UCLouvain

École polytechnique de Louvain

Analysis of a news generating tool in different contexts

Author: **Jeremie Bogaert**
Supervisors: **François-Xavier Standaert, Antonin Descampe**
Reader: **Axel Legay**
Academic year 2020–2021
Master [120] in Computer Science and Engineering

Abstract

During last years, fake news generators have done big progress and are now able to product really convincing results. In this work, we evaluate the quality of the news outputted by a state-of-the-art-news generator depending of the input data it is given. The goal is to check if the news are of worst quality when the generator works on topics far from the ones it was trained on, and if this change in quality is detected by both human beings and machine learning tools. To achieve this goal, we start by building our database. The original news come from crawling and the generated ones are obtained by giving some part of the original ones to our news generator. We then set up an experiment plan to collect human evaluations and we build two classifiers to do our automated evaluation. We observe that the news generated on new topics are of worst quality than the other ones, and that it is at least detected by machine learning tools. For the human evaluation, the results tend to be similar but they are less significant and ask for further investigations. We conclude that it shows that a state-of-the-art news generator, Grover, is not really durable in time. We finish by insisting on all the questions it raises, like if it is the case for other generators, when and how much to re-train the generator or what to do to mitigate the problem.

Acknowledgements

I thank my girlfriend, Mathilde, for all the support she gave me, the efforts she put in canalising my stress and all the attentions she had during the whole semester.
I thank my father for all the interesting discussions about the statistical bias of a human experiment.
I thank my whole family for the support they gave me, in particular during the writing period.
I thank Clément Massart for the help he gave me during the creation of the website.
I thank all the participants of the human experiment, for the time and efforts they put into it.
I thank my friends, who helped me to release the pressure when it was needed.
Finally, i thank my promoters, François-Xavier Standaert and Antonin Descampe, who helped me a lot during the whole semester, and who allowed me to participate to the redaction/presentation of a first paper at the WIC international conference.

Contents

Part 1

Motivation and context of the study

In the last past years, the fake news threat has been growing a lot [1], raising the need of efficient detectors to deal with the problem. Fake news detection being at a cross road between multiple disciplines [2], diverse approaches have been and are still tried to build efficient detectors. We can for example cite the graph based approach [3], based on the spreading of the news, or the text based approach, that aims at understanding how is a fake news written [4] [5]. On top of that, to help the research, the adversarial approach is widely used [6], in particular in the second domain. It can be seen as putting yourself in the skin of your adversary to try to understand what he is able to do and how he can do it. Applied to fake news detection, it consists in building the best fake news generators as possible. Even if it may seem counter intuitive to generate fake news to solve the problem, being able to generate well written fake news also means being able to assess the quality of the detectors. Currently state-of-the-art generators use a mix between natural language processing (NLP) and deep learning. The resulting "neural fake news" have proven to be hard to detect, whether by human beings or with automatic detection tools [6].

However, the research on the fake news field is still facing multiple challenges. Among them, there is the question of the adversarial behavior [7] of the fake news classifiers. A paper analyzing this problem in particular was written and presented at the WIC international conference during the semester (cf appendix A). Another recurrent problem, that was also faced during this preceding work, is that it is really hard to define what a fake news is [8][9]. It makes the task of data labeling difficult and it can have impact on the model's training. Finally, a third problem that, to the best of my knowledge, is not really discussed in the literature is that the news topics are changing every day. It means that after being trained, news generators could be asked to generate about new topics, topics

that they have not been trained on. This task is done day after day by human journalist, but can we expect the same from these automated tools ? It leads to the first question we will try to answer in this work: **"Is a state-of-the-art news generator able to output news of the same quality when it generates about new topics ?"**. Such a question is obviously key, because depending of the answer we could have to retrain it often to be able to get correct results. It is even more important when one has an idea of the cost to train such models [10].

To answer this question, we will assess the quality of the generated news in different contexts. The choice of generated/original rather than fake/reliable is important, because it prevents one of the major problems mentioned above: the data labeling. We are therefore not going to talk about "neural fake news", but rather about "neural news" in a broader sens. In such a classification in generated/original news rather than fake/reliable ones, we can work with a high number of data since we can label them more easily. It also means that we can generate them on our own using a state-of-the-art news generator and evaluate the quality of the results without having to check the facts in it. Of course, such a quality evaluation will not be based on the liability of the news anymore, but rather on how it is written, if it is self-coherent, etc.

In short, this work will evaluate the quality of the generated news, without distinguishing fake and reliable ones, in different contexts. "Different contexts" is considered to be different similarities between the input we give to the model in order to generate a news on one hand, and the data that were used to train it on the other hand. As the purpose of news generator is mainly to be able to fool people, the assessment of the news's quality will be done by both human beings and machine learning tools. It leads to the second question that this work will try to answer : **"If there is a quality difference between the news generated in different contexts, is it observed both by human beings and by machine learning tools ?"**.

This work will detail every step that was done in order to answer both of our questions. The first part will present the creation of our experiment database. The second part will talk about the experiment plan to collect the human and automated (i.e. done with machine learning tools) evaluations. The last part will show the results we obtained and will discuss their significance. Finally, we will conclude by a small section about all the possible further works.

Part 2

Background

You can find below the different prior knowledge needed to understand everything presented in the work. The sections concerning tools that were used as black box do not have mathematical explanation (as it was not taken into account to use these tools and the work is understandable without having to consider the maths), but you are free to check the related papers if you want more insight about how the tools are built mathematically.

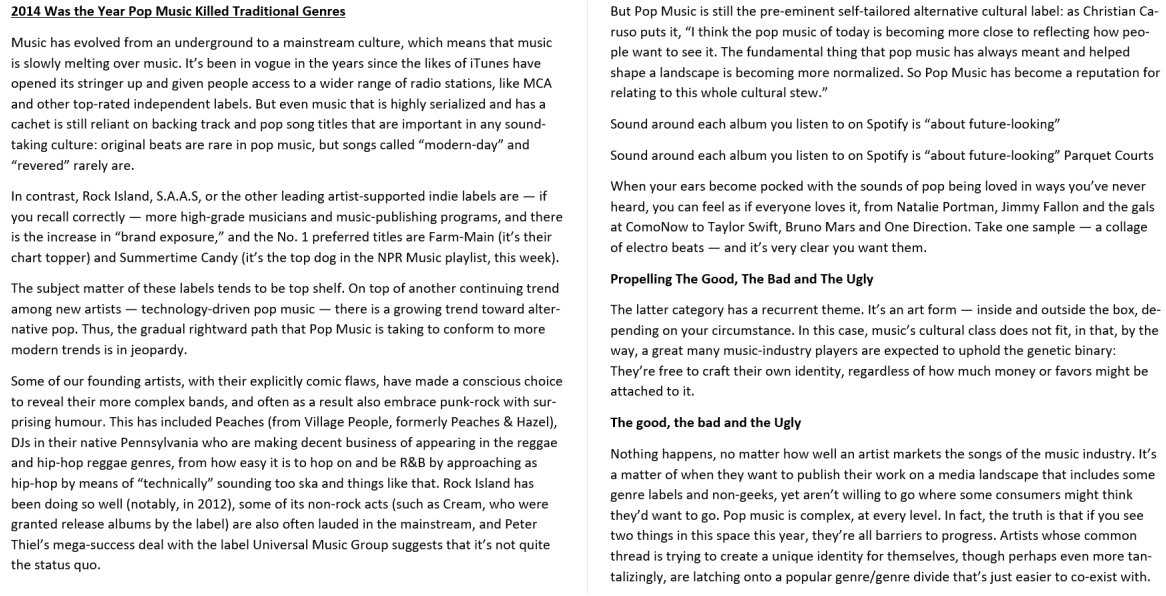
2.1 Text generator

This section contains explanations about the news generator that was used named Grover as well as all the prior knowledge needed to understand the analysis of its database.

2.1.1 Grover

Introduced in the paper [6], *"Grover"* is a state-of-the-art news generator that can create well written news even with minimal inputs. Even if it is supposed to get slightly better results when it is given some meta data like the url, the author's name or a small summary of the needed news, it is able to output a generated news given only a title and a date. Figure 2.1 shows an example of what kind of news Grover is able to generate given only these two information. Note that the titles and subtitles were manually highlighted to ease the reading. We can also notice that some sentences corresponds to meta data and not to the news in itself. It can be used to insert a publicity, picture, tweet, reference, etc. manually after the generation to make the news look even more real [1].

[1] We didn't modified anything and just used the raw text outputted by Grover. It means that we kept the meta data in it and that we didn't replaced them by pictures, tweets, etc.



2014 Was the Year Pop Music Killed Traditional Genres

Music has evolved from an underground to a mainstream culture, which means that music is slowly melting over music. It's been in vogue in the years since the likes of iTunes have opened its stringer up and given people access to a wider range of radio stations, like MCA and other top-rated independent labels. But even music that is highly serialized and has a cachet is still reliant on backing track and pop song titles that are important in any sound-taking culture: original beats are rare in pop music, but songs called "modern-day" and "revered" rarely are.

In contrast, Rock Island, S.A.A.S, or the other leading artist-supported indie labels are — if you recall correctly — more high-grade musicians and music-publishing programs, and there is the increase in "brand exposure," and the No. 1 preferred titles are Farm-Main (it's their chart topper) and Summertime Candy (it's the top dog in the NPR Music playlist, this week).

The subject matter of these labels tends to be top shelf. On top of another continuing trend among new artists — technology-driven pop music — there is a growing trend toward alternative pop. Thus, the gradual rightward path that Pop Music is taking to conform to more modern trends is in jeopardy.

Some of our founding artists, with their explicitly comic flaws, have made a conscious choice to reveal their more complex bands, and often as a result also embrace punk-rock with surprising humour. This has included Peaches (from Village People, formerly Peaches & Hazel), DJs in their native Pennsylvania who are making decent business of appearing in the reggae and hip-hop reggae genres, from how easy it is to hop on and be R&B by approaching as hip-hop by means of "technically" sounding too ska and things like that. Rock Island has been doing so well (notably, in 2012), some of its non-rock acts (such as Cream, who were granted release albums by the label) are also often lauded in the mainstream, and Peter Thiel's mega-success deal with the label Universal Music Group suggests that it's not quite the status quo.

But Pop Music is still the pre-eminent self-tailored alternative cultural label: as Christian Caruso puts it, "I think the pop music of today is becoming more close to reflecting how people want to see it. The fundamental thing that pop music has always meant and helped shape a landscape is becoming more normalized. So Pop Music has become a reputation for relating to this whole cultural stew."

Sound around each album you listen to on Spotify is "about future-looking"

Sound around each album you listen to on Spotify is "about future-looking" Parquet Courts

When your ears become pocked with the sounds of pop being loved in ways you've never heard, you can feel as if everyone loves it, from Natalie Portman, Jimmy Fallon and the gals at ComoNow to Taylor Swift, Bruno Mars and One Direction. Take one sample — a collage of electro beats — and it's very clear you want them.

Propelling The Good, The Bad and The Ugly

The latter category has a recurrent theme. It's an art form — inside and outside the box, depending on your circumstance. In this case, music's cultural class does not fit, in that, by the way, a great many music-industry players are expected to uphold the genetic binary: They're free to craft their own identity, regardless of how much money or favors might be attached to it.

The good, the bad and the Ugly

Nothing happens, no matter how well an artist markets the songs of the music industry. It's a matter of when they want to publish their work on a media landscape that includes some genre labels and non-geeks, yet aren't willing to go where some consumers might think they'd want to go. Pop music is complex, at every level. In fact, the truth is that if you see two things in this space this year, they're all barriers to progress. Artists whose common thread is trying to create a unique identity for themselves, though perhaps even more tantalizingly, are latching onto a popular genre/genre divide that's just easier to co-exist with.

Figure 2.1: Example of a news outputted by Grover

Grover has 3 different versions presented in [6]. The first one is Grover-base that has 12 layers and 124 million parameters, on par with GPT and BERT-Base [11] [12]. The second is Grover-Large, that has 24 layers and 355 million parameters, on par with BERT-Large. The last one is Grover-Mega, that has 48 layers and 1.5 billion parameters, on par with GPT2 [13]. Since we used the open source version of Grover (`https://github.com/rowanz/grover`), we used Grover-base that is available in the github repository. Grover can also be used to detect fake news, but it was only used in generation mode in our case. The code used to setup Grover in this mode is available in Appendix A.

2.1.2 Latent Dirichlet Allocation

First introduced in paper [14], latent dirichlet allocation (LDA) is a topic modeling tool. It allows to get the topics of a text corpus as well as the words associated with each topic. As explained in the paper : *"It is a generative probabilistic model for collections of discrete data such as text corpora. LDA is a three-level hierarchical Bayesian model, in which each item of a collection is modeled as a finite mixture over an underlying set of topics. Each topic is, in turn, modeled as an infinite mixture over an underlying set of topic probabilities."* In other words, LDA assumes that every text is a mixture of topics and that every word in the text appears with a certain probability for each of these topics. Since the model was only used as a

black box, the equations behind the model are not shown in this work, but if you want more information about it, you are free to read [14]. We used LDA to extract some of the topics of Grover's training database. The only parameter we used is λ, but it is not important to understand the work nor the results. It is linked to what are considered the "top words" for a topic. A big value of lambda will just look at the total number of appearance of the word inside the topic, while a lower value will consider more the word's frequency of appearance in the topic. More details can be found in [15] [16].

2.2 CommonCrawl

Commoncrawl (`https://commoncrawl.org/`), is a website that stores huge amount of information available online. It exists specifically to allow machine learning tasks that need a lot of training data from the web. Since 2016, Commoncrawl released *CC-news*, which is a dataset only containing news from all around the world. This dataset is the one that allowed to build Grover's training database, *Realnews*. We used commoncrawl to get the news needed to do our automated evaluation. The steps needed to use it are detailed in the Appendix B.

2.3 About Natural Language Processing

Natural language processing (NLP) is a field in a cross road between linguistics, computer science and artificial intelligence. The goal of NLP is to study the interactions between computers and human language, in particular how to program computers to process and analyze large corpus of natural language data. Note that some of the part of this section are inspired by Quentin Carbonelle's master thesis [17].

2.3.1 Pre-processing

The first required step to build any machine learning model working with text data is to transform the data into usable inputs[18]. This is called pre-processing and it is done in multiple steps in NLP. Its goal is to clean up the data while losing as few information as possible. It is also to transform the textual data into numbers to make them usable by machine learning models. Of course, different tasks often require a different pre-processings and the following sections will only present the pre-processing methods that were used during this work. It consists in the following steps:

- Tokenizing into sentences

- Part Of Speech tagging
- Cleaning
- Lemmatizing
- Tokenizing into words
- Vectorizing

Tokenizing

The first step to pre-process a text is often to separate it into smaller pieces. "Smaller pieces" can mean words or sentences, depending of what the task requires. In this work, we first tokenized the text into sentences. After some other steps, we did it again to split the sentences into words just before the vectorization.

Part Of Speech Tagging

The second step is to analyze each sentence one by one to determine what is the goal of each word in it. Such a process is called Part Of Speech (POS) Tagging, and its purpose is to get the grammatical nature of each term. To do it, we used the pre-trained POS tagger available in the nltk package (`http://www.nltk.org/book/ch05.html`). Of course, even state-of-the-art POS tagger are not perfect since they usually get results around 95% accuracy and this one is not an exception. There are still many researches done for the field to help getting better POS results, in particular for difficult languages [19] [20], since it is considered to be one of the mandatory steps to do for any NLP task [21].

Stemming/Lemmatizing

Stemming is the process of reducing inflected words to their word stem or base form while lemmatizing is the process of reducing inflected words to their lemma. The only difference between these two terms is that a lemma is always a meaningful word while a stem is not supposed to (but can) be meaningful. This step is done after the POS tagging to make use of the grammatical nature of the words in order to find more accurate base forms. This step is one of the staple of pre-processing [22] and is key because it allows to check if two words share the same base form and therefore the same meaning. For example :

- The words "user" and "users" are two different words, but they share the same base form, which is "user"

- The words "gone" and "goes" share the same base form, which is "go"

Note that the two base forms above are both lemmas, and that two words can share the same base form while none of them is in that form as we can see in the second example. Being able to see that such words share the same meaning is important to better understand the sentence while not having to look at all of the possible inflected variations of each word in it.

Cleaning

The next step is to clean the data. Its goal is to get rid of all the useless or the meaningless information in the text. It can be words, punctuation, upper case letters, emojis, etc. It may seem surprising to not start by the cleaning, but keeping every word during the POS tagging allows to have a better understanding of the sentence and therefore better results. To clean the data, one can first put every word in lower case to avoid some problems like considering the first word in every sentence differently than the others for example. One can then get rid of all the useless characters. Again, different tasks may consider different characters as useless, but in this case, only the alphabetical characters were kept. It means that the punctuation, numbers, emojis, special characters, etc. were removed from the text. The last part of data cleaning is to remove the useless words. Even if they are only composed of alphabetical character, words like "to", "from", "of", etc. are mostly noise to machine learning models since they appear everywhere in the text while giving little to no information. Such words are called stopwords and they were also removed from the news during the pre-processing. On top of that, all the words composed of 2 or less characters were deleted from the text as well.

Vectorization

Now that the input texts are cleaned, we need to transform them into what a machine learning model can understand, numbers. To do that, one have to convert the text to a vector. That step is called vectorization and can be done in multiple ways. Again, the method depends on the needs and the features the model wants to focus on. Even if new and probably more powerful vectorization methods are currently in development([23]), we chose to use two relatively "old-school" ones in this work: tf-idf and words embedding.

2.3.2 Term Frequency-Inverse Document Frequency

The term frequency-inverse document frequency (tf-idf) is a frequently used vectorization technique [24]. It uses two factors, namely the term frequency and the inverse document frequency to assign a score to each word in a corpus of documents.

In other words, it considers how important is the word in the given document as well as how rare it is to see the word in a document of the corpus. Its general equation is

$$tfIdf(t, d, C) = tf(t, d) * idf(t, C) \tag{2.1}$$

where t is the term, d is the document and C is the whole corpus. The tf and idf functions can take various forms, as long as tf is proportional to the word frequency in the document and idf is inversely proportional to the number of documents that contains the term. In this work, the equations of tf and idf are the ones from the sklearn package available at `https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html`.

2.3.3 Word embedding (Word2vec)

Word embedding is the concept of mapping words to vectors of real numbers. Word2Vec [25] designates a type of word embedding in which the models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. It takes as input a corpus of text and produces a vector space where each unique word in the corpus is assigned to its corresponding vector in the space. The vector space allows two things. First, it depicts some semantic meaning and relationship between the words. For example, one can say that in the vector space:

$$king - man + woman \approx queen \tag{2.2}$$

where king, man, woman and queen are the vectors corresponding to these words respectively. It follows that similar vectors in this space represent words that tend to have similar meanings.

The similarity of two words is usually computed with the cosine similarity [2] between the corresponding vectors in the embedding space. If V_1 and V_2 are the embedding vectors of the words w_1 and w_2 respectively and θ is the angle between these two vectors, the cosine similarity can be computed as:

$$Cos\_similarity(w_1, w_2) = cos(\theta) = \frac{V_1.V_2}{||V_1|| \cdot ||V_2||} \tag{2.3}$$

Again, since this section is not considering the equations and models behind the method but only describing it, more information can be found in [25] for people who would want more insights about it.

[2] The cosine similarity is the cosine of the angle between the two vectors.

2.4 About the visualisation tools

This section is partially inspired by Quentin Carbonelle's master thesis. [17].

2.4.1 T-distributed Stochastic Neighbor Embedding

To analyze a database, it is often interesting to visualize how similar are its elements. In our case, since we want to compare news, we need to be able to visualise how close are two texts. A common way to do it is to first vectorize the texts (cf section 2.3.1 Vectorization) and then use a high-dimensional visualization tool. T-distributed Stochastic Neighbor Embedding (t-SNE), introduced in [26], is a nonlinear dimensionality reduction technique well-suited for the visualization of high-dimensional datasets. The t-SNE algorithm can be described in two main steps.

1. First, it constructs a probability distribution over pairs of high-dimensional data points. The pair's probability is proportional to the similarity between the two elements.
2. Then, it tries to fit a similar probability distribution over the points in a low-dimensional map. It usually uses 2 as low dimension size to be able to represent the data on a 2D grid.

As t-SNE was used as a black box, no math is needed to understand the results we got with it. If you want more mathematical insight, you are free to look at [26].

2.5 About the metrics

2.5.1 P-value

The P-value is widely used in null hypothesis significance testing. It is the probability of obtaining test results at least as extreme as the results actually observed, under the assumption that the null hypothesis is correct[3]. It follows that having a very small p-value means that observing at least as extreme outcome is very unlikely if we accept the null hypothesis. In other words, it means that having a low p-value tells us to reject the null hypothesis. P-values are used in section 4.2.2 to illustrate the supposed independence between some variables.

[3]The null hypothesis is a default hypothesis that a quantity to be measured is zero. Typically, the quantity to be measured is the difference between two situations. In our case, the two situations are the results with and without a certain variable. It allows to check if the results are dependant of that variable.

2.5.2 Accuracy

The accuracy is a commonly used machine learning metric [27] that shows the ratio between the number of correct predictions and the total number of predictions. In other words, if we only predict once for each input sample, the accuracy represents the percentage of well classified inputs.

$$Accuracy = \frac{\#Correct\ predictions}{\#Total\ predictions} \tag{2.4}$$

As it does not weight the correct prediction from different classes differently, it is mostly used when the number of samples belonging to each class is the same. Since it is the case in this work, the accuracy will be a very useful metric.

2.5.3 Mean absolute error

The mean absolute error[4] (MAE) is a statistical measure of error. For a model that has to predict a value for n different inputs, it is computed as the sum (over all the samples) of the absolute value of the difference between the expected and the actual prediction, divided by n.

$$MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} = \frac{\sum_{i=1}^{n} |e_i|}{n} \tag{2.5}$$

2.5.4 Cross entropy

To learn from observations, a classifier needs to know and to quantify how close are its actual prediction from the expected ones. To do it, it uses what is called a loss function. A typical loss function for classification tasks is the cross entropy[28] that allows to get a metric of how similar are two probability distributions. The general form of the cross entropy is defined as

$$H(p, q) = -\sum_{x} p(x) log(q(x)) \tag{2.6}$$

where p is the correct distribution of classes and q is the estimated one. In this work, we used two particular forms of the cross entropy loss function : the binary one and the sparse categorical one.

[4]Usually, a more common error to use is the MSE (mean squared error), but training the models with MAE as error metric gave better results in our case.

Binary cross entropy

If we only have two classes (generated/original in our case), namely C_1 and C_2, the equation becomes :

$$H(p,q) = -\sum_x p(C_1) * log(q(C_1)) \ + \ p(C_2) * log(q(C_2)) \tag{2.7}$$

where $p(C_1)$ and $q(C_1)$ are the expected and actual probability of item x being of class 1. Note that since there are only two classes, and because p and q are both probability distributions, it is possible to replace $p(C_1)$ by $1 - p(C_2)$. The same goes for q.

Sparse categorical cross entropy

To explain what is the sparse categorical cross entropy, we first have to clarify what is the categorical one. The categorical cross entropy is basically the same as the classical version presented above except that it works with vectors. In that version, every expected probability p(x) is a bit vector with only one of its values equal to 1. Every actual probability q(x) is a vector with all its elements summing to 1. The only difference between sparse categorical cross entropy and categorical cross entropy is the format of the expected labels. It means that the actual predictions are still encoded as vectors, but since each data entry can only belong to one class (i.e. can only have one expected prediction), the expected predictions are one-hot encoded. It follows that it can be used to train a model that outputs probability vectors by giving it the expected classes. It will be particularly helpful when training our neural network in section 3.4.2.

2.6 About the classification models and their tuning

This section is partially inspired by Quentin Carbonelle's master thesis. [17].

2.6.1 Classifier

The output of a binary classifier is a value between 0 and 1, called the prediction probability. It represents the confidence of the classifier to label the input as class 1. In our context, the generated (resp. original) news are of class 0 (resp. 1). Therefore, an output with a value around 0.5 means that the classifier is not confident at all about its prediction, while an output close from 0 or 1 means that the classifier is confident about its prediction. The returned probability often needs

to be converted to a binary value and to do it, one needs to define a classification threshold. It means that a value above the threshold will be considered as class 1, and below as class 0. This is practical when one is not interested in the probability value, but in the class only. The threshold is problem-dependent and we will fix it to the most common value: 0.5. It follows that every value below 0.5 indicates a 'generated' prediction, while all the other values indicates an 'original' prediction.

2.6.2 K-fold cross validation

The K-fold cross validation is a validation method used to compare the results on the training and test sets. It solves the problem encountered if the data are only split in train and test once, which is that the split could be non representative of the data. We could have really easy examples to classify in the training set and all the hardest one in the test set, or the opposite. In k-fold cross validation, the data are split in k subsets. You can then train your model on k-1 parts and test its results on the last one. Finally, after using each part as the test set, you can average the results to measure the average accuracy your model is able to reach on a test set. Figure 2.2 shows an illustration of how a 5 fold cross validation is done.

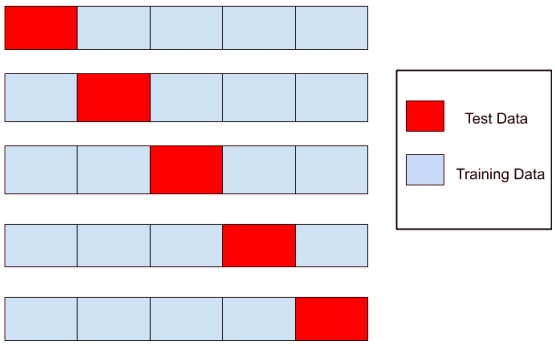


Figure 2.2: Representation of a 5-fold cross validation [5]

2.6.3 Logistic regression

The logistic regression is a statistical model. We used its basic form that models a binary dependent variable with a logistic function. To understand what is a

[5]Image taken from https://www.mltut.com/k-fold-cross-validation-in-machine-learning-how-does-k-fold-work/

logistic regression, it is first important to understand what is the logit or the log-odds function. The logit function is the logarithm of the odds and can be written as :

$$logit(p) = \frac{p}{1-p} \tag{2.8}$$

where p is a probability. The logistic regression assumes a linear relationship between the input variables $\mathbf{x} = (x_1, x_2, ..., x_n)$ and the logit of the prediction probability y. It means that a logistic regression aims at learning the w_i and ϵ parameters in the following equation:

$$logit(y) = \frac{y}{1-y} = w_1 x_1 + ... + w_i x_i + \epsilon = \mathbf{wx} + \epsilon \tag{2.9}$$

The model is fit by minimizing the binary cross-entropy (cf. section 2.5.4) on the training samples. To prevent from overfitting [6] we usually use regularization methods. Such methods add diverse penalties (l1, l2 or a linear combination of both) depending of the weights of the model to prevent them from being too big. When adding the l1 penalty, defined as $\lambda \sum_{i=1}^{p} w_i^2$, the model is called a Lasso regression, while when adding a l2 penalty, define as $\lambda \sum_{i=1}^{p} |w_i|$, it is called a Ridge regression [7].

2.6.4 (Bi-directional) Long Short Term Memory

Long short-term memory (LSTM) [29] is a particular type of recurrent neural network (RNN) [30]. A recurrent neural network is a neural network with feedforward and feedback connections, allowing loops and persisting information. Figure 2.3 shows an illustration of a RNN :

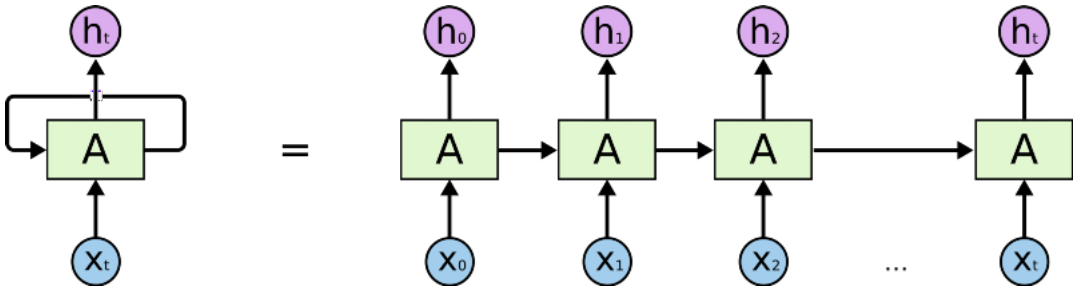


Figure 2.3: Illustration of an RNN architecture [8]

[6] To overfit means to fit too specifically to the train samples. It follows that the learned model gives really good scores when training, but does not generalize well to new data.

[7] Note that both these penalty need a careful choice for the λ parameter.

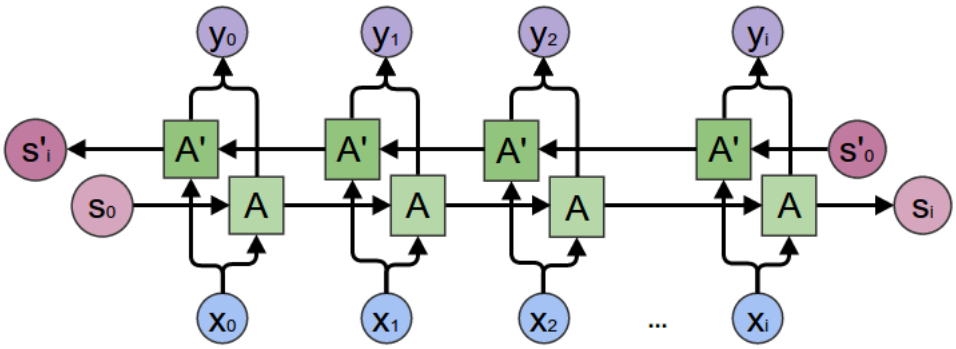


Figure 2.4: Illustration of a Bi-RNN architecture[9]

LSTM differs from regular RNN by the composition of its units. A LSTM unit is mainly composed of 4 parts : a cell, an input gate, an output gate and a forget gate. The cell is the part that remember the values over time and the three gates are in charge of regulating what enters and what gets out of the cell. More information about LSTM and RNN can be found in [31].

A bi(-directional) RNN [32] is a network that combines feedforward and feedback connection. It is particularly useful in text processing when to understand a word in a sentence, it is often interesting to analyse the words next to it [12]. Since feedforward connections allow to have information about the words before and feedback connections give insight about the words after, using both of them helps to understand the whole sentence better. It follows that a bi-RNN, allows to make use of a better context to analyse each word. Finally, since LSTM is a particular case of RNN, a bi-LSTM is a particular case of bi-RNN where both RNN's are replaced by LSTM networks.

[8]Image taken from `https://medium.com/@raghavaggarwal0089/bi-lstm-bc3d68da8bd0`
[9]Image taken from `https://medium.com/@raghavaggarwal0089/bi-lstm-bc3d68da8bd0`

Part 3

Methodology

3.1 Goal of the work

As said in the first section of this work, the dynamic evolution of the news topics is something that is often forgotten when evaluating news generators. Since the news topics are evolving days after days [33], the news generators need to be able to generalize their knowledge out of the topics they were trained on. This point is even more important as the cost of training machine learning models [10] tends to be higher and higher with the quick evolution of the models. This work aims at answering two questions:

1. **"Is a state-of-the-art news generator able to output news of the same quality when it generates about new topics ?"**
2. **"If there is a quality difference between the news generated in different context, is it observed both by human beings and by machine learning tools ?"**

To answer these questions and check if a current state-of-the-art news generator, Grover (cf. section 2.1), is able to generalize its knowledge, we will assess its performance in "different contexts". These contexts corresponds to input further and further from the data that were used to train Grover. We define 4 of such generation contexts with different rules based on the similarity they have with Grover's training database.

1. Using all the data and meta data available from the database for selected entries.
2. Using only the title and the date from the database for selected entries.

3. Using only the title and the date from news not present in the database but treating similar topics at similar dates as the news of the database.

4. Using only the title and the date from news not present in the database and treating topics not present in the database, at dates not covered by the database.

The first one is there to see what are supposedly the best results we can get using the news generator, since it corresponds to the exact same data that were used by Grover to learn how to generate texts. The second one allows to assess if it is possible to generate well written news with less input data. The choice of the title and the date as input data is motivated in section 3.2.1. The third one's goal is to see whether the model has indeed learned to write news on topics similar from its training ones. The last part is the key motivation of this work and aims at evaluating how easy it is for a state-of-the-art news generator to generalize its knowledge to new topics. We will refer to the news corresponding to these 4 contexts as the news of category 1, 2, 3 and 4 respectively.

To assess the quality of the generation, we will do a human evaluation on one hand, and an automated (i.e. with machine learning tools) evaluation on the other hand. To reach this goal, we split our study in three parts: The first part explains the creation of our database. The second part concerns the human quality evaluation while the last part concerns the automated evaluation.

3.2 Database creation

This section aims at explaining all the steps that were done to build the database we will use to assess the quality of the generation. This database was created in three steps: The first step is the study of Grover's database to isolate topics it was trained on. The second step is the search of human written news[1] corresponding to the characteristic we presented in the above section. The last one is the generation using Grover with the title and the date of the corresponding original news as input.

3.2.1 Grover and its database

The choice of Grover as a news generator was motivated by the fact that it is easy to use, can generate state-of-the-art news given only a date and a title and is

[1]that we will qualified as the "original" news during the rest of this work

an open source tool available on Github (`https://github.com/rowanz/grover`). You can find a little description of it in section 2.1.1, and for a bigger one, you are free to look at [6].

To build the 4 parts of our database corresponding to the 4 news category we presented in section 3.1, we first need to take a look at the database Grover was trained on. The database is called Realnews and is made of 37GB of data. It is composed of 32 millions news in a jsonl format. Every news in it contains the following fields.

- url : The url of the news on the web
- url_used : The url that was used by commoncrawl to find the news
- title : The title of the news
- text : The news itself
- summary : A small summary of the news
- authors : The author(s) of the news
- publish_date : The date at which the news was published
- domain : The news's website's domain
- warc_date : The date of the warc file used by Commoncrawl
- status : One of the field used by commoncrawl
- split : Train or validation, depending if the article was used to train Grover or to test its performances.

The red fields are linked to the way the data were collected using commoncrawl (cf. section 2.2), the green one is related to the training of Grover[2] and the blue ones are defining the news in itself.

We chose to focus on the title and the date for mainly 2 reasons. The first one is that Grover is able to generate the missing field itself [6] if needed and that we are therefore not forced to use all the blue parameters. The second one is that it seems easier to represent what a good publish date and a good title can be to generate

[2]Since we consider the whole database to be part of Grover's training, we did not distinguish the news using this field.

a news rather than what is a good author, domain name or url for example. On top of that, these data are not always available for the original news, in particular when collecting them with commoncrawl. Since we could not guarantee that we will have an easy access to these fields for every news, and to minimize the disparity in our database, we prefered to use only two fields that are easily accessible for all the news : the title and the date.

Since we are going to focus on these fields, let's analyse them in a little more details. First, we start by analysing the publish_date field. Figure 3.1 shows its distribution in the database.

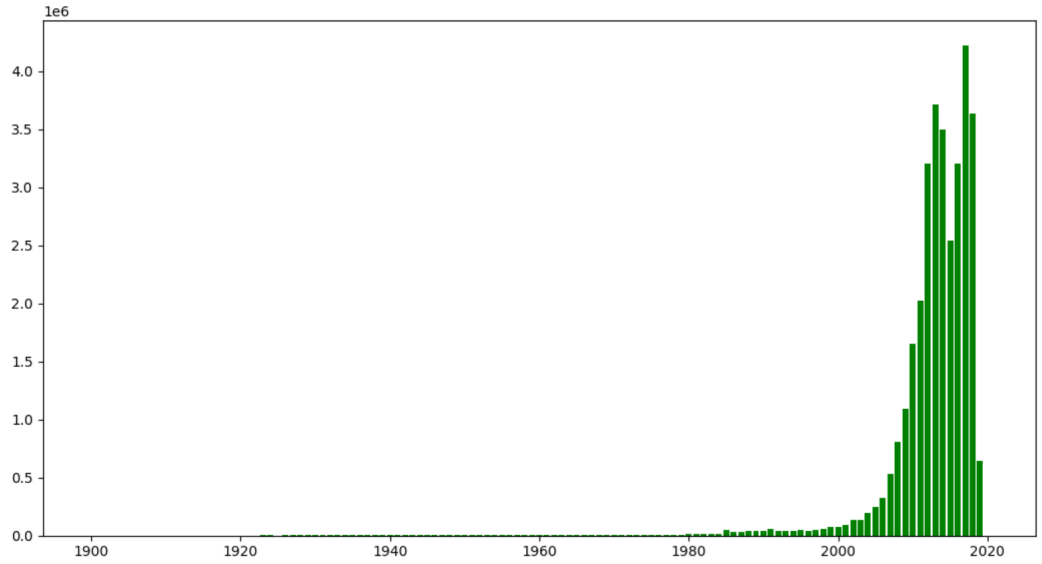


Figure 3.1: Distribution of the publish_date field's value in the database

We can see that the news were published roughly between 2000 and 2019. Some news were published between 1980 and 2000 as well, but their proportion is really low compare to the former ones. The main part of the database is composed of recent news, published between 2010 and 2019. Since Grover was released in 2019, it seems logical that there are no news published later than that in its database.

We can now analyse the titles. We will use them to find some topics of the database, as it was found to be too computationally intensive to use the main text of the news. We start by using a simple wordcloud to see what kind of words are the most present in the titles. Of course, we are not interested in stop words (cf. 2.3.1 Cleaning), so we remove them before building it. Figure 3.2 shows the

obtained wordcloud.

Figure 3.2: Wordcloud built with the top 5000 words among 1 million randomly selected titles

Even after removing the stop words, it does not provides us a lot of information about the topics of the news. At best, it's vague (*"trump", "family", "business", ...*), at worst it is simply useless (*"motley", "video", "review", "one", "back", ..."*). We need another way to find the main topics of the database. To do it, we chose to use latent Dirichlet allocation or LDA (cf. 2.1.2) which is a model that allows to extract topics from text corpus. Using LDA [3] to find 25 topics among the million titles we chose at random in the database, we get a map of topics where two topics appearing together intersect. Figure 3.3 shows such a map.

We can already see on the bottom left that some topics tend to appear together quite often while some others, on the top right are never combined with any other topic. Of course, LDA does not tell us the name of these topics, but it gives the words associated to them. For example, we can see the top words for topic 1 in figure 3.4. These words help us to guess the topic. In this case with the words *man, police, arrested, death, murder*, etc. we can say that it is related to criminality. However, it is sometimes more difficult to guess the topic based on the top words. For example, the top words for topic 11 are shown in figure 3.5

[3] We used a python package named ldavis [16] to do it and all the figures below come from an html file that ldavis build to help visualize the topics. The file should be made available on github soon.

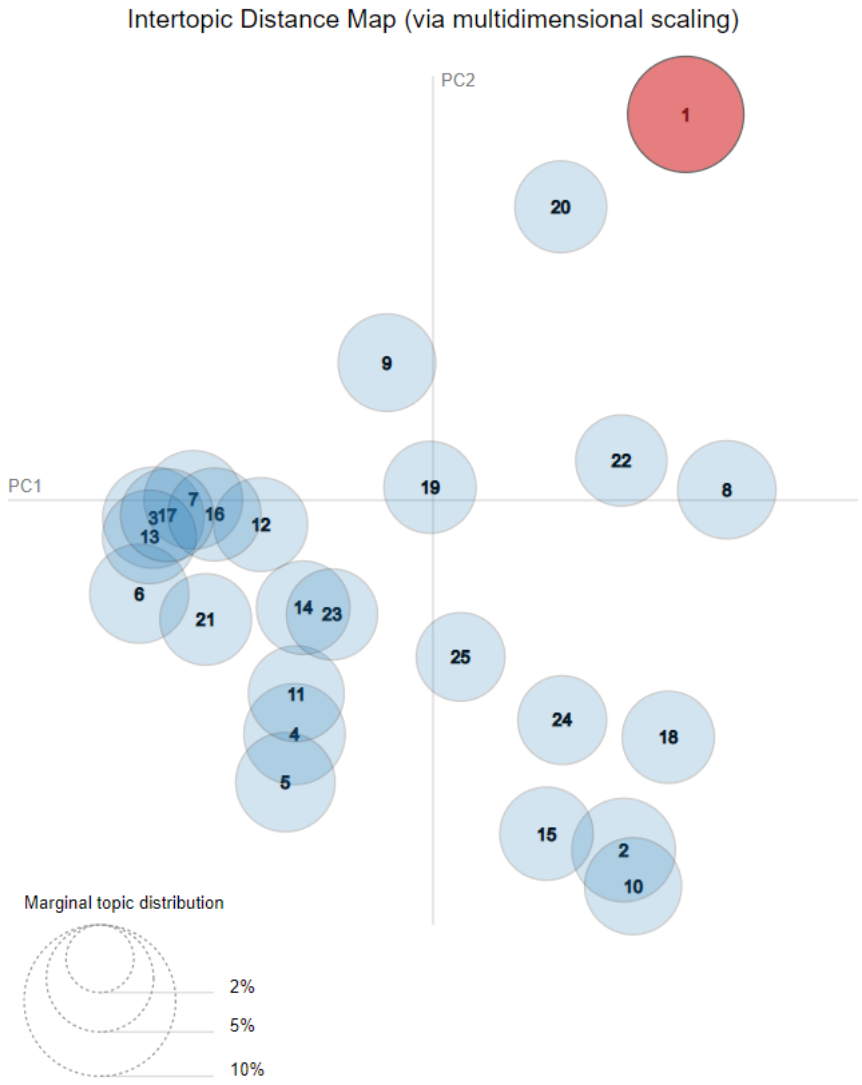


Figure 3.3: Intertopic Distance Map

Here it is really difficult to find a label for this topic. It is unfortunately one of LDA's defaults. We can use it to get some topics present in the database, but with some hard to label ones, it is impossible to have an exhaustive topic list. Therefore, we will only use LDA to get some of the topics of the database, but not all of them since it is nearly impossible to label them all. However, since we know the date of the latest news of the database (april 2019), we can build our 4th category (i.e. the one with news that treat different topics than Grover's database) simply by

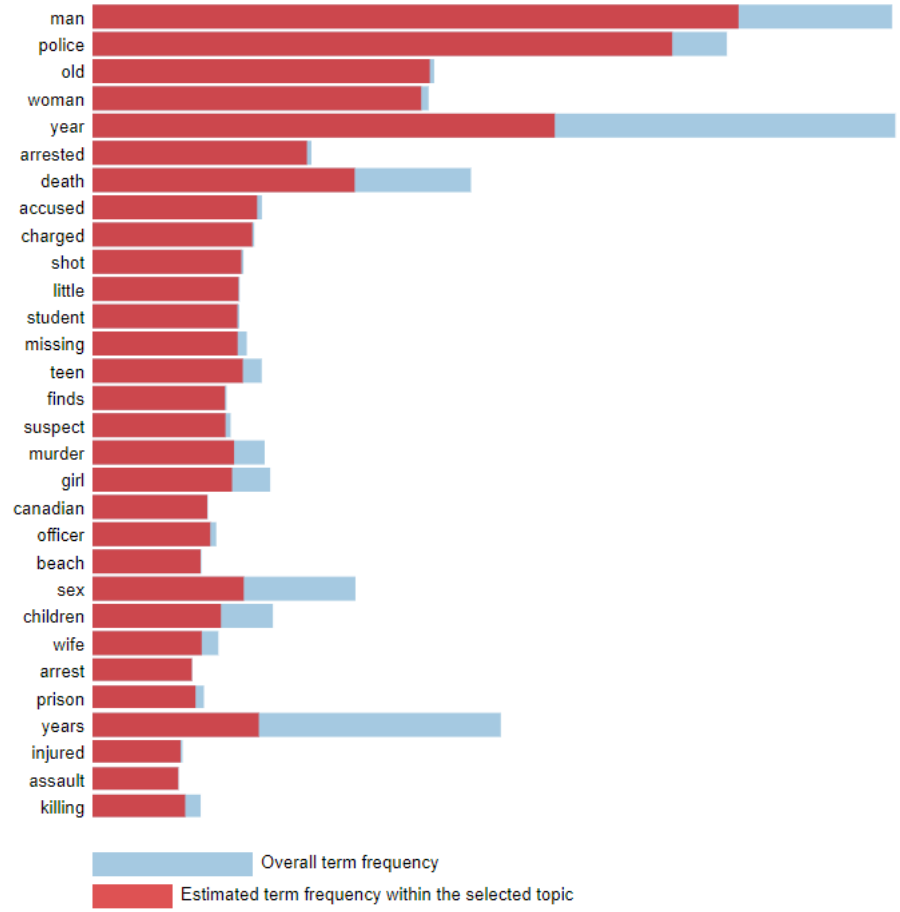


Figure 3.4: Top 30 words for topic 1

choosing news about topics that happened after this date and know that they will not be present in Grover's database. Using LDA to get some topics as well as the top words for each of them, we at least managed to find the following ones: *Black Friday, Facebook, Wall street, Football, American senate, Air crash, Olympic games, Android, Healthcare, Army, Music, Immigration, Christmas, Education* and *Canada*.

3.2.2 Creating our experiment database

Now that we have our news generator and that we know some of the topics it was trained on, we can start building our experiment database. To do it, we need generated and original articles from the 4 categories we already discussed in section 3.1. As a reminder, each category is built respectively:

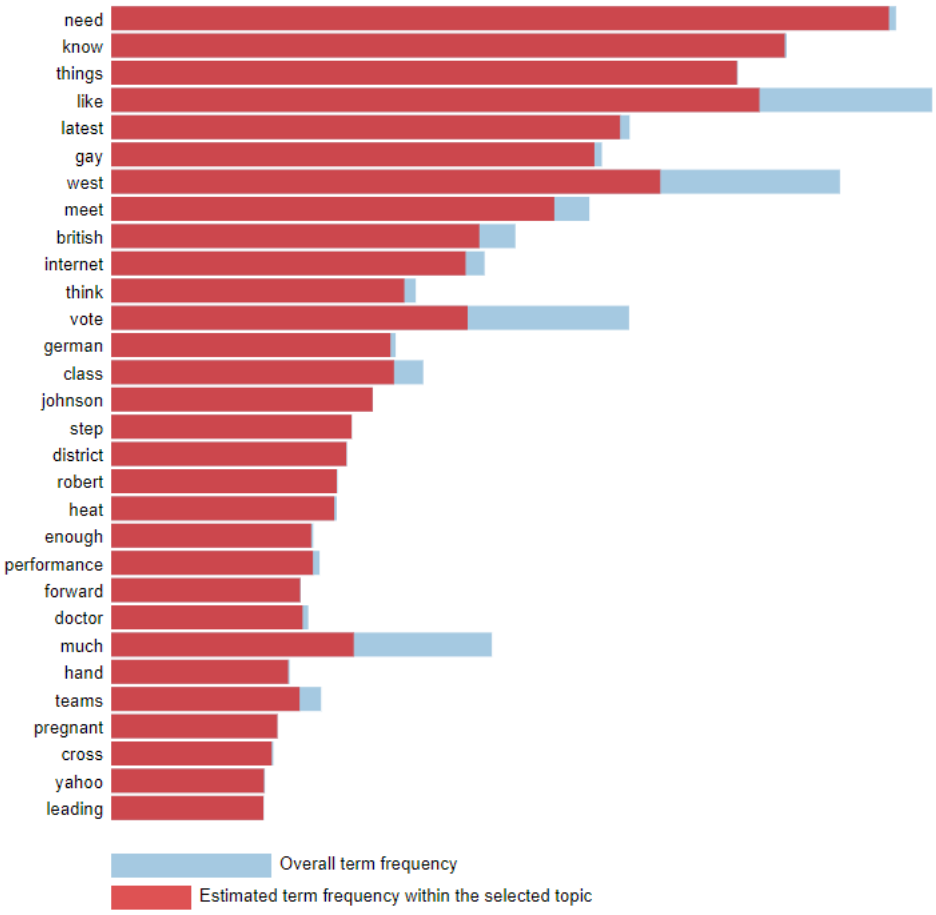


Figure 3.5: Top 30 words for topic 11

1. Using all the data and meta data available from the database for selected entries.

2. Using only the title and the date from the database for selected entries.

3. Using only the title and the date from news not present in the database but treating similar topics at similar dates as the news of the database.

4. Using only the title and the date from news not present in the database and treating topics not present in the database, at dates not covered by the database.

Where "the database" refers to Grover's training database, *Realnews*. In short, it means that we first have to find original news, extract the fields we need for the generation (all of them for category 1 and only the title and the date for the other

categories) and then generate the corresponding news based on these fields using Grover.

Since the topics of the last category have to be different from the ones of the other categories anyway, it was decided to set different topics for each of them [4]. The topics that were chosen were found using LDA as mentioned above and are the following :

- Category 1: *Black Friday, Facebook, Wall street, Football* and *American senate*
- Category 2: *Air crash, Olympic games, Android, Healthcare* and *Army*
- Category 3: *Music, Immigration, Christmas, Education* and *Canada*
- Category 4: *Joe Biden's election, Covid19, Beiruth explosion, Georges Floyd's death* and *Capitol invasion*

Before going further, one must note some key points. First, the topics for the three first categories were chosen arbitrarily and mostly because they were relatively easy to identify using the LDA tool described above. Other topics could have been chosen, as long as they belong to Grover's database. Next, as we saw in Figure 3.3, the topics are not disjoint so it is possible to find titles (and therefore news) composed of multiple topics. If it is the case, the news's assigned topic is simply one among them, and a news is only selected ones. It means that even if their exist, for example, an article talking about *Black friday* and *Android*, it will be assigned to only one of these topics. Finally, the topics for the human and the automated evaluation are the same, but the news are not. This is simply due to the fact that the quantities of news we need for these two evaluations are different and there is therefore no interest in trying to give the same news for both.

The topics for the fourth category were chosen because they happened after April 2019, which is the latest publish_date present in Grover's database. An effort was put on finding quite different topics, but as we need a decent number of original news, they have to be quite popular. We chose to include a topic close from the database's ones (Joe Biden's election that is close from trump's election or us elections in general), but also topics that are most likely really far from what Grover has been trained on (Covid19 for example).

[4] We could argue that since the news are not about the same topics it will be difficult to compare them. To take care of that, we will check latter if the evaluation gives different results for different topics among the same category, or if on the contrary we only get different results depending of the category, but same results for every topic among it.

Now that we know what news we are looking for, we can start getting the original ones. We will then extract some of their information (all of them for category 1 and only the date and the title for the other categories) to generate news with Grover and evaluate their quality. Finding the original news for the two first categories is rather easy, since they can just be collected in Grover's database. The two other categories need a bit of crawling to find original news on the chosen topics. Since the crawling for the human and the automated evaluation were different, they will be described separately on their own section below.

Manual crawling (human evaluation)

The crawling for the human experiment was done manually since we used only 50 original news from each category and crawling 100 (50 for the 2 last categories) news was faster to do manually than automatically [5]. To ensure not having news too far from the database format, a particular attention was put on checking their length and avoiding the news with multiple sections. For the last category, we tried to select titles that help to understand the topic. For example, the titles for the news about *Georges Floyd's death* all have information that help to understand what is the key idea of the topic (i.e that he was black, killed by a policeman and probably completely innocent). The same goes for all the other topics of the fourth category. Finally, we made sure that all the news of the last category were published after may 2019, which was not too difficult since none of the event had actually happened before this date. The website's links for the crawled news used during the human evaluation are available in *jebogaert*'s github repository. Thanks to this crawling and the other 100 original news that come from Grover's database (the news for the categories 1 and 2), we now have 200 original news (50 for each category). We can now extract their title and publish_date (or simply keep all the data for category 1) and use them as input for Grover, by following the code in Appendix A. It will allow us to get 200 generated news as well, 50 for each category.

Automated crawling (automated evaluation)

In the same fashion as the section above, this section will only describe how we got the original news for the two last categories [6] To get them, we used commoncrawl (cf section 2.2) to have enough data to train a machine learning classifier. Since

[5] It is also due to the fact that the human evaluation was done before the automated one and since it was 4 weeks long, it had to be launched quite early. At that time, we were not familiar enough with commoncrawl to collect the news this way.

[6] Since the original news from the two first ones are in Grover's database like for the human evaluation.

commoncrawl stores all the data into warc archives, the first step was to get access to them. After following the steps described in the appendix B that are mostly bash and python commands to open and read the files, one can have access to a tremendous amount of news. We had to filter them to be sure that they respected the conditions we put on the two last categories. To do it, we checked the elements presented below:

- **The article's date**: Since we only want news before (resp. after) May 2019 for category 3 (resp. 4), we checked the date field extracted thanks to the news-please package (available at `https://github.com/fhamborg/news-please`). News without date field were automatically discarded.
- **The title's language**: We used the package pycld2 (available at `https://pypi.org/project/pycld2/`) to automatically detect the language of the news's titles. To avoid any confusion, we only selected news where the only language of the title was English.
- **The text's language** : After being sure that the title was only in English, we also checked that the main text of the news was in the same language.
- **The news's topic**: To avoid complication, the topic extraction was only done using specific words for each topic. The list of these words for every topic can be found on *jebogart*'s github repository. Note that we only checked for the words in the title rather than in the entire news since it is more representative of the topic and less computationally intensive.
- **Possible duplicate**: Since no effort is made by commoncrawl to avoid the duplicates, we had to assess that no title is appearing more than once because it would mean that we got the same news multiple times [7].

Following these steps, we were able to collect 1000 original news for each category. It represents 200 news for each topic [8]. After the generation (cf Appendix A), we have a full database of 8000 news, 4000 original and 4000 generated ones. Since we will mostly train our classifiers on each category separately, we will have 2000 news to train on for each of them. As a reminder, our goal is not to reach

[7] Of course, this method is not perfect since two news could be the exact same ones but with a different title. However, comparing every news would have been too computationally intensive and could have been prone to the same errors. News where only the authors or the newspaper's name at the end differs would also have been considered different while they are actually not. Therefore, we consider our method to be sufficient to build our database.

[8] It may seem a bit short, but it is due to long crawling and generation time. On top of that, we wanted to have the same number of news for each topic and some of them are more difficult to find news about

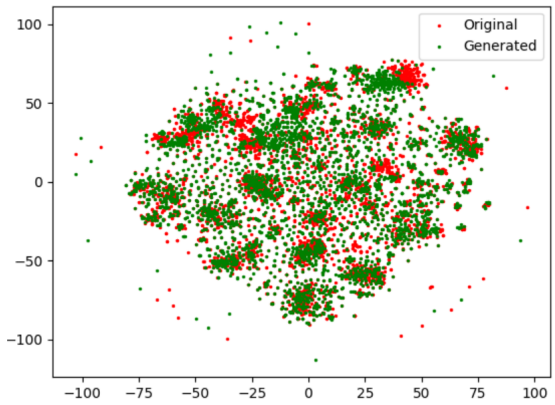


Figure 3.6: Scatter plot of our automated evaluation database after dimensionality reduction using t-SNE

state-of-the-art results, but rather to see if there is a difference in the classification accuracy between news of different categories. Since it is often useful to visualize a database before running machine learning models on it, we can visualize our full database using t-SNE (cf section 2.4.1) to see if the news are similar from each other. Figure 3.6 shows a scatter plot of the whole database after dimensionality reduction.

We can see that the original and corresponding generated news tend to be close from each other [9]. There are some outliers, but the big majority of the news are in the middle of the plot. We can also see that on the top of the plot, we can distinguish the red and green clusters more than in the other areas. We can also visualize the news for each category separately. Table 3.1 shows a scatter plot after dimensionality reduction for each category.

We can see that the news generated for the 4th category are further from their corresponding original ones that for the other categories. It is already a first interesting observation, as it mean that the two classes (generated/original) should be more easily separable for the 4th category than for the other ones. Finally, we can take a look at a scatter plot showing all the original or generated news by category. Table 3.2 shows such plots. We can observe than every category is split in 5 clusters, more or less distinct depending of the category, probably corresponding to the different topics.

[9]Note that since the green points where plotted after the red ones, seeing green points with a red border means that the generated and original news were plotted nearly at the exact same place.

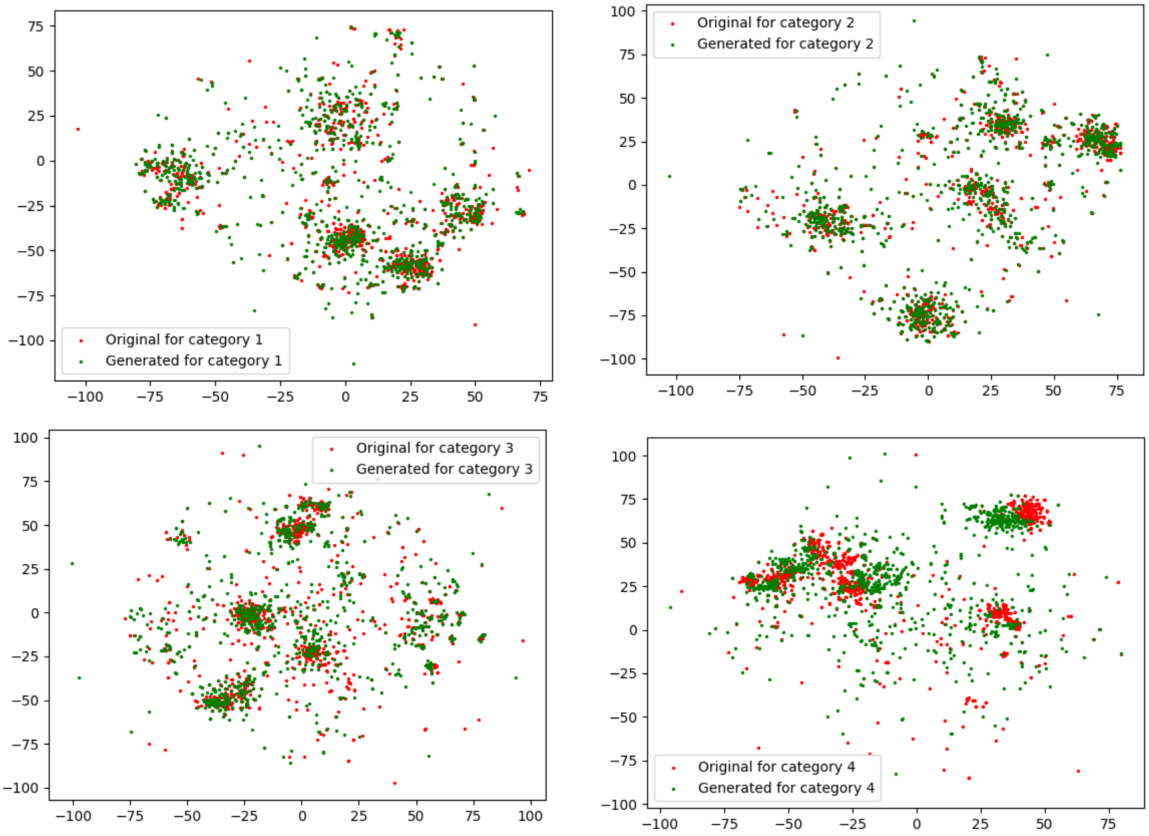


Table 3.1: Scatter plot after dimensionality reduction using t-SNE for each category

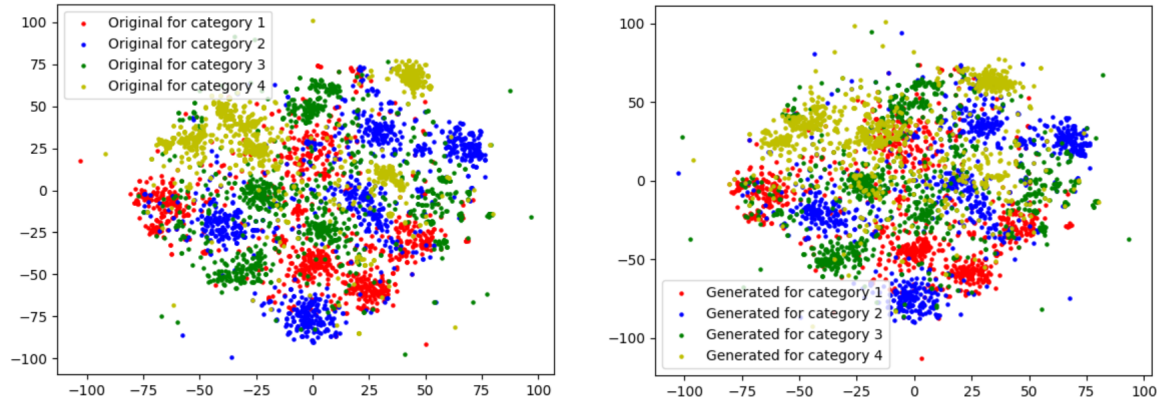


Table 3.2: Scatter plot after dimensionality reduction unsing t-SNE for original (left) and generated news (right)

3.3 Human evaluation

To assess the quality of the news generator, we setup a human experiment. By generation quality, we mean the similarity between a generated news and the corresponding original one. Therefore, we consider that a news of good quality will be hard to identify by human beings, while a news with poor quality will be classified easily as generated. It follows that the goal of the experiment is to collect human predictions about the news's label of our previously created database. The following sections will present all the setup that was done to have an experiment as useful as possible.

3.3.1 Experiment plan

The problem in human experiments is that many effects can superpose themselves and add a lot of noise to what we want to observe. In our case, since the news are from different categories, on different topics and are possibly also written with different styles, when someone answer that he thinks a news is generated, it is difficult to know exactly why he thinks it. On top of that, some effects are not controllable, like how two different persons perceive the same news, what is their mindset when they evaluate it, what is their prior knowledge of the topic, etc. Therefore, to have the maximum chances to be able to draw conclusions on the experiment, we can't give random texts to random users and observe the results because we would probably just observe a big cloud of noise. To have a meaningful human evaluation, a precise experiment plan was built to avoid superposing effects as much as possible.

First of all, to avoid having to big of an individual effect, we assumed that it was better to ask the same person multiple times rather than asking different persons every time. We therefore tried to get participants that would accept to evaluate a certain amount of news[10] We managed to get 30 participants for the experiment. To be able to see if prior knowledge is impacting, we distributed them in 3 groups of 10 persons. The first group is composed of people from social sciences, the second is composed of students in computer sciences and engineering, and the third one is composed of computer sciences and engineering professors and assistants. Since we have 400 news (50 original and 50 generated for each category) to evaluate, it means that every person will be in charge of 40 news. Because of time constraint for the participant, we had to split the experiment in 4 weeks. To

[10]Note that since we want to get evaluations on all our news, decreasing the number of participant increases the number of news every participant has to evaluate. We therefore had to go for a trade-off between the number of participants and the number of news each of them evaluates.

summarize, every of the 30 participant reviewed 10 news every week during 4 weeks.

But we also have to be sure that these participants all receive news with similar characteristics to assess. Otherwise, we could draw conclusions only based on who reviewed the news rather than on the news's characteristics themselves. To understand what precaution we took, it is first important to now how to identify every news in our database. Every news can be identify as a tuple (A, B, C), where A is the topic, B is its number inside the topic (from 1 to 10), and C is either G for generated or O for original. For example, the news (3, 7, G) is the generated version of the 7th news of the 3rd topic. Knowing that, we can now express easily the precaution we took:

- All 3 groups got the exact same partition of news. It means that the first person of group 1 reviewed the same news as the first person of group 2 and 3 and that they got the same news on the same weeks.
- Every user received exactly 2 news from every topic, one original and one generated. It follows that every user received exactly 10 news from every category, 5 original and 5 generated. It also follows that every user received 20 generated articles and 20 original ones during the whole experiment.
- No user can get the generated news and the original one on the same topic during the same week. It follows that every week, a user received news on 10 different topics.
- No user can receive the generated news and the corresponding original one. It follows that every user will assess news with 40 different titles and publish dates.
- For every group, the total number of news of each category is the same every week (25).
- No participant knows the partition of news, the topics, the news generator nor the different categories. No participant knows his number among his group, and there is no way to know information about other participants. The only available information is that there are original and generated news, but no precision on the ratio was given.

This experiment plan allowed to minimize the variance when possible and to randomize when it was not. All these precautions will allow us to get comparable results for the different variables, and to separate the different controlled source of variability. For the uncontrolled source, we have to assume that they are randomized so that they are not superposing. The results of the human experiment are presented in section 4.2.

3.3.2 Website and collected information

To be able to collect the human evaluations, a website was created. It reused the source code of an old project, done by Clément Massart, that was modified a bit for our needs. The website (available at this link `https://digital-journalism.elen.ucl.ac.be/`) is composed of different pages. Among them, a welcome page which contains a short explanation of the experiment, a log-in page and the main page, the news one. All the pages not presented in this section are available in the appendix C. This section will mostly detail the page that allowed us to collect the news's quality evaluation, the news page. It is composed of 10 news that change every week during the experiment's period. Figure 3.7 shows a screen shot of the page.

Figure 3.7: Main page of the website

When clicking on one of the news, one can see the main text but not the title of

the news immediately. It was done to collect an evaluation before and after seeing the title to check if the title has an impact on the user's perception of the news. Figure (3.8) shows a screen shot of such a page.

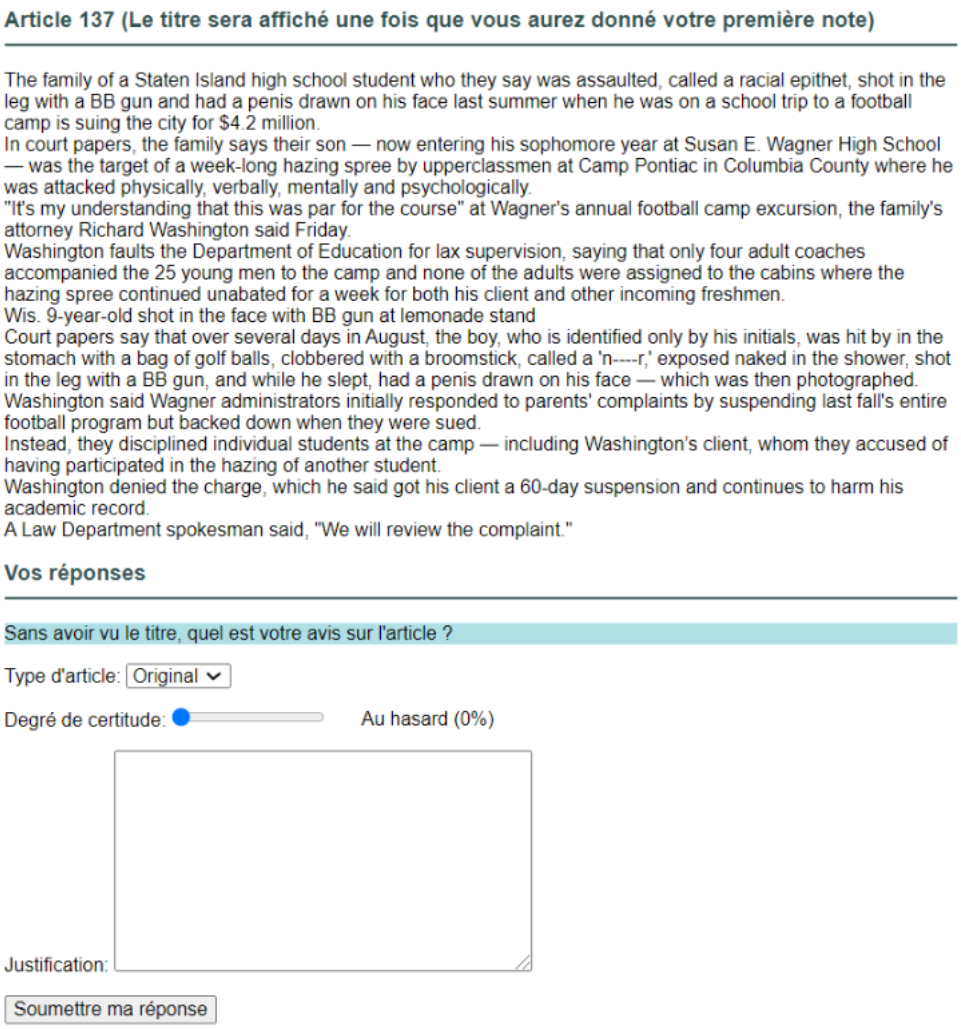


Figure 3.8: Evaluation page before seeing the title

As seen in the picture, the participants are asked to give a label to the news, either generated or original, and to give a confidence level for their choice. The possible confidence levels are the following :

- At random : 0-10%

- Really no sure : 11-30%
- Not sure : 31-45%
- Mitigate : 46-55%
- Sure : 56-70%
- Really sure : 71-90%
- No doubt : 90-100%

The user can also justify its choice with raw text. This justification will allow us to check the motivation of the participant's choice and to understand what makes a well generated news in a qualitative way as well. The first time a participant is asked to give an answer, he does not see the title. After validating his first answer, he can see the title and re-evaluate the news. Figure 3.9 shows the page when seeing the title. He can modify this second answer as much as he wants, but the first one can't be modified [11].

As said in the first part of this section, the 10 news were changing every week for each of the 30 participant and the experiment lasted 4 weeks. It means that in total, 1200 answers were collected using this website and each of them is done before and after seeing the title.

3.4 Automated evaluation

In parallel of the human experiment, an automated evaluation was done to check if the two evaluations would give the same results, or if, on the contrary, they would show totally different things. This section will show everything that was done to setup the automated evaluation, from the pre-processing to the models that were tried.

3.4.1 Preprocessing

As usual in machine learning task related to text [18], we start by pre-processing the data. Every steps we did are presented in section 2.3.1. The only thing we had to do first was to encode the text as utf-8 to avoid having unicode character in our

[11] This precaution was taken to avoid having access to the title and then go back to answer the "no title" part of the evaluation

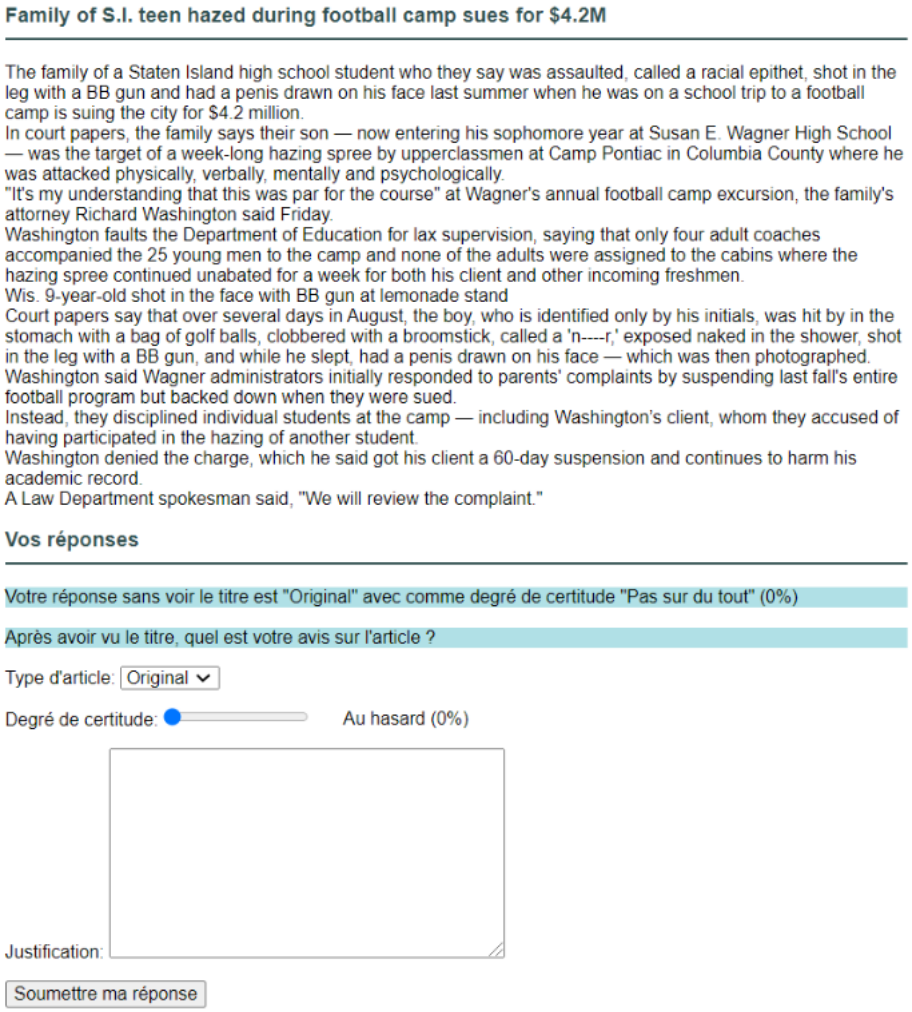


Figure 3.9: Evaluation page after seeing the title

inputs. We then tokenized, POS-tagged, cleaned and lemmatized the main text of every news. Since the vectorization method is model specific, it was done after the pre-processing for each model and it will be presented separately in the next sections.

3.4.2 Models and their tuning

Note that each of these models was tuned using a gridsearch[12] and with a 5 fold cross validation (ref background).

[12] A gridsearch is the method consisting in computing the results for all the possible combination of suggested parameters and keeping the one that gives the best results.

Logistic regression

As already said in the background section (ref background), the logistic regression model is a staple in text analysis [34]. We first started by applying the common vectorization method for it, tf-idf [35] using the TfidfVectorizer() function. We then fit a logistic regression model using the LogisticRegression() function. Both these functions come from the sklearn packages. To tune the whole model at once, we built a pipeline that allowed to do a gridsearch on every parameters at the same time. For the vectorizer, the following parameters were tried :

- **use_idf**: Whether we should use the inverse document frequency(idf) factor or not. If not, only the term frequency factor is used. We tried both true and false.
- **max_features**: The number of words to consider in the vocabulary. If the corpus contains more than max_features different words, they are first sorted by number of appearance. All the words after the max_features position are then replaced by the "OOV" (out of vocabulary) token. This is mainly done to avoid considering the words that appear few times in the corpus and therefore prevent overfitting. We tried the values 2500, 3500 and 5000.
- **max_df**: Allows to ignore terms that have a document frequency strictly higher than the given max_df. It prevent from using words that appear in a lot of documents. We tried the values 0.85, 0.9 and 0.95.

For the logistic regression, the following parameters were tried :

- **penalty**: Whether we should use the l1 or l2 penalty, as defined in section 2.6.3. It is also possible to take a linear combination of the both penalty by using 'elasticnet'. We tried the values 'l1', 'l2' and 'elasticnet'
- **C**: The inverse of the regularization strength. Smaller values specify stronger regularization. We tried the values 1.0, 0.95, 0.9, 0.6 and 0.3
- **max_iter**: The maximum number of iterations given to the solver to converge. We tried the values 250, 500 and 1000
- **l1_ratio** : Only used when the 'elasticnet' penalty is set. Setting this parameter to 0 is equivalent to using the 'l2' penalty, while setting it to 1 is equivalent to using the 'l1' penalty. In all the other cases, it is a combination of the l1 and l2 penalties. We tried the values 0.1, 0.5 and 0.9

Long Short Term Memory

LSTM is also a really common model in text analysis [36] [37]. To be able to run it, we first modified our pre-processed data in a particular way. Since the model works on sequences of the same length, we had to equalize the length of all the news. We did it by removing the words after the nth one for the too long news, and by adding zeros at the end for the too short ones. This is called padding and we chose a value of 200 for the sequence's length because it is was showed the best results. Once padded, we can feed the sequences to the LSTM model. The background section (ref background) explains briefly how the model works. Our architecture is composed of the following layers :

1. **Embedding layer**: The layer that transform our sequences to vectors using word embedding as presented in section 2.3.3.

2. **Bidirectional LSTM layer**: The actual Bi-LSTM layer, as presented in the section 2.6.4.

3. **Dropout layer**: A layer that helps to prevent over-fitting by dropping out some neurons during training. More precisely, at each training stage, individual nodes are either dropped with probability 1-p or kept with probability p. It follows that some neurons (chosen at random with probability p) will not be trained at certain stages. Incoming and outgoing edges to a dropped-out node are also removed.

4. **Dense layer with relu activation**. A layer where all the neurons are linked to all the neurons of the previous and the next layer. The relu activation function [38] means that a neuron will be activated with an input value only if this input is greater than 0.

5. **Dense with softmax activation** This serves as our output layer. The softmax activation function allows to have the prediction for every class[13].

Note that there exists other possible architectures. Recent paper reached state-of-the-art results in text classification using convolutionnal layers [37] and attention mechanism [39] for example. We chose to opt for a quite simple architecture because we are more interested in comparing the results for the different categories than getting the best possible results.

We tried the following parameters to fine tune the model:

- Embedding size: The size in the embedding space as explained in section 2.3.3. We tried the values 20, 25, 35, 50, 100, 150 and 200

[13]Two in our case since we only have the 'generated' and the 'original' classes.

- Batch_size: The number of training examples used in each iteration. We tried the values 1, 5 and 10
- Epoch: The number of passes of the entire training dataset that needs to be completed before stopping the learning. Note that when the callback parameter is active, it is possible to stop the training before the last epoch. We tried the values 3, 5 and 10.
- Callback: The callback is the fact to stop training the network after a certain condition is respected. In our case, this condition was the fact that the error on the test set was increasing. It means that we stopped training the network when the test error was raising to prevent over-fitting. We tried both with and without this callback parameter.
- Dropout rate: The probability to deactivate the neurons in the dropout layer. We tried the values 0.1, 0.2, 0.3, 0.4 and 0.5
- Loss function: The function used by the network to see how far from the expected predictions are its current ones. We tried using the binary cross entropy as well as the sparse categorical cross entropy, both defined in section 2.5.4

Part 4

Results and analyse

This section details all the results we obtained, first for the automated evaluation and then for the human one. It also contains an analysis to answer the two questions asked during the first part of the work : "Is the quality of the news of the 4th category worst than for the other ones ?" and "If there exists such a quality difference, is it observable by both human and machine learning tools?"

4.1 Automatic evaluation results

This section presents the best parameters and the accuracy value we obtained for these parameters. It also analyzes the learning curves to try to answer our first question in the best possible way.

4.1.1 Best parameters

For the logistic regression model, the best parameters are different for every category. They will be presented all at once in figure 4.1.

We can briefly observe that they are close from each other. Remember that the value of these parameters have little importance as the goal of the fine tuning was mostly to assess that we are close to the best accuracy we can reach with the model for each category.

For the LSTM model, we chose to not include the best parameters for every category since they were very different from one curve to another, and it does not make much sens. However, we reached at least the second best accuracy for every category when using the same set of parameters, which is detailed below. Therefore,

Parameter \ Category	1	2	3	4
C	1	1	1	1
l1_ratio	0.1	0.1	0.1	0.1
max_iter	500	500	500	500
penalty	l1	l1	l1	l2
max_df	0.95	0.95	0.95	0.95
max_features	5000	2500	2500	5000
use_idf	False	False	True	True

Table 4.1: Best parameters for each category for the logistic regression model

we chose to use these same parameters for each category in the LSTM model.

- Embedding size : 25
- Batch size : 5
- Epoch : 5
- Callback : Yes
- Dropout rate : 0.3
- Loss function : Sparse categorical cross entropy

One interesting observation we can do is the low embedding size compared to what is usually done [40]. It is possibly due to our quite low number of news.

4.1.2 Accuracy

We can check the accuracy we were able to reach with these parameters. Figure 4.1 and figure 4.2 show the learning curves for the logistic regression and the LSTM respectively. The learning curve represents the accuracy obtained on the test set when using only x examples during the training. The values are presented in table 4.2 and 4.3 for respectively the logistic regression and LSTM.

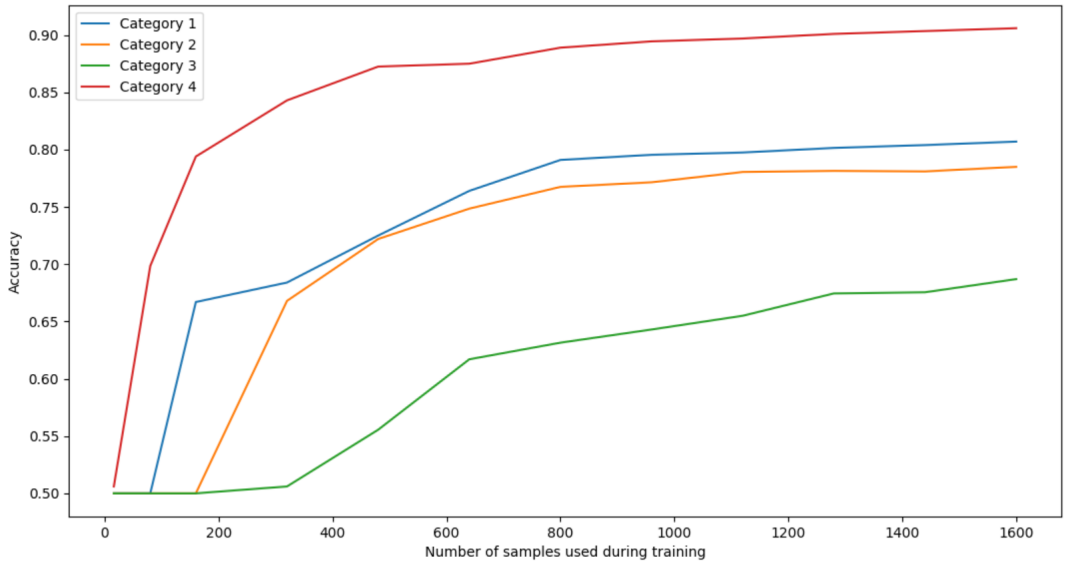


Figure 4.1: Learning curve for the logistic regression

	16	80	160	320	480	640	800	960	1120	1280	1440	1600
Category 1	0.50	0.50	0.67	0.68	0.73	0.76	0.79	0.80	0.80	0.80	0.80	0.81
Category 2	0.50	0.50	0.50	0.67	0.72	0.75	0.77	0.77	0.78	0.78	0.78	0.79
Category 3	0.50	0.50	0.50	0.51	0.56	0.62	0.63	0.64	0.66	0.67	0.68	0.69
Category 4	0.51	0.70	0.79	0.84	0.87	0.88	0.89	0.89	0.90	0.90	0.90	0.91

Table 4.2: Accuracy values corresponding to the learning curves of the logistic regression for the 4 categories

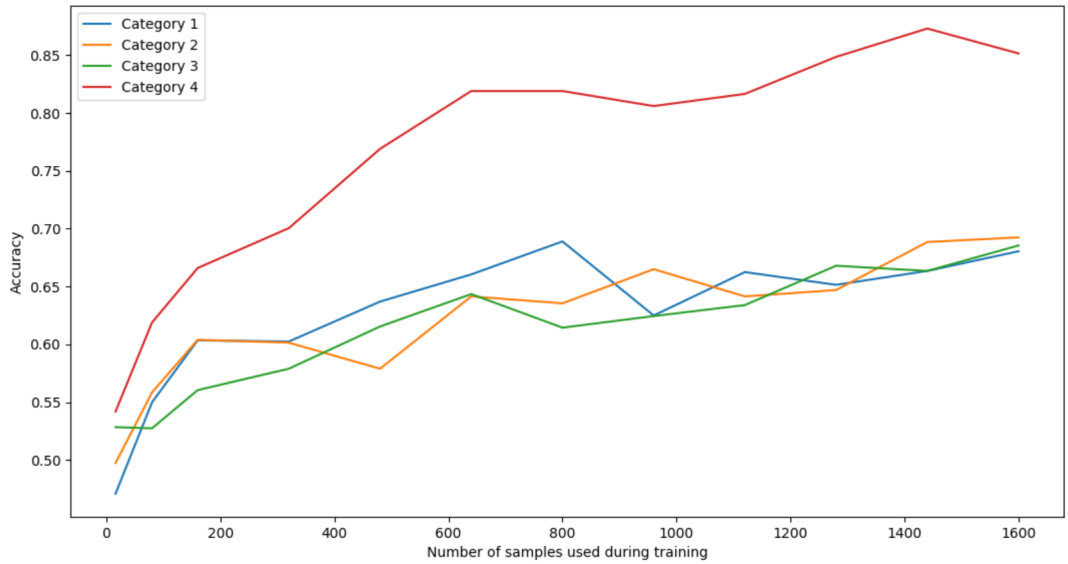


Figure 4.2: Learning curve for the LSTM

	16	80	160	320	480	640	800	960	1120	1280	1440	1600
Category 1	0.47	0.55	0.60	0.60	0.64	0.66	0.69	0.63	0.66	0.65	0.66	0.68
Category 2	0.50	0.56	0.60	0.60	0.58	0.64	0.64	0.66	0.64	0.65	0.69	0.69
Category 3	0.53	0.53	0.56	0.58	0.62	0.64	0.61	0.62	0.63	0.67	0.66	0.69
Category 4	0.54	0.62	0.67	0.70	0.77	0.82	0.82	0.81	0.82	0.85	0.87	0.85

Table 4.3: Accuracy values corresponding to the learning curves of the LSTM for the 4 categories

We can see that classifying news of category 4 leads to way better results than for the 3 others. To be sure that the difference is significant, we can compute a confidence interval by using the following formula:

$$z * \sqrt{\frac{x * 1 - x}{n}} \tag{4.1}$$

where z = 1,96 if we use a 95% confidence interval and n = 400 since we used 400 test sample to compute the accuracy[1]. Table 4.4 shows the confidence interval for the last accuracy value (i.e the accuracy using 1600 training samples) of each category. Table 4.5 shows a more visual illustration of the different intervals

[1]Note that this formula is only useful when assumed that the accuracy follows a Gaussian law

Model / Category	Logistic regression	LSTM
Category 1	0.77 - 0.85	0.63 - 0.73
Category 2	0.75 - 0.83	0.64 - 0.74
Category 3	0.64 - 0.74	0.64 - 0.74
Category 4	0.88 - 0.94	0.82 - 0.88

Table 4.4: Confidence interval for the last accuracy value in each category.

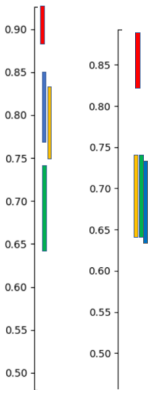


Table 4.5: Visual illustration of the confidence intervals for the 4 category, for the logistic regression (left) and for LSTM (right)

We can see that the difference in accuracy is significant for category 4 in both the LR and the LSTM model. It confirms the observations we made in section (ref analysis of the db) on the database where we saw that the news of category 4 tends to be further from their corresponding original news than the news of every other category. It also answer our first question: **Yes, the quality of the news of the 4th category is worst than for the other ones.** when evaluating the news in an automated way. Surprisingly, category 3 is significantly harder to classify when using a logistic regression. It is even more surprising as the LSTM does not seem to show the same results concerning category 3. It could be due to difference between the two vectorization method used for the models.

4.2 Results of the human evaluation

Thanks to the human experiment that was set up in section 3.3.1, we managed to collect 1200 answers. As a reminder, all of these answers where collected with and without the knowledge of the title to check if it impacts the prediction. The next

sections will detail and analyse the obtained results to check if we can made the same conclusion than for the automated evaluation.

4.2.1 Quality of the classification

When analyzing the results of the experiment, the first thing we want to do is to check if there is a difference in the answers for the 4 categories. To start answering that question, we will first compute two metrics: the accuracy and the "weighted accuracy"[2] for each category.

Accuracy

As presented in section 2.5.2, the accuracy is defined as the fraction of correctly classified news among the total number of news seen. We start by computing that metric for all the news, to check if the participants are often answering correctly. We can also separate the accuracy in two parts, namely on the original and on the generated news, to see if people tend to classify better one of the two classes. Table 4.6 shows the accuracy value obtained.

	Total	Original	Generated
Without title	0.706	0.777	0.647
With title	0.712	0.793	0.62

Table 4.6: Accuracy obtained on the different part of the dataset, both before and after seeing the title

This table shows us that knowing the title is not helping to classify the news correctly. It is also showing that the accuracy on the generated news is lower than on the original ones. It means that it is more common to think a generated news is generated than the opposite. Since the accuracy on the generated news is quite low, it means that they are quite difficult to detect and that their quality is therefore not too bad on average.

We can now check how big is the accuracy for each category. Since the title does not seem to make any difference, from now on this section will only present the results when not knowing the title. Table 4.7 shows the table of accuracy by category for all, generated and original news

[2]We use this term to design an accuracy metrics where the value obtained is multiplied by the coefficient of certainty

Dataset / Category	Total	Original	Generated
Category 1	0.673	0.773	0.573
Category 2	0.7	0.767	0.633
Category 3	0.72	0.787	0.653
Category 4	0.753	0.78	0.726

Table 4.7: Accuracy obtained for different categories and on different part of the dataset

We can clearly see that the accuracy on the generated news is increasing, while the one for the original news tends to be constant. This is a key point for two reasons: First, it respects our intuition that as we try to generate news in more and more difficult contexts, the quality of the results becomes worst. But it also shows that the variation in accuracy is not due to the topics themselves, or the accuracy would have changed for both the original and the generated news. In the same fashion as what was done for the automated part, we can check the confidence intervals to see if the difference is significant or if the tendency could be due to some random effects. Table 4.8 shows the value of the lower and the upper bound when computing the confidence interval with the same equation as for the automated section. Figure 4.3 shows a visual illustration of the confidence intervals:

Category	
Category 1	0.58 - 0.76
Category 2	0.61 - 0.79
Category 3	0.63 - 0.81
Category 4	0.67 - 0.84

Table 4.8: Confidence interval for the accuracy value in each category for the human evaluation

We can see that, on the contrary from the automated part, the difference is not significant. It does not mean that the difference is not existing, but only that we can't prove that it exists with a sufficient certainty with the data we have. The differences could appear due to some random effects or some experiment parameters. This effect is probably strengthened by the fact that each of these accuracy value is obtained by computing the average value over only 100 answers. Since the range between the lower and the upper bounds is inversely proportional to $\sqrt{n}$ (where n is the number of answers), it is possible that our experiment was evaluating too few news.

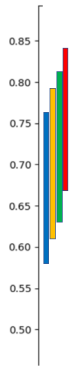


Figure 4.3: Visual illustration of the confidence interval for the accuracy obtained during the human evaluation

Weighted accuracy

Because it can also be interesting to see if the certainty of people is evolving, table 4.9 shows the weighted accuracy score for the whole dataset and for original and generated news alone:

	Total	Original	Generated
Without title	33.478	44.293	22.662
With title	30.442	45.32	15.563

Table 4.9: Weighted accuracy on the different part of the dataset

Again, the presence of the title doesn't seem to impact the weighted accuracy. The following results will therefore only present the weighted accuracy before seeing the title. We can see that the metric is lower for generated news than for other ones. We can check in Figure 4.10 what we obtain when computing it for every class :

Dataset / Category	Total	Original	Generated
Category 1	30.003	45.153	14.853
Category 2	30.587	43.56	17.613
Category 3	32.693	42.847	22.54
Category 4	40.627	45.613	35.64

Table 4.10: Weighted accuracy on the different part of the dataset and for different categories

Again, we can see that the weighted accuracy is raising for news on category 4. Not only are people better detecting the generated news of that category, but they are also more sure about their choice. The results are too close to be significant, but still, we observed two tendencies (for the accuracy and for the weighted accuracy) going in the same direction and following our basic intuition. The next section will investigate the experiment parameters to try to give further information about these tendencies.

4.2.2 Independence

Since we saw that the difference between the categories was not significant, the observed tendency could be due to another parameter than the category. This section aims at verifying that there was no parasitic effect that would have significantly influenced the results. If we don't find such an effect the logical conclusion would be that if the tendencies observed in the two above section really exist[3], then the most plausible explanation is that they are due to the way the news were generated[4], since all our observations lead to that conclusion.

All of the following sections analyse the possible dependencies between the participant's prediction and one of the experiment parameter. The independence is tested via a chi square test [41] outputting a p-value. That p-value tells us whether or not we should reject the null hypothesis. In our case, the null hypothesis is "the answer and the variable x are independent". It follows that having a low p-value (< 0.05) means rejecting the null hypothesis and therefore assuming that the participant's prediction is dependant of the variable. If the p-value is higher than 0.05, we have no reason to say that the answer is dependant from the tested parameter[5].

As the p-values were computed on cross tables corresponding to the evaluation of participants from different groups on different categories, topics, etc., each section will present tables with the p-value for each combination of possible parameters. On top of that, since every evaluation was collected twice, once before and once after seeing the title, we will present two such tables every time. To avoid confusion, here is a little example. If we want to check the independence between the real and

[3]That would mean that they are not due to some random effects

[4]Note that in such a conclusion, we don't know whether the tendency exists but is judged as not significant with the collected data, or if it doesn't exist and is simply due to random effects. We can only say that if it exists, the most logical explanation is that it is due to the difference between the categories

[5]Note that having a p-value > 0.05 does not proves that the variables are independent, it just doesn't statistically proves that they are dependent.

predicted label, we could have different results for each category and for each group. It represents 12 possible cases, and we have a different cross table for each of them. The first (resp. last) one corresponds to the evaluation of news of category 1 (resp. 3) by the participant of group 1 (resp. 3) for example. By computing its p-value, we could say whether or not the predicted label is independent of the expected one (i.e. being independent would mean that the prediction are distributed as they would have been if the participants answered randomly) for the participants of group 1 (resp. 3) on news of category 1 (resp. 3).

We chose to show the results this way to compare the possible differences between different values of one parameter when the other is fixed. With this notation, we can also compare the p-values when the user knows the title and when he doesn't to see if it changes something. Finally, note that this part does not make use of the coefficient of certainty given during the experiment, but only uses the participant's binary predictions (original or generated).

Between answer and label

This part allows us to check if people answered dependently of the real label of the text. The answer (A) is the label the participant gave to the news, and the label (L) is its real label (i.e original or generated). Obviously, we hope that these two variables are not independent or it would mean that people are answering at random. If that is the case, it can either mean that the news are really well generated, or that the participants did the evaluation really poorly and we would have no way to know in which case we are. Table 4.11 shows the format of the cross table comparing the actual and the expected predictions.

Label / Answer	Generated	Original	All
Generated	a	b	a+b
Original	c	d	c+d
All	a+c	b+d	a+b+c+d

Table 4.11: Contingency table format to evaluate the dependency between answer and real label. Since every group evaluates 50 generated and 50 original news of each category, a+c and b+d are both equal to 50.

Table 4.12 shows the p-values obtained for each combination of group and category when the user does not know the title. Table 4.13 shows the p-values obtained for similar combination when the user knows the title.

Group \ Category	Category 1	Category 2	Category 3	Category 4
Group 1	$1.530 * 10^{-1}$	$4.339 * 10^{-2}$	$2.679 * 10^{-3}$	$1.371 * 10^{-3}$
Group 2	$3.341 * 10^{-7}$	$1.062 * 10^{-6}$	$3.16 * 10^{-6}$	$1.519 * 10^{-6}$
Group 3	$5.202 * 10^{-4}$	$3.935 * 10^{-6}$	$3.341 * 10^{-7}$	$2.970 * 10^{-11}$

Table 4.12: P-values computed by a chi square test. The test was done on cross tables built with the answers from different groups on different news categories. The answers were collected before the participants saw the title.

Group \ Category	Category 1	Category 2	Category 3	Category 4
Group 1	$6.079 * 10^{-2}$	$6.371 * 10^{-2}$	$4.891 * 10^{-3}$	$1.230 * 10^{-3}$
Group 2	$5.196 * 10^{-6}$	$1.062 * 10^{-6}$	$2.092 * 10^{-5}$	$6.048 * 10^{-8}$
Group 3	$1.543 * 10^{-3}$	$1.579 * 10^{-6}$	$3.162 * 10^{-6}$	$7.704 * 10^{-11}$

Table 4.13: P-values computed by a chi square test. The test was done on cross tables built with the answers from different groups on different news categories. The answers were collected after the participants saw the title.

We can first notice that group 1 tends to be more mitigated than the two other groups, which could mean that the predicted label is also group dependant. Actually, all the p-values are below 0.05[6] except for the first group on category 1[7]. This is the case both when the participants have seen the title and when they didn't. Since this result is a bit surprising, we can analyse the cross tables of group 1 for news of category 1. The first value in every cell is the one before seeing the title, the second is the one after seeing it.

Answer \ Label	Generated	Original	All
Generated	24 | 23	16 | 13	40 | 36
Original	26 | 27	34 | 37	60 | 64
All	50 | 50	50 | 50	100 | 100

Table 4.14: Contingency table between answer and real label for category 1 and group 1 with and without seeing the title

[6] 0.05 was the limit to determine whether we consider the two variables as dependent or not.

[7] Note that having a really low p-value does not mean that the participants are classifying correctly. For example, if a group missclassifies all the news of one category, we get a cross table with a and d equal to 0. Doing an chi square test on such a table would give the lowest possible p-value while all the news are missclassified.

We can indeed see that the participants from group 1 have difficulties to classify the generated news of category 1. It leads to a distribution of predicted labels close to 50/50 among them, and therefore to a p-value higher than usual. Beside this particular case, all the p-values being lower that 0.05, we can assume that the participants are not answering independently of the real label of the news. We will therefore consider that the answer is dependent from the real label.

Between groups

The group is the group among the one the user was placed to evaluate the news. Since the 3 groups were composed by putting people from the same domain of work together, it is possible that it has impacted our evaluation. As a reminder, every group is composed of 10 persons, and each group is reviewing the same news on the same weeks.

Since the participants have a different prior knowledge[8], it can be interesting to see if that is something that can make a difference during the evaluation. Since we want to check if the link between the answer and the label, A ∩ L, is the same[9] for each group, our cross table format is the one presented in table 4.15, where A is the answer and L is the Label:

Group / A ∩ L	Group 1	Group 2	Group 3	All
(Gen, Gen)	a	b	c	$\sum$ (Gen, Gen)
(Gen, Orig)	d	e	f	$\sum$ (Gen, Orig)
(Orig, Gen)	g	h	i	$\sum$ (Orig, Gen)
(Orig, Orig)	j	k	l	$\sum$ (Orig, Orig)
All	$\sum G_1$	$\sum G_2$	$\sum G_3$	$\sum$

Table 4.15: Contingency table format to evaluate the group dependency of link between answers (A) and labels (L). Since every group evaluates 100 news of each category, $\sum G_1$,$\sum G_2$ and $\sum G_3$ are all equal to 100. The sum of all the elements, $\sum$, is therefore equal to 300

Table 4.16 shows the p-values obtained for each category when the user do not

[8]Their prior knowledge is assumed to come from their domain of work, but that is not completely correct since two people working in the same domain could have very different prior knowledge. However, we focused on what could be called "the prior knowledge shared by all the group members" that we named "the prior knowledge" to keep it simple.

[9]We actually don't even look at how dependent are the expected and the predicted label, we just want to check if the dependency between these two variables is the same among each group

know the title. Table 4.17 shows the p-values obtained for each category when the user knows the title.

Category	Category1	Category 2	Category 3	Category 4
	$2.407 * 10^{-1}$	$3.556 * 10^{-1}$	$4.958 * 10^{-1}$	$1.394 * 10^{-1}$

Table 4.16: P-values computed by a chi square test. The test was done on cross tables built with the combination of Answer and Label for different groups. The answers were collected before the participants saw the title.

Category	Category1	Category 2	Category 3	Category 4
	$6.124 * 10^{-1}$	$8.898 * 10^{-2}$	$8.383 * 10^{-1}$	$1.882 * 10^{-1}$

Table 4.17: P-values computed by a chi square test. The test was done on cross tables built with the combination of Answer and Label for different groups. The answers were collected after the participants saw the title.

We can see that no p-value is below 0.05. We have no reason to say that the link between the predicted and the expected label is dependent of the group. It does not mean that it is independent, only that we can't be sure that it is dependent with our data. By the way, since the p value is really close from 0.05 for the category 2 when seeing the title, it shows a big disparity and it is therefore possible that some groups are classifying better some of the topics of category 2 (maybe because it is linked to their common prior knowledge). We will come back to it in the following sections.

Between categories

Being able to study the variation of the prediction for the different categories is one of the key point of this work. We will therefore check if we can be sure that the relation between the answer and the label is varying among the different categories. Our cross tables have the format presented in table 4.18:

However, this format considers that we evaluate the change in the relation Answer/Label for both the generated and the original news. But two original news from 2 different categories are only differing on their topic, while two generated news also differs on the way they are generated. The topic dependencies are analyzed in the next section. We will therefore rather study in more details the table with the format described by table 4.19, that helps to analyze more precisely the change in the relation Answer/Label when the generation is changing:

Category / A ∩ L	1	2	3	4	All
(Gen, Gen)	a	b	c	d	$\sum$ (Gen, Gen)
(Gen, Orig)	e	f	g	h	$\sum$ (Gen, Orig)
(Orig, Gen)	i	j	k	l	$\sum$ (Orig, Gen)
(Orig, Orig)	m	n	o	p	$\sum$ (Orig, Orig)
All	$\sum C_1$	$\sum C_2$	$\sum C_3$	$\sum C_4$	$\sum$

Table 4.18: Contingency table format to evaluate the category dependency of the link between answers (A) and labels(L). Since every group evaluates 100 news of each category, the sums for each column are all equal to 100. The sum of all the elements, $\sum$, is therefore equal to 400

Category / A ∩ L	1	2	3	4	All
(Gen, Gen)	a	b	c	d	$\sum$ (Gen, Gen)
(Orig, Gen)	i	j	k	l	$\sum$ (Orig, Gen)
All	a+i	b+j	c+k	d+l	$\sum$

Table 4.19: Contingency table format to evaluate the category dependency of the link between answers (A) and labels(L) for generated news. Since every group evaluates 50 generated news of each category, the sums for each column are all equal to 50. The sum of all the elements, $\sum$, is therefore equal to 200

Table 4.20 shows the p-values obtained for each group when the user do not know the title. Table 4.21 shows the p-values obtained for each group when the user knows the title.

Group	Group 1	Group 2	Group 3
Gen + orig	$8.131 * 10^{-1}$	$9.939 * 10^{-1}$	$4.744 * 10^{-1}$
Gen only	$1.598 * 10^{-1}$	$8.929 * 10^{-1}$	$1.887 * 10^{-1}$
Orig only	$9.953 * 10^{-1}$	$7.501 * 10^{-1}$	$2.808 * 10^{-1}$

Table 4.20: P-values computed by a chi square test. The test was done on cross tables built with the combination of Answer and Label for different categories. The answers were collected before the participants saw the title.

A first observation can be that the p values when considering both the generated and the original news are very high. However, when only considering the generated news, the p-values are dropping. Based on our observations in the previous section (cf. section 4.2.1 Accuracy), their probably exists a link between the detection of

Group	Group 1	Group 2	Group 3
Gen + orig	$9.293 * 10^{-1}$	$9.632 * 10^{-1}$	$8.960 * 10^{-2}$
Gen only	$3.290 * 10^{-1}$	$5.092 * 10^{-1}$	$2.437 * 10^{-2}$
Orig only	$9.645 * 10^{-1}$	$8.716 * 10^{-1}$	$1.302 * 10^{-1}$

Table 4.21: P-values computed by a chi square test. The test was done on cross tables built with the combination of Answer and Label for different categories. The answers were collected after the participants saw the title.

the generated news and the categories, but we can't prove it since the p-values are not below 0.05. We can see that the p-values for the generated news are always lower than for the original ones[10], except for Group 2 when not knowing the title. We can check the corresponding cross table in figure 4.22.

A ∩ L \ Category	Category 1	Category 2	Category 3	Category 4	All
(Gen, Gen)	33|30	33|33	33|32	36|37	135|132
(Orig, Gen)	17|20	17|17	17|18	14|13	68
All	50|50	50|50	50|50	50|50	200|200

Table 4.22: Contingency table between the answer and category for group 2 when evaluating generated news and without seeing the title

We can see that the Group 2 gets high p-values because he is answering well for every category[11].

Again, we can see a tendency, but we have no statistical proof. We could justify this observation in multiple ways, among them there is :

- The facility to detect the generated news is independent from the category
- Some users are better than the other to detect generated news, independently of the category. It can be totally independent of their domain, and assign as their personal prior knowledge. On the contrary, some user might have done the experiment poorly.

[10]It means that the answers of the participants for the generated news tend to have bigger category disparity than for the original news.

[11]This result is a bit surprising since the news from category 1 and 2 at least are supposed to be harder to detect. It is possible that some people in group 2 were really good at detecting generated news, specially because group 2 is the one of computer science and engineering student and that they could be familiar with news generation.

- Their isn't enough data to be completely sure that the result is significant. The tendency exists but we would need more data to prove its existence.

Since all the tendencies are following our intuition and are similar to what was observed with the automated evaluation (cf. section 4.1), we will assume that the explanation is a mix between the two last possibilities. It will be the most plausible explanation if we don't find any other dependency during the next sections. Note that we can't prove that we are in this case, we can only suppose it based on all the things we observed and that are going in our direction.

Between topics in each category

To be sure that we can draw conclusions on the categories rather than on the topics [42], we have to check if the relation between the answer and the label is the same for every topic among a category. Since there are 5 topics in every class, the cross table used to compute the p-value had the format described by table 4.23:

Topic / A ∩ L	1st	2nd	3rd	4th	5th	All
(Gen, Gen)	a	b	c	d	e	$\sum$ (Gen,Gen)
(Gen, Orig)	f	g	h	i	j	$\sum$ (Gen,Orig)
(Orig, Gen)	k	l	m	n	o	$\sum$ (Orig,Gen)
(Orig, Orig)	p	q	r	s	t	$\sum$ (Orig,Orig)
All	$\sum T_1$	$\sum T_2$	$\sum T_3$	$\sum_i T_4$	$\sum T_5$	$\sum$

Table 4.23: Contingency table format to evaluate the topic dependency of the link between answers (A) and labels(L). Since their is 20 news by topic, the sums for each column are all equal to 20. The sum of all the elements, $\sum$, is therefore equal to 100

Table 4.24 shows the p-values obtained for each group and category when the user do not know the title. Table 4.25 shows the p-values obtained for each group and category when the user knows the title.

As we can see from the p-values, we can't prove that there is a topic dependency. And as we thought, the disparity among the answer for different topics is the biggest for the group 2 on category 2 when not knowing the titles. It is therefore possible that one of the topics in category 2 was better known by participants of Group 2, and that it leads to them having a bigger accuracy score in this category than any other group. It follows that the tendencies observed for the other groups are not respected for group 2, since they perform differently on at least 1 category based on the topics.

Group \ Category	Category 1	Category 2	Category 3	Category 4
Group 1	$4.901 * 10^{-1}$	$5.414 * 10^{-1}$	$7.742 * 10^{-1}$	$6.200 * 10^{-1}$
Group 2	$7.615 * 10^{-1}$	$1.541 * 10^{-1}$	$9.544 * 10^{-1}$	$7.590 * 10^{-1}$
Group 3	$9.844 * 10^{-1}$	$1.585 * 10^{-1}$	$5.523 * 10^{-1}$	$9.974 * 10^{-1}$

Table 4.24: P-values computed by a chi square test. The test was done on cross tables built with the answers from different groups on different news topics. The answers were collected before the participants saw the title.

Group \ Category	Category 1	Category 2	Category 3	Category 4
Group 1	$9.836 * 10^{-1}$	$8.987 * 10^{-1}$	$6.160 * 10^{-1}$	$4.569 * 10^{-1}$
Group 2	$8.697 * 10^{-1}$	$4.838 * 10^{-1}$	$9.656 * 10^{-1}$	$6.613 * 10^{-1}$
Group 3	$9.775 * 10^{-1}$	$7.310 * 10^{-2}$	$7.629 * 10^{-1}$	$9.971 * 10^{-1}$

Table 4.25: P-values computed by a chi square test. The test was done on cross tables built with the answers from different groups on different news topics. The answers were collected after the participants saw the title.

Summary

The last sections showed that the main dependency is between the answer and the label. We were not able to prove that any other dependency exists. However, this is still an important result, because it tells us that there is no proof of a clear influence from any parameter. It means that the experiment did not have any obvious bias, and since we have to explain the tendency seen in section 4.2, we will follow what we said in section 4.2.2, Between categories: There is a dependency, but we can't prove it because the data is too noisy. First, there is probably a high disparity between the users themselves. Some can have facilities to detect some topics, or to spot the critical points to analyze the news. On the contrary, some users might have done the experiment poorly and answered nearly randomly. It is also possible that some participants did not understood what was asked and based their judgment on a fake/reliable basis. This noise is increased even more by the relatively low number of evaluated news. Some solution to get rid of these effects could be to check if there are any outliers among the participants, and delete them from the analysis. However, one must be careful when doing it because the whole experiment would not be balanced anymore[12]. It would need some careful modification and assumptions to keep everything correct. This is mainly why such

[12]For example, if one removes a participant from a group, then the 3 groups would not be symmetrical anymore.

WASHINGTON (Reuters) - U.S. Chief of General Staff John Campbell said on Friday that some commanders in the international army called on President Donald Trump to intervene in Libya after the bombing of a nuclear plant in Iran raised serious concerns about the trajectory of U.S. military action there.

FILE PHOTO: The President of the United States addresses a joint session of Congress on foreign policy at the White House in Washington, U.S., March 26, 2017. REUTERS/Carlos Barria/File Photo

Campbell's comments came after a U.S. Air Force major said on Thursday that he and some generals were "seem to think" the Trump administration was considering action to help protect Nigerien President Muammar Gaddafi's forces, but said it was premature to declare that, as German Chancellor Angela Merkel had warned.

In a speech to the Military Officers Association of America in Washington on Thursday, Chief of General Staff Lt. Gen. Mary Odum had put the U.S. ambassador to Libya, Rear Admiral John Kirby, and the military chief of staff, General Michael McConnell, in "paranoia" about the escalating tensions.

U.S. defense officials have been trying to build up international pressure on Gaddafi since it was installed in Tripoli to end the 47-year-old dictator's 42-year rule. They say this largely lies in the perception that Gaddafi was ever capable of trying to overthrow an elected government in Libya.

Last week at a NATO summit in Brussels, Trump was infuriated by accusations that NATO was acting too slowly to support Gaddafi's forces, saying he was "very much fearful" about a possible world war.

"The president is too paranoid about the situation in the Middle East," Campbell told the Association for Professional Leadership.

The fact that the United States did not intervene militarily, he said, underscored the United States' readiness for another attack and described the attack as "a grave violation of international law.

Figure 4.4: News with the title "Trump on Beirut explosions: Some U.S. generals "seem to think it was an attack"", correctly identified by everybody as generated with an average certainty score of 80

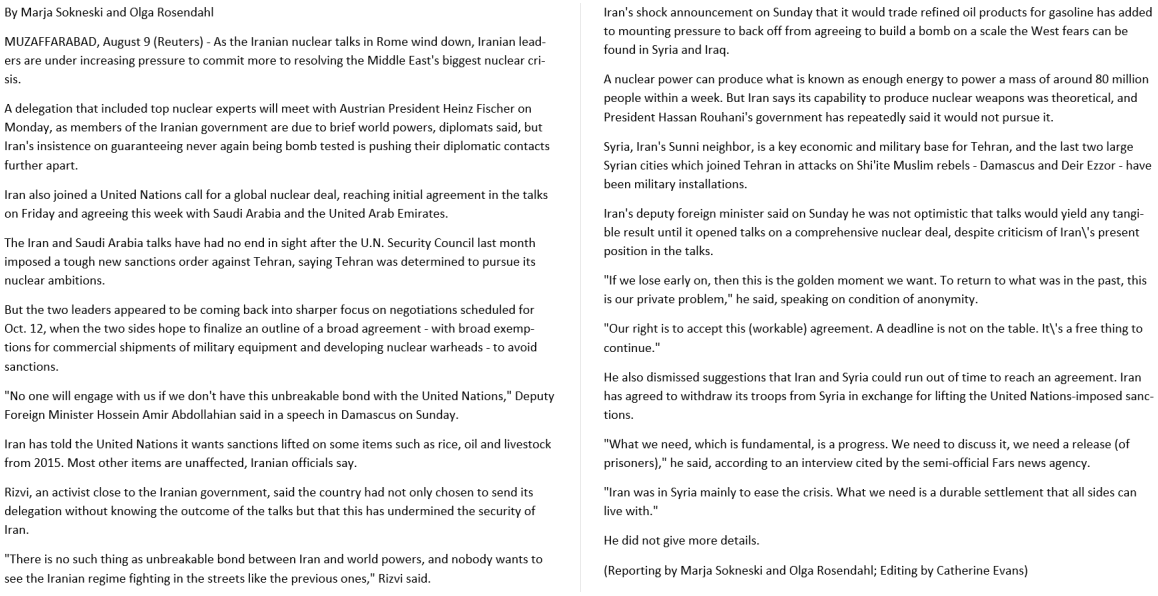
a solution was not part of our analyze. The database should be made available on *jebogaert*'s github for further works after asking the participants for permission.

4.2.3 Best and worst news

Finally, we can take a look at typical generated news that were correctly or incorrectly classified and take a look at the justifications. For example, figure 4.4 and figure 4.5 show news that were respectively identified and not identified as generated:

Among the justifications, we can find :

- Nigerien President Muammar Gaddafi, but he is syrian
- The text is asynchronous



By Marja Sokneski and Olga Rosendahl

MUZAFFARABAD, August 9 (Reuters) - As the Iranian nuclear talks in Rome wind down, Iranian leaders are under increasing pressure to commit more to resolving the Middle East's biggest nuclear crisis.

A delegation that included top nuclear experts will meet with Austrian President Heinz Fischer on Monday, as members of the Iranian government are due to brief world powers, diplomats said, but Iran's insistence on guaranteeing never again being bomb tested is pushing their diplomatic contacts further apart.

Iran also joined a United Nations call for a global nuclear deal, reaching initial agreement in the talks on Friday and agreeing this week with Saudi Arabia and the United Arab Emirates.

The Iran and Saudi Arabia talks have had no end in sight after the U.N. Security Council last month imposed a tough new sanctions order against Tehran, saying Tehran was determined to pursue its nuclear ambitions.

But the two leaders appeared to be coming back into sharper focus on negotiations scheduled for Oct. 12, when the two sides hope to finalize an outline of a broad agreement - with broad exemptions for commercial shipments of military equipment and developing nuclear warheads - to avoid sanctions.

"No one will engage with us if we don't have this unbreakable bond with the United Nations," Deputy Foreign Minister Hossein Amir Abdollahian said in a speech in Damascus on Sunday.

Iran has told the United Nations it wants sanctions lifted on some items such as rice, oil and livestock from 2015. Most other items are unaffected, Iranian officials say.

Rizvi, an activist close to the Iranian government, said the country had not only chosen to send its delegation without knowing the outcome of the talks but that this has undermined the security of Iran.

"There is no such thing as unbreakable bond between Iran and world powers, and nobody wants to see the Iranian regime fighting in the streets like the previous ones," Rizvi said.

Iran's shock announcement on Sunday that it would trade refined oil products for gasoline has added to mounting pressure to back off from agreeing to build a bomb on a scale the West fears can be found in Syria and Iraq.

A nuclear power can produce what is known as enough energy to power a mass of around 80 million people within a week. But Iran says its capability to produce nuclear weapons was theoretical, and President Hassan Rouhani's government has repeatedly said it would not pursue it.

Syria, Iran's Sunni neighbor, is a key economic and military base for Tehran, and the last two large Syrian cities which joined Tehran in attacks on Shi'ite Muslim rebels - Damascus and Deir Ezzor - have been military installations.

Iran's deputy foreign minister said on Sunday he was not optimistic that talks would yield any tangible result until it opened talks on a comprehensive nuclear deal, despite criticism of Iran\'s present position in the talks.

"If we lose early on, then this is the golden moment we want. To return to what was in the past, this is our private problem," he said, speaking on condition of anonymity.

"Our right is to accept this (workable) agreement. A deadline is not on the table. It\'s a free thing to continue."

He also dismissed suggestions that Iran and Syria could run out of time to reach an agreement. Iran has agreed to withdraw its troops from Syria in exchange for lifting the United Nations-imposed sanctions.

"What we need, which is fundamental, is a progress. We need to discuss it, we need a release (of prisoners)," he said, according to an interview cited by the semi-official Fars news agency.

"Iran was in Syria mainly to ease the crisis. What we need is a durable settlement that all sides can live with."

He did not give more details.

(Reporting by Marja Sokneski and Olga Rosendahl; Editing by Catherine Evans)

Figure 4.5: News with the title : "Beirut explosions create a dilemma for the world", wrongly identified as original by everybody with an average certainty score of 95

- Geographic problems, Libyan president can't be Nigerian
- Mixing topics,
- Impossible information: "It was installed in Tripoli to end the 47-year-old dictator's 42-year rule". It would mean that it was set when the dictator was 5, not really possible
- The OTAN is not supporting Gadaffi.

Among the justifications, we can find :

- Everything looks believable
- The authors are named
- The same authors are named in the beginning and in the end

This tells us some major things about how people perceive that a news is generated. First of all, coherence is a key point. For example, the first one was identified mainly because it was mixing topics and saying contradictory information. The second one, on the contrary was identified as original mainly because the name of the authors at the beginning and at the end was the same.

The second observation is that some people are sometimes relying on little details, while some other will process every line in the text to find inconsistencies. This is the typical kind of unpredictable factor than can induce a lot of noise in the collected data and it can be one of the reason why we are not able to prove the dependency from the category. It therefore comforts us is our choice of assuming that there is a dependency from the category, but the data are too noisy to prove it.

Part 5

Conclusion and possible further works

In conclusion, by first finding a news generator, creating our database and setting up a human and an automated evaluation, we were able to answered both of the questions asked in the beginning. **Yes, the news of the 4th category have a worst quality**, and **yes, based on our observations, it is verified both by human and by machine learning tools**. It leads to a bigger conclusion: Grover (and potentially other news generator) tends to have difficulties to create news on new topics. Grover is not able to generalize correctly its knowledge. It may seem quite obvious, but remember that since the news's topics are changing day after day, it means that such generator would have to be retrained quite often on new topics to be able to output decent results. This is even more true as we only assessed the fact that it is easy to see if it was automatically generated, not even if it seemed fake or not which is going a step further.

Knowing that, we could imagine a lot of possible further works. Since our experiment results were not completely significant, one can think about redoing a similar experiment in a bigger scale with even more controlled parameters to be able to confirm the tendency we saw on the accuracy curves. It could be even more interesting to redo it in a few months to see if the generated news are even worst, or simply extend the experiment to other news generators (potentially better version of Grover since we used the base version) and see if they all share the same problem.

Such a claim also asks for potential solution. One of them could be to retrain the model at some point. It follows that studying when to do it and how many news from a topic are enough to generate correct results about it are also some key questions.

Bibliography

[1] Burkhardt Joanna M. “History of fake news”. In: *Library Technology Reports* 53, no 8 (2017), pp. 5–9.

[2] Lazer David MJ et al. “The science of fake news”. In: *Science* 369 (6380) (2018), pp. 1094–1096.

[3] Monti Federico, Frasca Fabrizio, and Eynard Davide et al. “Fake news detection on social media using geometric deep learning”. In: *arXiv preprint* arXiv:1902.06673 (2019).

[4] Pérez-Rosas Verónica, Kleinberg Bennett, and Lefevre Alexandra et al. “Automatic detection of fake news”. In: *arXiv preprint* arXiv:1708.07104 (2017).

[5] Reis Julio CS, Correia André, and Murai Fabrício. “Supervised learning for fake news detection”. In: *IEEE Intelligent Systems* vol. 34, no 2 (2019), pp. 76–81.

[6] Rowan Zellers et al. “Defending Against Neural Fake News”. In: *NeurIPS* (2019), pp. 9051–9062.

[7] Goodfellow Ian J., Shlens Jonathon, and Szegedy Christian. “Explaining and harnessing adversarial examples”. In: *arXiv preprint* arXiv:1412.6572 (2014).

[8] Thorsten Quandt et al. “Fake news”. In: *The international encyclopedia of Journalism Studies* (2019), pp. 1–6.

[9] Edson C. Tandoc Jr., Zheng Wei Lim, and Richard Ling. “Defining “fake news””. In: *Digital Journalism* 6(2) (2018), pp. 137–153.

[10] *Energy considerations for training deep neural networks*. 2019. URL: `https://ekamperi.github.io/machine%5C%20learning/2019/08/14/energy-considerations-dnn.html`.

[11] Alec Radford et al. “Improving language understanding by generative pre-training”. In: *OpenAI blog* (2018).

[12] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint* arXiv:1810.04805 (2018).

[13] Alec Radford et al. “Language models are unsupervised multitask learners”. In: *OpenAI blog* vol. 1 no 8 (2019), p. 9.

[14] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. “Latent Dirichlet Allocation”. In: *Journal of Machine Learning Research* 3 (2003), pp. 993–1022.

[15] Chuang Jason, Christopher D. Manning, and Jeffrey Heer. “Termite: Visualization techniques for assessing textual topic models.” In: *Proceedings of the international working conference on advanced visual interfaces* (2012).

[16] Sievert Carson and Kenneth Shirley. “LDAvis: A method for visualizing and interpreting topics”. In: *Proceedings of the workshop on interactive language learning, visualization and interfaces* (2014), pp. 63–70.

[17] Carbonnelle Quentin. “Robustness of fake news detectors”. In: *Ucl master thesis* (2020), pp. 2–13.

[18] Uysal Alper Kursat and Gunal Serkan. “The impact of preprocessing on text classification”. In: *Information Processing & Management* vol. 50 no. 1 (2014), pp. 104–112.

[19] Thavareesan Sajeetha and Sinnathamby Mahesan. “Word embedding-based Part of Speech tagging in Tamil texts”. In: *IEEE 15th International Conference on Industrial and Information Systems (ICIIS)* (2020).

[20] Purnamasari K. K. and I. S. Suwardi. “Rule-based Part of Speech Tagger for Indonesian Language”. In: *IOP Conference Series: Materials Science and Engineering* Vol. 407. No. 1 (2018).

[21] Church Kenneth Ward. “A stochastic parts program and noun phrase parser for unrestricted text”. In: *International Conference on Acoustics, Speech, and Signal Processing IEEE* Vol. 407. No. 1 (1989), pp. 695–698.

[22] Lovins Julie Beth. “Development of a stemming algorithm”. In: *Mech. Transl. Comput. Linguistics* vol. 11 no. 1-2 (1968), pp. 22–31.

[23] Mersinias Michail, Afantenos Stergos, and Chalkiadakis Georgios. “CLFD: A Novel Vectorization Technique and Its Application in Fake News Detection”. In: *Proceedings of The 12th Language Resources and Evaluation Conference* (2020), pp. 3475–3483.

[24] Qaiser Shahzad and Ali Ramsha. “Text mining: use of TF-IDF to examine the relevance of words to documents”. In: *International Journal of Computer Applications* vol. 181 no. 1 (2018), pp. 25–29.

[25] Mikolov Tomas et al. “Distributed representations of words and phrases and their compositionality”. In: *arXiv preprint* arXiv:1310.4546 (2013).

[26] Laurens van der Maaten and Geoffrey Hinton. “Visualizing Data using t-SNE”. In: *Journal of Machine Learning Research* 9 (2008), pp. 2579–2605.

[27] Huang Jin and Ling Charles X. “Using AUC and accuracy in evaluating learning algorithms”. In: *IEEE Transactions on knowledge and Data Engineering* vol. 17 no 3 (2005), pp. 299–310.

[28] Zhang Zhilu et Sabuncu Mert R. “Generalized cross entropy loss for training deep neural networks with noisy labels”. In: *arXiv preprint* arXiv:1805.07836 (2018).

[29] Gers Felix A., Schmidhuber Jürgen, and Cummins Fred. “Learning to forget: Continual prediction with LSTM”. In: *Neural computation* vol. 12, no 10 (2000), pp. 2451–2471.

[30] Medsker Larry R. and Jain L. C. “Recurrent neural networks”. In: *Design and Applications* vol. 5 (2001).

[31] Sherstinsky Alex. “Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network”. In: *Physica D: Nonlinear Phenomena* vol. 404 (2020), p. 132306.

[32] Schuster Mike and Paliwal Kuldip K. “Bidirectional recurrent neural networks”. In: *IEEE transactions on Signal Processing* vol. 45 no 11 (1997), pp. 2673–2681.

[33] Zhang Xuchao, Zhao Liang, and Chen Zhiqian et al. “Trendi: Tracking stories in news and microblogs via emerging, evolving and fading topics”. In: *IEEE International Conference on Big Data (Big Data)* IEEE, 2017 (2017), pp. 1590–1599.

[34] Pranckevičius Tomas and Marcinkevičius Virginijus. “Comparison of naive bayes, random forest, decision tree, support vector machines, and logistic regression classifiers for text reviews classification”. In: *Baltic Journal of Modern Computing* vol. 5 no. 2 (2017), p. 221.

[35] Arora Monika, Mittal Vrinda, and Aggarwal Priyanka. “Enactment of tf-idf and word2vec on Text Categorization”. In: *Proceedings of 3rd International Conference on Computing Informatics and Networks: ICCIN 2020. Springer Singapore* (2021), pp. 199–209.

[36] Zhou Peng, Qi Zhenyu, and Zheng Suncong et al. “Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling”. In: *arXiv preprint* arXiv:1611.06639 (2016).

[37] Zhou Chunting, Sun Chonglin, and Liu Zhiyuan et al. “A C-LSTM neural network for text classification.” In: *arXiv preprint* arXiv:1511.08630 (2015).

[38] Ramachandran Prajit, Zoph Barret, and Le Quoc V. “Searching for activation functions”. In: *arXiv preprint* arXiv:1710.05941 (2017).

[39] Liu Gang and Guo Jiabao. “Bidirectional LSTM with attention mechanism and convolutional layer for text classification”. In: *Neurocomputing* vol. 337 (2019), pp. 325–338.

[40] Patel Kevin and Bhattacharyya Pushpak. “Towards lower bounds on number of dimensions for word embeddings”. In: *Proceedings of the Eighth International Joint Conference on Natural Language Processing* vol. 2: Short Papers (2017), pp. 31–36.

[41] Mchugh Mary L. “The chi-square test of independence”. In: *Biochemia medica* vol. 23 no. 2 (2013), pp. 143–149.

[42] Bozarth Lia and Budak Ceren. “Toward a better performance evaluation framework for fake news classification”. In: *Proceedings of the international AAAI conference on web and social media* (2020), pp. 60–71.

Appendix A

Code to setup Grover in Generation mode

You can fin below the code that was used to setup Grover in generation mode. It is a slightly different version from what was presented in Grover's github repository, and you should probably follow the instructions there first.

```
curl -o ~/miniconda.sh -O
 ↪  https://repo.continuum.io/miniconda/Miniconda3-4.5.4-Linux-x86_64.sh
 ↪  && \
chmod +x ~/miniconda.sh && ~/miniconda.sh -b -p ~/conda &&
 ↪  \
rm ~/miniconda.sh && ~/anaconda3/bin/conda install -y
 ↪  python=3.7

conda create -y -n grover python=3.7 && \
source activate grover && \
pip install -r requirements-gpu.txt

python download_model.py base

PYTHONPATH=$(pwd) python sample/contextual_generate.py
 ↪  -model_config_fn lm/configs/base.json -model_ckpt
 ↪  models/base/model.ckpt -metadata_fn
 ↪  sample/your_input_file.jsonl -out_fn
 ↪  your_output_file.jsonl
```

(

Where you can replace "your_input_file" and "your_output_file" by the name of your input and output files. The format for inputs and outputs in .jsonl, you

can find examples in the /sample directory. The requirements-gpu.txt file is a bit different from the one available on github. It was modified becaue of dependencies issues but it is possible that the basic one works fine in your setup. My requirements-gpu.txt file contains:

```
pandas==0.24.2
regex==2019.4.14
h5py==2.9.0
numpy==1.16.2
tensorboard==1.13.1
tensorflow-gpu==1.13.1
tensorflow==1.13.1
tqdm==4.31.1
requests==2.22.0
pyyaml
```

Appendix B

How to access news using Commoncrawl

The crawling was done using a WSL bash Ubuntu 18.04, but any other linux distribution should be able to follow the exact same steps. First, list all the accessible warc files available by using the following command in a linux terminal (WSL bash works fine as well). Since the archives are stored on Amazon's server, one has to use an aws command. If you don't have an AWS account, you can still access the files by adding the –no-sign-request option at the end of this command.

```
aws s3 ls --recursive s3://commoncrawl/crawl-data/CC-NEWS/
```

The command above returns a list of approximately 15 000 lines. If one wants to save these lines in a file, one can use » file_to_store at the end of the above command. Each line has the following format.

```
date hour warc_number index
```

For example, one of the line has the following fields :

- Date : 2021-06-01
- Hour : 20:05:04
- Warc_number : 1072799899
- Index : crawl-data/CC-NEWS/2021/06/CC-NEWS-20210601162055-00190.warc.gz

Which corresponds to an archive containing news collected on the 1st of june 2021 at 8 pm. To access the collected archive, one has to download it using

```
wget https://commoncrawl.s3.amazonaws.com/the_wanted_index
```

Where you can replace "the_wanted_index" by the index field of any of the 15 000 lines listed with the first command. This line downloads a .warc.gz file in the current directory. The following python code allows to iterate trough the .warc.gz archive.

```
with open(file_name, "rb") as stream:
    for record in ArchiveIterator(stream):
        news = from_warc(record)
```

Where you can replace file_name by the name of the downloaded file. The last line of the code allows to transform a record into a news object. It uses the package "News-please", available on github (link newsplease) and allows to extract data like the date or the main text and to access them in an easy way. The ArchiveIterator method is part of the warcio.archiveiterator package.

Appendix C

Website

This section contains screenshots of the pages that where not presented in the paper. The other pages can be found in section (ref website). The website is available at this link (link website)

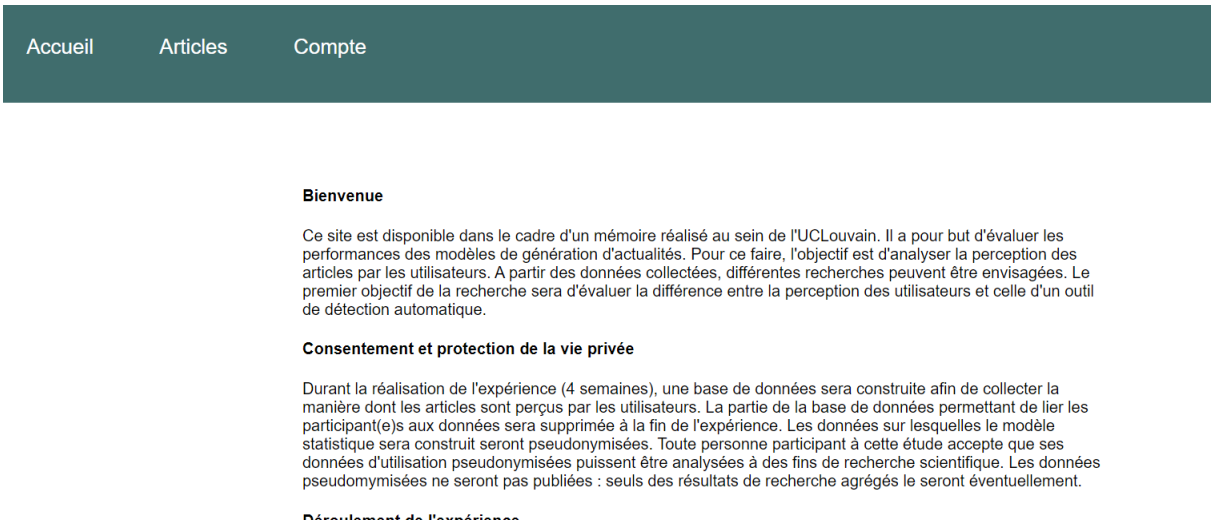


Figure C.1: Welcome page part 1

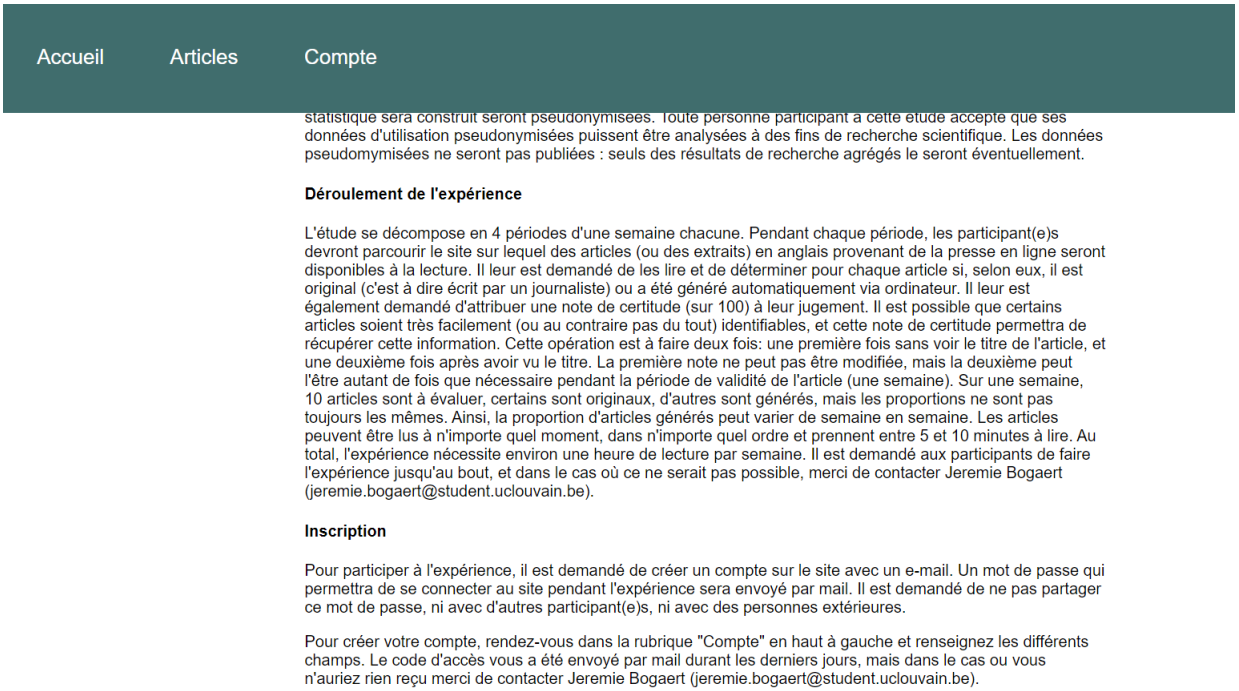


Figure C.2: Welcome page part 2

Figure C.3: Connection page

Appendix D

Paper presented at the WIC conference

Can Fake News Detection be Accountable? The Adversarial Examples Challenge

(Extended Abstract)

Jérémie Bogaert, Quentin Carbonelle,
Antonin Descampe François-Xavier Standaert

UCLouvain, Louvain-la-Neuve, Belgium

Abstract

Automated fake news detection is an important challenge in view of the increasing ability of statistical language models to generate large amounts of (possibly fake) articles, so that recognizing them manually becomes unrealistic. Yet, the reliable deployment of such automated detection tools would require ensuring that they are accountable. Algorithmic accountability is known to be difficult to reach, especially when adversarial behaviors aim to make algorithms deviate from their expected mode of operation. In this paper, we illustrate with a case study that this challenge is further amplified in contexts where the labeling of the articles is prone to errors, which is the case of fake news detection.

1 Introduction

The proliferation of fake news on social media platforms has created an increasing need of solutions to detect them. While the generation of fake news and the necessary fact checking that it implies started as a mostly manual process (e.g., discussed in [17]), recent advances in natural language generation with machine learning algorithms have amplified the risk of so-called neural fake news [19]. The massive amount of articles that such tools can generate makes manual fact checking unrealistic and raises the question of their automated detection. As recently surveyed in [1, 20], solutions combining natural language processing and machine learning are attractive for this purpose, due to their ability to capture various features of newspaper articles. Besides, the sensitive nature of the fake news detection problem also requires that its deployment comes with guarantees of algorithmic accountability [3]. But as discussed in [2], such guarantees are hard to obtain in contexts where adversarial behaviors can target the robustness of machine learning classifiers, which is the case of fake news detection.

In this paper, we therefore study the robustness of fake news detection against adversarial examples [4]. For this purpose, we first build a database of fake and reliable news. As already observed in the literature, this step is inherently challenging since there is no strict definition of what a fake news is [6]. We deal with this difficulty by combining a public dataset of fake news (`https://www.kaggle.com/mrisdal/fake-news`), which were scraped from blacklisted websites over a period of 30 days around the 2016 US election, and reliable news collected from the New York Times and the Guardian over approximately the same period. We additionally explore both data sets to show that they cover similar topics. While inevitably imperfect, we then use this database to show that standard machine learning classifiers can detect fake news with a good accuracy, but are also easy to fool with adversarial examples. Interestingly, it appears that the difficulty to define what a fake news is (possibly combined with label errors in the training sets) gives adversaries additional opportunities to generate fake news classified as reliable. So our results question again the possibility to

rely on accountable algorithms for sensitive tasks that can be targeted by adversarial behaviors and suggest different research avenues related to fake news in general.

We note that the topic of fake news is widely multidisciplinary [7] and even the more technical topic of automated fake news detection is already covered by a broad literature (e.g., the already mentioned [1, 19, 20] but also [5, 11, 13, 12] to name a few). Our contribution is not to improve automated fake news detection algorithms but to illustrate the difficult interplay between such algorithms and the need of algorithmic accountability when adversarial behaviors are considered. As a natural starting point in this direction, we show that there exist (for now simple) examples of fake news detectors that are weak against such adversarial behaviors, raising the question of how to prevent them, with technical and non-technical means. In other words, we put forward an under-discussed risk for the reliable deployment of such systems.

The rest of the paper is structured as follow. We start by describing the database we used for our investigations and discussing its unavoidable limitations in Section 2. We follow by showing simple examples of supervised fake news detection tools that can detect fake news with reasonable accuracy in Section 3. We finally exhibit how to craft adversarial examples against these classifiers in Section 4. We conclude by analyzing the impact of these findings and tracks for further research in Section 5.

2 Building a fake news database

Research on fake news detection has often been limited by the quality of existing datasets and their specific application contexts [12]. As already mentioned, this is mostly due to the difficulty of precisely defining what a fake news is, making their labeling for supervised learning challenging [6]. Besides, our goal to investigate adversarial examples is typically calling for "not too short" texts, excluding popular databases such as [18]. To the best of our knowledge, the database that comes the closest to our needs is the fake news corpus (`https://github.com/several27/FakeNewsCorpus`). However, preliminary investigations suggested quite significant dissimilarities between the topics of reliable and fake articles (namely, football for fake articles and politics for reliable ones). This similarity makes it unsuitable for our purposes, since it implies risks to classify the articles based on their topic more than their fake/reliable nature. We therefore prepared our investigations by attempting to build a better database.

Concretely, we started from a public dataset of 3500 fake news from Kaggle (`https://www.kaggle.com/mrisdal/fake-news`), which were scraped from 207 blacklisted websites over a period of 30 days around the 2016 US election. Some preliminary processing was applied to remove unwanted features of the articles (e.g., meta-data that is not in the original articles and was added by the blacklisted websites). No website contributed to more than 1% of the database to ensure that imperfect scraping, preprocessing or labeling for some websites cannot significantly impact the overall results. We then tried to build a complementary database of reliable news, covering the same period of time. For this purpose, we used the API developed by the New York Times and The Guardian, and scraped papers about World news and US news for the intended period. We had to slightly increase the window of time (Oct. 16 to Dec. 4 for the reliable news vs. Oct. 26 to Nov. 25 for the fake news), in order to collect 3500 articles (1500 from the New York Times, 2000 from The Guardian).

We performed a preliminary exploration of the topics covered by this database using the Term Frequency–Inverse Document Frequency (TF-IDF) statistic. It is a standard tool for information retrieval or summarization, which indicates the relevance of the words in some documents [10]. We first launched it on the full database to identify the most relevant words overall. Next, we launched it on the fake news corpus, leading to the results of Figure 1 (where the X axis lists the 40 most relevant words of the full

database). It confirms the coverage of the US elections (e.g., with Trump and Clinton in the first places), but also highlights the weight of some neutral words like "said".

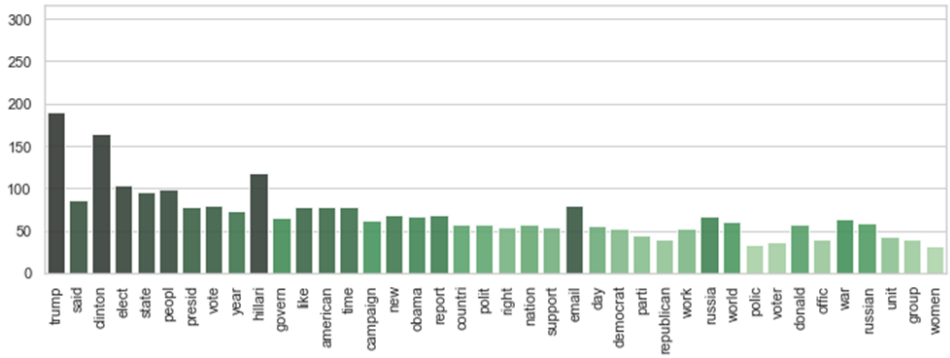


Figure 1: Evaluation of the fake news database's topics with TF-IDF score.

We finally computed the same TF-IDF scores for the reliable news, which are given in Figure 2 and confirm the coverage of similar topics. They also highlight some specific features of the fake news corpus already: for example the more frequent use of the (misspelled) first name Hilari or the importance of the word e-mail (relating to an ongoing affair of Mrs. Clinton's e-mail leaks during the 2016 elections).

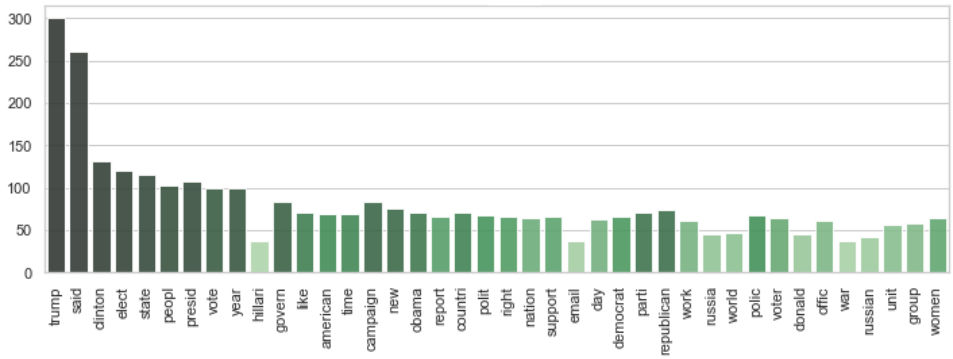


Figure 2: Evaluation of the reliable news database's topics with TF-IDF score.

In order to confirm that the resulting (7000-paper) database did not contain obvious parasitic patterns, we additionally visualized the data by feeding the t-distributed Stochastic Neighbor Embedding (t-SNE) tool of [16] with the vectors output by the TF-IDF transform.* t-SNE projects each high-dimensional object towards a two-dimensional point in such a way that similar objects are modeled by nearby points and dissimilar objects are modeled by distant points with high probability. As illustrated in Figure 3, the distribution of the topics is similar for fake and reliable articles while, for example, the separation between World and US (reliable) news can be distinguished in the right part of the figure. It suggests that the topics do not create obvious (parasitic) ways to discriminate fake and reliable news. So while the evaluations in this section do admittedly not provide any formal guarantee that no such patterns exist, we assume in the following that this database is good enough for our purposes.

* Reduced to 500 dimensions thanks to truncated Singular Value Decomposition (SVD). We also tried larger number of dimensions but it did not change our main observations.

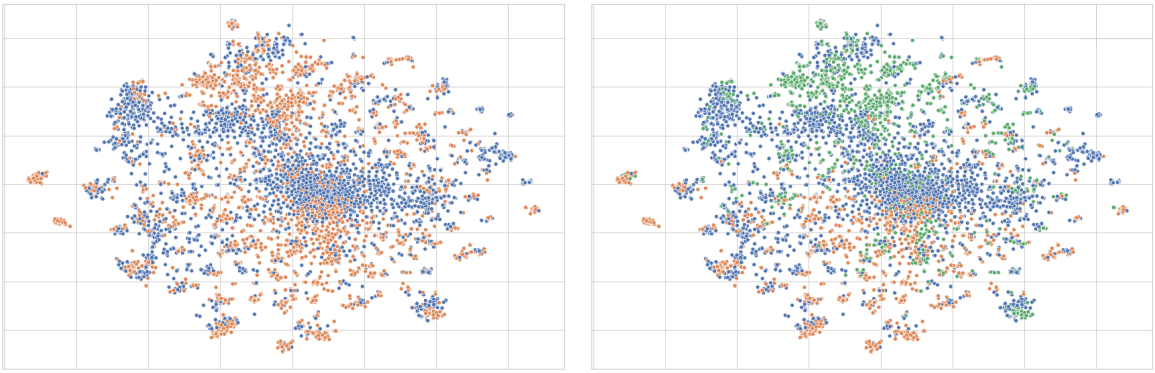


Figure 3: Visualization of the news' topics with t-SNE. Left: fake news (●) and reliable news (●). Right: fake news (●) and reliable World (●) & US news (●).

3 Exemplary classifiers

The next step of our investigations was to build statistical classifiers for fake and reliable news, thanks to supervised machine learning. We considered various options for this purpose: logistic regression, naive Bayes, random forests and Long Short-Term Memory (LSTM). For place constraints, we focus our following descriptions on logistic regression and LSTM (the other classifiers did not significantly affect our main conclusions).

Concretely, all our classifiers were built starting with the same preprocessing: we removed punctuation, non-alphanumeric characters, multiple whites spaces, websites, stop words and short words). For the logistic regression, we then vectorized the articles as bag of words using TF-IDF while for the LSTM we used a Word2Vec model trained on our full dataset [9]. As Google's Word2Vec (`https://code.google.com/archive/p/word2vec/`), we fixed the embedding to 300-dimensional vectors. The rationale behind this choice is that the LSTM can take advantage of the words' order. The model hyperparameters were then tuned using a 5-fold cross-validation.

The accuracy of these classifiers is illustrated in Figure 4. It is significantly better than a random guess in both cases, and reaches $> 90\%$ for the logistic regression (we expect that the LSTM would reach this accuracy or even improve it with more training samples). Despite further optimizations are certainly possible, we assume these values to be sufficient for trying to decrease them with adversarial examples.

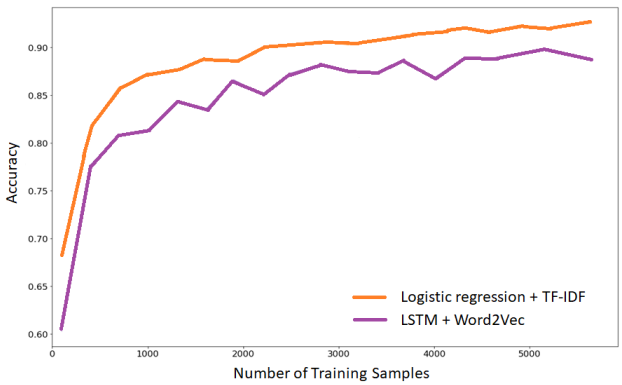


Figure 4: Learning rate of exemplary fake news detectors.

4 Crafting adversarial examples

An adversarial example is an input, unknown to the target machine learning algorithm, that makes it deviate from its public specifications and is purposely generated by an adversary having an interest in such a deviated behavior [4]. In the context of fake news detection, the goal of the adversary is to force the misclassification of a fake news as reliable, despite not changing its semantic content from the human perception viewpoint. The main question we tackle next is whether such adversarial examples can be crafted for the detectors of the previous section. In this context, the adversarial goal could vary from producing such adversarial examples at a low rate, possibly taking advantage of some manual processing, or to produce them at a high rate, automatically. As a first step to show the existence of a risk, we consider the easiest (low rate with manual intervention) context. We briefly discuss the relaxation of this context at the end of the section. The threat model could also vary from white box access to the models (i.e., knowing their parameters) to only black box access (i.e., only being able to observe input/output pairs). We discuss both options in the following.

4.1 Methodology

The high-level approach we used to craft adversarial examples at low rate is in 3 steps:

1. Identify Reliable Hot Words (RHW), which are the words having high probability to push the classification of any article towards the reliable class.

2. Identify the Target's Fake Hot Words (TFHW), which are the words having high probability to push the classification of a target article towards the fake class.

3. Human intervention to replace TFHW by RHW in a semantic-preserving manner.

The identification of the RHW and TFHW was performed using high-level ideas similar to [8, 2]. In the white box setting, we applied the Fast Gradient Sign Method (FGSM) which essentially boils down to maximizing the classifier's loss function, then using the gradient information to add or remove words independent of their position.[†] This gradient can be computed analytically for the logistic regression, and we used the numerical estimation provided by TensorFlow (`https://www.tensorflow.org/`) for the LSTM. In the black box setting, we used a more straightforward approach where we just removed words one by one and feed the classifier with the modified articles in order to identify words that impact the detection probability the most.

4.2 Experimental results

We will discuss the type of results we obtain with Example 1, which is initally detected as a fake news with 90% probability by the logistic regression, and 65% probability by the LSTM. As aforementioned, the first step in trying to fool the detection is to identify RHW and TFHW. RHW are identified on the reliable news corpus while TFHW are identified on the target article only. We illustrate this step with our black box approach applied to the beginning of the example (namely, the words 'In what is being described as another 'bizarre'), where the short words 'In', 'what', 'is', 'being' and 'as' do not impact the classification, while removing the words 'described' and 'bizarre' respectively decrease its probability of being detected as a fake by 2.8% and 3.5%. By extending such an approach to the whole article (and the whole corpus of reliable news for RHW), we can then build lists of TFHW and RHW.

[†] This position could be exploited in the case of the LSTM, which we leave as a scope for further investigations (a direct exploitation would result in a quite computationally intensive strategy).

In what is being described as another 'bizarre' attempt to sabotage her own campaign, Hillary Clinton has desecrated a series of beloved US symbols, including punching a bison, setting fire to the Stars & Stripes and spitting at Jerry Seinfield. The Presidential hopeful seems determined to make a series of unprovoked errors, not least of which was agreeing to Bill hosting a sleepover for a group of Girl Guides. Short of dressing the Statue of Liberty in a Burka, Mrs Clinton has lurched from one PR blunder to another. Commented one journalist: 'The Presidential race is entering the final furlong and if Mrs Clinton was horse – and before you can say Benghazi – she's gone from bookie's favourite to an ingredient at the local glue factory'. Having already become the unwitting focus of various health scares and FBI investigations, Mrs Clinton's campaign is as orderly as a Marx Brothers movie. Her lead in the polls has been cut as video emerges of her lighting a cigar with a rolled up Bill of Rights, then proceeding to take a dump on the White House lawn. Hillary's erratic behaviour has seen her sing the Star-Spangled Banner in Korean, dress as Oprah Winfrey for Halloween and pebble-dash Mount Rushmore. Remarked a flummoxed advisor: 'She keeps doing the unthinkable – like making Donald Trump electable'. Share this story...

Exemple 1: Sample of the fake news database.

The main high-level observations that can be extracted from these lists are:

1. That they contain both semantically tainted words (e.g., 'Hilary', 'FBI', 'Donald') and semantically neutral words (e.g., 'said', 'commented', 'campaign').
2. That there is a higher proportion of semantically tainted words for TFHW.
3. That the lists of (ordered) words identified as RHW or TFHW for the logistic regression and the LSTM, significantly overlap but are not identical.

We can then build adversarial examples, e.g., for the (simpler) LSTM:

[...] 'bizarre' attempt to sabotage her own campaign, ~~Hillary~~ **Mrs** Clinton has desecrated
[...] Mrs Clinton's campaign is as ~~orderly~~ **neat** as a Marx Brothers movie [...]

By changing only two words (which do not affect the meaning of the article), it is now classified as a fake with only 45% probability (so as reliable with 55% probability). Interestingly, exactly those changes are not sufficient to misclassify the article with the logistic regression (which still classifies it as a fake, but with a probability reduced from 90% to 55%). Yet, another change of the second word does the job (suggesting that as usual with adversarial examples, they have a certain level of transferability):

[...] 'bizarre' attempt to sabotage her own campaign, ~~Hillary~~ **Mrs** Clinton has desecrated
[...] Remarked a flummoxed ~~advisor~~ **minister**: 'She keeps doing the unthinkable [...]

Since based on manual interventions, we did not craft such examples for large number of articles. We repeated the process for 5 fake news and succeeded to force a misclassification by changing a maximum of 15 words. The LSTM classifier was usually fooled with slightly less words which we assume is due to its initially lower accuracy.

Overall, and despite drawing general conclusions based on such small-scale examples is hard, one important observation is that in contrast with the case study of [2] where adversarial examples had to be mostly based on neutral words (to avoid being easy to spot by the human perception), fooling a fake news detection algorithm can be done with more tainted words (e.g., by changing 'Hillary' into 'Mrs' in our example). In other words, the fact that the fake or reliable nature of an article does not have a definition as clear as the topics used in the case study [2] makes the adversary's task easier.

4.3 Towards automation

Our handmade experiments naturally raise the question whether adversarial examples can be automated. As a first step in this direction (i.e., to evaluate the sensitivity of

our fake news detectors to semantically-neutral modifications), we evaluated a greedy strategy where we just substituted words by synonyms (sometimes causing syntax problems). One example obtained for the logistic regression is given below:

> [...] symbols, including punching a ~~bison~~ **buffalo** [...] [...] a group of ~~Girls~~ **Woman** guides
> [...] various health scares and FBI ~~investigations~~ **inquiry**, Mrs Clinton's campaign [...]

Out of 100 test articles, we could misclassify 22% (resp., 32%) with the LSTM (resp., logistic regression) classifier. Using advanced statistical language models should lead to much more adversarial opportunities, which we leave as an open problem.

5 Conclusions and open problems

Advances in digital media are resonsible for journalism to loose the monopoly on information production and dissemination, and raise new legitimacy concerns (e.g., regarding whether journalism can still offer quality and reliable news in the digital era) [15]. Algorithmic accountability is one of the emerging (and difficult to reach) goals aiming to mitigate such concerns [3]. In this work, we confirm that ensuring algorithmic accountability with robustness against adversarial behaviors is especially challenging in contexts such as fake news detection where the definition of optimization criteria is inherently fuzzy due to the lack of well defined classes. Our results are preliminary in many respects and therefore suggest various directions for further investigations. First, the difficult collection of a fake news / reliable news database would be improved by designing a tool able to automatically generate large amounts of fake news from reliable ones (e.g., by biasing them in a given direction). It would allow a better analysis of the syntactic or semantic patterns that fake news detection can exploit. Second, and as already mentioned, the generation of adversarial examples would benefit from automation in order to enable larger-scale experiments. Third, the evaluation of countermeasures (i.e., fake news detection tools able to cope with adversarial examples) is an important long-term goal as well. Yet, early discussions in [4, 2] suggest that a purely technical solution may not be possible. The perspective of such negative conclusions therefore questions the need of complementary approaches, e.g., relying on the trust in the journalists who write stories more than on content, as suggested in [14].

References

[1] Nadia K. Conroy, Victoria L. Rubin, and Yimin Chen. Automatic deception detection: Methods for finding fake news. Proceedings of the Association for Information Science and Technology, 52(1):1–4, 2015.

[2] Antonin Descampe, Clément Massart, Simon Poelman, François-Xavier Standaert, and Olivier Standaert. Automated news recommendation in front of adversarial examples and the technical limits of transparency in algorithmic accountability. AI & Society, 2021.

[3] Nicholas Diakopoulos. Algorithmic accountability. Digital Journalism, 3(3):398–415, 2015.

[4] Ian J. Goodfellow, Patrick D. McDaniel, and Nicolas Papernot. Making machine learning robust against adversarial inputs. Commun. ACM, 61(7):56–66, 2018.

[5] Naeemul Hassan, Fatma Arslan, Chengkai Li, and Mark Tremayne. Toward automated fact-checking: Detecting check-worthy factual claims by claimbuster. In KDD, pages 1803–1812. ACM, 2017.

[6] Edson C. Tandoc Jr., Zheng Wei Lim, and Richard Ling. Defining "fake news". Digital Journalism, 6(2):137–153, 2018.

[7] David M. J. Lazer, Matthew A. Baum, Yochai Benkler, Adam J. Berinsky, Kelly M. Greenhill, Filippo Menczer, Miriam J. Metzger, Brendan Nyhan, Gordon Pennycook, David Rothschild, Michael Schudson, Steven A. Sloman, Cass R. Sunstein, Emily A. Thorson, Duncan J. Watts, and Jonathan L. Zittrain. The science of fake news. Science, 359(6380):1094–1096, 2018.

[8] Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. Deep text classification can be fooled. In IJCAI, pages 4208–4215. ijcai.org, 2018.

[9] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In ICLR (Workshop Poster), 2013.

[10] Juan Ramos. Using tf-idf to determine word relevance in document queries. In Proceedings of the First Instructional Conference on Machine Learning, pages 29–48, 2003.

[11] Natali Ruchansky, Sungyong Seo, and Yan Liu. CSI: A hybrid deep model for fake news detection. In CIKM, pages 797–806. ACM, 2017.

[12] Karishma Sharma, Feng Qian, He Jiang, Natali Ruchansky, Ming Zhang, and Yan Liu. Combating fake news: A survey on identification and mitigation techniques. ACM Trans. Intell. Syst. Technol., 10(3):21:1–21:42, 2019.

[13] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. Fake news detection on social media: A data mining perspective. SIGKDD Explor., 19(1):22–36, 2017.

[14] David Sterrett, Dan Malato, Jennifer Benz, Liz Kantor, Trevor Tompson, Tom Rosenstiel, Jeff Sonderman, and Kevin Loker. Who shared it?: Deciding what news to trust on social media. Digital Journalism, 7(6):783–801, 2019.

[15] Jingrong Tong. Journalistic legitimacy revisited. Digital Journalism, 6(2):256–273, 2018.

[16] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. Journal of Machine Learning Research, 9(86):2579–2605, 2008.

[17] Chris J. Vargo, Lei Guo, and Michelle A. Amazeen. The agenda-setting power of fake news: A big data analysis of the online media landscape from 2014 to 2016. New Media Soc., 20(5):2028–2049, 2018.

[18] William Yang Wang. "liar, liar pants on fire": A new benchmark dataset for fake news detection. In ACL (2), pages 422–426. Association for Computational Linguistics, 2017.

[19] Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. Defending against neural fake news. In NeurIPS, pages 9051–9062, 2019.

[20] Xinyi Zhou and Reza Zafarani. A survey of fake news: Fundamental theories, detection methods, and opportunities. ACM Comput. Surv., 53(5):109:1–109:40, 2020.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**