Newsjam: Automatically Summarizing French News about COVID-19

Shane Kaszefski-Yaschuk **Joseph Keenan** **Adelia Khasanova**
Maxime Méloux **Mahamadi Nikiema**

Institut des Sciences du Digital
Pôle Herbert Simon
13, rue Michel Ney
54000 Nancy

Abstract

We introduce *Newsjam*, an automatic summarization tool for French COVID-19 news articles. To this purpose, we implement two extractive summarization methods: Latent Semantic Indexing and K-means clustering on contextual word embeddings. We then evaluate our tool using three evaluation metrics and two different corpora, an existing one as well as a new one that we have built ourselves. Finally, after testing all of our methods, we choose the most optimal to be implemented in our complete pipeline, going from text scraping and classification to summarization, and ultimately posting the summaries to Twitter along with the original articles via a bot.

1 Introduction

The ongoing COVID-19 pandemic has caused people from around the world to feel fatigued from the overload of pandemic related news articles being released every day (Savage, 2020). One study collecting data from 11 different countries found that more than 26 million coronavirus related articles have been posted since the beginning of the pandemic (Krawczyk et al., 2021).

These observations have prompted us to create *Newsjam*, a COVID-19 news summarization tool aimed at reducing news fatigue by keeping only the main points of pandemic-related information and aggregating them in a single place, therefore reducing the need to rummage through the plethora of available information.

Our project consists of four interconnected parts:

- A web scraper to locate and scrape relevant COVID-19 articles.
- An article classification model to separate the COVID-19 articles into two categories: French news and international news.
- A backend architecture that automatically summarizes these articles.
- A bot that automates the previous three parts by regularly fetching new articles, feeding them to the classifier and summarizer, and then posting the output summary as a single tweet on Twitter.

The last part implies that our generated summaries must abide by the maximum tweet length of 280 characters.

The rest of the report is organized as follows: Section 2 is a literature review of summarization and text classification methods. Section 3 outlines the methods we chose to collect, classify, and summarize the data. Section 4 describes how we tested and evaluated the various methods we implemented. Sections 5 and 6 discuss and analyze our evaluation results. Section 7 details the implementation of our full pipeline, using the most successful methods tested and adding a Twitter posting bot. Finally, the paper concludes in Section 8.

2 Literature Review

2.1 Text Summarization

Automatic text summarization refers to generating a condensed version of a text document through automated means while preserving key information. There are two primary text summarization methods, extractive and abstractive (Saggion and Poibeau, 2013). Extractive summarization centers around identifying key sentences in the text and putting them together verbatim, whereas abstractive summarization involves generating novel text. Extractive approaches include assigning importance scores to sentences using topic modeling, k-means clustering, and latent semantic indexing (Allahyari et al., 2017). Primary approaches seen in abstractive text summarization include the use of deep learning, RNN encoder-decoders, Gated Recurrent Units, and Long Short-Term Memory (Suleiman and Awajan, 2020).

Text summarization comes with a few key

challenges. During the testing stage, reference summaries are needed for evaluation. However, datasets often contain poor reference summaries or do not contain them at all, making evaluation unreliable or impossible (Suleiman and Awajan, 2020). Other challenges include the occurrence of out-of-vocabulary (OOV) words that are absent from the training dataset but are central to understanding a document, and finding metrics able to evaluate a summarized or paraphrased fragment on both a syntactic and semantic level (Soto, 2021).

Given the multiple challenges encountered in text summarization, the question arises as to how to validate the results. The most commonly used metric is ROUGE (Recall-Oriented Understudy for Gisting Evaluation), which compares a generated summary to a reference one (typically created by a human) and calculates the number of overlapping units (Lin, 2004a). ROUGE is however far from ideal: Dorr et al. (2005) found that the metric was sensitive to the type of summarization. ROUGE scores also tend to be higher for summaries that are longer or generated by using supervised learning approaches (Shulter, 2017).

2.2 Text Classification

The goal of text classification is to classify documents into different categories. There are many different methods for performing this task, such as Decision Trees, Logistic Regression (LR), Naive Bayes (NB) and Support Vector Machine (SVM) (Dalal and Zaveri, 2011). NB is one of the oldest methods of text classification, which is based on Bayes' Theorem and determines the probability that each document belongs to a given class (Kowsari et al., 2019). It can use several distributions, such as the normal (Gaussian), Bernoulli and multinomial distributions. LR is a simple but effective binary classifier for text classification, which calculates the probability of a document belonging to two different classes. SVM is a supervised learning model that was also originally built as a binary classifier, but was later extended to support multiple classes (Kowsari et al., 2019). Lastly, Decision Trees calculate the probability of different 'children' belonging to the 'parent' of a tree (Magerman, 1995). All of these methods are inherently probabilistic, so we wish to test several of them to see which one gives the most accurate results for this project.

3 Methodology

3.1 Scraping

Several French news websites have been selected for scraping. One custom scraper is required for each website due to their different underlying structure. Our project currently contains two working Python scrapers for two sites: one for *Actu*[1], using the *selenium* library, and the other for *L'Est Républicain*[2] using the *beautifulsoup* library.

For each website, we retrieve articles tagged with the labels "COVID-19", "Coronavirus", or "Santé" (French for 'health'). Summaries, which can usually be found in a highlight section that is presented before the main text, are also extracted from each article.

In both cases, the output of these scrapers is a JSON file that contains the same fields as the ML-SUM corpus (Scialom et al., 2020). These fields have been chosen for data consistency as MLSUM is also part of our experimental setup (see 4). The scrapers work on an iterative basis and only fetch new articles that have not yet been retrieved.

Before publicizing the summaries of our scraped articles on the planned Twitter bot, the legal aspect needs to be taken into account. In a practical setting, this would need to be done on a case-by-case basis to ensure full compliance with the wishes of the owners of the intellectual property contained on these news sites. Currently, however, *Newsjam* is a proof of concept that is intended to be run a few times for demonstration purposes and is not meant to be used in a commercial setting.

3.2 Annotation

Part of our pipeline consists of implementing a binary news classifier, with the aim to create two distinct Twitter bots so that readers could choose whether they want to familiarize themselves with updates on France or the rest of the world. As of now, only the bot for French articles has been implemented.

To this purpose, we first came up with agreed-on annotation guidelines (see Appendix B) for tagging articles as local (France) or global. The latter category includes not only news about other countries but also global events, such as decisions made by the World Health Organization.

[1]`https://actu.fr/societe/coronavirus`
[2]`https://estrepublicain.fr/sante/coronavirus`

One annotator (Annotator A) then annotated the entire *L'Est Républicain* section of our built corpus. Due to a lack of time, we weren't able to tag the *Actu* sub-corpus. However, two other annotators (Annotators B and C) also annotated 50 articles of the *L'Est Républicain* sub-corpus in order to be able to compute inter-annotator agreement.

The Ao, S, Scott's π and Cohen's κ inter-annotator agreement coefficients were computed for each pair of annotators. The results are summarized in table 1.

All three pairs of annotators have similar inter-annotator agreement scores for all three computed coefficients, typically around 0.5. Ideally, if all three annotators had annotated the whole corpus, one could have then looked into each annotator's tags to determine which annotation set to keep for training our model. In the absence of full dataset tagging and due to the fact that all of the scores were similar, we decided to keep Annotator A's annotations as they span the most articles.

The resulting dataset, to be used for classification training and testing, is rather imbalanced (65% of articles are local and 35% are global).

Coefficient	A & B	A & C	B & C
A_o	0.660	0.660	0.840
IAA_S	0.500	0.500	0.500
IAA_π	0.505	0.534	0.520
IAA_κ	0.505	0.519	0.513

Table 1: Inter-annotator agreement for each pair of annotators

3.3 Article Classification

Our article classification is entirely done in Python.

The pre-processing of the corpus for classification begins with noise removal (punctuation and irrelevant special characters), stopwords removal, and lemmatization. We then implement three different methods for classification: Multinomial Naive Bayes (MNB), Logistic Regression (LR), and Support Vector Machine (SVM). Theoretically, the MNB classifier is based on Bayes' theorem and assumes strong (naive) independence of features (Kowsari et al., 2019). The LR classifier uses a logistic function to model and predict the dependent variable that can take the value of 0 or 1. Lastly, the idea behind the SVM classifier is to represent data entries as points in a high dimensional space and find a hyperplane with the largest margin to separate as many points as possible.

For model-specific issues, one way to improve the performance is hyperparameter tuning, which can be done with the *GridSearch CV* function. MNB may benefit from tuning the α parameter, that represents Laplace smoothing and helps tackle the issue of zero probabilities. For our LR model, we tasked GridSearch CV with finding the best *C* parameter. It appeared to be a lower value ($C = 0.1$). Lower values tend to be more fit for imbalanced datasets as they require more regularization and more weight to the complexity penalty to avoid overfitting. Regarding the SVM model, we tuned the *C* parameter as well to find a balance between the minimum margin and accounting for outliers in the data.

3.4 Summarization

Our project currently implements two extractive approaches for summarization, Latent Semantic Indexing and K-means clustering on contextual word embeddings, implemented in Python. While these methods are not state-of-the-art, they provide a solid starting point for us to get a better idea of how text summarization methods work. Furthermore, we believe the use of these methods is beneficial due to the possibility of being able to see what is happening in every step of the process, unlike the black boxes of neural networks. It should also be noted that these methods exhibit how one can still obtain admirable text summarization results while using methods that are not as advanced as the current state-of-the-art.

3.4.1 Latent Semantic Indexing

Latent Semantic Indexing (LSI) is a technique initially introduced for automatic document classification (Borko and Bernick, 1963) and information retrieval (Deerwester et al., 1990), but it was later found to be efficient for automatic text summarization (Gong and Liu, 2001; Ozsoy et al., 2011) and its evaluation (Steinberger et al., 2004; Steinberger and Jezek, 2009). We choose to implement LSI due to its popularity, historical significance and relative simplicity.

Our algorithm replicates that of Gong and Liu (2001), including for sentence selection. Our initial implementation was based on Chakravarthy (2020) as a starting point, but has since been rewritten to be easier to maintain and extend. For each article, we choose the optimal number of topics n_{topics} by measuring C_v topic coherence (Röder et al., 2015) implemented in *gensim*'s topic coherence pipeline. We pick:

$$n_{topics} = \underset{k \in [\![2,10]\!]}{\arg\max}\, C_v(k)$$

. Further analysis would be required to determine the minimum and maximum topics an article can have, but 2 and 10 were picked as bounds based on our intuition that there should only be a low amount of significant topics in a given news article. Our implementation uses the Python modules *spacy* and *gensim*, respectively for tokenization/segmentation and LSI/topic coherence modeling.

Sentences are then categorized by topic, and the output summary is generated by looping through the topics and choosing the top score from each one, until we reach the aforementioned maximum summary length of 280 characters. Highly-ranked sentences are skipped in favor of lower-ranked ones if the former would make the summary go over the character limit and the latter would not.

We decide to apply LSI to the TF-IDF (term frequency-inverse document frequency) of a list of keywords generated by removing punctuation and stopwords from the article and stemming the remaining words. Our intent in applying LSI to keywords rather than the raw text was to eliminate noise that could be caused by stopwords and to apply topic modeling to the most semantically loaded words in a document. While theoretically, TF-IDF is supposed to take into account the amount of times a word is repeated over many documents, in practice, we only apply TF-IDF to one document at a time, and we felt it was more effective to filter out the stopwords ourselves.

3.4.2 Word embeddings and k-means clustering

K-means clustering is an alternative way to model topics within a document, which has successfully been applied to text summarization before (Shetty and Kallimani, 2017).

Our implementation uses contextual word embeddings as the raw input of k-means clustering (Gupta, 2020; ialifinaritra, 2021), which are generated on a sentence basis using the pre-trained models FlauBERT (Le et al., 2019) and CamemBERT (Martin et al., 2020). We arbitrarily choose $n_{clusters} = 5$ for k-means clustering, but further research is necessary to determine an optimal value.

We then use the same algorithms as in 3.4.1 for selecting sentences and building the final summary, with the modification that scores are generated for individual words (and not sentences) by computing the cosine similarity between a word's embedding and topic centroids. For each cluster, the top words are picked and mapped back to the original sentence that contains them.

The choice of K-means clustering is motivated by the simplicity and ease of implementation of the method, while embeddings are chosen with the intuition that they capture additional semantic information. Specifically, we use contextual embeddings with the hope that they would perform better than a bag-of-word model such as TF-IDF.

Our implementation uses the Python modules *transformers* for BERT-like models and *sklearn* for K-means clustering.

4 Experimental setup

4.1 Datasets

After careful consideration, we decided to create our own corpus. The goal of our project is to provide people with the most up to date information about the coronavirus pandemic. In order to do this successfully, we need to constantly scrape websites to get new information. If we did not make our own web scrapers along with our own, dynamic corpus, this would not be possible. Furthermore, when discussing the coronavirus pandemic, many words such as *pandémie*, *rappel*, and *autotest* occur much more frequently than in regular French articles. Since our project is tailored towards this pandemic specifically, we expect a tailor-made corpus to provide better accuracy for summarizing novel articles compared to a more general news-based corpus.

Our corpus, created via scraping news websites, currently contains 895 articles extracted from *Actu* and 1,703 articles from *L'Est Républicain* along with their summaries.

In addition, we use the French version of the MultiLingual SUMmarization corpus (MLSUM) (Scialom et al., 2020), which is made up of over 400,000 news articles from *Le Monde*, split into train, test, and validation sets. The title and content of each article are stored along with a corresponding summary for evaluation, which usually comes from a highlights section found at the beginning of each article. Our first models are evaluated on the test set of this corpus, which contains 15,828 articles.

We choose this dataset to evaluate our program due to its semantic proximity with ours (recent news articles, in French, extracted from online newspapers) as well as the large amount of data it contains, especially at a stage where our own corpus was not yet created.

4.2 Article Classification Evaluation

In a machine learning process, it is often insufficient to split the dataset into training and testing subsets and assume that the model will always perform well on unseen data. For this reason, we used the *cross_val_score* method from the *sklearn* library to further evaluate our classifier. This method divides data into a user-defined number k of sets, out of which $k-1$ are used for training and the remaining one is used for testing. The choice of the testing set changes with each execution. We ran a five-fold validation on all models and collected the values of all four metrics: accuracy, precision, recall, and F1-Score.

Since MNB classifiers are known to struggle with correctly predicting classes when trained with imbalanced datasets (Kowsari et al., 2019), we resampled the data to balance the classes in an exact 1:1 proportion. Namely, we kept the data points from the smaller class as is, and randomly drew an equal number of data points from the "majority class." Therefore, our dataset became smaller but perfectly balanced. The LR and SVM models were evaluated before and after the tuning.

4.3 Summarization Evaluation

To evaluate our models, we use ROUGE (Lin, 2004b) as well as the much more recent, state-of-the-art metric BERTScore (Zhang et al., 2020). Both metrics calculate precision, recall, and F1-score but via two different means. The three scores are proportions ranging from zero to one, where a higher score indicates higher performance. For our evaluation, we are focusing on the F1-score, which can be calculated on two different forms of our data:

- The *standard score* is computed on pairs consisting of the generated summary and the reference summary.
- The *keyword score* is computed on pairs consisting of stemmed keyword-only versions of the generated and reference summaries. Keyword extraction is described in 3.4.1.

To our knowledge, it is not common practice for text summarization models to evaluate keyword versions of summaries. Our goal in adding this evaluation is to reduce the risk of score inflation due to stopword similarity. We also expect stemming to increase the opportunity for matching word subsequences between our generated summaries and the references, which is the same motivation for the METEOR evaluation metric (Lavie and Agarwal, 2007) to include a Porter stemmer module.

4.3.1 ROUGE-L

The ROUGE-L score (Lin, 2004b) is a specific version of ROUGE denoted by the hyphenated 'L', which stands for "Longest Common Subsequence" (LCS). As implied by it's name, this version of ROUGE tries to find the longest common sequence of words between a generated summary and a reference one. Scores are then computed based on the LCS length in a pair of summaries. In other words, if two summaries have a long LCS in common, they will receive high scores and vice versa. The choice of ROUGE-L was motivated by its prevalence, which allows us to compare the performance of our models with the existing literature.

Our implementation uses the Python package *rouge_score*.

4.3.2 BERTScore

BERTScore (Zhang et al., 2020) is a metric originally created for text generation and translation, but due to it being based on comparison of a generated and a reference text, it can also be applied to summarization. We use it (as implemented in the Python module *bert_score*) as our second evaluation metric. This choice is motivated by the fact that it uses BERT contextual embeddings (Devlin et al., 2019) to compare the deeper, semantic content of the summaries rather than their surface-level similarity using n-grams, like in many versions of ROUGE (Lin, 2004b) and BLEU (Papineni et al., 2002). BERTScore computes scores via cosine similarity of the contextual embeddings.

BERTScore is only able to compute a score when the reference and generated summaries have the same number of sentences. When it is not the case, scores are automatically set to 0. For our built corpus, this happens for over 65% of generated summaries, which brings the average score down significantly. In order to offer a fairer comparison metric, we also compute the adjusted BERTScore (Adj-BERTScore), which is the average of the BERTScore scores of articles in a dataset, but only computed over the non-zero values.

4.3.3 Execution time

Due to limited computing resources, time, and the relatively large size of the MLSUM corpus, we find it important to measure the average execution time of our summarization methods. This also allows us to track speed improvements made over time by optimizing various aspects of our program.

For each dataset and summarization method, we summarize the first 100 articles of the test split of the dataset and measure the average elapsed time per article. This process is repeated 10 times to mitigate the statistical variance of time measurements, and we report the average value of this execution time. A lower value reflects a more useful summarization method for our purposes. All time measurements were applied on the same machine, and GPU processing was enabled whenever possible.

5 Results

5.1 Article Classification Results

Our article classification results are in Table 2. All measures are computed relative to 1, the local label.

At first, the MNB classifier generated a considerable number of false negatives and had less than ideal accuracy at 72.1%. After tuning the Laplace smoothing, α, parameter the MNB's performance improved across the board. While re-sampling boosted the precision of the model from 70.8% to 88.3%, other metrics plummeted below acceptable levels. Overall, MNB and LR each provide maximum values depending on the metric. When it comes to accuracy, the tuned LR and SVM models show the best performance with a negligible difference, 85.7% and 85.2% accordingly. The precision metric is lead by the tuned MNB with 92.2%. The highest recall is recorded by our initial MNB model at 97.7%. Finally, the top F1-Score goes to the tuned LR with 89.3%.

5.2 Summarization Results

Our summarization results are shown in Table 3.

We generally observe similar scores on the *Actu* corpus and the *MLSUM* corpus. We also find that the LSI model slightly outperforms the k-means clustering implementations with respect to the ROUGE-L scores. This is also true for the BERTScore, but less marked.

The runtimes of each model are shown in Table 4. In addition to getting better evaluation scores, the LSI model also runs much faster than either k-means model.

For these reasons, the LSI model is the default one chosen when implementing our full pipeline, which is discussed in Section 7.

6 Discussion

Since there is no one method that performs high and above all the others for article classification, we must focus on one metric that we deem to be the most important in order to make our choice. Thus we will focus on the *F1-Score* metric. We justify this choice with our goal to maximize the amount of true positives while minimizing the amount of false positives and false negatives, which translates to minimizing the misidentification of local and global articles in our model. Given this choice, we believe that the tuned LR method will work best for our article classification.

For our summarization evaluation, as mentioned in Section 4.3.2, many of the scores for BERTScore were invalid due to the reference and generated summaries not having the same amount of sentences. Thus, even though the adjusted BERTScore only represents 35% of our data, these scores better reflect the quality and consistency of summaries generated by our model. Considering there is such a vast difference between the adjusted BERTScore and the ROUGE-L scores, the question must be asked as to whether or not there was an issue with

Method	**Accuracy**	**Precision**	**Recall**	**F1**
Multinomial Naive Bayes (MNB)	72.1	70.8	**97.7**	82.1
MNB (resampled)	67.1	88.3	38.6	53.1
MNB (tuned)	83.1	**92.2**	81.1	86.2
Logistic Regression (LR)	83.8	88	87.3	87.7
LR (tuned)	**85.7**	88.3	90.5	**89.3**
Support Vector Machine (SVM)	82.8	83.3	91.3	87.1
SVM (tuned)	85.2	87.5	90.6	89.1

Table 2: Article classification evaluation

Method	ROUGE-L	Keyword ROUGE-L	BERTScore	Adj-BERTScore
MLSUM corpus, testset (15,828 articles)				
LSI	0.1507	0.1147	—	—
FlauBERT + k-means	—	—	—	—
CamemBERT + k-means	—	—	—	—
Built corpus (895 + 1,703 = 2,598 articles)				
LSI	**0.1589**	**0.1566**	**0.2636**	**0.7993**
FlauBERT + k-means	0.0879	0.0821	0.2374	0.7198
CamemBERT + k-means	0.0902	0.0850	0.2385	0.7231

Table 3: Summarization evaluation (only F1-Scores reported)

MLSUM corpus	**Exec. time**
LSI	**4.79s**
FlauBERT + k-means	14.77s
CamemBERT + k-means	18.98s

Built corpus	**Exec. time**
LSI	**4.57s**
FlauBERT + k-means	15.53s
CamemBERT + k-means	19.83s

Table 4: Summarization execution time

the calculation of the ROUGE scores. After investigating ourselves, we do not know of any overarching issues in the ROUGE score calculations but we have found some interesting examples when investigating a few generated summaries along with their scores.

While looking deeper into the *Actu* LSI examples of the data, we found that the summaries chosen by our model outlined the articles well and matched a quality reference summary. In the *MLSUM* LSI examples, we noticed a recurring issue where we felt that our summary matched the article, but received a low ROUGE-L score due to a poor reference summary. This seems inevitable when working with a corpus of this magnitude, but it is important to note because it exhibits a way in which the scores may not always reflect the quality of a generated summary. In this matter, a quick look at a few examples seems to show that BERTScore better represents the quality of our models.

To further look into the question of how relevant computed scores are compared to human evaluation, one would need one or several native French speakers to manually annotate each generated summary as good or bad, and to compare how those scores relate to ROUGE-L and BERTScore. This could be done as an extension to our project.

Another issue that we encountered is the relatively slow speed of our model. Despite manually optimizing our programs as much as possible through the use of parallel GPU computations and running our summarizers on fast dedicated systems, evaluating a single summarization method on our relatively small created corpus took up to 24 hours. This made it difficult to adjust different parameters and fine-tune our models, which might have allowed us to get better overall scores.

Finally, one issue that has affected both datasets is poor summary selection, which may be due to the performance of the summarization methods themselves. One example below shows an instance of a quality summary given a poor score. Further examples can be found in Appendix A.

Quality Summary | Poor Score:

- MLSUM Test Dataset Article 54[3]
- *Reference Summary*: "Le suspect principal, un employé des services de la ville, a tiré « à l'aveugle ». Il est lui aussi décédé."
- *Generated Summary*: "Douze personnes ont été abattues vendredi 31 mai par un tireur dans un bâtiment municipal de Virginia Beach (Etat de Virginie), station balnéaire de la côte est américaine."
  - ROUGE-L
    - Standard F1-Score: 0.151
    - Keyword F1-Score: 0.066
  - BERTScore
    - Standard F1-Score: 0.157
    - Keyword F1-Score: 0.119
- This example is important, because it shows that our model is able to select high quality

[3]https://bit.ly/3n9qRuK

summaries, but that they will not always be evaluated as such, because some of the reference summaries in the MLSUM corpus are not up to the expected gold standard.

As for the classification part of our task, we have confined ourselves to traditional, supervised methods, such as Multinomial Naive Bayes, Support Vector Machine, and Logistic Regression. We are well aware that the state-of-art approaches now revolve mainly around deep learning (DL). It has also been established in scientific reviews that unsupervised models do demonstrate superior performance. XLNet-Large and XLNet showed consistently good classification results on multiple 'classic' datasets, such as IMDB and Yelp reviews (Minaee et al., 2021). For instance, for the SST-2 dataset, the best result yielded by a traditional model (Naive Bayes) was 81.8, while the best result yielded by a DL model (XLNet) was 97.0.

Even with the success of DL models today, there are a few key reasons we chose to use simpler statistical models over DL for all of the models in this project. DL models come with a handful of unique challenges that may not be the case for traditional supervised models. For instance, most of these models cannot easily be interpreted (Minaee et al., 2021). Poor interpretability makes it difficult to pinpoint exactly why one DL model outperforms another one (Minaee et al., 2021). Furthermore, we cannot always understand what a model has learned to achieve a desirable benchmark to use this insight further on. Another issue is that our datasets may not be big enough to train a DL model.

When choosing a model, we aimed at understanding the motivation first. Since for now, we planned to create an MVP (minimum viable product), we went with more lightweight methods. Taking all this into account, we still see a benefit in moving to DL in the future to boost the robustness of each component of our pipeline.

7 Pipeline Implementation

For the Twitter bot implementation, as stated above, we chose to use LSI as our summarizer, and the tuned logistic regression model as our classifier. With these models, we were able to create an automated implementation of our project. Our pipeline automatically grabs news articles and classifies them based on whether they are French-specific or global. Once an article classified as 'French' is found, our model summarizes it and posts the summary to Twitter. Once a summarized news article is posted, the Twitter bot then replies to the tweet with a link to the original article. Although the summarization model defaults to LSI, there is also the option to switch between the other models described in this paper.

The Twitter bot[4] was created using the Twitter API and the Python library *tweepy*. Until permission is acquired from the sources used, the Twitter account will remain private in order to avoid issues with copyright.

Although the pipeline from article to tweet is fully automated, it must first be run manually. In the future, it would be beneficial to automate it further by scheduling the program itself to run at a certain time of day, or when a new article is posted. Currently, it is also only able to grab articles from *actu.fr*, but adding more news sources in the future is a possibility.

8 Conclusion and Future Works

Throughout this project, we have built a web scraper and created a dynamic COVID-19 corpus on French news articles, manually annotated for local relevance. We then created text classification and text summarization models. Finally, everything was combined to create a complete pipeline.

Our summarization task currently contains two different methods (with one method having two separate models, CamemBERT and FlauBERT) and two evaluation metrics (with two variants each). These have been tested on the corpus we have built as well as another, much larger one for training.

Our classifier evaluation results are satisfactory, with a maximum F1-Score close to 90% in the tuned LR model. On the other hand, our summarization results are mixed, but after adjusting BERTScore to drop empty scores, our LSI model achieves a maximum score of 79.93%.

Lastly, we have implemented a full pipeline for our project, which automatically selects news articles pertaining to COVID-19 in France using our classifier, summarizes them and posts them to Twitter. In the future, the overall performance and usefulness of the project could be improved by further automation of the pipeline, the addition of more news sources, and optimization of our models.

The entirety of *Newsjam*'s code and results can be found on GitHub[5].

[4]https://twitter.com/newsjam_fr
[5]https://github.com/pie3636/newsjam

References

Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D. Trippe, Juan B. Gutierrez, and Krys Kochut. 2017. Text summarization techniques: A brief survey. In *Proceedings of arXiv, USA*, pages 1–9.

Harold Borko and Myrna Bernick. 1963. Automatic document classification. *Journal of the ACM (JACM)*, 10(2):151–162.

Srinivas Chakravarthy. 2020. Document Summarization Using Latent Semantic Indexing.

Mita K Dalal and Mukesh A Zaveri. 2011. Automatic text classification: a technical review. *International Journal of Computer Applications*, 28(2):37–40.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Bonnie J. Dorr, Christoph Monz, Stacy President, Richard Schwartz, and David Zajic. 2005. A methodology for extrinsic evaluation of text summarization: Does rouge correlate? In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 1–8.

Yihong Gong and Xin Liu. 2001. Creating generic text summaries. In *Proceedings of Sixth International Conference on Document Analysis and Recognition*, pages 903–907.

Akanksha Gupta. 2020. Understanding Text Summarization using K-means Clustering.

ialifinaritra. 2021. French Text Summarization. Original-date: 2020-10-07T21:22:46Z.

Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Medu, Laura Barnes, and Donald Brown. 2019. Text classification algorithms: A survey. *Information*, 10(4):150.

Konrad Krawczyk, Tadeusz Chelkowski, Daniel J Laydon, Swapnil Mishra, Denise Xifara, Benjamin Gibert, Seth Flaxman, Thomas Mellan, Veit Schwämmle, Richard Röttger, Johannes T Hadsund, and Samir Bhatt. 2021. Quantifying Online News Media Coverage of the COVID-19 Pandemic: Text Mining Study and Resource. *Journal of Medical Internet Research*, 23(6):e28253.

Alon Lavie and Abhaya Agarwal. 2007. METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231, Prague, Czech Republic. Association for Computational Linguistics.

Hang Le, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux, Benjamin Lecouteux, Alexandre Allauzen, Benoît Crabbé, Laurent Besacier, and Didier Schwab. 2019. Flaubert: Unsupervised language model pre-training for french. *CoRR*, abs/1912.05372.

Chin-Yew Lin. 2004a. Looking for a few good metrics: Rouge and its evaluation. In *Ntcir Workshop*.

Chin-Yew Lin. 2004b. Rouge: A package for automatic evaluation of summaries. In *ACL 2004*.

David M. Magerman. 1995. Statistical decision-tree models for parsing.

Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamé Seddah, and Benoît Sagot. 2020. CamemBERT: a tasty French language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7203–7219, Online. Association for Computational Linguistics.

Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. 2021. Deep learning–based text classification: A comprehensive review. *ACM Comput. Surv.*, 54(3).

Makbule Ozsoy, Ferda Alpaslan, and Ilyas Cicekli. 2011. Text summarization using latent semantic analysis. *J. Information Science*, 37:405–417.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA. Association for Computational Linguistics.

Michael Röder, Andreas Both, and Alexander Hinneburg. 2015. Exploring the space of topic coherence measures. *WSDM 2015 - Proceedings of the 8th ACM International Conference on Web Search and Data Mining*, pages 399–408.

Horacio Saggion and Thierry Poibeau. 2013. Automatic text summarization: Past, present and future. In *Multi-Source, Multilingual Information Extraction and Summarization*, pages 3–21. Springer, Berlin, Heidelberg.

Maddy Savage. 2020. Coronavirus: How much news is too much?

Thomas Scialom, Paul-Alexis Dray, Sylvain Lamprier, Benjamin Piwowarski, and Jacopo Staiano. 2020. Mlsum: The multilingual summarization corpus. *arXiv preprint arXiv:2004.14900*.

Krithi Shetty and Jagadish S. Kallimani. 2017. Automatic extractive text summarization using k-means clustering. In *2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT)*, pages 1–9.

Natalie Shulter. 2017. The limits of automatic summarisation according to rouge. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 41–45.

William Soto. 2021. X-pareval: A multilingual metric for paraphrase evaluation. Master's thesis, Université de Lorraine.

Josef Steinberger and Karel Jezek. 2009. Evaluation measures for text summarization. *Computing and Informatics*, 28:251–275.

Josef Steinberger, Karel Jezek, et al. 2004. Using latent semantic analysis in text summarization and summary evaluation. *Proc. ISIM*, 4:93–100.

Dima Suleiman and Arafat Awajan. 2020. Deep learning based abstractive text summarization: Approaches, datasets, evaluation measures, and challenges. *Mathematical Problems in Engineering*, 2020:1–29.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert.

A Examples of LSI model outputs

1. Poor Summary | Poor Score
   - MLSUM Test Dataset Article 21[6]
   - *Reference Summary*: "Le journalisme est un métier qui nécessite parfois de faire face aux situations les plus extrêmes. Aujourd'hui, c'est à une expérience en lisière de l'anthropophagie que je dois me livrer, puisque ma mission consiste à aller déguster le « Chirac », un burger portant le nom d'un ancien président de la République amateur de tête de veau."
   - *Generated Summary*: "Le vaillant cobaye qui m'accompagne et moi mordons à pleines dents dans cet agglomérat de fromage de Normandie dégoulinant, de pommes confites et de sauce à la moutarde."
     - ROUGE-L
       - Standard F1-Score: 0.146
       - Keyword F1-Score: 0.038
     - BERTScore
       - Standard F1-Score: 0.018
       - Keyword F1-Score: 0.036
   - This is an example where our generated summary is quite poor as the main subject of the article, a burger named the "Chirac", is never even mentioned, but a very in-depth description is included without any context.

2. Quality Summary | Quality Score
   - Built Corpus Actu Article 36[7]
   - *Reference Summary*: "A partir de ce lundi 11 octobre 2021, le port du masque en extérieur n'est plus obligatoire dans le Pas-de-Calais. Explications."
   - *Generated Summary*: "Le port du masque en extérieur n'est plus obligatoire dans le Pas-de-Calais. A partir de ce lundi 11 octobre 2021, le port du masque en extérieur n'est plus obligatoire dans le Pas-de-Calais."
     - ROUGE-L
       - Standard F1-Score: 0.738
       - Keyword F1-Score: 0.743
     - BERTScore
       - Standard F1-Score: 0.711
       - Keyword F1-Score: 0.521
   - This last example is one where we have a high quality generated summary that receives a relatively high F1-Score. The summary itself could still be improved by being less repetitive, which may in turn improve the score. This problem of repetitiveness is one we often found with our generated summaries from this corpus and we plan to look into it further.

[6] https://bit.ly/3GaDM7x
[7] https://bit.ly/3FbJLaO

B Annotation Guidelines

- Local (1):
  1. Article is directly relevant to someone who lives in France (i.e. affects daily life in France)
  2. Article only mentions France or does not mention any country but it's clear that it's about France (due to mention of departments, etc.)
  3. Article mainly discusses France but mentions other countries only for comparison/data
  4. Article may mention other countries but highlights travel restrictions to enter France (labeled as 'Local' due to Local guideline 1; as this could be relevant to a French citizen returning to France or to a French citizen that has international family visiting)

- Global (0):
  1. Article is not tied to a specific country but is relevant to anyone in the world (i.e. article about vaccine efficacy, variants, etc.)
  2. Article mainly explains situations in other countries, can mention France but not as the main focus of the article
  3. Article focused on travel restrictions to enter foreign countries
  4. Some articles consist of hourly update/daily highlights format where some updates are France focused while others are worldwide focused. In situations like this, label the article as global since a large amount of the info is not constrained to the French situation.