Text Mining and Search, a.y. 2024/2025

Aarne-Thompson-Uther Tale Type Index

Alice Brunazzi[1], **Alessandro Della Beffa**[2] **and Daniele Lepre**[3]

[1]*Bachelor's degree in Banking and Finance, Università degli Studi Milano-Bicocca*
[2]*Bachelor's degree in Mathematics, Università degli Studi di Milano*
[3]*Bachelor's degree in Banking and Finance, Università degli Studi Milano-Bicocca*

Abstract

This analysis aims to classify folktales genres and generate concise summaries through text mining techniques. The adopted approach seeks to enhance computational narratology by exploring the potential and limitations of automated text analysis in traditional storytelling.
The research employs machine learning approaches to extract meaningful patterns and structures from folktales; the results provide insights into the effectiveness of different methodologies and contribute to the understanding of narrative analysis through automated tools.

Keywords: *Aarne-Thompson-Uther,* ATU*, abstractive summarization, extractive summarization, fairy tale, folklore, folktale, natural language processing,* NLP*, text classification, text mining*

Revised: January 28th, 2025

Contents

1 Introduction 1
- 1.1 Goal of the project . . . 1
- 1.2 Research questions . . . 1

2 Data sets description 1
- 2.1 ATU data set (`atu_df`) . . . 1
- 2.2 AFT data set (`aft_df`) . . . 1

3 Data preparation and text preprocessing 2
- 3.1 Data preparation . . . 2
- 3.2 Text preprocessing and feature extraction . . . 2

4 Classification 2
- 4.1 *tf-idf* . . . 2
  Model training and hyperparameter tuning • Validation
- 4.2 Non-contextual word embeddings . . . 3
  Model training and hyperparameter tuning
- 4.3 Conclusion . . . 3

5 Text summarization 3
- 5.1 Data selection . . . 4
- 5.2 Abstractive summarization . . . 4
- 5.3 Extractive summary . . . 4
- 5.4 Evaluation metrics . . . 4
- 5.5 Conclusion . . . 4

6 Contact us 5

1. Introduction

1.1. Goal of the project

The primary objective of this project is to employ text mining techniques to classify and analyze folktales, utilizing structured data sets sourced from esteemed web catalogs of mythology and folklore. The classification is based on the *genre* to which each folktale is associated through its attributed ATU index. The subsequent analysis aims to produce extractive summaries of 22 selected representative folktales and compare the results with purposely produced generative summaries of the same folktales.

1.2. Research questions

- How effectively can text mining techniques be used to classify folktales into predefined categories based on narrative structures and thematic elements?
- To what extent can extractive and generative summarization detect meaningful events and turning points in the stylistically varied narratives of international folktales?

2. Data sets description

This study is carried out on the data sets meticulously collected and curated by Joshua Hagedorn in the Trilogy [1] project, who in turn derived the material from two sources: the web catalog Folktexts: A library of folktales, folklore, fairy tales, and mythology [2] by prof. D.L. Ashliman, and the book *Motif-index of folk-literature* [3] by Stith Thompson [4]. Designed for use in text mining and statistical analysis, these data sets provide a valuable foundation for future developments in the formal and structured study of mythological and folkloric literary traditions.

The following data sets descriptions summarize the exhaustive documentation available in the documentation provided in Trilogy Data Dictionary (see [1]).

2.1. ATU data set (`atu_df`)

Derived from the Aarne-Thompson-Uther Index (published by Hans-Jörg Uther) [5], the data set has 2247 rows, one for each tale type from the index. It contains the following variables:

- `chapter`: the highest level groupings (also referred to as *genre*) into which tale types are classified. It is the classification label and includes *Anecdotes And Jokes*, *Animal Tales*, *Formula Tales*, *Religious Tales*, and *Tales Of Magic*;
- `atu_id`: the tale type identifier. This data set contains one row per `atu_id`;
- `tale_type`: a brief summary of the plot of the tale type.

2.2. AFT data set (`aft_df`)

The Annotated Folk Tales data set contains folktales that have been marked as characteristic of a specific tale type from the ATU. It contains 1518 rows, one per tale, and only includes stories with a specified `atu_id`. Amongst its variables are:

- `atu_id`: the tale type identifier, as in `atu_df`;
- `tale_title`: the specific name of the tale;
- `provenance`: the country, region, tribe or author from which the tale comes;
- `text`: the narrative text of the tale itself.

3. Data preparation and text preprocessing

The preliminary steps outlined below were guided by the three main principles outlined in Han, J., Kamber, M., & Pei, J. (2011) [6]: **data integrity**, **analytical relevance** and, to a minor extent, **scalability**.

3.1. Data preparation

The project has been developed in Python: the data sets `atu_df` and `aft_df` were imported into a Jupyter Notebook and then manipulated using libraries such as Pandas. The data sets were merged into one (`aft`) via the shared field `atu_id`, so each tale `text` (in `aft_df`) could be associated to the `chapter` of its type.

No data were missing in the fields of interest `tale_title`, `text` and `chapter`. Rows with duplicate `tale_title` or `text` were also checked, and surprisingly found in 7 occurrences; after verifying their consistency in the target `chapter`, we decided to retain all the samples regardless, and avoid reducing the already limited number (1518) of texts at our disposal.

For classification purposes only, the classes of the target `chapter` were reduced from the original 7 to 5 by reincorporating two under-represented classes, *Other Animals And Objects* (64 samples) and *Other Tales Of The Supernatural* (72 samples), into their respective *genres*: *Tales Of Magic* and *Animal Tales*. The five classes retained are those listed in subsection **2.1** (see `chapter`), the largest being *Tales Of Magic* (450 samples), and the two smallest *Animal Tales* (280 samples) and *Formula Tales* (52 samples). For operational reasons, the classes have been encoded as numbers from 0 to 4 in a new variable `y`.

In addition, for each tale, we have prepended the title (`tale_title`)—considered, if known, to be an integral part of the tale—to the text (`text`), *e.g.*:

> **Cinderella or The Little Glass Slipper.** No sooner were the ceremonies of the wedding over …

Finally, the 1518 texts were partitioned into a training set (90%) and a validation set (10%) using stratified sampling.

3.2. Text preprocessing and feature extraction

To improve the quality of the textual data and to facilitate effective feature extraction, preprocessing steps were applied.

The typical text normalization operations were performed entirely using the library spaCy, which provides state-of-the-art pre-trained processing pipelines available in multiple languages and architectures. We chose the CPU-optimized English language model `en_core_web_lg`, which by default includes a tokenizer, a part-of-speech (POS) tagger, a syntactic dependency parser, a lemmatizer[1], and a named entity recognition (NER) component. Applying the pipeline to one or more texts produces a special `doc` object containing all the outputs resulting from the processing, including those of interest to us: the extracted tokens, their lemmas, and many other and more advanced features such as (non-contextual) embedding vector representation of both terms and documents.

In addition, we integrated the `textrank` component PyTextRank [7] (spaCy's third-party extension for TextRank algorithms) into the pipeline, which is later needed for extractive summarization.

4. Classification

For the classification of the tales according to *genre*, we essentially tried two text representation models (bag-of-words with simple *tf-idf*, and non-contextual embedding) and four classification models (support vector machines [SVM], Naive Bayes, *k*-nearest neighbours [*k*-NN], and random forests [RF]).

We also tried to improve the embeddings of the texts as the average of the embedding vectors of their terms weighted with the corresponding *tf-idf* value, but as we did not observe any improvements in the performance of the classifiers, the experiment is not further discussed here.

4.1. *tf-idf*

All the tools mentioned as being used in this subsection and the subsections derived from it belong to the scikit-learn library, to which we refer here once and for all.

The *tf-idf* matrix was computed with `TfidfVectorizer` using the texts from the training set reduced to the lemmatized forms of the terms, after removing non-alphabetical tokens (numbers, special characters, punctuation) and stop words. The choice of classifiers was guided by common practices in text classification (see, *e.g.*, this dedicated page of scikit-learn). Two models were trained, a linear SVM (`LinearSVC`) and the variant `ComplementNB` of the Naive Bayes model—specifically suggested as preferable in the case of unbalanced classes.

The training involved the optimization of the parameters specified in the subsequent subsection 4.1.1, as described there.

4.1.1. Model training and hyperparameter tuning

The parameters, a descriptive list of which follows, have been optimized with exhaustive search over grids of values specified via the `GridSearchCV` class:

- `ngram_range` (`TfidfVectorizer` parameter): the ranges of *n*-values for different *n*-grams to be extracted. Set to $(1, 1)$ (only unigrams) or $(1, 2)$ (unigrams and bigrams);
- `min_df` (`TfidfVectorizer` parameter): the threshold (or cut-off) of the minimum (integer) document frequency for terms not to be ignored when building the *tf-idf* vocabulary. Set to vary 1–10;
- `C` (only used in `LinearSVC`): regularization parameter. Set to an evenly spaced logarithmic grid $[10^{-3}, 10^{3}]$;
- `alpha` (only used in `ComplementNB`): additive smoothing parameter. Set as `C`.

These are the best resulting parameters and their corresponding evaluation metrics:

	`LinearSVC`	`ComplementNB`
`ngram_range`	$(1, 1)$	$(1, 2)$
`min_df`	3	4
`C`	1	-
`alpha`	-	0.139
Mean accuracy	0.796	0.738
Mean weighted F_1	0.793	0.726

Table 1. Best parameters

4.1.2. Validation

`LinearSVC` has been selected as the final classifier due to its superior performance.

Table 2 and figure 1 respectively show the results and the confusion matrix observed on the held-out validation set:

	Precision	Recall	F_1 score	Support
Anecdotes And Jokes [0]	0.74	0.81	0.77	31
Animal Tales [1]	0.88	0.86	0.87	35
Formula Tales [2]	1.00	0.60	0.75	5
Religious Tales [3]	0.78	0.62	0.69	29
Tales Of Magic [4]	0.81	0.90	0.85	52
Accuracy			0.81	152
Macro average	0.84	0.76	0.79	152
Weighted average	0.81	0.81	0.81	152

Table 2. Classification report

The chosen classifier, based on a number of features drastically reduced by the exclusion of bigrams and the appropriate choice of `min_df`, generalizes the model well over unknown tales, achieving

[1] spaCy's lemmatizer and NER, while excellent, sometimes mistake capitalized common nouns for named entities: this known problem has been fixed by modifying the lemmatizer's source code according to the guidelines of this Medium article by Jade Moillic

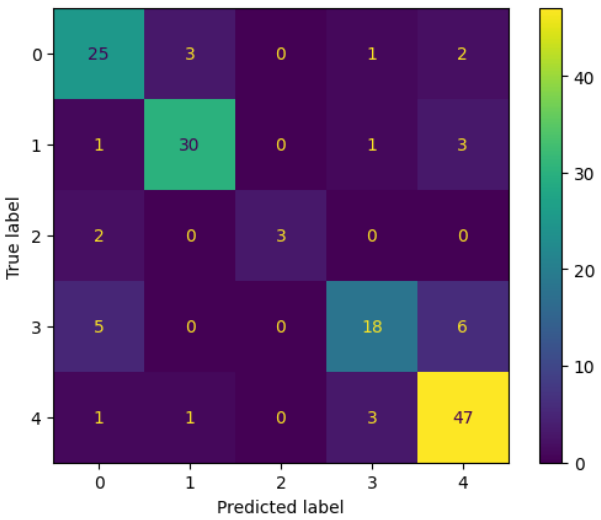

Figure 1. Confusion matrix

overall higher accuracy values than those observed in the training phase.

On the basis of the single and averaged values of precision and recall—slightly lower than accuracy—the model can also be said to be reasonably robust to class imbalance, the only exception being the class *Formula Tales* (largely underrepresented), for which 3 out of 5 tales are correctly classified.

4.2. Non-contextual word embeddings

The non-contextual document embeddings were generated by the `tok2vec` component of the `en_core_web_lg` pipeline: the vector is accessible through the `vector` field of the `doc` object representing the processed document, and has a dimension of 300.

In this second part of the classification study, each document in the training set and later in the validation set was hence represented as a dense numerical vector of 300 features. *A priori*, all traditional machine learning models are suitable for such data, so we have limited ourselves to those we consider to be the most widely used. Since some of them are based on the calculation of distances between samples, we standardized the features to have zero mean and unit variance.

4.2.1. Model training and hyperparameter tuning

As before, different hyperparameters of the classifiers were optimized via exhaustive search on predefined grids, similar to those seen in subsection 4.1.1.

In the following, we only report the hyperparameters and results of the estimated classifiers:

	Linear SVM	*k*-NN	Gaussian SVM	RF
`C`	0.007	-	19.307	-
`n_neighbors`	-	5	-	-
`gamma`	-	-	0.001	-
`min_samples_leaf`	-	-	-	1
`max_features`	-	-	-	None
`max_leaf_nodes`	-	-	-	1000
Mean accuracy	0.672	0.502	0.669	0.575
Mean weighted F_1	0.669	0.464	0.666	0.546

Table 3. Best hyperparameters

The linear SVM was chosen for its robustness in high-dimensional spaces and its ability to handle imbalanced classes through the `class_weight="balanced"` parameter. Its results suggest that the model can effectively leverage the dense embedding vectors, but there is room for improvement, particularly in balancing precision and recall across classes.

The *k*-NN algorithm was tested as a non-parametric baseline due to its simplicity and interpretability. The grid search evaluated different numbers of neighbors and weight functions, however, the model struggled to generalize. The poor performance indicates that *k*-NN may not be well-suited for embeddings with hundreds of features, likely due to the curse of dimensionality and the lack of clear boundaries among classes in the vector space.

The SVM with Gaussian kernel and its ability to learn non-linear decision boundaries slightly underperformed its linear counterpart, though its ability to model non-linear relationships makes it a promising candidate for future experiments with additional tuning.

Lastly, the random forest was evaluated due to its flexibility and robustness to overfitting. The lower performance suggests that the model struggles with the high dimensionality of the embedding vectors.

4.3. Conclusion

The effectiveness of the classification of the folktales based on non-contextual embeddings—apart from being rather variable between classifiers, with accuracy values of 0.50–0.67—is, above all, considerably lower than the effectiveness of the classification based on *tf-idf*, which achieves an accuracy of 0.81.

This result seemed remarkable to us, as we assumed that the modelling of refined semantic relations between the narrative lexicons of the tales would contribute much more effectively to the identification of their *genre* than the mere statistical characterization based on terms. Given the narrative characteristics of folktales and the meaning of their classification (essentially a classification by subject), there was reason to believe that embeddings, by capturing the semantic similarity between, *e.g.*, *cat*, *donkey*, *dog*, *rooster* and *fox*, would favour the common identification of *The Town Musicians of Bremen* and the many Aesop's fables as *Tales Of Animals*.

Now, although this is true in principle, we believe that the contribution of the high similarity thus detected, more or less occasionally, between the vectors of individual terms of two tales is almost completely reduced by averaging these vectors to obtain the one representative of the whole document. This natural mitigating effect would be all the greater the greater the number of terms—and our texts can be reasonably considered long.

Furthermore, it is reasonable to believe that even if the concept of *animal* could not be grasped—thus linking *cat*, *donkey*, *dog*, *etc.*—terms belonging to the same semantic sphere (*animal*, *beast*, *paw*, *fur*, *etc.*) would be present and repeated in both texts; these correspondences would also be identifiable via *tf-idf* bag-of-words.

On the other hand, since word embeddings do not provide any information about the weight or importance of the terms within the document, it may happen that a magic tale like *Puss in Boots*—which contains terms such as *cat* only barely and secondarily—is classified as *Tale Of Animals*. Thus we try to explain the much more satisfactory result of our *tf-idf* model.

Future work could explore alternative embedding techniques, namely contextual embeddings like BERT.

5. Text summarization

Text summarization aims to condense information while retaining the essential content and meaning. To address this task, we employed a two-step summarization approach.

First, we generated abstractive summaries using a transformer-based model, which serve as a reference or 'ground truth' for evaluating subsequent extractive summaries. Once the generative summaries were created, we set the number of sentences in each extractive summary to be the same as in the corresponding abstractive summary so that, if the abstractive summary of, *e.g.*, *Cinderella* contained ten sentences, the extractive summary would be constrained to have the same number of sentences. This choice primarily provides a logical basis for the desired comparison, and secondarily fixes an otherwise

free hyperparameter for the TextRank summarization algorithm. We used the ROUGE metric for evaluation, measuring the overlap between generated and reference summaries.

5.1. Data selection

Due to the high computational time required to generate abstractive summaries, we decided to focus primarily on a smaller selection of texts.

Specifically, we opted to summarize historically relevant authors and works of Italian folktale literature by selecting 22 folktales by Giovanni Boccaccio, Giovan Francesco Straparola or Giambattista Basile from their respective masterpieces: the *Decameron*, *Le piacevoli notti* ('the facetious nights') and *Lo cunto de li cunti* ('the tale of tales').

5.2. Abstractive summarization

The abstractive summarization process was implemented using the pre-trained transformer-based model BART [8] from Hugging Face, a model that has been specifically trained for summarization tasks. This model generates summaries by interpreting the input text and creating entirely new sentences that capture the essence of the original content, rather than directly extracting sentences.

The summarization pipeline followed these steps:

1. Tokenization and text chunking: the input texts were tokenized using the BART tokenizer. For texts exceeding the limit of 512 tokens, the input was split into smaller chunks to be summarized independently: this partition was performed by reducing the original texts in chunks consisting of an equal number of processed sentences (these are already available as a result of spaCy's pipeline)[2].
2. Summarization: the model was run separately on each text (or segment of long texts); the summaries of the chunks of long texts were subsequently concatenated into the final resulting one. Chunking longer texts instead of abruptly truncating the exeeding tokens allows to maintain possibly relevant content, at the cost of losing semantic dependencies among the chunks.
3. Parameters: the summarization model is run via the `pipeline` function with the following parameters:
    - `max_length`: the maximum number of tokens in the output summary; set to 128;
    - `min_length`: the minimum number of tokens in the output summary, constraining always a minimum amount of text; set to 64;
    - `num_beams`: number of beams for beam search to enhance the quality of the summaries while increasing computation time; set to 4 as suggested in most use cases.

This approach ensured that even lengthy texts could be summarized effectively while maintaining coherence and minimizing truncation of important content. The generated abstractive summaries were then used as reference points for evaluating extractive summaries.

5.3. Extractive summary

Extractive summarization aims to generate summaries by selecting and rearranging sentences directly from the source text, preserving the original wording. Unlike abstractive summarization, this approach does not generate new phrases or sentences, identifying instead the most relevant portions of the text based on specific criteria.

For our study, we utilized the TextRank algorithm of the previously added PyTextRank component (see subsection 3.2), which scores sentences based on their relevance to the overall text using graph-based ranking techniques. The preprocessing of texts is automatically handled by the spaCy pipeline. For each of the 22 selected folktales, an extractive summary was generated as follows.

[2]For texts where chunking left a small remainder of sentences (less than 25% of the sentences of a full chunk), this remainder was truncated.

The desired number n of sentences (`limit_sentences` parameter) was set to match that of the corresponding abstractive summary previously generated; hence, the n sentences were then selected based on their computed rank; finally, these sentences were concatenated and sorted according to their order of appearance in the original text.

The ranking algorithm, implemented in the `summary` function, also accepts a `limit_phrases` parameter, which works similarly to `limit_sentences`. To evaluate its impact on the summary length and quality, we fine-tuned this parameter on a logarithmic grid spanning 32 values between 1 and 1000: the best parameter was set to be the one maximizing the average of the ROUGE-1, ROUGE-2 and ROUGE-L metrics (for which see the following subsection 5.4) with respect to the abstractive summary.

By following these steps, we were able to generate extractive summaries that maintain the structure and key information of the original text while ensuring conciseness and clarity.

5.4. Evaluation metrics

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) metrics [9] are elementary yet widely used methods to evaluate the goodness of an extractive summary with respect to a human-written counterpart (in our case, the abstractive summary):

Specifically, we computed the ROUGE-1, ROUGE-2 and ROUGE-L scores:

- ROUGE-1 measures the overlap of unigrams (individual terms) between the summaries
- ROUGE-2 measures the overlap of bigrams (pairs of consecutive terms) between the summaries
- ROUGE-L assesses the longest common subsequence (LCS) between the summaries, which helps to evaluate the fluency and sentence structure of the summaries.

For the 22 selected folktales we obtain the following:

	ROUGE-1	ROUGE-2	ROUGE-L
1	0.671	0.500	0.557
2	0.726	0.610	0.530
3	0.667	0.409	0.414
4	0.548	0.348	0.356
5	0.598	0.386	0.400
6	0.385	0.197	0.250
7	0.569	0.346	0.338
8	0.531	0.349	0.361
9	0.556	0.362	0.394
10	0.481	0.232	0.237
11	0.540	0.382	0.446
12	0.551	0.344	0.389
13	0.575	0.421	0.430
14	0.589	0.334	0.372
15	0.583	0.340	0.382
16	0.496	0.360	0.416
17	0.594	0.423	0.491
18	0.597	0.350	0.349
19	0.576	0.338	0.405
20	0.494	0.247	0.285
21	0.522	0.262	0.288
22	0.556	0.374	0.398
Average	0.564	0.360	0.390

Table 4. ROUGE scores for each folktale (row)

5.5. Conclusion

Overall, the average overlap of unigrams (single tokens) is 0.564; this measure suggests that, *a priori*, about half of the vocabulary used in the extractive summaries coincides with that of the abstract summary, and is an assessment of the similarity of the two.

The effectiveness of the extractive summary to synthetize folktales should of course be assessed individually by human judgement, and for this purpose we provide the two summaries of the second folktale, the one with the highest ROUGE scores, in the appendix [6].

Finally, we would like to make a few observations that we feel are relevant to the evaluation of these summaries: the texts we have chosen date from the 14th (Boccaccio), 16th (Straparola) and, at most, 17th (Basile) centuries. Even in translation, these stories are characterized by long and complicated syntax and a predilection for subordination. The average length of a sentence (between two full stops) is thus greater in extractive summaries—where the text appears *verbatim*—than in abstract ones—where the phrasing is obviously influenced by modern syntax.

This implies that an extractive summary, while containing by construction the same number of sentences as an abstractive one, may have a significantly higher number of words, altering both the conciseness of the former and the discrepancy with the latter (and thus the ROUGE values).

6. Contact us

You can contact us at the following adresses:
✉ a.brunazzi@campus.unimib.it
✉ a.dellabeffa@campus.unimib.it
✉ d.lepre@campus.unimib.it

Appendix

The abstractive summarization of the tale with index 2—*Lidia e Pirro* (Giovanni Boccaccio, *Decameron*, VII, 9) is:

> Nicostratus, a wealthy patrician, married Lydia, a woman of great distinction and unsurpassed beauty. Lydia found herself paying ever more attention to one of her husband's servants, Pyrrhus. He was attracted to her as well, and gladly would have accepted her invitations to love, but the old man gave them no opportunity.
>
> Pyrrhus and Lydia were able to pluck the fruits of their love at regular intervals, even without the help of their enchanted pear tree. From that time forth Nicostratus relaxed his watchful vigil over his young wife, and thus Pyrrhus, Lydia and their children lived happily ever after.

The extractive summary of the same tale is:

> Nicostratus, a wealthy patrician, married Lydia, a woman of great distinction and unsurpassed beauty. Thus, it is quite understandable that Lydia found herself paying ever more attention to one of her husband's servants, Pyrrhus by name, who was elegant, handsome, young, and energetic. Looking down, what did he behold but Pyrrhus and Lydia making fervent love. Nicostratus climbed down from the tree, only to find Pyrrhus and Lydia seated discretely on a garden bench. From that time forth Nicostratus relaxed his watchful vigil over his young wife, and thus Pyrrhus and Lydia were able to pluck the fruits of their love at regular intervals, even without the help of their enchanted pear tree.

References

[1] j-hagedorn, *Trilogy*, 2020. [Online]. Available: https://github.com/j-hagedorn/trilogy.

[2] WVS, *Folklore and mythology, electronic texts*, 2025. [Online]. Available: https://sites.pitt.edu/~dash/folktexts.html.

[3] Wikipedia, *Motif-index of folk literature*, 2025. [Online]. Available: https://en.wikipedia.org/wiki/Motif-Index_of_Folk-Literature.

[4] Wikipedia, *Stith thompson*, 2025. [Online]. Available: https://it.wikipedia.org/wiki/Stith_Thompson.

[5] Wikipedia, *Aarne–thompson–uther index*, 2025. [Online]. Available: https://en.wikipedia.org/wiki/Aarne%E2%80%93Thompson%E2%80%93Uther_Index.

[6] J. P. Jiawei Han Micheline Kamber, *Data mining - concepts and techniques*, 2011. [Online]. Available: https://archive.org/details/the-morgan-kaufmann-series-in-data-management-systems-jiawei-han-micheline-kambe.

[7] Spacy, *Spacy-pytextran*, 2025. [Online]. Available: https://spacy.io/universe/project/spacy-pytextrank.

[8] Huggingface, *Facebook-bart large cnn*, 2024. [Online]. Available: https://huggingface.co/facebook/bart-large-cnn.

[9] freecodecamp, *An intro to rouge, and how to use it to evaluate summaries*, 2025. [Online]. Available: https://www.freecodecamp.org/news/what-is-rouge-and-how-it-works-for-evaluation-of-summaries-e059fb8ac840/.