sentence selection procedure in a way that differs from the original approach where a weighted sum of the sentence metrics was employed. With only two instead of three metrics, using the minimum value of either novelty or relevance, resulted in a better selection overall in terms of informativeness and relevance. The score of a sentence $S_i$ in a document $D$ is hence defined as:

$$\text{score}(D, S_i) = min(novelty(D, S_i), relevance(D, S_i)) \tag{6.1}$$

Table 6.1 shows how the results change for each dataset. Values in parentheses are the relative change to the original values, results that did not change are omitted for clarity. The fluency values improve on all datasets. On the Opinosis dataset, the informativeness and relevance increases, on the other two datasets it decreases. This is because the latter

two are news articles, where the position score works very well. Still, removing it allows the system to work on datasets from different domains.

| Dataset | Inf. | Rel. | Abs.* | Flu.* | Coh. |
|---|---|---|---|---|---|
| Opinosis | 10.43 (+1.34) | 9.73 (+1.89) | 70.31 (+9.07) | 62.83 (-20.37) | 63.77 (-1.38) |
| DUC2004 | 28.35 (-2.02) | 4.59 (-2.10) | 58.00 (-0.72) | 113.21 (-10.4) | 62.32 (-2.47) |
| Multi-News | 10.60 (-1.18) | 7.91 (-1.29) | 65.63 (+0.60) | 101.18 (-13.99) | 61.46 (-5.33) |

**Table 6.1:** The results for each dataset after changing the weighted sum of the original three metrics to the minimum of novelty and relevance. (*lower is better)

## 6.2 Sentence encoding

Both the extractive and the abstractive part are highly-dependent on good sentence representations. Both metrics from the former are based on inter-sentence similarity which makes the sentence representation the biggest lever for improving the result. The latter uses the sentence embeddings mainly for clustering similar sentences. These sentence clusters are then fused using the word-graph. Therefore, without a good representation, unrelated sentences might be merged into the same cluster which could result in low-quality unrelated sentence fusions.

During the implementation of the proposed system, several different sentence embedding techniques were used. To identify the most appropriate technique, the scores on the *Semantic textual similarity (STS)* benchmark[1] were consolidated. The goal in STS is to predict whether a pair of sentences is semantically similar or not. Table 6.2 shows the Spearman rank correlation of different sentence embedding models on the *STSb* dataset.

| Model | STS spearman rank |
|---|---|
| Avg. BERT embeddings[20] | 46.4 |
| Avg. GloVe embeddings[69] | 58.0 |
| SkipThought[42] | 62.2 |
| InferSent[16] | 66.87 |
| Universal Sentence Encoder[13] | 74.9 |
| Sentence BERT[74] | 77.0 (79.23) |
| Sentence RoBERTa[74] | 77.8 (79.10) |

**Table 6.2:** STS benchmark results for different sentence embeddings. Values as reported by Reimers and Gurevych [74] and Perone et al. [70]. Values in parentheses are obtained from the *large* version of the system.

[1] `http://ixa2.si.ehu.es/stswiki/index.php/STSbenchmark`

The Bi-GRU approach from Nayeem et al. [62] is not included in this table, but considering that the underlying architecture is very similar to the *InferSent* system (which uses a *Bidirectional Long Short-Term Memory* (Bi-LSTM) architecture), it is save to assume that the performance is similar or lower than InferSent's. The highest-scoring values are reported by the *Sentence BERT/ RoBERTa* system from Reimers and Gurevych [74]. It works by fine-tuning the BERT model (see Section 2.7 for details) to produce semantically meaningful sentence representations using a triplet network [33]. Several pre-trained models can be obtained from the authors' GitHub page[2]. It is to note that the model works very well on clean sentences but struggles with very short or noisy sentences as they appear in the Opinosis dataset. This is due to the system's training on sentences from the English Wikipedia, which contain almost no noise and are of a high quality in general.

To improve the system, the *Sentence BERT base* model was employed. Although the large BERT model and the RoBERTa model do have a higher STS score, they are not considered due to high hardware requirements. The abstractive part benefited the most from these embeddings, as clusters with more similar sentences meant less unrelated sentence fusions. Table 6.3 shows the influence of the new embeddings on the overall result. Although the information content contained in the summaries increases, the fluency score gets a lot worse. Ranking the sentence fusions by their perplexity will improve the fluency again.

| Dataset | Inf. | Rel. | Abs.* | Flu.* | Coh. |
|---|---|---|---|---|---|
| Opinosis | 17.32 (+6.89) | 5.20 (-4.53) | 61.89 (-8.42) | 189.67 (+126.84) | 68.18 (+4.41) |
| DUC2004 | 28.35 (+0.00) | 4.72 (+0.13) | 60.27 (+2.17) | 196.96 (+83.75) | 64.78 (+2.46) |
| Multi-News | 11.56 (+0.96) | 8.34 (+0.43) | 66.84 (+1.21) | 168.58 (+67.40) | 60.49 (-0.97) |

**Table 6.3:** Improvements in the results for each dataset after switching to the Sentence BERT model. (*lower is better)

## 6.3 Sentence Fusion Word Graph

The most challenging aspect of the word-graph is not the sentence fusion itself but the ranking of the candidate fusions. For this, Nayeem et al. [62] used the keyphrase-based approach from Boudin and Morin [11] which focuses on producing a diverse set of informative sentences without taking fluency into account. To overcome this issue and improve both informativeness and fluency, the fusion candidates are ranked in a two-step procedure: First, the sentences are grouped based on the keyphrases they contain. Then the $n$ best sentences out of each group are identified to ensure that a diverse set of information is preserved. To rank the sentences, the SLOR metric (see Section 3.2.2.1) as computed by a language model is used to ensure only grammatically correct sentences are selected. The LM employed for this is OpenAi GPT-2 from Radford and Salimans [73] (see Section 2.7).

[2] `https://github.com/UKPLab/sentence-transformers`

| Dataset | Inf. | Rel. | Abs.* | Flu.* | Coh. |
|---|---|---|---|---|---|
| Opinosis | 14.43 (-2.89) | 6.17 (+0.97) | 54.27 (-7.62) | 81.20 (-108.47) | 68.00 (-0.18) |
| DUC2004 | 30.25 (+1.90) | 6.69 (+1.97) | 46.93 (-13.34) | 120.42 (-76.54) | 65.37 (+0.59) |
| Multi-News | 11.78 (+0.22) | 9.12 (+0.78) | 42.93 (-23.91) | 113.70 (-54.88) | 61.51 (+1.02) |

**Table 6.4:** Improvements in the overall result for the candidate sentences after ranking the sentence fusions based on the normalized perplexity of OpenAi GPT-2. (*lower is better)

The SLOR score is lower for words which are very likely to follow after another and a low SLOR score of a sentence is a reasonable indicator of good grammaticality. Table 6.4 shows the improvements in informativeness and fluency on each of the datasets.

## 6.4 Lexical substitution

Whether or not a target word should be substituted is computed using the appropriateness equation (see Equation 4.9). It relies on the word embeddings of the surrounding words of the target and only allows substitution if the cosine distance does not change significantly. This leads to sentences that are equally or only slightly worse in terms of grammaticality and fluency. To find sentences which are more fluent than the original sentence, an approach based on the normalized perplexity of a language model is proposed. Similar to the SLOR metric defined in Equation 3.7, the perplexity is then normalized with the unigram probability of a sentence, in order to compensate for uncommon words in an sentence. A sentence $S$ with a target word $t$ can be paraphrased using a substitution candidate $s$ if the normalized perplexity of $S_{original}$ is smaller or equal to the normalized perplexity of $S_{substituted}$. An appropriateness value greater than zero is used to select substitutions:

$$appropriateness(s,t) = perplexity(S_{original}) - perplexity(S_{substituted}) \tag{6.2}$$

| Dataset | Inf. | Rel. | Abs.* | Flu.* | Coh. |
|---|---|---|---|---|---|
| Opinosis | 22.88 (+8.45) | 7.38 (+1.21) | 68.41 (+14.14) | 64.88 (-16.32) | 57.14 (-10.86) |
| DUC2004 | 35.90 (+5.65) | 6.97 (+0.28) | 62.08 (+15.15) | 114.88 (-5.54) | 65.31 (-0.06) |
| Multi-News | 12.47 (+0.69) | 8.48 (-0.64) | 44.48 (+1.55) | 93.99 (-19.71) | 67.06 (+5.55) |

**Table 6.5:** Improvements in the fluency and abstractiveness (lower is better) values after augmenting the lexical substitution component to use normalized perplexity instead of neighbouring context-embedding distance.

Table 6.5 shows the improvement in terms of fluency and abstractiveness for the datasets used in the evaluation. To implement this, the LM GPT-2 (see Section 2.7) from Radford and Salimans [73] is employed.

## 6.5 Keyphrase extraction

The extracted keyphrases have a high influence on the candidates as the assigned weights and the keyphrases included in a sentence are used for the sentence selection. Therefore, using a high-quality system for the extraction is an obvious candidate for improvement. Campos et al. [12] propose a feature-based system by the name *Yet Another Keyword Extractor* (YAKE) which uses a similar approach to RAKE, but instead of only focusing on word degree and frequency (see Section 4.3.5) it also focuses on word casing, relatedness to the context and how often a word occurs in different sentences.

YAKE is achieving state-of-the-art or close to state-of-the-art performance on several extraction datasets and outperforms RAKE on all of them[3]. The effect on the generated summaries' informativeness and relevance can be seen in Table 6.6.

| Dataset | Inf. | Rel. | Abs.* | Flu.* | Coh. |
|---|---|---|---|---|---|
| Opinosis | 18.19 (-4.69) | 13.41 (+6.03) | 84.47 (+16.06) | 84.00 (+19.12) | 62.79 (+6.65) |
| DUC2004 | 40.93 (+5.03) | 6.74 (-0.23) | 87.18 (+25.1) | 67.28 (47.60) | 63.16 (-2.15) |
| Multi-News | 11.97 (-0.50) | 13.13 (+4.65) | 92.25 (+47.77) | 89.85 (-4.14) | 61.53 (-6.53) |

**Table 6.6:** Changes to the results for each dataset when using YAKE instead of RAKE. (*lower is better)

Using YAKE keyphrases makes the system a lot less abstractive, which improves the fluency and the relevance scores.

## 6.6 True-case sentences

One issue with the candidate sentences that is often overlooked, as common metric do not take it into account is word casing. The system's output is currently all lower-cased, because the word graph needs lower-cased words to fuse sentences on any word position. Lower-cased text is more difficult to read and might convey false information, as proper-nouns or named-entities might not be recognized by the reader.

To restore the original case information, a *true-casing* component is proposed. True-casing is defined as the process of "restoring case information to badly-cased or non-cased text" [49]. In the context of this thesis, a combination of POS-tagging and rule-based matching is proposed, which can easily be integrated in the summarization pipeline. To evaluate whether this is effective, an experiment has been conducted to recreate the original case of all lower-cased documents of the DUC2004 dataset. A random baseline as well as a statistical approach based on the ideas of Lita et al. [49] is used for comparison. Table 6.7 shows how effective both approaches are.

[3] `https://github.com/LIAAD/yake/blob/master/docs/YAKEvsBaselines.jpg`

| Random Baseline(%) | Rule-based POS-tagging(%) | Statistical approach(%) |
|---|---|---|
| 0.449 | **0.993** | 0.983 |

**Table 6.7:** Percentages of correctly ordered words of the DUC-2004 dataset. The proposed rule-based POS-tagging approach outperforms the approach from Lita et al. [49].

The proposed approach has an edge over the approach from Lita et al. [49], which alone might not be enough to choose one over another. Due to the fact that each word is already tagged in the word graph, however, the proposed approach integrates very well in the summarization pipeline and runs considerably faster than the statistical approach.

## 6.7 Improving text coherence

Text coherence is an important quality aspect for a human reader and has even been included in the official DUC guidelines in 2005[4]. Still, many systems neglect this or, as is the case in many extractive single-document systems, simply use the original sentence position as a relative indicator. The abstractive approach by Nayeem et al. [62] does not feature a text coherence component, resulting in summaries with randomly chosen sentences. To improve this, two methods for ordering the sentences have been evaluated.

**Next sentence prediction** using BERT. One of the training tasks of BERT is to predict whether a sentence $A$ is likely to precede a sentence $B$. Given a set of randomly ordered sentences, the next sentence probability is calculated for each sentence combination. The ideal sentence order is computed by finding the sentence order which maximizes the next sentence probability.

**Maximize follow-up sentence similarity** following the proposal of Lapata and Barzilay [44]. They define a coherence metric for a document $D$ consisting of $N$ sentences $S_1, ..., S_N$ as the sum of the normalized sentence similarity:

$$coherence(D) = \frac{\sum_{i=1}^{N-1} cos(S_i, S_{i+1})}{N-1} \tag{6.3}$$

The ideal sentence order is therefore the order that maximizes the follow-up similarity or *coherence* of a given text. The similarity is defined as the cosine similarity of two Sentence BERT representations. For both methods, a linear problem was defined to maximize either the follow-up-similarity or the next-sentence probability. This was then solved for all permutations of a list of sentences using PuLP.

To assess which method is superior, a small experiment has been conducted. The task is to order the sentences in a set of randomly shuffled documents from the DUC 2004 dataset. The methods are scored on the relative amount of correctly ordered follow-up sentence pairs. Correctly ordered means that one point is awarded for each sentence pair

[4] `https://duc.nist.gov/duc2005/quality-questions.txt`

$s_i, s_{i+1}$ that matches the original sentence pair $o_i, o_{i+1}$. Table 6.8 shows how they compare to each other and a randomly shuffled baseline.

| Random Baseline(%) | BERT next sentence(%) | Follow-up similarity(%) |
|---|---|---|
| 0.123 | 0.152 | **0.187** |

**Table 6.8:** Percentages of correctly ordered sentences from a randomly shuffled documents of the DUC 2004 dataset.

Seeing that the follow-up sentence similarity approach is slightly superior in this experiment, it is included in the system's architecture as the second-to-last step. It is to note that sentence ordering is a challenging problem and both methods are only slightly better than a random baseline.

## 6.8 Bringing it all together

To conclude the Analysis and Discussion chapter and summarize the findings of this thesis, this section provides an overview on how each proposal improves the system and its output. The final results, as well as the effects of each iterative improvement are displayed in Table 6.9. Table 6.10 shows how the final version of the system compares to other systems. In line with previous tables, the best values on each dataset are highlighted in bold letters.

| System | Inf. | Rel. | Abs.* | Flu.* | Coh. |
|---|---|---|---|---|---|
| **Opinosis** | | | | | |
| Original | 9.09 | 7.84 | 61.24 | 83.20 | 65.15 |
| + generalized extractive | 10.43 | 9.73 | 70.31 | **62.83** | 63.77 |
| + sentence encoding | 17.32 | 5.20 | 61.89 | 189.67 | 68.18 |
| + improved word-graph | 14.43 | 6.17 | **54.27** | 81.20 | 68.00 |
| + perplexity substitution | **22.88** | 7.38 | 68.41 | 64.88 | 57.14 |
| + improved keyphrases | 18.19 | 13.41 | 84.47 | 84.00 | 62.79 |
| + improved coherence | 18.19 | **13.41** | 84.47 | 84.00 | **68.42** |
| **DUC 2004** | | | | | |
| Original | 30.37 | 6.69 | 58.72 | 123.61 | 64.79 |
| + generalized extractive | 28.35 | 4.59 | 58.00 | 113.21 | 62.32 |
| + sentence encoding | 28.35 | 4.72 | 60.27 | 196.96 | 64.78 |
| + improved word-graph | 30.25 | 6.69 | **46.93** | 120.42 | 65.37 |
| + perplexity substitution | 35.90 | **6.97** | 62.08 | 114.88 | 65.31 |
| + improved keyphrases | 40.93 | 6.74 | 87.18 | 67.28 | 63.16 |
| + improved coherence | **40.93** | 6.74 | 87.18 | **67.28** | **74.59** |
| **Multi-News** | | | | | |
| Original | 11.78 | 9.20 | 65.03 | 115.17 | 66.79 |
| + generalized extractive | 10.60 | 7.91 | 65.63 | 101.18 | 61.46 |
| + sentence encoding | 11.56 | 8.34 | 66.84 | 168.58 | 60.49 |
| + improved word-graph | 11.78 | 9.12 | **42.93** | 113.70 | 61.51 |
| + perplexity substitution | **12.47** | 8.48 | 44.48 | 93.99 | 67.06 |
| + improved keyphrases | 11.97 | 13.13 | 92.25 | 89.85 | 61.53 |
| + improved coherence | 11.97 | **13.13** | 92.25 | **89.85** | **78.56** |

**Table 6.9:** Results of the original approach in comparison to the results obtained with iterative improvements. (*lower is better)

| System | Inf. | Rel. | Abs.* | Flu.* | Coh. |
|---|---|---|---|---|---|
| **Opinosis** | | | | | |
| TextRank [56] | 14.03 | **20.30** | 100.0 | 135.28 | 66.87 |
| Opinosis [26] | **27.09** | 18.85 | 92.05 | 83.76 | **73.06** |
| Pointer-Generator-MMR [45] | 3.20 | 10.24 | 91.03 | 98.10 | 63.54 |
| Proposed system | 18.19 | 13.41 | **84.47** | **84.00** | 68.42 |
| **DUC 2004** | | | | | |
| Lead-100w | 34.83. | 9.42 | 100.0 | 53.74 | 73.21 |
| Submodular [48] | 35.82 | 13.68 | 100.0 | 47.81 | 73.59 |
| RegSum [34] | **44.25** | **14.66** | 100.0 | 44.74 | **81.34** |
| DDP [43] | 42.49 | 14.56 | 100.0 | **40.43** | 78.77 |
| Proposed system | 40.93 | 6.74 | **87.18** | 67.28 | 74.59 |
| **Multi-News** | | | | | |
| Transformer [79] | 8.79 | 8.69 | 36.86 | 100.84 | 58.22 |
| AllSummarizer [1] | **13.85** | **14.87** | 100.0 | **70.39** | 66.64 |
| Hi-Map [23] | 2.38 | 8.70 | **36.11** | 120.54 | 59.16 |
| Proposed system | 11.97 | 13.13 | 92.25 | 89.85 | **78.56** |

**Table 6.10:** Results for the final version in comparison to several influential peer systems. (* lower is better)

# 7 Conclusion and future work

This thesis investigated whether unsupervised summarization systems can compete with supervised systems in the field of multi-document summarization. To evaluate this, a system has been developed that combines the extractive approach from Joshi et al. [37] and the abstractive approach from Nayeem et al. [62]. The system was employed to generate summaries for three diverse datasets - Opinosis, DUC2004 and Multi-News. These documents were chosen to cover a variety of document lengths and compression rates. The results were evaluated on four quality aspects as identified by Grusky et al. [29]. To further improve the generated summaries, several improvements on the summarization system have been proposed to include state-of-the-art techniques and make the summaries more readable.

The chosen sentence representation had a great impact on the performance of the model and the use of the Sentence BERT model [74] improved both the extractive selection and the abstractive generation. The extractive part was improved, as the model highly depends on sentence similarity and the abstractive part benefited from highly-similar sentences in each cluster, which improved the sentence fusion. Other current large-scale models such as BERT [20] and GPT-2 [73] have been found to benefit the system to produce coherent and fluent text.

Although the proposed system can compete with other summarization systems in many aspects (see Table 5.1 and Table 6.9), the conducted experiments show that there is a trade-off between the abstractiveness of a summary and the information included in it. Highly-abstractive summarization tends to result in low information content, which is in line with the findings of Zhang et al. [96]. They state that many of the abstractive systems (e.g. Pointer-Generator [79]) with high information content are near-extractive. Given that the information content is the most important aspect of a summary, extractive systems outperformed the proposed system.

The proposed system is quite complex, so it is very likely that not all components are optimized to their full potential. Reducing the complexity while still maintaining the important aspects will surely benefit the system. Besides, for each component there are a number of parameters and thresholds that are only optimized locally. An end-to-end grid search optimization was not performed due to time constraints but might benefit the system's performance as well.

Another important topic that was not covered in this thesis is aspect-based summarization. This can most easily be observed on the generated summaries of the Opinosis dataset, where contradictions are very common. A generated summary might be *The bathroom is very large. The bathrooms were tiny.*, which is a problem that aspect-based clustering might solve.

One way to enhance the sentence fusion component could be to merge *multiword expressions* (MWEs) as ShafieiBavani et al. [81] proposed. PPDB 2.0 also provides support for MWEs, which could be used to merge sentences where the only connection is a figure of speech. A similar effect might be archived using a neural sentence simplification model [91] to simplify the sentences in each cluster before summarization. Incorporating an information extraction system such as *ClauseIE* [19] could help by linking personal pronouns with named entities in a sentence cluster.

One of the biggest challenges that remains for this system is the sentence selection. Selecting a subset of sentences from the large pool of paraphrased and fused sentences in a way that the selection forms a coherent text, is still an unsolved task. Coming up with a way to select sentences that build upon each other should yield a major improvement in text quality. Maximizing the coherence and fluency on a document level instead of on a sentence level, might improve this.

With the current rate of improvements in NLP, it is to be expected that in the near future even better sentence representations and language models will be published. Incorporating these new systems might benefit the approach presented in this thesis as well.

# Bibliography


[1] Abdelkrime Aries, Djamel Eddine Zegour, and Khaled Walid Hidouci. "AllSummarizer system at MultiLing 2015: Multilingual single and multi-document summarization". In: *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Prague, Czech Republic: Association for Computational Linguistics, 2015, pp. 237–244. DOI: 10.18653/v1/W15-4634.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: (2014). arXiv: 1409.0473.

[3] Siddhartha Banerjee, Prasenjit Mitra, and Kazunari Sugiyama. "Multi-document abstractive summarization using ILP based multi-sentence compression". In: *IJCAI International Joint Conference on Artificial Intelligence* (2015), pp. 1208–1214. ISSN: 10450823. arXiv: 1609.07034.

[4] Regina Barzilay and Mirella Lapata. "Modeling local coherence". In: *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics - ACL '05*. Morristown, NJ, USA: Association for Computational Linguistics, 2005, pp. 141–148. DOI: 10.3115/1219840.1219858.

[5] Regina Barzilay and Kathleen R. McKeown. "Sentence fusion for multidocument news summarization". In: *Computational Linguistics* 31.3 (2005), pp. 297–327. ISSN: 08912017. DOI: 10.1162/089120105774321091.

[6] Yoshua Bengio et al. "A Neural Probabilistic Language Model". In: *Journal of Machine Learning Research* 3.Feb (2003), pp. 1137–1155. ISSN: ISSN 1533-7928.

[7] Lidong Bing et al. "Abstractive multi-document summarization via phrase selection and merging". In: *ACL-IJCNLP 2015 - 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, Proceedings of the Conference* 1 (2015), pp. 1587–1597. arXiv: 1506.01597.

[8] Christopher M. Bishop. *Pattern recognition and machine learning*. Springer, 2006. ISBN: 9780387310732.

[9] Florian Böhm et al. "Better Rewards Yield Better Summaries: Learning to Summarise Without References". In: *Proceedings of the 2019 Conference on Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Hong Kong, China, 2019. arXiv: 1909.01214.

[10] Piotr Bojanowski et al. "Enriching Word Vectors with Subword Information". In: *Transactions of the Association for Computational Linguistics* 5 (2017), pp. 135–146. ISSN: 2307-387X. DOI: 10.1162/tacl_a_00051. arXiv: 1607.04606.

[11] Florian Boudin and Emmanuel Morin. “Keyphrase extraction for n-best reranking in multi-sentence compression”. In: *NAACL HLT 2013 - 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Main Conference* (June 2013), pp. 298–305.

[12] Ricardo Campos et al. “YAKE! collection-independent automatic keyword extractor”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 10772 LNCS. 2018, pp. 806–810. ISBN: 9783319769400. DOI: 10.1007/978-3-319-76941-7_80.

[13] Daniel Cer et al. “Universal Sentence Encoder for English”. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Brussels, Belgium: Association for Computational Linguistics, 2018, pp. 169–174. DOI: 10.18653/v1/D18-2029.

[14] Eric Chu and Peter J. Liu. “MeanSum: A Neural Model for Unsupervised Multi-document Abstractive Summarization”. In: *International Conference on Machine Learning* (2018). arXiv: 1810.05739.

[15] Arman Cohan and Nazli Goharian. “Revisiting Summarization Evaluation for Scientific Articles”. In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. : Portorož, Slovenia: European Language Resources Association (ELRA), 2016, pp. 806–813.

[16] Alexis Conneau et al. “Supervised Learning of Universal Sentence Representations from Natural Language Inference Data”. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, 2017, pp. 670–680. DOI: 10.18653/v1/D17-1070.

[17] Zihang Dai et al. “Transformer-XL: Attentive Language Models beyond a Fixed-Length Context”. In: 2019, pp. 2978–2988. DOI: 10.18653/v1/p19-1285. arXiv: 1901.02860.

[18] Hoa Trang Dang. “Evaluation of Question-Focused Summarization Systems”. In: *Proceedings of the Workshop on Task-Focused Summarization and Question Answering*. Sydney, Australia, 2006, pp. 48–55.

[19] Luciano Del Corro and Rainer Gemulla. “ClausIE”. In: *Proceedings of the 22nd international conference on World Wide Web - WWW ’13*. New York, New York, USA: ACM Press, 2013, pp. 355–366. ISBN: 9781450320351. DOI: 10.1145/2488388.2488420.

[20] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North*. Minneapolis, Minnesota, USA: Association for Computational Linguistics, 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.

[21] H. P. Edmundson. “New Methods in Automatic Extracting”. In: *Journal of the ACM* 16.2 (1969), pp. 264–285. ISSN: 00045411. DOI: 10.1145/321510.321519.

[22] Gunes Erkan and Dragomir R. Radev. “LexPageRank : Prestige in Multi-Document Text Summarization”. In: *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2004), pp. 365–371.

[23] Alexander Fabbri et al. “Multi-News: A Large-Scale Multi-Document Summarization Dataset and Abstractive Hierarchical Model”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, 2019, pp. 1074–1084. DOI: 10.18653/v1/P19-1102.

[24] Katja Filippova. “Multi-Sentence Compression: Finding Shortest Paths inWord Graphs”. In: *Coling 2010 - 23rd International Conference on Computational Linguistics, Proceedings of the Conference*. August. Association for Computational Linguistics, 2010, pp. 322–330. DOI: 10.3115/977035.977047.

[25] Mahak Gambhir and Vishal Gupta. “Recent automatic text summarization techniques: a survey”. In: *Artificial Intelligence Review* 47.1 (Jan. 2017), pp. 1–66. ISSN: 15737462. DOI: 10.1007/s10462-016-9475-9.

[26] Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. “Opinosis: A Graph Based Approach to Abstractive Summarization of Highly Redundant Opinions”. In: *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. Beijing, China: The COLING 2010 Organizing Committee, 2010, pp. 340–348.

[27] Jade Goldstein et al. *Multi-document summarization by sentence extraction*. 2000. DOI: 10.3115/1567564.1567569.

[28] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT Press, 2016. ISBN: 9780262035613.

[29] Max Grusky, Mor Naaman, and Yoav Artzi. “Newsroom: A Dataset of 1.3 Million Summaries with Diverse Extractive Strategies”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, 2018, pp. 708–719. DOI: 10.18653/v1/n18-1065. arXiv: 1804.11283.

[30] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2016, pp. 770–778. ISBN: 978-1-4673-8851-1. DOI: 10.1109/CVPR.2016.90.

[31] G. E. Hinton and R. R. Salakhutdinov. “Reducing the dimensionality of data with neural networks”. In: *Science* 313.5786 (2006), pp. 504–507. ISSN: 00368075. DOI: 10.1126/science.1127647.

[32] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9.8 (1997), pp. 1735–1780. ISSN: 08997667. DOI: 10.1162/neco.1997.9.8.1735.

[33] Elad Hoffer and Nir Ailon. “Deep metric learning using triplet network”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 9370 (2015), pp. 84–92. ISSN: 16113349. DOI: 10.1007/978-3-319-24261-3_7. arXiv: 1412.6622.

[34] Kai Hong and Ani Nenkova. “Improving the Estimation of Word Importance for News Multi-Document Summarization”. In: *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. Gothenburg, Sweden: Association for Computational Linguistics, 2014, pp. 712–721. DOI: 10.3115/v1/E14-1075.

[35] Matthew Honnibal and Mark Johnson. "An Improved Non-monotonic Transition System for Dependency Parsing". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, 2015, pp. 1373–1378. DOI: 10.18653/v1/D15-1162.

[36] Eduard Hovy et al. "Automated Summarization Evaluation with Basic Elements." In: *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*. Genoa, Italy: European Language Resources Association (ELRA), 2006.

[37] Akanksha Joshi et al. "SummCoder: An Unsupervised Framework for Extractive Text Summarization Based on Deep Auto-encoders". In: *Expert Systems with Applications* (2019). ISSN: 0957-4174. DOI: 10.1016/J.ESWA.2019.03.045.

[38] Dan Jurafsky and James H. Martin. *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition*. 2nd ed. Pearson Prentice Hall, 2009, p. 988. ISBN: 0135041961.

[39] Dan Jurafsky and James H. Martin. *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition*. 3rd ed. dr. Pearson Prentice Hall, 2019.

[40] Uday Kamath et al. "Attention and Memory Augmented Networks". In: *Deep Learning for NLP and Speech Recognition*. Cham, Switzerland: Springer International Publishing, 2019, pp. 407–462. DOI: 10.1007/978-3-030-14596-5_9.

[41] Katharina Kann, Sascha Rothe, and Katja Filippova. "Sentence-Level Fluency Evaluation: References Help, But Can Be Spared!" In: *Proceedings of the 22nd Conference on Computational Natural Language Learning*. Brussels, Belgium: Association for Computational Linguistics, 2018, pp. 313–323. DOI: 10.18653/v1/K18-1031.

[42] Ryan Kiros et al. "Skip-Thought Vectors". In: *Advances in Neural Information Processing Systems 28*. Ed. by C Cortes et al. Curran Associates, Inc., 2015, pp. 3294–3302.

[43] Alex Kulesza and Ben Taskar. "Determinantal point processes for machine learning". In: *Foundations and Trends in Machine Learning* 5.2-3 (2012), pp. 123–286. ISSN: 19358237. DOI: 10.1561/2200000044. arXiv: 1207.6083.

[44] Mirella Lapata and Regina Barzilay. "Automatic evaluation of text coherence: Models and representations". In: *IJCAI International Joint Conference on Artificial Intelligence*. Edinburgh, Scotland, 2005, pp. 1085–1090.

[45] Logan Lebanoff, Kaiqiang Song, and Fei Liu. "Adapting the Neural Encoder-Decoder Framework from Single to Multi-Document Summarization". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, 2018, pp. 4131–4141. DOI: 10.18653/v1/D18-1446.

[46] Piji Li et al. "Salience estimation via variational auto-encoders for multi-document summarization". In: *31st AAAI Conference on Artificial Intelligence, AAAI 2017* (2017), pp. 3497–3503.

[47] Chin-Yew Lin and Eduard Hovy. "Manual and automatic evaluation of summaries". In: *Proceedings of the ACL-02 Workshop on Automatic Summarization -* . Vol. 4. Phildadelphia, Pennsylvania, Association for Computational Linguistics, 2002, pp. 45–51. DOI: 10.3115/1118162.1118168.

[48] Hui Lin and Jeff Bilmes. "A Class of Submodular Functions for Document Summarization Extractive Document Summarization The figure below represents the sentences of a document". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1* (2011), pp. 510–520.

[49] Lucian Vlad Lita et al. "tRuEcasIng". In: *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - ACL '03*. Vol. 1. Morristown, NJ, USA: Association for Computational Linguistics, 2003, pp. 152–159. DOI: 10.3115/1075096.1075116.

[50] Yinhan Liu et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: (2019). arXiv: 1907.11692.

[51] H. P. Luhn. "The Automatic Creation of Literature Abstracts". In: *IBM Journal of Research and Development* 2.2 (1958), pp. 159–165. ISSN: 0018-8646. DOI: 10.1147/rd.22.0159.

[52] Thang Luong, Hieu Pham, and Christopher D. Manning. "Effective Approaches to Attention-based Neural Machine Translation". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, 2015, pp. 1412–1421. DOI: 10.18653/v1/D15-1166.

[53] Shulei Ma, Zhi-Hong Deng, and Yunlun Yang. "An Unsupervised Multi-Document Summarization Framework Based on Neural Document Model". In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, Japan: The COLING 2016 Organizing Committee, 2016, pp. 1514–1523.

[54] Christopher D. Manning, Hinrich Schütze, and Gerhard Weikurn. *Foundations of Statistical Natural Language Processing*. 1st ed. MIT Press Ltd, 1999. ISBN: 0262133601. DOI: 10.1145/601858.601867.

[55] Oren Melamud, Jacob Goldberger, and Ido Dagan. "context2vec: Learning Generic Context Embedding with Bidirectional LSTM". In: *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Berlin, Germany: Association for Computational Linguistics, 2016, pp. 51–61. DOI: 10.18653/v1/K16-1006.

[56] Rada Mihalcea and Paul Tarau. "TextRank: Bringing Order into Text". In: *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. Barcelona, Spain: Association for Computational Linguistics, 2004, pp. 404–411.

[57] Tomas Mikolov et al. "Distributed representations ofwords and phrases and their compositionality". In: *Advances in Neural Information Processing Systems* (2013). ISSN: 10495258. arXiv: 1310.4546.

[58] Tomas Mikolov et al. "Efficient Estimation of Word Representations in Vector Space". In: *1st International Conference on Learning Representations, {ICLR}*. Scottsdale, Arizona, USA, 2013. arXiv: 1301.3781.

[59] Tomas Mikolov et al. "Recurrent neural network based language model". In: *Eleventh annual conference of the international speech communication association*. Makuhari, Chiba, Japan, 2010.

[60] Tom M Mitchell. *Machine learning*. Vol. 1st ed. McGraw-Hill Education Ltd, 1997. ISBN: 978-0070428072.

[61] Frederic Morin and Yoshua Bengio. "Hierarchical probabilistic neural network language model". In: *AISTATS 2005 - Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics*. 2005, pp. 246–252. ISBN: 097273581X.

[62] Mir Tafseer Nayeem, Tanvir Ahmed Fuad, and Yllias Chali. "Abstractive Unsupervised Multi-Document Summarization using Paraphrastic Sentence Fusion". In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, pp. 1191–1204.

[63] Ani Nenkova, Rebecca Passonneau, and Kathleen Mckeown. "The Pyramid Method: Incorporating human content selection variation in summarization evaluation". In: *ACM Transactions on Speech and Language Processing* 4.2 (May 2007), 4–es. ISSN: 15504875. DOI: 10.1145/1233912.1233913.

[64] Jun Ping Ng and Viktoria Abrecht. "Better summarization evaluation with word embeddings for ROUGE". In: *Conference Proceedings - EMNLP 2015: Conference on Empirical Methods in Natural Language Processing* (Aug. 2015), pp. 1925–1930. arXiv: 1508.06034.

[65] Daniel W. Otter, Julian R. Medina, and Jugal K. Kalita. "A Survey of the Usages of Deep Learning in Natural Language Processing". In: (2018). arXiv: 1807.10854.

[66] Karolina Owczarzak et al. "An Assessment of the Accuracy of Automatic Evaluation in Summarization". In: *Proceedings of Workshop on Evaluation Metrics and System Comparison for Automatic Summarization*. Montréal, Canada: Association for Computational Linguistics, 2012, pp. 1–9.

[67] Kishore Papineni et al. "BLEU". In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*. Morristown, NJ, USA: Association for Computational Linguistics, 2001, p. 311. DOI: 10.3115/1073083.1073135.

[68] Rebecca J. Passonneau et al. "Automated Pyramid Scoring of Summaries using Distributional Semantics". In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Sofia, Bulgaria: Association for Computational Linguistics, 2013, pp. 143–147.

[69] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "GloVe: Global vectors for word representation". In: *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*. Doha, Qatar: Association for Computational Linguistics, 2014, pp. 1532–1543. ISBN: 9781937284961. DOI: 10.3115/v1/d14-1162.

[70] Christian S. Perone, Roberto Silveira, and Thomas S. Paula. "Evaluation of sentence embeddings in downstream and linguistic probing tasks". In: (June 2018). arXiv: 1806.06259.

[71] Matthew Peters et al. "Deep Contextualized Word Representations". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana, USA: Association for Computational Linguistics, 2018, pp. 2227–2237. DOI: 10.18653/v1/n18-1202. arXiv: 1802.05365.

[72] Maxime Peyrard and Judith Eckle-Kohler. "Supervised learning of automatic pyramid for optimization-based multi-document summarization". In: *ACL 2017 - 55th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*. Vol. 1. Vancouver, Canada: Association for Computational Linguistics, 2017, pp. 1084–1094. ISBN: 9781945626753. DOI: 10.18653/v1/P17-1100.

[73] Alec Radford and Tim Salimans. *Improving Language Understanding by Generative Pre-Training*. 2018.

[74] Nils Reimers and Iryna Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2019. arXiv: 1908.10084.

[75] Lior Rokach and Oded Maimon. "Clustering Methods". In: *Data Mining and Knowledge Discovery Handbook*. New York: Springer-Verlag, 2005, pp. 330–331. DOI: 10.1007/0-387-25465-X_15.

[76] Stuart Rose et al. "Automatic Keyword Extraction from Individual Documents". In: *Text Mining: Applications and Theory*. Chichester, UK: John Wiley & Sons, Ltd, Mar. 2010, pp. 1–20. ISBN: 9780470749821. DOI: 10.1002/9780470689646.ch1.

[77] Gaetano Rossiello, Pierpaolo Basile, and Giovanni Semeraro. "Centroid-based Text Summarization through Compositionality of Word Embeddings". In: *Proceedings of the MultiLing 2017 Workshop on Summarization and Summary Evaluation Across Source Types and Genres*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2017, pp. 12–21. DOI: 10.18653/v1/W17-1003.

[78] Raphael Schumann. "Unsupervised Abstractive Sentence Summarization using Length Controlled Variational Autoencoder". In: (Sept. 2018). arXiv: 1809.05233.

[79] Abigail See, Peter J. Liu, and Christopher D. Manning. "Get To The Point: Summarization with Pointer-Generator Networks". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, 2017, pp. 1073–1083. DOI: 10.18653/v1/P17-1099.

[80] Elaheh ShafieiBavani et al. "A Graph-theoretic Summary Evaluation for ROUGE". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, 2018, pp. 762–767. DOI: 10.18653/v1/D18-1085.

[81] Elaheh ShafieiBavani et al. "An Efficient Approach for Multi-Sentence Compression". In: *Proceedings of The 8th Asian Conference on Machine Learning*. Ed. by Robert J Durrant and Kee-Eung Kim. Vol. 63. Proceedings of Machine Learning Research. The University of Waikato, Hamilton, New Zealand: PMLR, 2016, pp. 414–429.

[82] Madhvi Soni and Jitendra Singh Thakur. "A Systematic Review of Automated Grammar Checking in English Language". In: (Mar. 2018). arXiv: 1804.00540.

[83] Josef Steinberger and Karel Jezek. *Using Latent Semantic Analysis in Text Summarization and Summary Evaluation*. 2004. DOI: 10.1177/0165551511408848.

[84] Josef Steinberger and Karel Ježek. "Evaluation measures for text summarization". In: *Computing and Informatics* 28.2 (2009), pp. 251–275. ISSN: 13359150.

[85] Josef Steinberger, Peter Krejzl, and Tomáš Brychcín. "Pyramid-based Summary Evaluation Using Abstract Meaning Representation". In: *RANLP 2017 - Recent Advances in Natural Language Processing Meet Deep Learning*. Varna, Bulgaria: Incoma Ltd. Shoumen, Bulgaria, Nov. 2017, pp. 701–706. ISBN: 9789544520496. DOI: 10.26615/978-954-452-049-6_090.

[86] Raphael Tang et al. "Distilling Task-Specific Knowledge from BERT into Simple Neural Networks". In: (2019). arXiv: 1903.12136.

[87] Simone Teufel and Hans Van Halteren. "Evaluating Information Content by Factoid Analysis: Human annotation and stability." In: *Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP '04)* (2004), pp. 419–426.

[88] J R R Tolkien. *The Fellowship of the Ring*. Vol. 1. London: George Allen & Unwin Ltd., 1954.

[89] Dung Tran Tuan, Nam Van Chi, and Minh Quoc Nghiem. "Multi-sentence Compression Using Word Graph and Integer Linear Programming". In: *Studies in Computational Intelligence*. Vol. 710. Springer, Cham, 2017, pp. 367–377. DOI: 10.1007/978-3-319-56660-3_32.

[90] Ashish Vaswani et al. "Attention is all you need". In: *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., 2017, pp. 5999–6009. arXiv: 1706.03762.

[91] Xiaojun Wan. "Automatic Text Simplification". In: *Computational Linguistics* 44.4 (Dec. 2018), pp. 659–661. ISSN: 0891-2017. DOI: 10.1162/coli_r_00332.

[92] Qian Yang, Rebecca J. Passonneau, and Gerard De Melo. "PEAK: Pyramid evaluation via automated knowledge extraction". In: *30th AAAI Conference on Artificial Intelligence, AAAI 2016*. AAAI Press, 2016, pp. 2673–2679. ISBN: 9781577357605.

[93] Zhilin Yang et al. "XLNet: Generalized Autoregressive Pretraining for Language Understanding". In: (2019). arXiv: 1906.08237.

[94] Michihiro Yasunaga et al. "Graph-based Neural Multi-Document Summarization". In: *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*. Vancouver, Canada: Association for Computational Linguistics, 2017, pp. 452–462. DOI: 10.18653/v1/K17-1045.

[95] Aston Zhang et al. *Dive into Deep Learning*. Creative Commons licence, published on http://www.d2l.ai, 2019.

[96] Fangfang Zhang, Jin-ge Yao, and Rui Yan. "On the Abstractiveness of Neural Document Summarization". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, 2018, pp. 785–790. DOI: 10.18653/v1/D18-1089.

[97] Yukun Zhu et al. "Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, Dec. 2015, pp. 19–27. ISBN: 978-1-4673-8391-2. DOI: 10.1109/ICCV.2015.11.

# List of Figures

# List of Tables