# The Effects of Pre-Processing on Text Summarisation

**Ben Taylor**

[redacted]

## Abstract

This paper explores how individual and combined preprocessing steps affect extractive text summarisation. Using the CNN/Daily Mail dataset, the impact of preprocessing was compared across two models: TextRank and BERTsum. Our findings show that each model's performance is greatly influenced by the preprocessing steps applied, with BERTsum favouring minimal preprocessing and TextRank preferring normalisation techniques such as stemming. The results highlight the importance of a nuanced, model-specific approach when designing preprocessing pipelines for extractive summarisation.

## 1 Introduction

Preprocessing techniques, used to clean and standardise data, are essential in almost all machine learning tasks. Yet the exact extent to which they impact text summarisation remains comparatively under-explored. With the ever-expanding use of Large Language Models (LLMS) like ChatGPT and Microsoft Copilot, the demand for efficient summarisation over a variety of corpora, both from corporations and the general public, grows. This paper investigates the effects of preprocessing on text summarisation to ascertain the best method or combination for the task.

Automatic text summarisation creates an abbreviated version of an original text whilst still maintaining many of the original's key points. Automatic text summarisation is typically split into two categories: abstractive(Moratanch and Chitrakala, 2016) and extractive(Mutlu et al., 2020). Abstractive generates new sentences that aim to capture the original text's meaning. Extractive takes a subset of sentences or phrases from the original input and creates a summary from them. This paper will focus solely on extractive summarisation as it is the more widely used of the two and relies on surface-level text features, which are directly affected by several pre-processing techniques, therefore making it easier to identify the effects.

The techniques explored in this paper are stopword removal, stemming, lemmatisation, word tokenisation, sentence segmentation, anaphora resolution, and lower casing. These were identified as commonly used techniques in text summarisation research(Gambhir and Gupta, 2017). Additionally, to assess the differences that preprocessing makes across summarisation paradigms, two different models were used: TextRank(Mihalcea and Tarau, 2004), a graph-based algorithm, and BERTsum(Liu, 2019), a deep learning transformer-based model. This dual approach allows for a more holistic view of how the effects of preprocessing on text summarisation vary by model.

## 2 Related Work

As the interest in Natural Language Processing grows, so does the research into improving its techniques, and text summarisation is no exception to this. WIDYASSARI et al.(Widyassari et al., 2022) identified the most commonly used preprocessing methods for text summarisation as stopword removal, stemming, tokenising, sentence segmentation, and word frequency analysis, along with several other influential methods such as POS (part of speech) tagging and bag-of-words.

Although preprocessing's impact on text summarisation for English texts has not been widely studied, the topic has been covered in papers focusing on Turkish(Salih and GUNAL, 2022) and Arabic(Elbarougy et al., 2020). Salih and Gunal analysed the effects of four preprocessing techniques alongside a new latent semantic analysis(Geetha and Deepamala, 2015) feature. Elbarougy et al. tailored preprocessing techniques to better tackle Arabic's linguistic complexities.

The effectiveness of summarisation varies from language to language due to differences in linguistic structure: morphologically rich languages (e.g. Turkish), non-configurational languages(Hale, 1983) (e.g. Warlpiri), and even languages without spaces between words (e.g. Chinese) all pose their own unique challenges. As such, researching English may offer a valuable insight into a best-case scenario thanks to it being morphologically simple and having a fixed word order. Thus, offering guidance to future studies on more complex languages.

## 3 Data

This study uses version 3.0 of the CNN/Daily Mail dataset(Hermann et al., 2015) from the Hugging Face library(See, 2017). A frequently used benchmark for summarisation research(See et al., 2017), the dataset consists of online news articles covering a variety of topics, each paired with a human-written summary. Version 3.0 introduced changes to the structure of the dataset to support summarisation and preserving named entities.

The dataset is an ideal candidate for several reasons. Its human written summaries, diverse topics, and pre-cleaned text offers a varied corpus with reliable points of comparison and minimal cleaning required.

Each sample in the data contains: an article (780 tokens on average), a summary written by the article author (average of 56 tokens), and a hashed source ID. For computational efficiency, a subset of 1000 articles from the full 300,000+ was used.

### 3.1 Ethical Considerations

While the CNN/Daily Mail dataset is composed of publicly available articles, it is important to acknowledge that it was collected without the original authors' explicit consent. As such, outputs created from this dataset should not be commercialised or misrepresented.

### 3.2 Limitations

The dataset solely contains news articles, potentially limiting generalisation to other domains. Furthermore, due to computational constraints, only a subset of the data is used, which may not represent the diversity of the complete corpus.

## 4 Methodology

In order to evaluate how preprocessing techniques affect the performance of extractive summarisation models, a number of text preprocessing pipelines were applied to a subset of the CNN/DailyMail dataset. Each of these pipelines were then summarised and evaluated using several metrics to compare effectiveness.

### 4.1 Dataset Preparation

The dataset sourced from Hugging Face is pre-cleaned and thus requires minimal preparation. The first 1000 articles were used to reduce computational demands. Sentence segmentation was applied to each article and used as a baseline for comparison against the other preprocessing methods as it is a common first step in extractive summarisation.

### 4.2 Preprocessing Pipelines

Seven preprocessing techniques were implemented as modular Python functions.

- Baseline (Sentence segmentation) - separates the text into a list of sentences
- Stop Word Removal - filters out low-value words such as "a" and "the"
- Stemming - removes suffixes and prefixes from words, helping to group similar words together
- Lemmatisation - reduces words to their base dictionary form
- Word Tokenization - splits the text into word tokens
- Anaphora Resolution - identifies what a pronoun/ referential expression refers to
- Lower Casing - changes all text to lower case

Each method was applied separately to the baseline data. These pipelines, consisting of both structural preprocessing and lexical transformations, were selected based on several factors such as how commonly used they are within similar research(Widyassari et al., 2022), their compatibility with summarisation models, and their ability to be applied alone and in combination with one another.

The pipelines were implemented through the use of NLTK(Bird et al., 2009) and spaCy(Honnibal et al., 2020) libraries.

#### 4.2.1 Combined Pipelines

Further study was conducted into which combination of the preprocessing pipelines produces the best results. A list of 20 unique combinations of 2 and 3 combined techniques was generated using itertools.combinations(). Stemming was removed from the list of combined techniques due to its similarity to lemmatisation and slightly worse performance. Just as with the individual pipelines, each combination was applied to the baseline, then summarised and evaluated on both models.

### 4.3 Summarisation models

TextRank and BERTsum were both implemented in Python. Using two separate models allows us to test the difference in performance across model types.

#### 4.3.1 TextRank

In TextRank, a sentence-similarity graph is created where sentences are nodes and the edges between nodes represent the sentence similarity. In this implementation, Term Frequency-Inverse Document Frequency(tf-idf) scoring was used to evaluate the similarity of sentences. This method is used as term overlap is sensitive to preprocessing, therefore making it a good candidate to highlight the effects of the different pipelines. Sentences in the graph are ranked using the PageRank(Page et al., 1999) algorithm, with the top 7% (percentage length of summary to article in the CNN dataset) of sentences being selected for the summary.

#### 4.3.2 Bertsum

Bertsum is a deep learning approach to summarisation adapted from the BERT model. This paper's implementation used distil-Bert(Sanh et al., 2020), a smaller and faster version of BERT-base. This implementation may not perform as well as the base version, but, requires far less time and memory to load and run.

BERTsum works by passing sentences through the distil-Bert model, which computes sentence embeddings from the average of the token embeddings in the model's final hidden layer. Cosine similarity is used to compare sentences against one another, with the most relevant being selected for the summary. Additionally, lazy-loading (where the model is only loaded when first called) and batch processing were implemented to reduce memory usage.

Bertsum offers a contrast to TextRank as it is less sensitive to shallow text changes, thus allowing us to explore how much preprocessing affects deep models.

### 4.4 Evaluation Methods

The generated summaries were evaluated against their human-written counterpart with several metrics. ROUGE scores (1,2, and L) were used to measure n-gram overlap between the summaries. Precision, recall, and F1 were calculated using BERTScore to evaluate the summary similarity beyond token matching. Compression score was used to assess conciseness by calculating the ratio of the summary length to the source length. Redundancy score was calculated using the number of repeated n-grams. These metrics provide a good perspective on the success of a summarisation by testing lexical, semantic, and efficiency measures.

## 5 Results

This section presents the results and findings from evaluating different preprocessing techniques on two extractive summarisation models (TextRank and BERTsum). Performance was evaluated using ROUGE scores (measuring lexical overlap), BERTScore metrics (measuring semantic similarity), compression, and redundancy.

### 5.1 Overview of Results

A table detailing each pipeline's key metrics can be found in Appendix A.

#### 5.1.1 ROUGE and BERTScore

For TextRank, stemming produced the highest ROUGE scores, however it produced the lowest BERTScores. This displays a trade-off between semantic similarity and selecting sentences with keywords that match the reference summaries. Interestingly, the baseline achieved the highest BERTScores whilst sharing identical ROUGE scores with lower casing, indicating that lower casing alone did not significantly impact the summarisation quality.

A few key differences were observed when using the BERTsum model; the baseline and lowercasing tied for the best ROUGE scores, while stemming ranked much lower than with TextRank. BERTsum generally produced higher BERTScores overall and favoured lighter preprocessing.

Across both models, Anaphora Resolution displayed high BERTScores, indicating its ability to preserve semantic fidelity.

These results highlight how some normalisation methods, like stemming and lemmatisation, can improve content overlap, but also come at the cost of degrading the semantic quality.

#### 5.1.2 Compression and Redundancy

Stopword removal produced the most compressed summaries across both models, but lowered the summary quality. Anaphora resolution provided best balance in terms of maintaining the content length while minimising repetition.

Overall the results show that while compression is a desirable trait, in the context of extractive summarisation, it may harm informativeness.

### 5.2 Individual Pipeline Visualisations

Figure 1 displays the evaluation metrics obtained from training the TextRank model on each pipeline. Appendix B contains the same graphs for the BERTsum model. The radar chart displays normalised metrics for each pipeline on a scale of 0-1, helping to visualise how each technique performs across all metrics.

### 5.3 Combined Pipelines

Of the 20 total combinations, Lemmatisation+Anaphora Resolution+Lower Casing performed the best with Lemmatisation+Anaphora

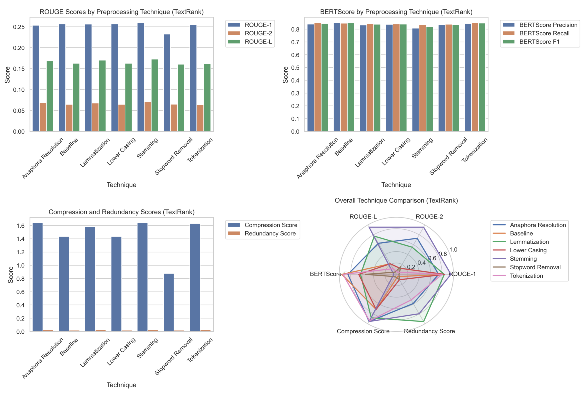


Figure 1: Results of the Effects of Preprocessing When Using TextRank

Resolution ranking 2nd. Lemmatisation appeared in all of the top six combinations, and anaphora resolution appeared in the top three, displaying both techniques strength in producing high quality summarisations. The figure in appendix C shows the top 10 combinations by Rouge-1 score.

## 6 Key Findings

From these results, several key findings can be observed.

Minimal preprocessing such as the sentence segmentation (the baseline) often performed just as well as other more complex techniques across both models.

Anaphora resolution offered consistently good scores across all metrics giving it the all around best performance.

Stemming and Stopword removal although effective for compression lead to degradation in ROUGE and BERTScores.

Lemmatisation outperformed stemming in preserving meaning and as such is likely a better option despite its more complex implementation and marginally slower runtime.

The effectiveness of preprocessing techniques vary by model type. TextRank benefited from lexical simplifications like stemming and lemmatisation. Whereas BERTsum -likely due to its full context attention mechanism- favoured less intrusive preprocessing.

Lemmatisation+Anaphora resolution offered the strongest trade-off between informativeness, compression and redundancy. With both techniques appearing frequently in many high-ranking combinations.

Adding tokenisation or lower casing to any combination often complemented stronger techniques without harming performance.

Overall the findings reinforce the importance of carefully selecting preprocessing technique/s based on the summarisation model and evaluation criteria. Additionally, over-preprocessing can harm informativeness and semantic nuance.

## 7 Conclusions and Future Work

### 7.1 Conclusions and Implications

This study investigated how preprocessing techniques affect extractive summarisation. Preprocessing techniques were investigated both individually and in combination using TextRank and BERTsum with each model reacting differently to the preprocessing techniques. TextRank benefited from simple text normalisation such as stemming. Whereas BERTsum favoured more minimal preprocessing. Additionally, anaphora resolution provided a well-balanced result across both models, performing well in all metrics. Stopword removal performed poorly in metrics relating to summary informativeness and is likely not a good choice for extractive summarisation.

As such, when designing a preprocessing pipeline for extractive summarisation, one must take into account the balance between informativeness, compression and redundancy, and be tailored to the model in use.

### 7.2 Limitations

Due to computational constraints, the experiments were conducted on a subset of the dataset with limited fine-tuning of the distilled BERTsum model meaning that its full potential was not explored.

Furthermore, the exclusive use of news articles means that it is unknown how well the results will generalise to other text domains.

### 7.3 Future Work

Future work on this topic could take on a variety of forms. Such as domain-specific analysis and working with scientific, social media or fiction-based corpora; utilising neural models, reinforcement learning, and long memory models; exploring adaptive preprocessing strategies where different techniques are used depending on the text's characteristics.

This study highlights that preprocessing for extractive summarisation is not one-size-fits-all and suggests that summarisation pipelines require a nuanced approach to preprocessing in order to reap the best results.

## References

Steven Bird, E Loper, and E Klein. 2009. Natural language processing with python o'reilly media inc. *O'Reilly Media Inc.*

Reda Elbarougy, Gamal Behery, and Akram El Khatib. 2020. *A Proposed Natural Language Processing Preprocessing Procedures for Enhancing Arabic Text Summarization*, pages 39–57. Springer International Publishing, Cham.

Mahak Gambhir and Vishal Gupta. 2017. Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review*, 47(1):1–66.

J Kamala Geetha and N Deepamala. 2015. Kannada text summarization using latent semantic analysis. In *2015 International conference on advances in computing, communications and informatics (ICACCI)*, pages 1508–1512. IEEE.

Ken Hale. 1983. Warlpiri and the grammar of non-configurational languages. *Natural language & linguistic theory*, 1:5–47.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *Advances in neural information processing systems*, 28.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spacy: Industrial-strength natural language processing in python. *Zenodo*.

Yang Liu. 2019. Fine-tune bert for extractive summarization. *Preprint*, arXiv:1903.10318.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411.

N Moratanch and S Chitrakala. 2016. A survey on abstractive text summarization. In *2016 International Conference on Circuit, power and computing technologies (ICCPCT)*, pages 1–7. IEEE.

Begum Mutlu, Ebru Sezer, and M. Akcayol. 2020. Candidate sentence selection for extractive text summarization. *Information Processing Management*, 57:102359.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab. Previous number = SIDL-WP-1999-0120.

BAL Salih and Efnan SORA GUNAL. 2022. The impact of features and preprocessing on automatic text summarization. *SCIENCE AND TECHNOLOGY*, 25(2):117–132.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *Preprint*, arXiv:1910.01108.

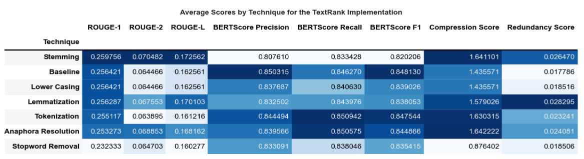
Abigail See. 2017. abisee/cnn_dailymail · Datasets at Hugging Face — huggingface.co. `https://huggingface.co/datasets/abisee/cnn_dailymail`.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Adhika Pramita Widyassari, Supriadi Rustad, Guruh Fajar Shidik, Edi Noersasongko, Abdul Syukur, Affandy Affandy, and De Rosal Ignatius Moses Setiadi. 2022. Review of automatic text summarization techniques methods. *Journal of King Saud University - Computer and Information Sciences*, 34(4):1029–1046.

## Appendix

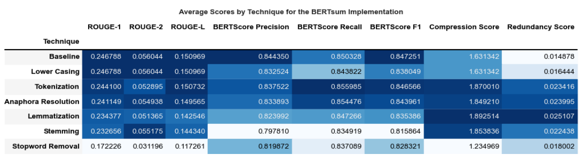
### A
### TextRank and BERTsums' Evaluation Metric Results



Average Scores by Technique for the TextRank Implementation

| Technique | ROUGE-1 | ROUGE-2 | ROUGE-L | BERTScore Precision | BERTScore Recall | BERTScore F1 | Compression Score | Redundancy Score |
|---|---|---|---|---|---|---|---|---|
| Stemming | 0.259756 | 0.070482 | 0.172562 | 0.807610 | 0.833428 | 0.820206 | 1.641101 | 0.026470 |
| Baseline | 0.256421 | 0.064466 | 0.162561 | 0.850315 | 0.846270 | 0.848130 | 1.435571 | 0.017786 |
| Lower Casing | 0.256421 | 0.064466 | 0.162561 | 0.837687 | 0.840630 | 0.839026 | 1.435571 | 0.018516 |
| Lemmatization | 0.256287 | 0.067553 | 0.170103 | 0.832502 | 0.843976 | 0.838053 | 1.579026 | 0.028295 |
| Tokenization | 0.255117 | 0.063895 | 0.161216 | 0.844494 | 0.850942 | 0.847544 | 1.630315 | 0.023241 |
| Anaphora Resolution | 0.253273 | 0.068853 | 0.168162 | 0.839566 | 0.850575 | 0.844866 | 1.642222 | 0.024081 |
| Stopword Removal | 0.232333 | 0.064703 | 0.160277 | 0.833091 | 0.838046 | 0.835415 | 0.876402 | 0.018506 |

Figure 2: TextRank Metrics



Average Scores by Technique for the BERTsum Implementation

| Technique | ROUGE-1 | ROUGE-2 | ROUGE-L | BERTScore Precision | BERTScore Recall | BERTScore F1 | Compression Score | Redundancy Score |
|---|---|---|---|---|---|---|---|---|
| Baseline | 0.246788 | 0.056044 | 0.150969 | 0.844350 | 0.850328 | 0.847251 | 1.631342 | 0.014878 |
| Lower Casing | 0.246788 | 0.056044 | 0.150969 | 0.832524 | 0.843822 | 0.838049 | 1.631342 | 0.016444 |
| Tokenization | 0.244100 | 0.052895 | 0.150732 | 0.837522 | 0.855985 | 0.846566 | 1.870010 | 0.023416 |
| Anaphora Resolution | 0.241149 | 0.054938 | 0.149565 | 0.833893 | 0.854476 | 0.843961 | 1.849210 | 0.023995 |
| Lemmatization | 0.234377 | 0.051365 | 0.142546 | 0.823992 | 0.847266 | 0.835386 | 1.892514 | 0.025107 |
| Stemming | 0.232656 | 0.055175 | 0.144340 | 0.797810 | 0.834919 | 0.815864 | 1.853836 | 0.022438 |
| Stopword Removal | 0.172226 | 0.031196 | 0.117261 | 0.819872 | 0.837089 | 0.828321 | 1.234969 | 0.018002 |

Figure 3: BERTsum Metrics

### B
### BERTsum Metric Result Visualisations

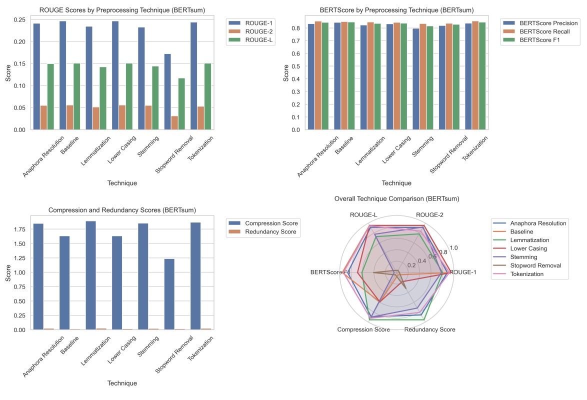


Figure 4: BERTsum Charts

## C
## Top 10 Preprocessing Combinations

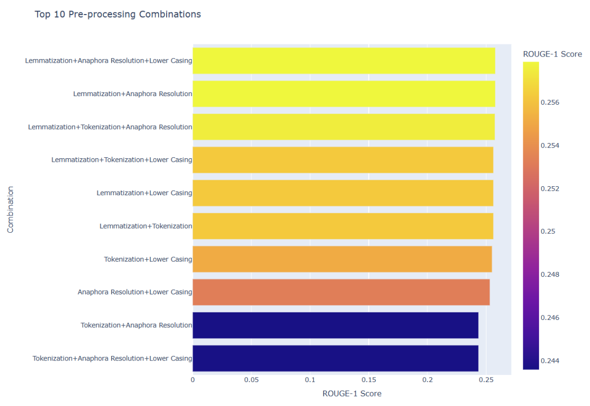


Figure 5: Top 10 Preprocessing Combinations by Rouge-1 Score