# Automatic Summarization of News Articles

## 1) Introduction

Automatic summarization is a popular task in the field of Natural Language Processing due to the vast number of applications for such systems. While this has been relevant for a long time already, it's becoming even more crucial nowadays. The number of new written material published about any topic every day is so high that it becomes infeasible for humans to keep themselves updated in their field by reading everything, so efficient and reliable ways of condensing information are needed more than ever. Aside from its practical use though, this task also holds a certain fascination. Intuitively, to summarize a text, one needs to understand it first, so giving a program the ability to extract the most important information in a text is both a challenge and one marker of how much progress has been made on the path towards teaching machines to understand text. While systems for automatic summarization are already adding value in many contexts that mainly require a rough overview of meaning, there are still various problems to be solved on the theoretical side, most of all regarding the generation of natural-sounding and cohesive language.

In this project, I compare two of the methods available for this task using a corpus of newspaper articles. In section 2 of the report, I will outline the task, some of the common approaches to it, and how it's usually evaluated. In section 3, I will describe my dataset and the methods I chose to work with. Sections 4 and 5 follow with my results and a discussion thereof, as well as some areas for future work. Lastly, I end with a conclusion and my references.

## 2) Background

Even though it's trending again now because of the new possibilities with large datasets, language models, and deep learning, this is not a new task, with the first paper about it dating back to 1958 (Luhn). Generally, there are several variations of the summarization task. Some of the main distinguishing characteristics are the type of input, level of detail, type of output, content, and language of the summary (Mahdavi & Nazari, 2019). The input can be either single documents or collections of several documents. Regarding the level of detail, there are indicative summaries, which only list the topics mentioned in a text, and informative summaries, which aim to contain all relevant information in a condensed form. The type of output, or rather the way the output is generated is one of the most important characteristics: Extractive summarizers use various criteria to identify the most important sentences of a text and extract these as a summary. Abstractive summarizers create a representation of what the text means and then generate a summary of new sentences containing the same meaning. There are advantages and disadvantages of both approaches, and I will get back to them further on. Regarding the content, there are general summaries, summaries focusing on a specific domain, and summaries driven by specific queries. Lastly, summaries can be monolingual, where text and summary are in the same language, multi-lingual, where texts of different languages are distilled into one summary, and cross-lingual, where the text is in one language, and the summary in another one.

There are also numerous different approaches to getting accurate summaries, so I will only give a very brief overview here. Mahdavi and Nazari group the work done in the field so far into 4 categories: Statistical, machine learning-, semantic-, and AI-based methods (2019). Statistical methods don't look at the meaning of words or the relations between them, but just the form and shape of a text. For example, Luhn (1958) used word frequency to summarize technical documents. Other examples for statistical models look at sentence position, the structure of a text (e.g. if a sentence is in the heading or not), and the presence of cue words to identify the most important sentences (Das & Martins, 2007).

Machine learning methods look at extractive summarization as a binary classification task, going over every sentence in the text and predicting whether it should be part of the summary or not (Mahdavi & Nazari, 2019). Such systems use several parameters to decide which sentences are important, for example word frequency, number of uppercase words, length of sentences, sentence position, or the sentence's phrase structure, presence of title words in the sentence, or the presence of priorly extracted or created lists of thematic words. Machine learning methods for automatic summarization range from relatively simple algorithms with predefined features (such as the ones mentioned above) to neural networks that learn some features themselves, or models of fuzzy logic, which can see beyond typical binary distinctions and separate sentences into important, average, and irrelevant ones (Mahdavi & Nazari, 2019). However, they are all supervised forms of learning and need large training corpora with reference summaries for every language, and ideally even every genre a system is supposed to work on.

Semantic methods don't look at the form or structure of texts to summarize them, but at the word meanings and relationships between words by utilizing dictionary/thesaurus knowledge or part-of-speech tagging to identify important sentences. Mahdavi and Nazari (2019) describe for example lexical chains, clustering methods, or graph-based methods. The latter two will be explained in more depth in section 3, as these are the approaches I chose for my project. The last group, which Mahdavi and Nazari refer to as AI-based, is a relatively recent development, based on large-scale datasets, advanced neural network architectures, and pre-trained language models. There has been promising recent work with encoder-decoder models based on CNNs or RNNs, bi-directional LSTMs, and using attention models (Mohd et al., 2020), as well as on fine-tuning BERT for summarization (e.g. Liu, 2019). Such approaches also gave rise to more work on abstractive summarization.

Reliable evaluation of automatically created summaries is problematic due to the fact that there's low inter-annotator agreement even among humans, which is not surprising given the fact that there are many ways to correctly summarize one and the same text. The measure that has become standard in the field is called ROUGE, which stands for recall-oriented understudy for gisting information. This compares a generated summary with a reference summary by measuring how many tokens in the reference summary also appear in the generated one. There are several variants, depending on whether the texts are split into unigrams (ROUGE-1), bigrams (ROUGE-2), the longest matching sequence (ROUGE-L), or skip-grams and unigrams (ROUGE-SU) (Ganesan 2018). This score is not ideal since it only regards the presence or absence of words/n-grams, but not the form and coherence of the summary. It also doesn't capture synonymy, so a summary can successfully contain the most important topics but still get a low ROUGE score if different synonyms than in the reference are used. This is primarily a problem in abstractive summarization though, so it's of no further concern for this project. Ganesan (2018) suggests some improvements, but ROUGE remains the standard despite its shortcomings, so I will also use it in this report for the sake of comparability.

## 3) Data and Methods

I chose to focus on the domain of newspaper articles for 2 reasons. First, it's one of the most common domains in the field of automatic summarization, maybe due to the relatively easy accessibility of large datasets. Second, because the relatively short genre of news articles allows me to make judgements about the summaries myself, which would be a lot more difficult with longer or more complex genres such as academic articles. The dataset I decided to work with is the CNN/Daily-Mail dataset, a collection of over 300 000 news articles with corresponding reference summaries (created by concatenating highlighted words/phrases), and one of the standard datasets in the field (Ruder, 2020). However, since I decided for unsupervised approaches that don't require training, I used only a fraction of the dataset in the end for evaluation.

For my project, I decided to work on extractive methods. They have the advantage of often being computationally simpler, and of automatically returning grammatically correct and complete sentences (although often in somewhat incohesive succession). The semantics-based methods appealed to me because they are quite intuitive and somewhat mirror how a human would go about picking the most important sentences in a text. Therefore, I implemented a graph-based and a cluster-based algorithm for comparison.

My implementation of the graph-based model uses the TextRank algorithm. TextRank is introduced by Mihalcea and Tarau (2004) and is based on graph-based systems for ranking webpages. It is a way to find connections between sentences by using the concept of recommendation. Sentences that are connected, in this case similar, recommend each other, and the weight of a recommendation depends on the rank of the recommending sentence itself (Mihalcea & Tarau, 2004). Applied to automatic summarization, this means that the system works under the assumption that important content generally appears multiple times throughout a text. Therefore, by finding out which sentences are closest connected to other sentences, and taking the top n of these, an extractive summary can be created.

For the practical implementation of this logic, I was guided by Joshi's description of using TextRank for multi-document summarization (2018) but adapted the logic to work on single documents. In the first step, I pre-process the data. Since the version of the dataset I'm using is somewhat messy in terms of formatting, I start by deleting additional punctuation, empty lines, etc., separating the files into articles and summaries, and reading them into two dataframes, one for the full articles and one for the reference summaries. In the second step, these cleaned-up texts are split into sentences, tokenized, and stop-words are removed. To get feature vectors for every sentence, pre-trained GloVe word embeddings with 100 dimensions are used and combined to sentence vectors by taking their average. Next, a similarity matrix is filled by calculating the cosine similarity between all possible sentence pairs in the text. This similarity matrix is then converted into a graph with help of the networkx library. Finally, the pagerank function is applied to the graph, extracting scores that indicate the importance, or connectedness, of every sentence. The original, unchanged sentences (including stopwords) are then sorted by these scores and the n highest-ranking ones make up the summary, depending on the summary length defined in the function call.

The second approach I implemented uses sentence embeddings collected with the skip-thought algorithm and simple k-means clustering to group the sentences by meaning similarity. This is inspired by Kushal Chauhan's work on email summarization (2018). There are various ways of encoding sentence meaning in embedding vectors. One common approach is to somehow combine the individual

embedding vectors of all words in the sentence, for example by concatenation, weighted summation, or taking the average, as described above. However, these approaches usually don't encode the semantic and syntactic relations between the words, such as word order. Therefore, I used a different option here. The skip-thought method is introduced by Kiros et al (2015) and adapts the logic behind skipgram word embeddings to the sentence level in order to get general, task-independent representations of sentence meaning. They use an encoder-decoder model consisting of RNNs with GRU layers. In the encoder, sentences are encoded to feature vectors of a specified length. Then, two decoders use this representation to predict the last and the next sentence. The model is optimized on the accurate prediction of the preceding and following sentence, which results in embedding vectors that are very similar for sentences that are similar in semantics and/or syntax (Kiros et al., 2015).

In my model, I used an implementation of this algorithm made available through pypi.org by Remi Cadene. The preprocessing steps are the same as in the first approach described. Then, the full vocabulary of the text (or texts) to summarize is extracted in order to train the sentence embeddings. I chose to use only the articles that get summarized to train the embeddings to avoid potential bias from other domains. However, training on a bigger and more varied dataset would also have advantages, so I will leave further experiments and optimization of this for future work. After the sentence embeddings are trained and extracted for each sentence, they are split into n clusters, depending on the target summary length, using scikit-learn's kmeans-clustering. In every cluster, the most central sentence as well as the average position of the cluster's sentences in the text are calculated. Finally, the summary consists of the most central sentences, sorted by their cluster's average sentence position.
In both methods, if the original text isn't longer than the target summary length, the full text is returned.

For evaluating the models, I use the standard ROUGE-scores as described above. They can be calculated both for individual samples, and for a list of samples. For easier comparison between the two approaches, the results are also visualized in a bar chart.

## 4) Results

I tested both models on a test set of 500 articles. In the first experiment, I use a target summary length of 3, since that corresponds most closely to the average length of the reference summaries. In a second experiment, I use a length of 15 to test how generating significantly longer summaries affects the scores.

The tables below show the results for experiment 1 (summary length 3) in yellow and experiment 2 (summary length 15) in orange for both methods. As can be seen, the results are almost identical between the two approaches, but differ greatly depending on which score one looks at, with rouge-1 (which looks at unigrams) and rouge-l (which compares the longest matching string) being significantly higher than rouge-2 (bigrams) throughout both methods and experiments. The graphs show the comparison between the TextRank-based and cluster-based models by rouge-1.

| TextRank | f1-score | | precision | | recall | |
|---|---|---|---|---|---|---|
| rouge-1 | 0.24 | 0.17 | 0.19 | 0.10 | 0.34 | 0.69 |
| rouge-2 | 0.07 | 0.07 | 0.05 | 0.04 | 0.09 | 0.27 |
| rouge-l | 0.24 | 0.22 | 0.19 | 0.13 | 0.34 | 0.68 |

*Table 1: Results for TextRank*

| Cluster | f1-score | | precision | | recall | |
|---|---|---|---|---|---|---|
| rouge-1 | 0.24 | 0.17 | 0.19 | 0.10 | 0.34 | 0.70 |
| rouge-2 | 0.07 | 0.07 | 0.05 | 0.04 | 0.10 | 0.28 |
| rouge-l | 0.24 | 0.21 | 0.19 | 0.13 | 0.35 | 0.69 |

*Table 2: Results for skip-thought/cluster*

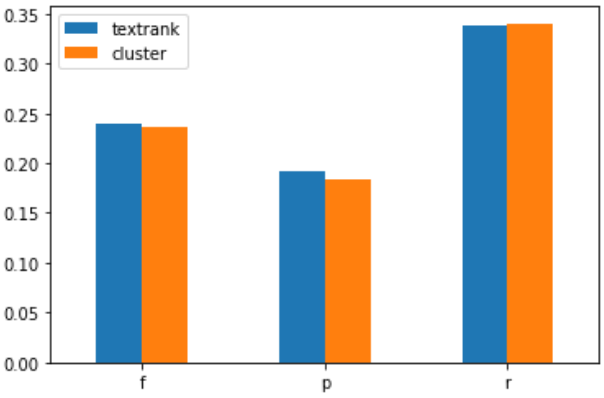


*Figure 1: Experiment 1*

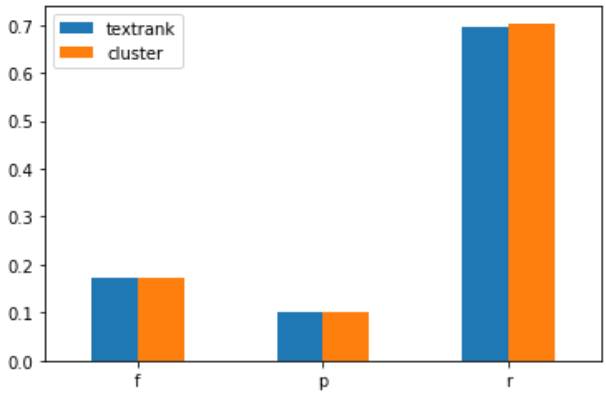


*Figure 2: Experiment 2*

The comparison between the two experiments yields very clear and intuitive results. Due to the longer summaries, recall is much higher in the second experiments, and precision significantly lower. This results in a lower f1-score for the longer summaries overall.

To put these results into perspective, state-of-the-art models (mostly abstractive methods based on deep learning) trained and evaluated on the full version of the CNN/DailyMail corpus reach an f1-score of around 40% on rouge-1, around 15-20% on rouge-2, and around 35-40% on rouge-l (Ruder, 2020).

However, due to the above-mentioned shortcomings of ROUGE, I also manually looked through some samples and their summaries for a more qualitative (and naturally subjective) evaluation. The first thing that becomes clear is that even when the number of sentences in the generated summaries correspond to the average length of the reference summaries, the generated summaries are on average longer. Evidently, when humans write summaries, they tend to use shorter sentences than in complete newspaper articles. Secondly, despite the similarity of the models' ROUGE-scores, the impression one gets when reading the summaries is not quite that similar. The summaries generated with the skip-thought embeddings and clustering are generally easier to read and understand. The main reason for this seems to be the fact that the extracted sentences are ordered in a more meaningful way in this version, given that the skip-thought/clustering variant sorts the sentences by the clusters' average sentence position in the text, whereas the TextRank version simply sorts the sentences by their rank in the graph, based on their similarity to other sentences. However, in this context, it should be noted that the reference summaries are also not ideal for quickly understanding the articles' main topic. As mentioned in the description of the dataset above, the summaries were manually created by concatenating highlighted words/phrases from the text, which does not result in the most natural language. Additionally, most of the articles in the dataset are relatively long, making it difficult to summarize them well in an average of under 3 sentences.

## 5) Discussion and Future Work

Overall, these results show that it is possible to generate useful, although not ideal, summaries by using relatively simple extractive and unsupervised methods. Even though the ROUGE-score does not give a complete evaluation of a summary, it is a good estimate of whether the main topics and keywords were covered. It also allows comparison with other systems, especially since I used a subset of the CNN/DailyMail corpus, a commonly used one for automatic summarization. As expected due to the relative simplicity of the algorithms I used and the timeframe and scope of this project, my results lie far below the state-of-the-art. However, they are still high enough to show that the models are successfully extracting useful information from the articles. This evaluation is confirmed by the impression I gained from manually looking through some examples. The generated summaries do extract the general topics and most important keywords from the texts, but clearly aren't close to anything a human summarizer would produce. However, both in the interpretation of the ROUGE scores and in the subjective comparison, the quality and type of reference summaries also has to be taken into account. As concatenations of manually highlighted words and phrases, they are neither completely extractive, nor completely abstractive, and also not very natural. So in this case, the gold standard we compare the generated results against might not be as "golden" as one could expect.

In this context, a note about the data in general might be in order. It was a bit of a challenge to choose a dataset that is freely available, has reference summaries, contains texts of a genre I could easily make sense of, and is used frequently enough in summarization tasks to make comparison with other models possible. Overall, the CNN/DailyMail dataset fulfilled these criteria nicely, but it had one clear disadvantage. The version I downloaded was rather messy, either by default or due to encoding issues. Even though I carefully pre-process the texts to clean them up as much as possible, they still contain random letters to varying degrees (that I can't remove without also removing other relevant letters). This does not impact the summarization task itself, but it adds to the chaotic impression of many of the generated summaries.

An unexpected result of this project is that the two methods I wanted to compare display almost identical performance. Assuming that the quality of sentence vectors is a deciding factor in how well a system captures the content of a text, I expected the skipthought/cluster approach to clearly outperform the one using TextRank. After all, it uses sentence vectors trained on in-domain texts, whereas the other one uses averages of general pretrained word embeddings. The fact that this did not happen brings up several points of interest for future work with this set-up. It would be interesting to find out whether the summarization method or the embedding method have the bigger influence on overall performance by repeating the same experiments once with averaged GloVe-embeddings as feature vectors for both TextRank and clustering, and once with skip-thought embeddings for both. Another area of interest would be the effect of more general or more specialized training data on the skip-thought embeddings. Training it once on a bigger number of newspaper articles from the same corpus and once on a more general or more varied set such as for example a Wikipedia dump, social media data, or news articles from other sources might yield interesting insights. In the same manner, different ways of combining word vectors into sentence vectors, such as concatenation or weighted sums could be explored to optimize this part of my TextRank implementation.
Regarding data, it would be insightful to experiment with different datasets and types of reference summaries. Especially longer and fully extractive ones could be interesting.

Apart from expansions of these models, I am still interested in further exploring the field of automatic summarization. While extractive approaches such as the ones above are useful as "safe bets" for extracting the general themes of a text, there seems to be more potential for natural-sounding and cohesive generated summaries in abstractive approaches. Therefore, I am curious about the benefits of encoder-decoder networks and large (pre-trained) language models on automatic summarization. If the scope and timeframe for this project weren't limited as they are, this would be a logical direction for expanding the project and for further comparison.

## 6) Conclusion

In this project, I implemented and compared two semantics-based unsupervised extractive methods for single-document automatic summarization on a portion of the CNN/DailyMail corpus. I showed that both an approach using pre-trained word embeddings and TextRank, and an approach using skip-thought sentence embeddings and clustering to summarize the articles perform very similarly according to the ROUGE metric. While they are clearly outperformed by more complex, mostly deep learning-based state-of-the-art models, the scores are high enough to indicate a useful level of information extraction. A more qualitative and subjective evaluation of random examples confirms this impression. The summaries are not necessarily easy to understand but enable the reader to grasp the main topic of an article, for example to decide whether the full text is of interest or not. The comparison between the two approaches does not show clear results, thus leaving several questions open for future work.

## 7) References

Chauhan, G. (2018). Unsupervised Text Summarization using Sentence Embeddings. Medium. https://medium.com/jatana/unsupervised-text-summarization-using-sentence-embeddings-adb15ce83db1

Das, D., Martins, A. F. T. (2007). A Survey on Automatic Text Summarization. http://www.cs.cmu.edu/~nasmith/LS2/das-martins.07.pdf

Ganesan, K. (2018). ROUGE 2.0: Updated and Improved Measures for Evaluation of Summarization Tasks. https://arxiv.org/pdf/1903.10318v2.pdf

Joshi, P. (2018, November 1). An Introduction to Text Summarization using the TextRank Algorithm (with Python implementation). Analytics Vidhya. https://www.analyticsvidhya.com/blog/2018/11/introduction-text-summarization-textrank-python/

Kiros, R., Zhu, Y., Salakhutdinov, R., Zemel, R. S., Torralba, A., Urtasun, R., Fidler, S. (2015). Skip-Thought Vectors. https://arxiv.org/pdf/1506.06726.pdf

Liu, Y. (2019). Fine-tune BERT for Extractive Summarization. https://arxiv.org/pdf/1903.10318v2.pdf

Luhn, H. P. (1958). The automatic creation of literature abstracts. *IBM Journal of Research and Development, 2 (2)*, 159-165, https://pdfs.semanticscholar.org/a6dc/c17c6f3dbc2d203ade9ff671a895a9dead7c.pdf?_ga=2.120969935.699216834.1603963909- 1662683943.1603810459

Mihalcea, R., Tarau, P. (2004). Textrank: Bringing order into text. *Proceedings of the 2004 conference on empirical methods in natural language processing*, https://www.aclweb.org/anthology/W04-3252.pdf

Mohd, M., Jah, R., Shah, M. (2020). Text document summarization using word embedding. *Expert Systems with Applications*, 143. https://doi.org/10.1016/j.eswa.2019.112958

Nazari, N., Mahdavi, M. A. (2019). A survey on Automatic Text Summarization. *Journal of AI and Data Mining, 7 (1),* 121-135. https://dx.doi.org/10.22044/jadm.2018.5742.1696

Ruder, S. (2020, October 28). *Summarization.* NLP-progress. http://nlpprogress.com/english/summarization.html