UCLouvain

École polytechnique de Louvain

Faking News Recommendation

An exploration of text representation

Author: **Simon Poelman**
Supervisors: **François-Xavier Standaert, Michel Verleysen**
Readers: **Antonin Descampe, Olivier Standaert, Clément Massart**
Academic year 2018–2019
Master [120] in Electrical Engineering

Abstract

Algorithms are making their way into many disciplines, this paper studies their use in journalism. We focus on the representation of text content for recommendation. We introduce a new text clustering technique based on graph modularity and inter-article similarities that outperforms classical clustering methods. We then explore the case of adversarial examples on text classifiers, writing manually articles to be incorrectly classified by the best performing algorithms on our dataset. We observe differences in difficulty according to the type of classifier.

Keywords : algorithm, machine learning, journalism, adversarial example, recommendation system, recsys, NLP

Acknowledgements

This thesis ends my five years of studying Engineering at the UCLouvain. While I put my heart and soul into this work, I would not have made a fraction of what I did were it not for all the help and advices given to me. For that, I would like to thank Pr. François-Xavier Standaert, Pr. Michel Verleysen, Pr. Antonin Descampe, Pr. Olivier Standaert and Clément Massart for their advices, their teachings and the considerable amount of time they devoted to me. I would also like to thank Pr. Julien Hendrickx and Dr. Stéphane Moniotte with whom I started a first thesis. Although logistics did not allow me access to the needed dataset in time, I learned a lot about the human heart and its complications. Thanks again to Pr. F.-X. Standaert, Pr. M. Verleysen for accepting me on a new subject so late in the academic year. Special thanks to my parents who helped me craft adversarial examples manually, and thanks to my friends for the support and help with detecting typos. Lastly, if anyone reading this is not metionned in the list above, thank you for reading. If any of the mentioned subject interests you I greatly recommend reading more on them, it gets more and more fascinating and thought provoking over time.

Contents

Chapter 1

Introduction

1.1 Background

This document is a story of data, a story of data and humans and their interactions. Today, when we think of a person working with large amounts of data, we often think of a data scientist, a programmer. But journalists have been working with large data sets and trying to extract useful information from them since long before the birth of the first computer. Technologies have been used in the past for news production and distribution [1] and still do today, which makes the jobs of data scientists and journalists closely intertwined.

Some examples of technology being used in the field of journalism are: Automated news generation [2] [3], bots to distribute news online [4], testing different titles to see which works best [2]. This technology has made itself essential with the augmenting number of data made available to the journalists, but also for the reader, with more and more new stories coming out, there is a crucial need for good filtering and recommandation. Escaping the mass ditribution of news through news aggregators (Google News for example) seems to not be an option either. In the past, Belgian publishers forced Google News to remove their content from the platform, which they did, but later, the publishers came back on their decision [5]. But analysis has shown that news application design in startups was driven by more technical challenges than journalistic goals [6].

But algorithms are not perfect, while they are considered as objective guides, they carry with them the bias of the data given to them. Carthy O'Neil, data scientist, said quite concisely: "Algorithms don't make things fair if you just blithely, blindly apply algorithms. They don't make things fair. They repeat our past practices, our patterns. They automate the status quo. That would be great if we had a perfect world, but we don't." [7] An example of such problems happened when an algorithm used in the American justice system classified black people as presenting risks of recidivism twice more often than white defendants [8]. Not only can the training data be biased, the decisions made by a programmer while coding also are. The problem is much too big for us to tackle all at once. In this paper, we will focus on the recommendation aspect.

1.2 Scope

Recommendation of news articles is somewhat different from other recommendation techniques, it cannot work like content recommendation systems which use collaborative filtering. Simply put, collaborative filtering works by exploiting similarities between user interests. For example, if a lot of people have been interested in contents A, B and C, the algorithm will recommend C to a person interested in A and B [9][10]. News recommenders cannot rely on user input on the content. News, per definition, are new. Data is thus not available to use that type of recommandation engine and there is a need to represent the content in a machine-interpretable way to then associate the content to the user [11]. The association of content with the user is then done by analyzing her habits and/or social medial profile [3][12]. It should be made clear that news recommandation is not only something referring to news applications, but also news and social networking sites. Since we lack data to analyze the association with content to the user, we focus on the content representation. We will implement and try to attack state-of-the-art content representation algorithms.

A question that can be raised is: "Does it matter if an article gets incorreclty recommended?". While it may not be obvious that getting recommended an article that you do not care about is a problem, we give two examples of ways people could use these weaknesses for personal gain. First, since writers are more and more paid in accordance of the success of their online news articles [3], a writer could well write an article that is represented as belonging to a more popular category to increase her earnings. Another use case is for political gain, divulging fake news has been associated with influencing elections [13], making those news reach a higher number of people is a good way to amplify the phenomenon. Also in the context of elections, making articles belong to certain categories can help target an audience in order to influence their decision, this has proven to be sought after by politicians (in the recent Cambridge Analytica scandal for example [14]). More vastly, we see that the goal of creating a personalized news paper for everyone [11], removes a big part of the information filtering done by following an editorial line.

1.3 Outline

We focus on the content representation part of the recommendation algorithms. The following chapter sets the theoretical background for the work done in the rest of the document. It is there to set a solid basis for understanding what comes after. The influence of the various parameters is not mentioned yet. The paramters will be discussed when the tool is mentioned in the subsequent chapters. The first of these chapters is a description of the methodology we followed to obtain our results. It starts with the creation of a dataset, the preprocessing and the choices made to represent that data. Since the currently used algorithms are not publicly available, we implement them in the later chapters. First, we implement clustering algorithms, a first way to find patterns in our data, we then implement classification algorithms and select the two best performing models to attack in the last chapter. The last chapter is a description of the methodology used to make the attack and a presentation of some of our results. We end the document by giving a slighly more enlighted opinion on this problem.

Chapter 2

Background

This section contains intuitive descriptions and explanations on state of the art algorithms and tools for Natural Language Processing (NLP), it is not meant to be complete but rather a description of the tools used in the rest of this document. The influence of the parameters for each tool is discussed in the section where the tool is used. First, some NLP specific tools are introduced, then we discuss a few methods for clustering and classification. Then, we introduce the concept of adversarial examples, a way to trick the previously introduced classification techniques to provide wrong outputs. We then explain how to find such examples and how to mitigate their impact.

2.1 NLP Tools

2.1.1 Word Embeddings

Word embeddings, and more specifically the Word2Vec model proposed in [15], allow the representation of words in a continuous vector space. The words represented in this space, keep some semantic meaning and the relationship between the word vectors is coherent with the relationship of the words in the language. Vector arithmetic has a correspondance in the semantic domain. We use the following 2 examples (in French, since we use a French model) to illustrate the result of this modelling.

Input	**Similarity**	**Output**
$\overline{roi} - \overline{homme} + \overline{femme}$	$\simeq$	$\overline{reine}$
$\overline{journalisme} - \overline{\text{éthique}}$	$\simeq$	$\overline{scoop}$

The similarity function above returns the most similar word vector to the input. The similarity used in that space is cosine similarity, defined as:

$$cossim(\mathbf{V}_1, \mathbf{V}_2) = \frac{\mathbf{V}_1 \cdot \mathbf{V}_2}{\|\mathbf{V}_1\|_2 \|\mathbf{V}_2\|_2}$$

2.1.2 Term Frequency - Inverse Document Frequency

Term Frequency - Inverse Document Frequency or TFIDF is widely used technique to identify salient words of a text. It can be used to represent by only a few words every document of a large corpus, as it has been done for the Iraq war logs [16].

TFIDF is a method that gives a score to words in a text as a function of their number of occurences in the text compared to the collection of documents. The intuitive explanation is that the more frequent a word is in a text, the higher its score will be. On the other hand, the more frequent a word is in the collection of texts, the lower its score will be.

Methods

There are numerous ways to implement this algorithm and we chose to test the parametrizations listed in Table 2.1 [17]. The following symbols are used:

- $n_{w,d}$: number of appearances of word w in document d
- $n_{w,D}$: number of appearances of word w in the corpus
- $\max_{w,d} = \max_i n_{wi,d}$: number of appearances of the most frequent word in d
- D: number of documents in the corpus
- D_w: number of documents containing word w
- W_d: number of words in document d

Number	TF	IDF
0	$n_{w,d}/W_d$	$\log(1 + D/D_w)$
1	$1+\log(n_{w,d})$	$\log(1 + D/D_w)$
2	$0.5 + 0.5 * n_{w,d}/\max_{w,d}$	$\log(1 + D/D_w)$
3	$0.5 + 0.5 * n_{w,d}/\max_{w,d}$	$\log(1 + D/(D_w * n_{w,D}))$

Table 2.1: TFIDF parametrization methods

Finally:

$$TFIDF(\mathbf{a}) = TF(\mathbf{a}) \times IDF(\mathbf{a})$$

An example of the five highest ranked words by parametrization 0 is shown below:

Article title	Gaza : Un nouveau tir de roquette en direction d'Israël menace la trêve
Article text	Une roquette a été tirée de Gaza mardi soir en direction d'Israël, menaçant une trêve qui semblait tenir au lendemain d'un nouvel accès de violence, Benjamin Netanyahu ayant prévenu qu'il était prêt à ordonner une offensive si nécessaire, à deux semaines des législatives. [...]
Keywords	Roquette, Gaza, Palestinien, Hamas, Israël

2.2 Clustering

2.2.1 k-Means

The clustering of vectors $v \in V$ a vector space by k-Means in k clusters aims at finding k centers and then assigning each data point to the cluster corresponding to the closest center. The algorithm, also known as Lloyd's algorithm, presented in [18] is well described in the literature and we will not describe it here. For n+1 data points x and an arbitrary number k of μ_j's, the cluster centers, the algorithm aims at minimizing the expression:

$$\sum_{i=0}^{n} \min_{j \in \{1,...,k\}} (||x_i - \mu_j||^2)$$

2.2.2 Latent Dirichlet Allocation

Proposed in [19], Latent Dirichlet Allocation is a method for text clustering. The hypothesis it poses is that each document is a sample from a distribution of topics, each topic being a sample from a distribution of words. This means that we can use LDA for clustering by choosing to group together documents that have the same topic in higher proportion. But it also means that LDA can be used as a dimensionality reduction technique, representing each article as a vector or the proportions of each topic in the article.
Let us explain how a document would be generated using this model: Figure 2.1 represents the model graphically: the top portion of the image represents the topics, we choose that there are K topics in the corpus of M documents. Each topic is assigned a multinomial probability distribution of words drawn from a Dirichlet distribution with parameter λ:

$$\forall k \in \{1, ..., K\} \text{ draw } \beta_k \sim Dirichlet(\lambda)$$

The second line of the figure shows that each of the M documents is itself represented as a multinomial distribution of topics, drawn from a Dirichlet distribution with parameter α:

$$\forall d \in \{1, ..., M\} \text{ draw } \theta_d \sim Dirichlet(\alpha)$$

Then, for the N word positions in a document d, a topic assignment $z_{d,n}$ is drawn from the distribution θ_d.

$$\forall n \in \{1, ..., N_d\} \text{ draw } z_{n,d} \sim Multinomial(\theta_d)$$

At last, the word at position n is drawn from the distributions of the assigned topic for that position.

$$\forall n \in \{1, ..., N_d\} \text{ draw } w_{n,d} \sim Multinomial(\beta_{z_{n,d}})$$

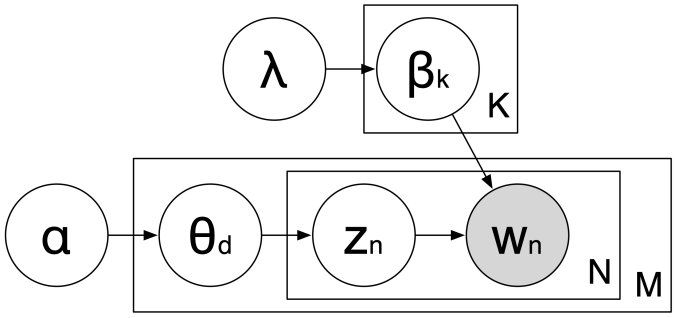

Figure 2.1: Latent Dirichlet Allocation plates representation (retrieved from [20])

Fixing λ, α and K the following algorithm allows to retrieve the distribution of words in topics, and the distribution of these topics in documents of a given corpus D.

Algorithm 1 Topic Assignment Using LDA

Choose λ and α parameters of a Dirichlet distribution
Choose K the number of topics present in the Corpus
$\forall n, d$ randomly choose $z_{n,d} \in \{1, ..., K\}$
while e<max epochs **do**
  for $d \in \{1, ..., M\}$ **do**
    for $n \in \{1, ..., N_d\}$ **do**
      set $z_{n,d}$ = nil
      for $k \in \{1, ..., K\}$ **do**
        $\mathrm{n}_{d,k} = \alpha +$ Number of times topic k is used in d
        $V(k, w_{d,n})$ = Number of times word w_n is used by topic k
        $\mathrm{aff}_{k,w_n} = (V(k, w_{d,n}) + \lambda) / \sum_i (V(k, i) + \lambda)$
      end for
      Draw $z_{n,d}$ s.t. $p(z_{n,d} = k) = \mathrm{n}_{d,k} \times \mathrm{aff}_{k,w_n} / \sum_j \mathrm{n}_{d,j}$
    end for
  end for
  e++
end while

2.2.3 Spectral Clustering

Spectral clustering is a method for graph clustering, it is especially useful for cases like the one depicted on Figure 2.2.

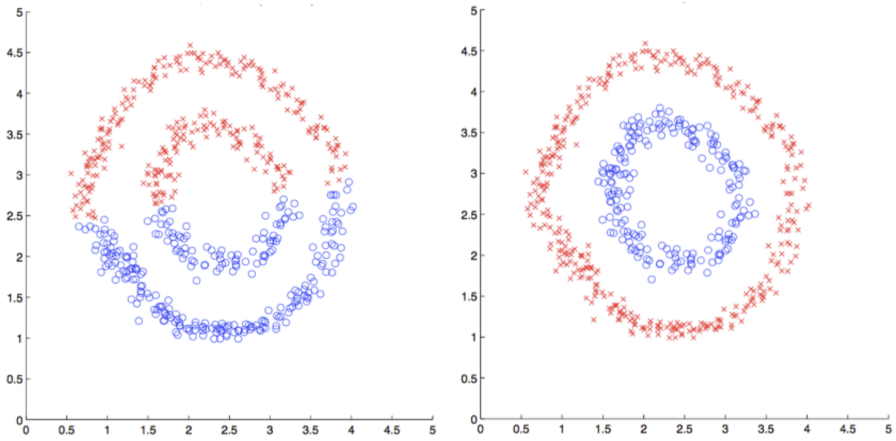

Figure 2.2: Clustering of concentric circles geometry with Kmeans(left) and with spectral clustering (right) (images taken from [21])

As it is shown, all data geometries are not suited to be represented by centroids in the space that they are in. This method, used in image processing, creates a graph by connecting neighbouring data points, then uses the following method to use the graph to cluster the original data [22]:

- Compute the similarity matrix $S \in \mathbb{R}^{(n \times n)}$, the matrix of inter-articles similarity, n is the number of articles.
- Fix the number of clusters k
- From the similarity matrix, create a graph by connecting neighbouring points under a certain threshold, or m-nearest neighbours. Create $W \in \mathbb{R}^{(n \times n)}$ such that:

$$w_{ij} = \begin{cases} s_{ij} & \text{if i and j are connected} \\ 0 & \text{otherwise} \end{cases}$$

- Create a diagonal degree matrix $D \in \mathbb{R}^{(n \times n)}$ such that $D_{ii} = \sum_j w_{ij}$
- Compute the unnormalized graph Laplacian $L = D - W$
- Compute the k eigenvectors of L with lowest eigenvalue: $u_i, i \in \{1, ..., k\}$
- Let $U \in \mathbb{R}^{(n \times k)}$ the matrix with columns $u_1, ..., u_k$ and write the y_i lines of U for $i = 1, ..., n$
- Run the K-Means algorithm on the y_i vectors to obtain clusters $A_1, ..., A_k$
- Obtain clusters $C_1, ..., C_k$ such that $C_j = \{i | y_i \in A_j\}$

2.2.4 Louvain

Another method to cluster data based on a graph where neighbouring data points are connected is the Louvain algorithm. Presented in [23], it is aimed at finding communities in networks. Representing the targeted network as a graph, each vertex being a member of the network and the edges between vertices representing their interactions, it aims at maximizing the modularity.
Defined in [24], modularity, a measure of comparison between the intra-communities links and the inter-communities links. It is a good indicator of the quality of division of the network in communities. It differs from most clustering algorithms, in the sense that the underlying structure of the data will fix the number of cluster the algorithm outputs, not the user.

Algorithm 2 presented below uses the change of modularity from moving an isolated node i in a community C as:

$$\Delta Q(i, C) = \left[\frac{\sum_{in} + k_{i,in}}{2m} - \left(\frac{\sum_{tot} + k_i}{2m}\right)^2\right] - \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right]$$

"where $\sum_{in}$ is the sum of weights of the links inside C, $\sum_{tot}$ is the sum of the weights of the links incident to nodes in C, k_i is the sum of the weights of the links incident to node i, $k_{i,in}$ is the sum of the weights of the links from i to nodes in C and m is the sum of the weights of all the links in the network"[23]. The obtained gain in modularity when moving a node from a community to another is computed by first removing the node from its community, then adding it to the other one.

2.3 Classification

2.3.1 Random Forest

Introduced in [25], the Random Forest Classifier algorithm works on the basis of majority votes of random decision trees. In order to get different decision trees when training, bootstrap aggregating (or bagging) is used [26]. The idea behind this technique is to treat the training set as a good enough representation of the observed process to estimate the probability distribution of said process from it. Samples are then drawn from that distribution to train the model. Random decision trees are then formed minimizing the Gini impurity of the split or maximizing its information gain. The impurity is defined as such:

$$Gini\_impurity = 1 - \sum_{c \in C} [p_c]^2$$

And the information gain, that can also be seen as the reduction of uncertainty on the class by the split:

$$IG = H(parent\_node) - \mu(H(child\_nodes)) \text{ with } H \triangleq -\sum_{c \in C} p_c \log p_c$$

where p_c is the probability of class c, H the entropy and μ an averaging function. Through the voting of different decision trees, the algorithm is robust against overfitting.

Algorithm 2 Louvain - Finding communities in a network

To each node assign a community, C_i is the community containing i
while change in communities occur **do**
  while Improvements in ΔQ are possible by moving one node **do** ▷ First phase
    for each node i **do**
      Compute $\Delta Q(i, C_i)$ ▷ As if i was isolated
      for each neighbour j of i **do**
        Compute $\Delta Q(i, C_j)$ ▷ As if i was isolated
      end for
      if $\max_j \Delta Q(i, C_j) > \Delta Q(i, C_i)$ **then**
        Move i to C_j
      end if
    end for
  end while
  Create a new graph with weight matrix A' ▷ Second Phase
  for $c \in C$ **do** ▷ Each community is a new node
    $A'_{cc} = \sum_{i \in c} \sum_{j \in c} A_{ij}$
  end for
  for $c_1, c_2 \in C, c_1 \neq c_2$ **do**
    $A'_{c_1 c_2} = \sum_{i \in c_1} \sum_{j \in c_2} A_{ij}$
  end for
  $A \leftarrow A'$
end while

2.3.2 Multinomial Naive Bayes

The Multinomial Naive Bayes algorithm is a pretty basic text classification algorithm that works quite effectively as far as text classification goes [27]. The way it estimates the class c in which to put a document d is as follows [28]: First, it estimates that the presence of individual words is an indicator of the class c of the document.

$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c)$$

Where $P(t_k|c)$ is the probability to observe word t_k in a c-class text. The Maximum A Poteriori estimate of the class of a document d is then:

$$\hat{c}_{MAP} = arg \max_{c \in C} \hat{P}(c|d) = arg \max_{c \in C} \hat{P}(c) \prod_{1 \leq k \leq n_d} \hat{P}(t_k|c)$$

and consequently:

$$\hat{c}_{MAP} = arg \max_{c \in C} \log(\hat{P}(c)) + \sum_{1 \leq k \leq n_d} \log(\hat{P}(t_k|c))$$

The estimator of the prior probability of a class appearing is quite evident:

$$\hat{P}(c) = \frac{M_c}{M}$$

where M_c is the number of documents of class c and M is the size of the training set. To avoid the problems raised by probabilities of never occurring terms in a class to be 0, a smoothing factor $\lambda \in]0, 1]$ is introduced such that:

$$\hat{P}(t_k|c) = \frac{T_{ct} + \lambda}{\sum_{t' \in V}(T_{ct'} + \lambda)}$$

where T_{ct} is the number of times word t appears in category c, V is the vocabulary of all total words. Creating a vector $b \in \mathbb{R}^{|C|}$ the number of different classes such that:

$$b = [\log(\hat{P}(c_1)), \log(\hat{P}(c_2)), ..., \log(\hat{P}(c_{|C|}))]^T$$

and a matrix $W \in \mathbb{R}^{|C|x|V|}$ such that:

$$W_{ij} = \log(\hat{P}(t_j|c_i))$$

The classification of a new article, given its word count vector $\mathbf{a} \in \mathbb{R}^{|V|}$ (a vector where each component is a count of the number of appearances of the term in the article) can then be written as:

$$arg \max_{c \in C} R_c \text{ where } R = W\mathbf{a} + b$$

2.3.3 Neural Networks

Neural networks come in various shapes and forms. We will here describe the Multilayer Perceptron (or MLP) for the sake of argument but current Natural Language Processing research also uses more complex models like the Recurrent Neural Networks (RNNs), the Long Short-Term Memory Networks (LSTMs) and Bidirectional Recurrent Neural Networks (BRNNs).

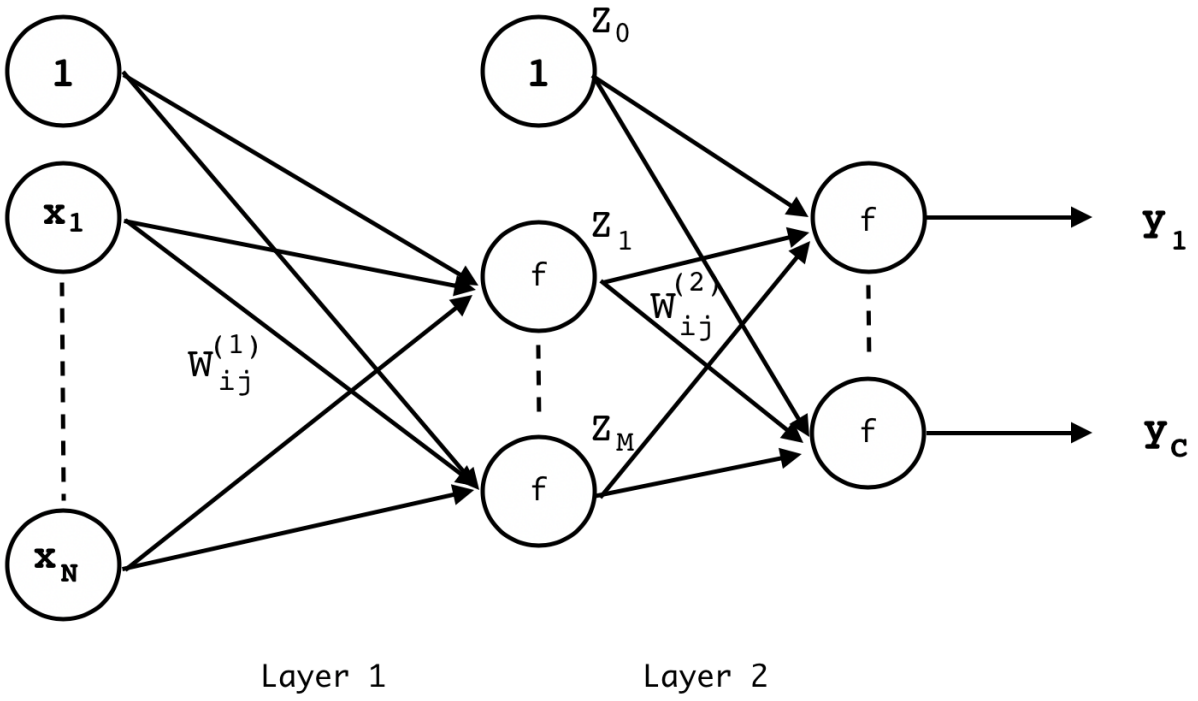

Figure 2.3: MultiLayer Perceptron Schematic

MultiLayer Perceptron

The MultiLayer Perceptron or MLP best described in pair with its schematic representation shown on Figure 2.3. (The notations, equations and the figure are mainly taken from or inspired by [29]). Let us call the circles on the schematic neurons and the set of connections between two columns of neurons and the column on the right: a layer. The schematic only presents two layers but the number of layers and of neurons for each layer is completely arbitrary. Each set of arrows merging before a neuron represents a linear combination of the neurons at the tail of each of the arrows. For example, the input of the bottom neuron of the first layer is a weighted sum of the input and the bias term 1, writing that input a_M:

$$a_m = \sum_j W_{mj}^{(1)} x_j \qquad \text{where } x_0 = 1$$

Each neuron outputs a nonlinear function of its input (except for the input neurons which have no input), this nonlinear function f is called the activation function of the neuron. The most common activation functions are:

- Rectified Linear Unit: $Relu(x) = max(0, x)$
- Logistic function: $logistic(x) = \frac{1}{1+\exp(-x)}$
- Hyperbolic Tangent: $tanh(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$

On the schematic, the activation function is the same for every neuron for the sake of simplicity, but it can be layer-specific (even neuron-specific but there is little to no interest

in doing that). Writing $z_i^{(l)}$ the output of neuron i of layer l:

$$z_i^{(l)} = f(a_i^{(l-1)}) = f(\sum_j W_{ij}^{(1-1)} z_j^{(l-1)}) \tag{2.1}$$

For a classification task, the output are usually passed through a $softmax$ function, allowing the outputs to be bounded between 0 and 1 and to sum up to 1. We call these numbers, the *confidence* of the network in its prediction.
Using equation 2.1, one can propagate the input through the network to find its ouput value. To train the network, a criterion to be optimized has to be chosen. After choosing an error function E, usually the sum of the squared errors, we can compute the gradient of that error and optimize the weights accordingly. The computation of the error gradient can be done this way:

$$\frac{\delta E}{\delta W_{ij}^{(l)}} = \frac{\delta E}{\delta a_i^{(l)}} \frac{\delta a_i^{(l)}}{\delta W_{ij}^{(l)}} = \frac{\delta E}{\delta a_i^{(l)}} z_j^{(l-1)}$$

writing $\frac{\delta E}{\delta a_i^{(l)}}$ as $\delta_i^{(l)}$ we identify two different cases:

- Output Layer:

$$\delta_i^{(l)} = \frac{\delta E}{\delta y_i^{(l)}} \frac{\delta y_i^{(l)}}{\delta a_i^{(l)}}$$

  The first fraction is the derivative of the error criterion, which is known. The second one is equal to $f'(a_i^{(l)})$ which is also known.

- Hidden Layer:

$$\delta_i^{(l)} = \sum_k \frac{\delta E}{\delta a_k^{(l+1)}} \frac{\delta a_k^{(l+1)}}{\delta a_i^{(l)}} = \sum_k \left[\delta_k^{(l+1)} \frac{\delta \left(\sum_{j \in (l)} W_{kj}^{(l+1)} z_j^{(l)} \right)}{\delta a_i^{(l)}} \right] = f'(a_i^{(l)}) \sum_k \left(\delta_k^{(l+1)} W_{ki}^{(l+1)} \right)$$

  The δ-term is dependent on the δ-term of the next layer. One must then compute the errors starting from the output layer towards the input, we call this back-propagation.

The weight are then adjusted using the error gradient, using conjugate gradients for example.

2.4 Adversarial Examples

Adversarial examples, first talked about in [30], were defined as imperceptible non-random perturbations to the input of a neural network that allowed to arbitrarily change its prediction. A powerful example, displayed on Figure 2.4, is in the case of image classification, or object detection in images: by cleverly changing some pixel values ever so slightly the model can be fooled to misclassify the image [31]. From these examples, real life attacks have also been made possible, for example by sticking a few stickers on a road sign as shown on Figure 2.5. A stop sign on the road can be made perceived as a speed limit sign

by the type of network that can be used in a self-driving car [32].

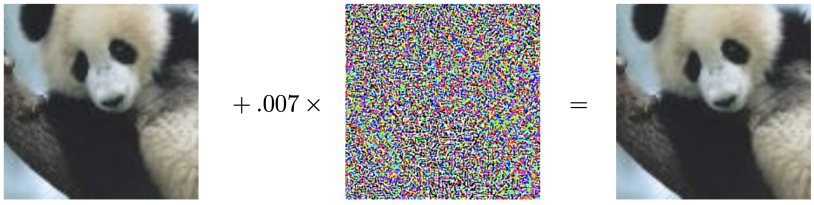

Figure 2.4: The picture on the left is classified as a panda but the one on the right as a gibbon (from [31])

Figure 2.5: Stop sign classified as a speed limit sign (from [32])

2.4.1 Attack

To attack a model with adversarial examples, it is crucial to understand where they emerge from. It seems that models are valid on a thin manifold in the input space, one requires to move outside of that thin comfort zone of the model to fool it [33]. Figure 2.6 shows how they could appear in the input data space.

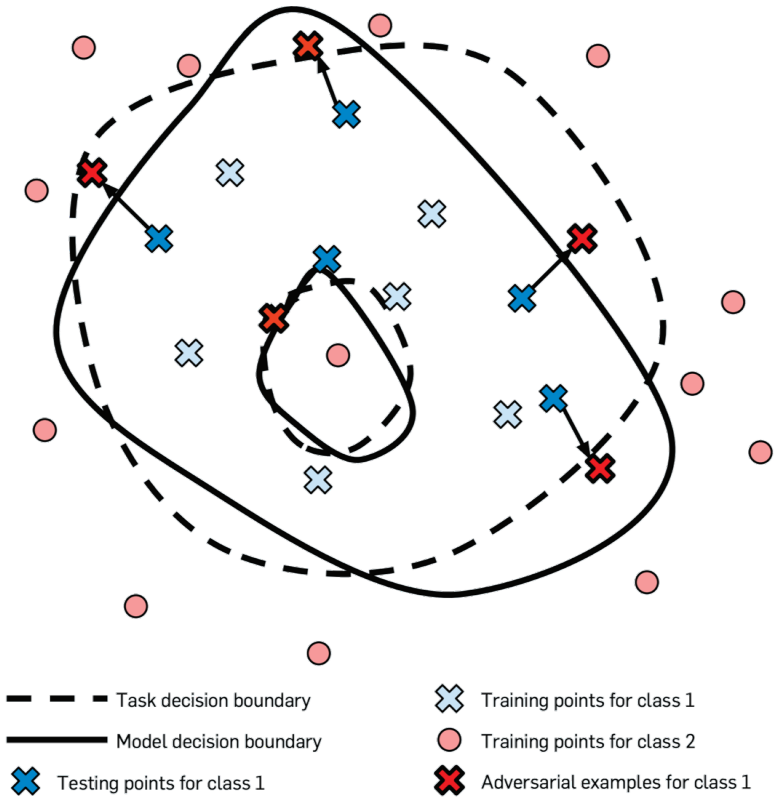

Figure 2.6: Adversarial examples (schematic from [34])

Adversarial examples are located close to decision boundaries, indeed, moving a point slightly to make it cross only of the two boundaries is interesting. Let us take the two-class

example of classifying pictures of dogs and cats:

- Either the image point crosses the task decision boundary (the dashed line on the graph) from the cat set to the dog set without crossing the model decision boundary (the solid line). This means that the picture is a picture of a dog but will be classified as a cat.
- If, on the other hand the image point moves across the model decision boundary but not the task decision boundary, it means that the cat picture will be classified as a dog even though it still shows a cat.

Note: The model can be fooled to output a wrong class with high confidence even if the perturbation is very small.

Adversary Capabilities

Defined in [35], the adversary capabilities against a Deep Neural Network can be summed up as his level of knowledge of the network. They are given below in descending order of attacking power:

- Training data and architecture: the adversary knows everything about the model, its number of layers, nodes, the training data used, all the weights and activation functions used. In short: everything.
- Network architecture: access to everything but the data used to train the network.
- Training Data: the adversary has access to a sample of the training data
- Oracle: The adversary can send an input to the model and observe the output (much like a chosen-plaintext attack)
- Samples: The adversary has access to pairs of input-output data from the target model.

In the first three scenarios, it is said that the attacker has access to the model as a white box. For the two other cases, as a black-box.

White Box attack

The following method requires the attacker to have access to the model architecture. If the attacker does not have access to such information, but has accesss to samples of the training set, other models can be trained on these samples and adversarial examples crafted fom these models, this is called transferability. The rate of success at which attacks can be transferred depends on the source and target models but even for the Neural Networks for which it is quite low [36] it is still significant enough to be viable as an attack technique.
Let us thus assume that the attacker has access to the model architecture, the Fast Gradient Sign (FGS) approach is a method to easily craft adversarial examples. It

consists in computing the gradient of the error made by the network (using a first-order approximation for example):

$$J(x', \theta) \approx J(x, \theta) + (x' - x)^T \nabla_x J(x)$$

Where J is the error function, θ the model parameters, x the input to be modified and x' the modified input. Since we want to maximize the error. We want to maximize:

$$J(x, \theta) + (x' - x)^T \nabla_x J(x)$$

Since $J(x, \theta)$ is constant we can choose:

$$x' = x + \epsilon \ sign(\nabla_x J(x))$$

Where ϵ is the maximum modification for each of the dimensions of the input. Other, more complex, techniques do exist but are out of the scope of this document.
What has been found is that not only do adversarial examples transfer quite well between machine learning methods, the direction taken to go from a naturally occuring sample to an adversarial example can be added onto other naturally occuring samples to create new adversarial samples.

2.4.2 Defense

The defense against adversarial examples is still an open research question, many attempts to mitigate this problem have been proven to be bypassable [37]. One of the more robust methods is adversarial training: generating adversarial data to train the model with, so that it can learn to not be fooled by such trickeries. But as the saying goes, the model is only as good as the data it is trained with. Since fast techniques to generate adversarial examples are required to produce high number of training data, more simple models (like FGS) are used to generate them. The model is then quite robust to the type of adversarial examples generated by the method used to train it, but still weak to other methods not used in training.

Chapter 3

Framework

This section describes the framework followed to obtain the results presented in the following two chapters. All the code used for the results presented in this document was written in Python. Figure 3.1 is a visual representation of this chapter and the two subsequent, each level corresponding to a section of this chapter and the last two stages represent the two following chapters of the document.

Every step from collecting the data to classifying it requires various design choices, each decision made at a certain stage will influence all the following. While it is essential to try and determine the optimal parameters for every step, trying to optimize all the parameters at the same time is unrealistic. That is why, for every step we analyse the influence of the model parameters.

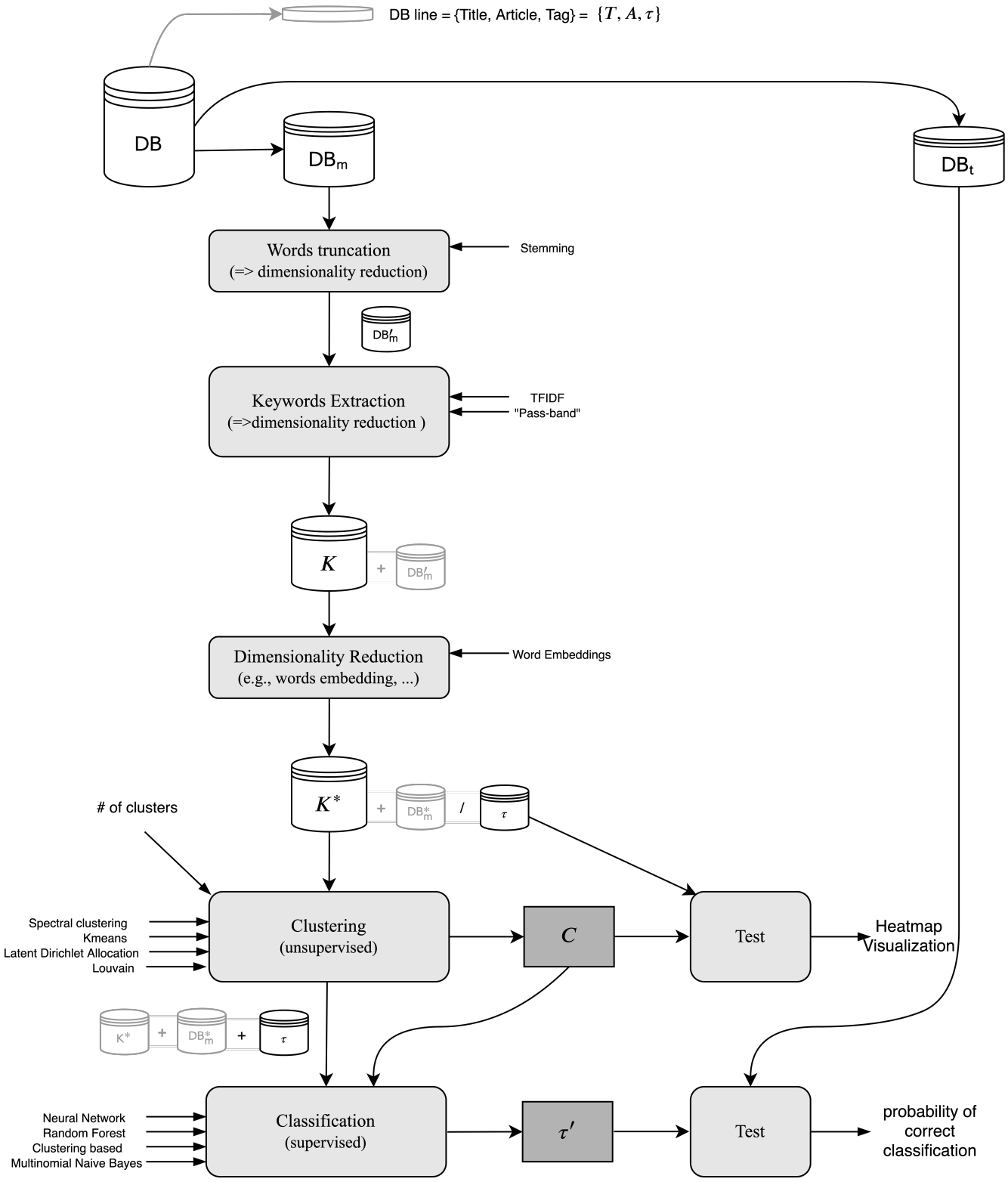

Figure 3.1: Framework

3.1 Database

Composition

Our dataset consists of 5232 articles from the Belgian news site Le Soir [38]. To collect them, we designed a web crawler that for each article collected its:

- Title
- Body
- Category

Each of these elements is first stored as a string of characters, waiting to be preprocessed. We randomly shuffle our dataset, then take 80% of it as our model set (DB_m) and the remaining 20 % as our test set (DB_t) which will be kept apart until final testing. All the numbers discussed hereafter are thus based on DB_m.

The article categories are a hierarchy of categories, subcategories and subsubcategories. We simplify them by taking the highest category to make our testing easier. Table 3.1 shows the way we name them for the rest of this document.

Name	**# of articles**	**Original sub-categories**
Culture	580	musiques: 122, livres: 162, cinéma: 145, ...
Médias	104	/
Belgique	772	politique: 186, société: 357, régions: 189
Économie	519	mobilité: 163, entreprises: 117, ...
Monde	772	Europe: 131, France: 196, Union Européenne: 132, ...
Sports	825	football: 499, cyclisme: 135
Autres	561	opinions: 523, sciences et santé: 9, techno: 16, ...

Table 3.1: Training categories simplified

We see that the category 'Autres' contains the subcategory 'opinions' which can be about anything and everything. This will probably be harder to classsify or form 'bridges' of data point between different categories. It would not be realistic to just discard it, but we keep this fact in mind dor the rest of this document.

3.1.1 Representation

To represent the articles contents and titles, we use the Bag Of Words (BOW) representation, meaning that the order of the words in the articles does not matter. Indeed, each string of text is represented as a vector as shown below:

"The dog is barking" vector representation

...	barking	...	dog	...	is	...	news	...	the	...
[0...0	1	0...0	1	0...0	1	0...	0	...0	1	0...0]

We represent each article with two vectors, corresponding to the title (**T**) and the text (**A**) of the article, and a scalar, representing its category ($\boldsymbol{\tau}$). In order to represent an article in such a way, we need to do some preprocessing on the raw text collected online.

3.2 Preprocessing - Word truncation

The first step to transform our articles into a pair of vectors and a scalar is to tokenize our strings of data into single-word strings [39]. The punctuation marks and numbers are then removed. This gives us 64680 different words, and each article can then be represented as a vector of size 64680. To reduce this number without losing crucial information we stem each word [40]. Stemming is the process of cutting off the suffix of a word to make any variations of the same stem equal to only one symbol. This allows the different conjugations of a verb to be taken as a single word for example. We create a dictionary containing stems and the words from the original dataset they represent, as an example, these are two entries from the dictionary:

- `dict['journali']`= ['journalier', 'journalières', 'journalière']
- `dict['journal']`= ['journal', 'journaliste', 'journalistes', 'journaux', 'journalisme']

The stemming of our words allows us to respresent our texts as vectors $A, T \in \mathbb{R}^{|W|}$ where $|W| = 37109$. We write $\mathbf{a} = T + A$ the vector of each articles title and text appended.
Note: The stemming of words does not necessarily return a word, another technique, called lemmatization aims "to return the base or dictionary form of a word" [41]. We found no open-source python algorithm to do this for the french language, so, when an actual word was needed in place of a stem, we took the most frequent word corresponding to it in our corpus.

3.3 Keywords Extraction - Dimensionality Reduction

The two following methods aim at reducing the number of considered words per article. The first one does it by selecting words of highest importance in an article while the second one simply removes words of least importance.

3.3.1 TFIDF

Notation

The transformation of a dataset by the representation of each article by a vector in the space of the N best-rated words for each article by TFIDF method M is denoted by **TFIDF_M_N**, if M is omitted, M=0.

Comparison of different parametrizations

It is impossible to determine a metric of what makes a good keyword without considering its use. We simply observe that, depending on the parametrization, the methods will select words in different parts of the word-usage histogram. These histograms are depicted on

Figure 3.2. The number on the horizontal axis corresponds to the frequency of occurence of the word, the more a word is used in the French language, the most left on the axis it will be represented.

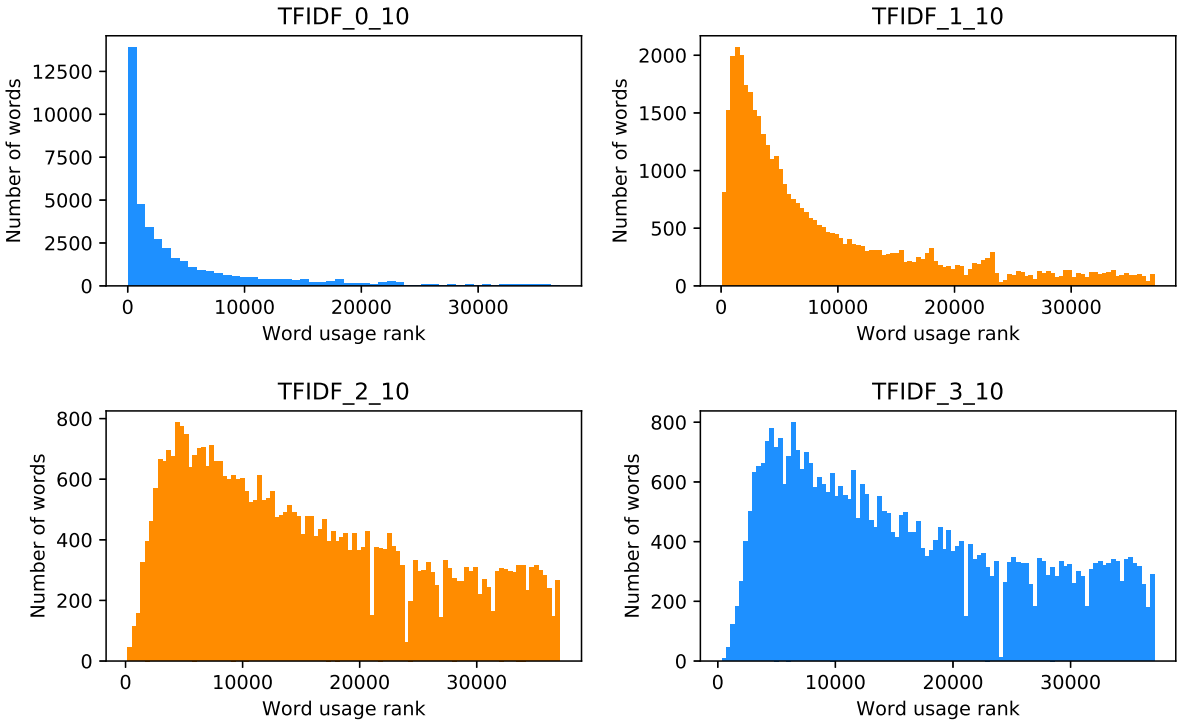

Figure 3.2: TFIDF word usage (10 words kept per article)

It appears that methods 2 and 3 will select words that occur rarely in the corpus, this is probably not desirable since a word appearing only once in our set of documents is probably rarely used in the language or too specific to link articles together. Method 0, on the other hand, might select frequent words too often, probably selecting stop words, words void of information about the discussed topic. The following method aims at removing such words.

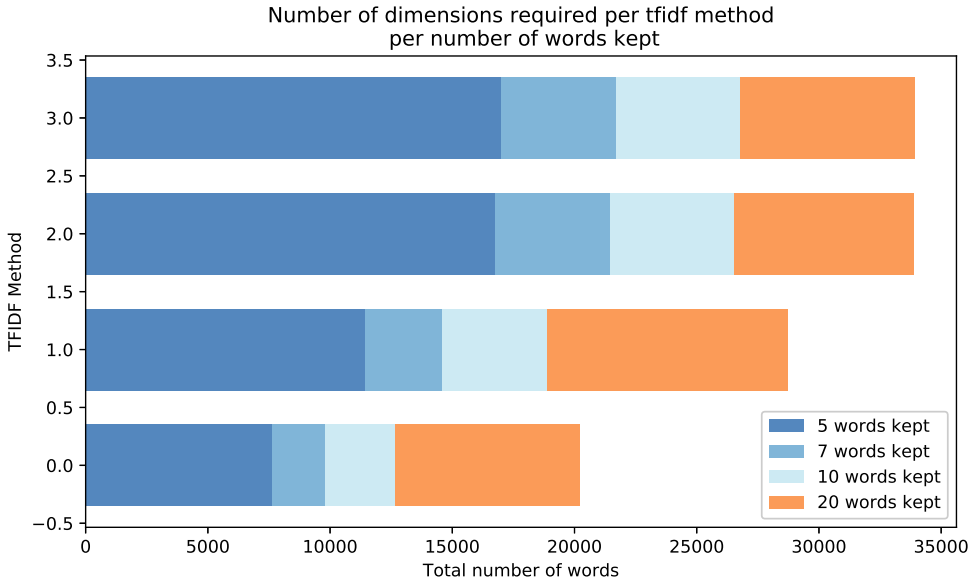

Figure 3.3: Word usage as a function of number of words kept per article

3.3.2 Irrelevant words removal

The graphs from Figure 3.2 are quite close to frequency-response plots of filters. Another technique to reduce the number of words used to represent an article is then to make a sort of band-pass filter. This corresponds to removing words appearing too often or too seldom. We assume that these words will not be too helpful for classifying articles, since we need words that link at least a few articles together but not too many.

Notation

The remaining dataset after the removal of the words present in more than P% of the text and less than N articles is noted as: **S_P_N**

3.4 Vector Densification - Dimensionality Reduction

Extracting keywords from the articles might have reduced the total number of different words (and consequently dimensions) we need to represent an article, but we have made our sparse vectors even sparser. An article of 500 words was represented using a vector of dimension ~ 37000, being filled at $\sim 99\%$ with zeros. Representing it with its five highest ranked tfidf words will make it use 5 dimensions out of the ~ 8000 (see Figure 3.3) thus leaving $\sim 99.9\%$ of the dimensions unused.
The main issue here is that there is no semantic distinction between words. If a copy of an article is produced by replacing every word with one of it's synonyms, the article vectors will be totally different, to bring the semantics into the vector domain, word embeddings are the most useful tool (see section 2.1.1)

3.4.1 Word Embeddings

Since computing a word embedding model is data- and computationally demanding, we do not compute a new vector model from scratch and take a precomputed model that we will use to generate a lookup table to match our words to vectors. We take Jean-Philippe Fauconnier's precomputed model of dimension 200 [42].
The articles can now be represented with Nx200 with $\sim 0\%$ sparsity, where N is the number of keywords kept per article. It should be noted that Principal Component Analysis can be used on the word embedding vectors to further reduce the dimensions but we chose to keep their original dimensions.

3.5 First results and discussion

As it is to be expected from Zipf's law [43], a lot of the required dimensions are used for words that only appear once in our corpus. To reduce the dimensionality of our data and have a consistant size of word vectors, we decide to cut the vector, keeping only the 20000 most frequently occuring words. Doing this only removes words appearing one single time and allows us to have a standard vector representation. For clustering we still use various representations of words, but for classification, we stick with the vectors of size 20000 to reduce the number of combinations to try. We designate these bag-of-words vector by the name *article vector*.

Other considered representations that did not make it into this document because they did not lead to sufficient results were:

- **Representations taking into consideration pairs or higher order relationships of words inside a text:** These representations, while interesting in the information they offer, made the high dimensionality problem even worse. While we lose information by neglecting the order of the words, it should be mentionned that
  - Word Embeddings are constructed based on the positions of words relative to one another
  - Higher order relationships of words are taken into consideration by Latent Dirichlet Allocation (see 4.2) and the MultiLayer Perceptron (see 5.3.4)
- **Representation of an article by its title only:** While a human being could probably classify article by simply reading their title, it was proven to be hard to do for two simple reasons:
  - Titles use a few words high in information, not often observed within other articles of our corpus. Thus, they do not constitute good links between articles. These words are often proper nouns, we would need a Named Entity Recognition (NER) module to be able to associate deeper meaning to single words. Such modules associate named entities with what they represent.
  - Since the words are seldom used, the sparsity is even worse. There is less text and thus less words used overall, but each title is often less than a dozen words long.

Chapter 4

Clustering

4.1 Goal and succes metric

The aim of clustering is to group similar documents together. Finding an automated way of assessing the quality of these clusters is not possible without determining the end goal of the clustering. In this case, the optimal goal is to obtain 7 clusters, each one containing all the articles from a category and only that category. For every algorithm allowing us to fix the number of clusters, we will fix it to 7 to have a basis for comparison. The way we will analyse and compare the results is via a heatmap as displayed in Figure 4.1.

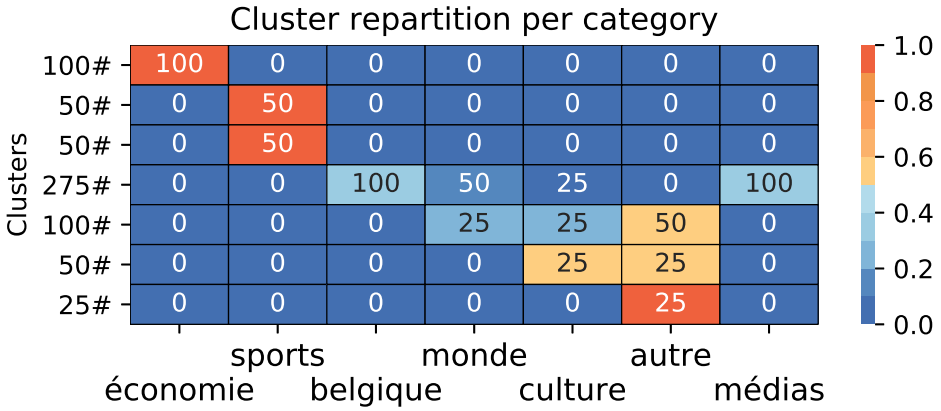

Figure 4.1: 'Heatmap' visualization scheme for clustering results

Horizontally, each cluster is represented with the number of articles in the cluster on the left. The color of a box is related to the proportion of the category in the cluster. It is appearant that seeing red in a cluster is a good sign, it means that it can discriminate between categories. Vertically, we observe the different clusters that a category is in, for example, the 'sports' category is in two different clusters, but no other category is present in that cluster. The media category on the other hand is located in only one cluster but that cluster is full of articles of different categories. Good indicators of the quality of the clustering scheme are the number of red boxes and making sure that no cluster with no red has a too large number of articles inside it (number on the vertical axis).

For each of the clustering methods proposed below, we will link the parameters and their influence with the theory layed in the previous chapter. We only show results of the parametrization that worked the best for each method. If the reader is interested to observe the influence of the parameters, Appendix A shows the results for various parameters. The number between brackets next to the name of the clustering method is clickable and will send the reader straight to the parameter selection appendix.

4.2 Latent Dirichlet Allocation [A.2]

As described in 2.2.2, Latent Dirichlet Allocation represents an article as a weigthed combination of topics. Since we desire to put each article in one cluster, we group articles having the highest weighted topic in the same cluster. The two parameters that we have influence over are the parameters of the Dirichlet distribution of words in topics λ and of the distribution of topics in documents α. By design, λ is the same for every topics and α is the same for all the documents. The influence of these parameters is depicted of Figure 4.2.

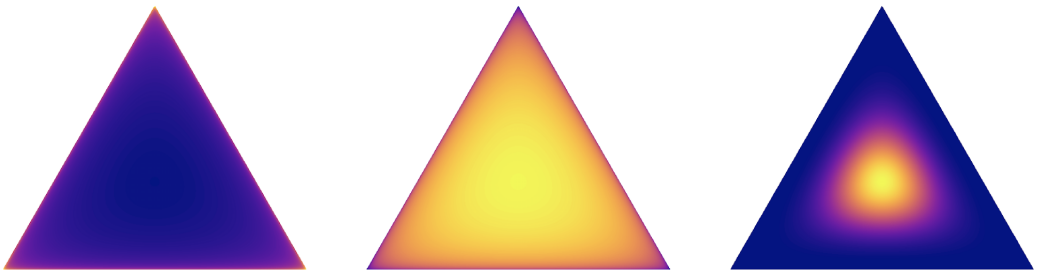

Figure 4.2: Influence of the Dirichlet parameter on the distributions (from left to right $\lambda = \{0.95, 1.1, 5\}$

Let us explain the relationship between α and the three triangle plots. Assume that we fix the number of topics to be equal to 3. Each corner of the triangle represents a topic, if an article is fully represented by a single topic, its point will be exactly on the corresponding corner of the triangle. On the other hand, if an article is an even representation of 3 topics, its point will be in the center of the plot. So the furthest a point is from a corner, the least the weight of the corresponding topic is important. The plots from Figure 4.2 represent the probability of apppearances of points on various locations of the triangle, the lighter the color, the higher the probability of appearance of a point. We then observe that a higher λ will lead in a document being more an even mixture of the different topics, and a low λ meaning that a document is composed of few topics with high weights. The reasoning for α is similar.

4.3 k-Means [A.1]

We can use TFIDF and k-Means in two different ways to cluster our articles:

- We select the N first words selected by TFIDF for each article and represent the articles in a new space containing only these words. Then we use k-Means in that space.
- We select the first word selected by TFIDF for each article and represent the article by the transformation from that word into the embeddings space. We then use k-Means in that space.

It is appearant that the first solution will still represent articles in too high number of dimensions and that the second solution, representing each article with a single word is too limiting to allow the algorithm to group them by category. A way to mitigate this would be to take the N first selected words by TFIDF and append their embeddings representation to represent each article as a vector of dimension Nx200. But again, it appears that if the first word of article 1 and the second word of article 2 are highly similar, the algorithm will not be able to grasp that similarity. The following section proposes a solution to that problem.

4.4 Distance Based

Distance metrics

To avoid the high dimensionality problem faced in Section 4.3, we also try clustering the articles using only distance metrics, by first creating an article-to-article distance matrix. There are many ways to compute distances between articles, the most common is cosine distance on the bag-of-words vectors representation of the articles. But we also propose other ways to compute inter-article distances. We try the following distances metrics:

- **DM_P_N_L** : We represent each article in its bag of words representation of words left after removing the words appearing in more than P% or less than N documents or composed of L letters or less. Each element of the article vector is equal to the number of times a word appears in the article. The distance between two articles is computed as the cosine distance between the two corresponding vectors.
- **TFIDF_M_X**: Using TF-IDF method M and keeping the X highest ranked words for each article, we create a new space containing the kept words for all the articles. Each element of the article vector is equal to the number of times a word appears in the article. The distance between two articles is computed as the cosine distance between the two corresponding vectors.
- **WE_200_X_Y**: We keep the X highest ranked words by TF-IDF and represent them in a word embeddings (WE) space of dimension 200. We then compute the distance between 2 articles as the average of Y pairs of different words presenting the lowest cosine distance.

Since it is hard to know which distance is the best for our use case we compare them using this metric:

$$\text{Score}_1 = \frac{1}{N} \sum_{\mathbf{a} \in \mathcal{A}} \mathcal{D}(\mathbf{a'}, \mathbf{a}) \text{ where } \mathbf{a'} := \arg\min_{\mathbf{a'} \neq \mathbf{a}} cdist(\mathbf{a'}, \mathbf{a})$$

and

$$\mathcal{D}(\mathbf{a'}, \mathbf{a}) = \begin{cases} 1 & \text{if } \mathbf{a} \text{ and } \mathbf{a'} \text{ belong to the same categories} \\ 0 & \text{otherwise} \end{cases}$$

This metric allows us to see if the chosen distance will lead us to the right path, we also use a second measure, Score_2 which neglects articles from the categories classified as 'Other' (see Table 3.1). The results for Score_1 and Score_2 for all the methods presented above are depicted in Appendix A.3. We observe that WE_200_50_30 works the best for our criterion and we will use that metric form now on unless specified otherwise.

Limitations

By analyzing the distances we obtain, we see that the distance an article and the one closest to it can vary greatly. The problem arising from that is that distances cannot be taken as absolute and no threshold can be fixed as the frontier between articles from a category and another one.

Curse of dimensionality

Since these distances are taken from high dimensionality spaces, they tend to concentrate around their expectation. To mitigate this problem we first assume that the distribution of distances is Gaussian (In reality it is actually a Rayleigh distribution [44]), to obtain a more uniform distribution we transform the distances by the cumulative distribution function of the standardized variable. Or :

$$\text{Modified} = cdf(\frac{d - \mu}{\sigma})$$

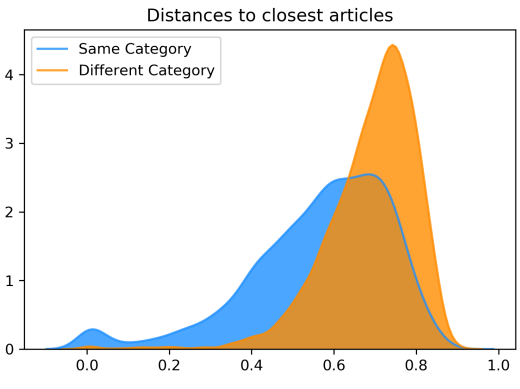

Figure 4.3: Distances to the closest article of the same category vs a different one (DM_90_10_2)

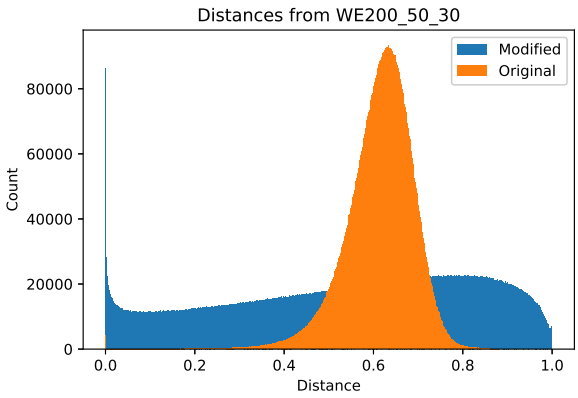

Figure 4.4: Distances taken from high dimension spaces concentrate around their expectation

4.4.1 Spectral Clustering [A.4]

Creating a graph where each article is connected to its k-nearest neighbours with an edge of weight one (or 1-d, where d is the distance between the articles corresponding to the vertices connected by that edge) then applying the Spectral Clustering algorithm described in Section 2.2.3 we can cluster our documents using the distance metrics presented above. The only parameter that we will make vary will be k, the number of neighbours to which an article is connected in the graph.

The best results observed were with k=100 as shown on Figure 4.5. The algorithm aims at finding disconnected portions of the graph, but it may be that articles link subgraphs of different categories together, the next section aims at fixing that problem.

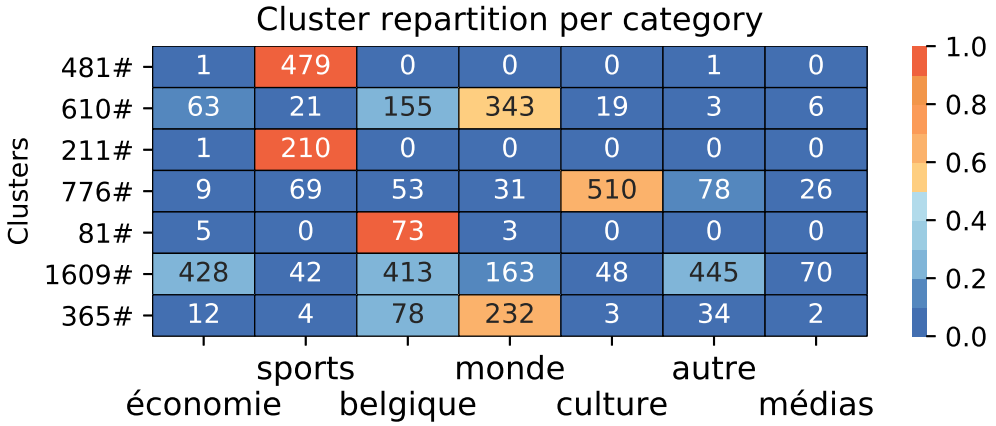

Figure 4.5: Results of Spectral Clustering for k=100

4.4.2 Louvain [A.5]

Assuming that connecting articles to their closest neighbours makes densely connected communities of articles of the same category linked together by noisy links due to semantic proximity of articles of different categories, the Louvain algorithm 2 is a good candidate to isolate these communities. Since the Louvain algorithm groups together vertices which have edges of high weights between them, we weight the edges between two vertices n_1 and n_2 as $1 - d(n_1, n_2)$. Where d represents the distance metric derived earlier.

The best results were obtained by connecting to each article the 50 closest to it. We will show the influence of some parameters below. The modification of distance proposed above influences the clusters in a positive way, as shown on Figure 4.6. We can observe that the number of clusters is not the same for both results, this is because the number of clusters returned by the algorithm is not fixed by the user but relying on the structure of the data itself. Exploring that property of the algorithm, we also try stopping it one iteration before it naturally stops. We observe that two groups of 2 clusters that would have merged during the last iteration were actually quite good in term of identifying categories. We also try stopping the algorithm 2 iteration before it stops yielding much more clusters, but again, being quite good at isolating a category (see Appendix A.5

Figure A.7).

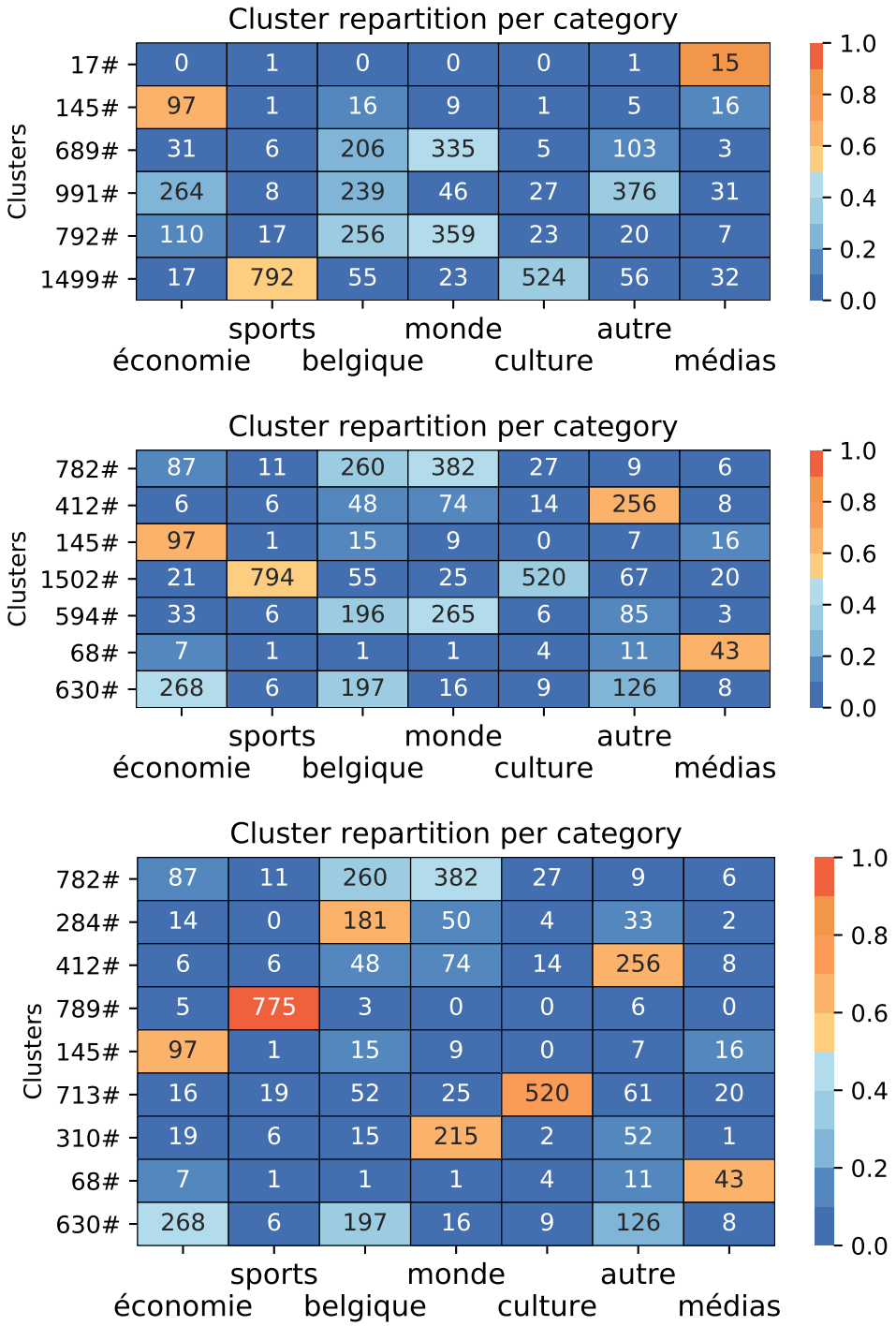

Figure 4.6: Results for Louvain using the non-modified distance (top) versus the unifmormized distance (middle) versus the result obtained when stopping the algorithm prematurely (with uniformized distance) (bottom)

4.5 Results

This section aims at comparing the different methods proposed above. Since Louvain performs better than classical Spectral Clustering on distance-based graphs, we will only present the results of KMeans, LDA and Louvain.

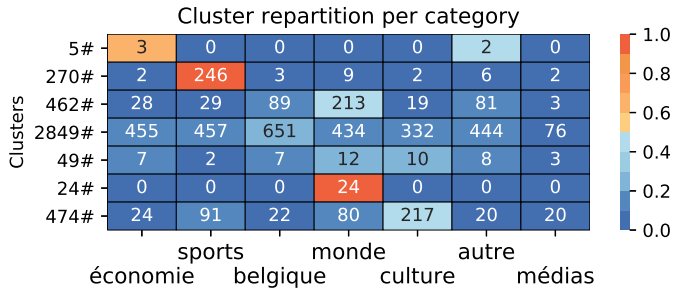

Figure 4.7: Best result for k-Means, using 1 word in embedding representation, TF-IDF parametrization 0

k-Means performs the worst of the solutions we tried, the first method proposed is trying to find clusters in much too high dimensions to work correctly, especially with the amount of data we have. The second solution, while lower in dimensions, represents whole articles with only one word. It is only shown here for the sake of comparison.

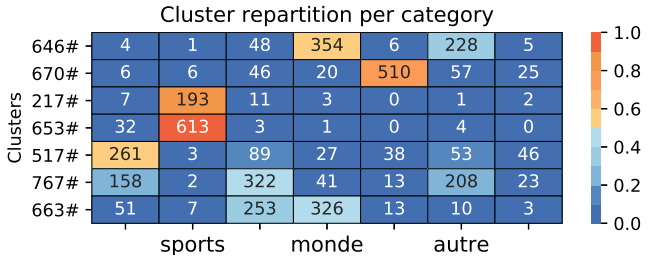

Figure 4.8: Best result for LDA, $\alpha = 5$ & $\lambda = 1$

Latent Dirichlet Allocation works quite well without needing to reduce the dimensions for it to work, this and the fact that it can assign mulitple weighted labels make it quite a strong solution for the task we are requiring. Having a mixture of topics can be useful for recommending similar articles. Which is precisely why the New York Times used it for its recommandation engine [45].

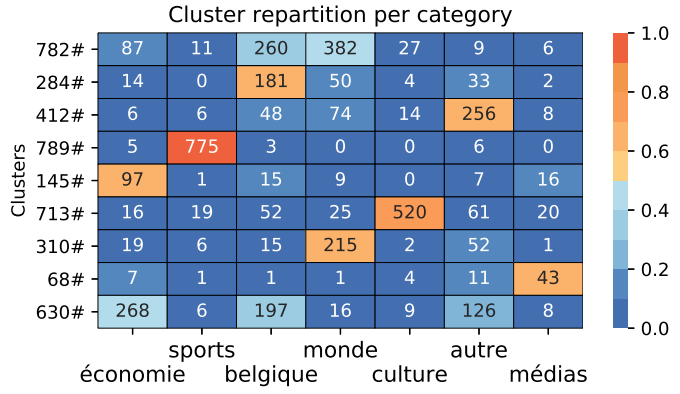

Figure 4.9: Best result for Louvain $k = 50$, distance normalized, algorithm stopped 1 iteration before convergence

Louvain performs quite well compared to the other methods, and the fact that the algorithm can be stopped a few iterations before it converges will be interesting to use it to perform classification.

Chapter 5

Classification

5.1 Goal and success metric

The goal of classification is to train a model to correctly assign its category to an article when presented to it. It has to be kept in mind that an article always belongs to one class and one only. Also as mentioned in 3.1, the 7 classes are not equal in terms of number of train articles and content differentiation from other categories. The metric we will use to assess the performance of our model is the probability of good classification, simply defined as:

$$P_{success}(model) = \frac{\#\text{ of articles correctly classified by } model}{\#\text{ of articles to classify by } model}$$

In total, we try 4 different classification methods on 4 different types of input vectors. The four types of inputs are presented in the next section.

5.2 Inputs

The first input, that we name *title + text*, is the bag of words representation of an article and its title. Only the 20 000 most frequent stems are considered. The second input, that we name *Louvain Clusters* is the output of the Louvain clustering algorithm when stopped after the first iteration, each article is represented as a "one-hot" vector of length equal to the number of clusters. The third one, *Clusters k-Means* is much similar to the second but the considered clustering method is k-Means. The last representation, *article + clusters*, is *title + text* to which *Clusters Louvain* is appended. The parameters to both the clustering methods will be selected on the basis of the results from the Direct method presented in the following section.

5.3 Classification methods

This section presents the four methods we used or implemented to classify the different inputs presented in the previous section. To select the hyperparameters of each method, we use 5-fold cross validation on the training set. The intermediary results for parameter selection can be found in Appendix B or by clicking the appendix number between the square brackets next to the method name in the titles.

5.3.1 Direct Method [B.1]

This method, derived from a simple technique that assigns the most common category is a cluster to all articles inside it, is extended to any vector, as described by the algorithm below.

Algorithm 3 Direct Method for classification

Let M be the modelling set
Let T be the test set
for $Vm \in M$ **do**
    vote(argmax$_{i \in |Vm|}$ Vm_i, category(Vm))++
end for
for $Vt \in T$ **do**
    Assign category(Vt) = argmax$_{category}$ vote(argmax$_{i \in |Vt|}$ Vt_i, category)
end for

Parameter Selection

For *Clustering Louvain*, we are looking at the influence of the k parameter (the number of closest neighbours connected to an article in the input graph of the Louvain algorithm). Figure 5.1 shows the probability of successful classification using the direct method for Louvain. The shade around the curve represents the 95% confidence interval.

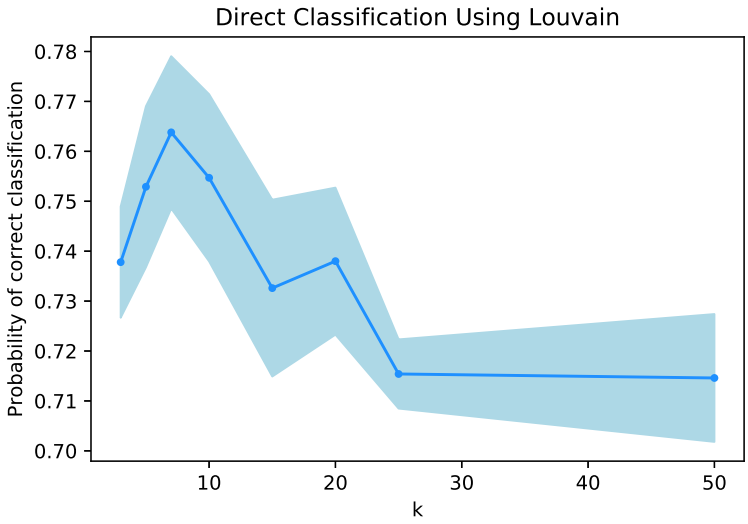

Figure 5.1: Louvain parameter influence on performance

We choose to go with k=7 since it presents the maximum performance and the highest lower bound of confidence interval.
For k-Means, applying the two methods presented in Section 4.3, we observe that applying the algorithm on the *title+text* vectors with $k = 100$ works better than setting higher values for k.

5.3.2 Random Forest [B.2]

Using the sklearn implementation of Random Forest [46], we try different numbers of trees and Gini's impurity and information gain as splitting objective functions. We choose to use Gini's impurity and 200 trees since the performance increase per tree is much lower past that number, this allows us to save computation time.

5.3.3 Multinomial Naive Bayes [B.3]

The only parameter wich influences the output of Multinomial Naive Bayes with a fixed input is the smoothing factor (see 2.3.2). The best value we find for our dataset is 0.09.

5.3.4 Neural Networks [B.4]

Various shapes and types of neural networks were tried, but no performance advantage was found by using the more complex recurrent network types. The different types and their parameters are shown in the Appendices. The MLP outperformed the other types without needing the information of the words position in the text. Although this came as a surprise, we explain that by the fact that recurrent networks are tools designed to identify more subtle text features like the meaning of sentences, which is not needed in our case. The selected MLP parametrization is described in Table 5.1.

Layer	**# of neurons**	**Activation Function**
L1	$141 \approx \sqrt{20000}$	*relu*
L2	$31 \approx \sqrt{141 \times 7}$	*relu*
L3	7	*tanh*

Table 5.1: Best MLP parametrization

5.4 Results and comparison

Our estimation of the probabilty of correct classification has been done by 5-fold cross validation on the full dataset. The values shown in Table 5.2 represent the 95% confidence interval. X [+/- Y] is our notation for a confidence interval [X-Y, X+Y]

Input \Method	**Direct**	**Random Forest**	**Naive Bayes**	**MLP**
Title + Text	0.4085 +/- 0.006	0.7847 +/- 0.010	0.8505 +/- 0.009	0.8213 +/- 0.013
Clusters Louvain	0.7638 +/- 0.015	0.7636 +/- 0.012	0.8269 +/- 0.013	0.7646 +/- 0.013
Clusters kMeans	0.6105 +/- 0.014	0.6110 +/- 0.015	0.6101 +/- 0.014	0.4291 +/- 0.024
article + clusters	0.4085 +/- 0.006	0.7849 +/- 0.010	0.8509 +/- 0.009	0.8509 +/- 0.009

Table 5.2: Classification results, 95% confidence interval

Looking at this table vertically, we observe that the text input is the input that generally works the best, with only the MLP gaining interesting performance from the clustering information. The direct method works best with the input containing only clustering information but it was designed that way.
This preference for title + text over article + clusters is explained by the way the algorithms work. For the direct method, the dimensions added by appending the cluster information will never be considered since their maximum value is 1. The performance is thus the same than for the *title + text* case. For random forest, it seems normal that using one of these cluster information dimensions will not allow for a more accurate split than the direct method. For Naive Bayes, the compared weight of the clusters is drowned in the weights of all the other words and their contribution to the classification is then quite low. The MLP on the other hand can adjust its weights to benefit from the added information provided by the clustering.

To conclude, we have two solutions that provide similar accuracy numbers: Naive Bayes and the MultiLayer Perceptron. We can direcly notice that performance is better than the one obtained by cross validation on the training set. This can be explained by the higher number of articles used for training, as shown by the learning curve on Figure 5.2. We can expect performance to improve as more samples are given to the algorithm to train on, but it is impossible when a change of slope/learning rate will occur.
Since the need for data is exponential to the number of dimensions, it is clear that a model would need much more data to reach higher performances. Recent improvements in language models have mainly been made by scaling up the training set [47].

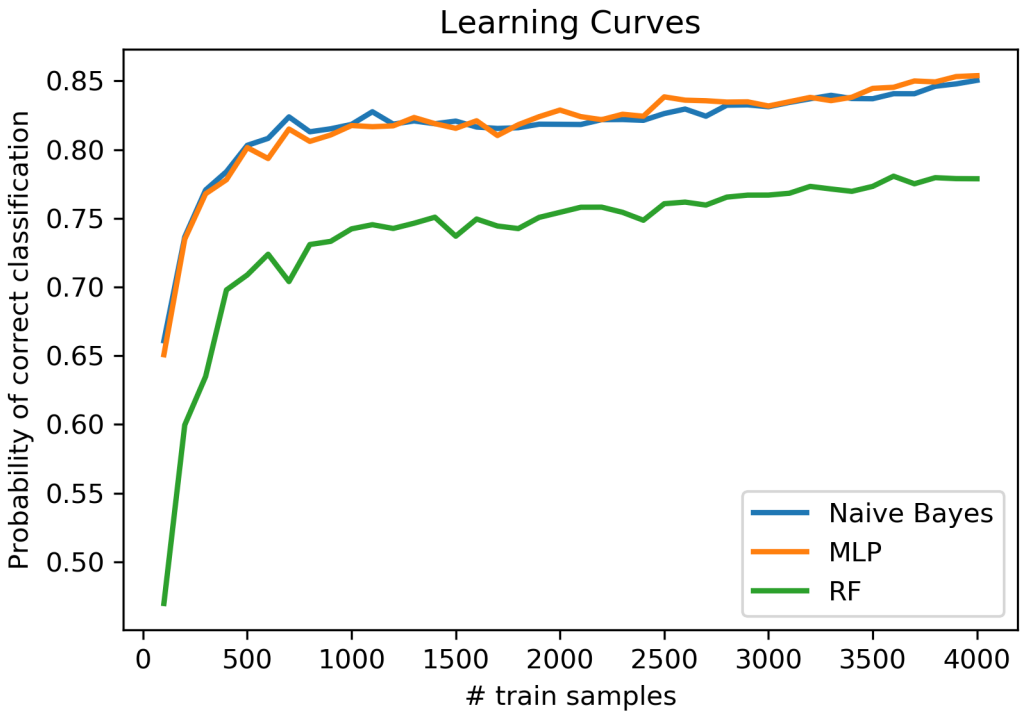

Figure 5.2: Learning curve: performance as a function of the number of training samples

Chapter 6

Stability

6.1 Goal and success metric

The goal of this chapter is to find weaknesses in the best performing models from the previous chapter. The goal is to make these models misclassify an article that they would otherwise classify correctly by slightly modifying the text. The success metric we use will be inherently subjective: we will consider an attack against a model successful if an article is misclassified without making dramatic changes to it. Examples will be put in the appendices for the reader to make his own appreciation. While usual adversarial example attacks aim at slightly modifying many elements of the input that then magnify due to a dot product with the weights, modifying text cannot be done in such a continuous fashion. It is then appearant that modifying text will be more perceivable than nudging the pixel values of an image by a fraction. Since our models take as input a count of words, our only option is to add words with not enough meaning to dramatically change the content of the article. Removing words is another option we keep in mind but that is it. Attacks on text have already been performed by inserting typos in the text, we feel it not realistic in the sense that typos are quite easily detectable by humans.
We first define precisely what type of attack we want to perform and the tools we are allowing ourselves to use to obtain the results we want. For both the following models, we attempt to find the minimal disturbance that can be done to a text for it to be misclassified. We also analyze how much a text has to be modified to be classified in a chosen category. We only show examples since we do not have tools to make these types of attacks automatically.

6.2 Attacks against MNB

6.2.1 White Box

With Multinomial Naive Bayes, if the model is trained without bootstrapping and we have all the training data at our disposal, we can retrieve the model parameters with close to 100% accuracy since the training is deterministic and order-independent. The only unknown is the smoothing factor, which would need to be estimated by looking at samples already classified by the model. We will thus focus on the situation where the attacker has access to the trained model with all its parameters.

As showed in section 2.3.2, the Naive Bayes classifier can be written as:

$$arg\max_{c \in C} R_c \text{ where } R = W\mathbf{a} + b$$

Let us write W_x the line of matrix W corresponding to category x, and b_x the corresponding element of vector b. It is appearant that for an article classified in category x, for any category y:

$$R_x = W_x\mathbf{a} + b_x > W_y\mathbf{a} + b_y = R_y$$

Since the attacker has no control on the training of the model, W_x, b_x, W_y and b_y will remain unchanged. The only way to misclassify a correctly classified article is then to modify it (changing its vector **a** to a vector **a'** in the process).

$$R'_x < R'_y \iff R'_x - R'_y < 0 \tag{6.1}$$

to achieve that we can look at the direction we should take to minimize the second inequality from equation 6.1.

$$\nabla R_{xy} = \frac{\delta}{\delta \mathbf{a}} R'_x - R'_y = W_x - W_y$$

Since the gradient is an upwards direction, we know that to classify an article **a'** from an article **a** is done by adding words with positive weights in $W_{yx} = W_y - W_x = -\nabla R_{xy}$ or removing words with negative weights in the same vector.
Better yet, since the attacker has full knowledge of the model, he can compute $R'_y - R'_x$, the amount of 'score' he needs to get to get the article misclassified. By score we mean $W_{yx}(a' - a)$. The goal of a real life attacker will probably be to misclassify a document while causing minimum disruption to the original text. Of course, measuring such a disruption implies having an accepted distance metric between documents.

6.2.2 Black Box

There are two ways to go at this problem.

- Try to estimate all the model parameters to fall back into the previously explained white box scenario.
- From an article, estimate the number of certain words to be added to it to move to another category.

Model Estimation

If the attacker knows that the targeted model, model_t is Multinomial Naive Bayes, she can train a new model, model_a with available samples or by querying model_t as an oracle and train model_a with its outputs. As shown on the learning curve at the end of the previous chapter (Figure 5.2). Most of the performance of the model is acquired by training the 500 first samples. Since the model is based on first order linear relationships between the number of appearance of words and a certain category, we assume that the parameters do not change drastically with the size of the training set. More specifically,

we are interested in how the difference of weights between categories evolve with the number of training samples. Even further, we are interested in knowing if the sign of each element of W^t_{xy} from model_t, is the same sign as the W^a_{xy} from model_a. We then derive a metric for the closeness of estimation m_1.

$$m_1(model_t, model_a) = \frac{1}{N} \sum_{n \in N} \frac{1}{|C|} \sum_{c_i \in C \setminus c_0} same\_sign(W^a c_0 c_i - W^t c_0 c_i)$$

where $same\_sign(x, y)$ is a function that returns 1 if its two inputs have the same sign, and 0 otherwise. Which indicates the percentage of correct directions the estimated model will give to an attacker. Figure 6.1 shows the evolution of m_1 as a function of the number of samples used to train $model_a$, these were not used in the training of $model_t$. Lambda is the smoothing factor, the one used to train $model_t$ was 0.09.

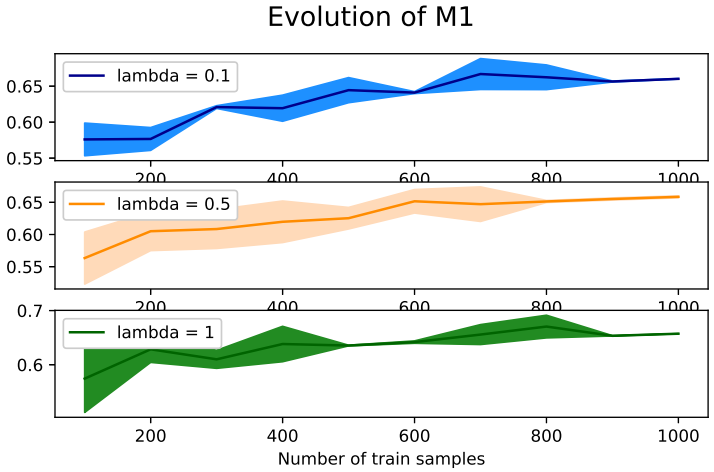

Figure 6.1: Evolution of m_1 score as a funtion of the number of training samples, the shae around the curve represent the 95% confidence interval

We can expect these number to be higher in a real-life situation where the attacker trains his source model with news articles found on the targeted new site, the chance to have access to articles that were used to train the model are quite high.

Single Article Attack

Let's now suppose that the attacker wishes to only attack one article specifically, classifying in category y an article originally classified in x by a classifier F. This can be done quite simply but requires many requests to the model, which can pose a problem. Indeed, if the requests to the model are monitored, the number of queries would be suspicious and the attack detected. From an article vector $\mathbf{a}$, for each component i of the vector, the attacker sends a classification request for the article $\mathbf{a}+k\mathbf{e}_i$, where $\mathbf{e}_i$ is a vector of length equal to $\mathbf{a}$, equal to 0 everywhere except for its i^{th} dimension, where it is equal to 1. The attacker varies k until he finds k and $k+1$ such that:

$$F(\mathbf{a} + k\mathbf{e}_i) \neq y$$

and

$$F(\mathbf{a} + (k+1)\mathbf{e}_i) = y$$

This will not work for words for which W_{yx} is negative, and will not work in certain cases where another category z is closer to that article and/or has a bigger W_{zx}. Knowing that, the attacker can now know certain words to add and how much of these words he would need to add to successfully misclassify the article. Creating a type of gradient not unlike the one found for the white box case.

6.3 Attacks against MLP

MultiLayer Perceptron have a more complex way of classifying articles than a Naive Bayes model, they use nonlinearities and relationships of orders higher than 1 between words and categories. Let's say adding word w_1 or word w_2 brings an article closer to a certain category y, it may very much be that adding w_1 and w_2 to the article will 'push it away' from category y. The first order methods used for Naive Bayes are in the risk of not working as good, but we will present a few ways the problem can be tackled.

6.3.1 White Box

The most common attack against an MLP from which all the parameters are know is to make a gradient ascent on the error function of the MLP. This corresponds to maximizing the output error with constant weights. Similarly to the attack proposed for the white box Naive Bayes case, we take words that do not have dramatically different meanings than the ones found in the original text but have relatively high importance in the gradient and add them to the text.

6.3.2 Black Box

Since the architecture of an MLP is much more complex than for Naive Bayes, estimating the hyperparameters and the parameters represents a much bigger task than what we could easily do in the previous section. Transferability (see 2.4.1) is a good option in this case, we do not explore that possibility because the impossibility of automation of the process. The track we will exploit relies on the claim that the nonlinearities in the Neural Nets are not so nonlinear. That claims relies on the fact that the rectified linear unit is linear for a good part, and that most of the interesting cases on sigmoids occur on the rising slope, which is almost linear. If that affirmation is valid, the following technique should work in some cases:

Word-by-word

This method can be seen as a sort of Stochastic Gradient Ascent. Each iteration, the attacker makes a step towards the target category, the assumption is that every step the attacker takes will not bring him further away from the target category.
Considering an article $\mathbf{a}$ correctly classified by a classifier F in category c that an attacker

wants to misclassify into a target category t. We propose a technique much similar to the *single article attack* porposed for a Multinomial Naive Bayes model:

$$\forall i \in \{1, ..., |\mathbf{a}|\} \text{find } k_i \in \mathbb{Z} \text{ such that:}$$

$$F(\mathbf{a} + (k_i - 1)e_i) = c \text{ and } F(\mathbf{a} + k_i e_i) = t$$

Then the attacker sorts the words by their associated k value in ascending order. Chooses a word to add to the text and reiterates until the article has been misclassified. This method requires a lot of requests to the target model but if done cleverly, the attacker can only compute the direction of words he would actually add to the text, and not all the possible words. Also, if the nonlinearities are neglected, the attacker could treat the k_i's as constants. This is not exact and would probably require more steps than needed but would not be as detectable by the targeted model, while testing we observed a majority of words keeping the same k value after adding and removing other words.

6.4 Results and discussion

We attacked a specific article from the *Belgium* category, it was chosen for its closeness with the *economy* category for the Naive Bayes classifier.

Flandre : des mesures préventives contre les attaques de loups
Le ministre flamand en charge de la nature et de l agriculture koen van den heuvel a prévu des mesures préventives pour des éleveurs dont les animaux pourraient subir des attaques de loup(s). La prévention des dommages causés par des loups a reçu un coup de pouce des autorités flamandes qui vont subventionner une série de mesures préventives. Des investissements passés peuvent être subsidiés s'ils remplissent les conditions. Ce nouveau règlement doit être prêt pour début avril. Les éleveurs qui équipent leur terrain d un enclos contre les loups pourront récupérer 80% de leurs investissements. [. . .] [48]

We tried attacking it in two different ways:

- White-box attack for Naive Bayes
- Black-box word-by-word attack for the MLP

The first result, showed on Table 6.1 was that adding the word 'normalement' (normally) did shift both articles into other categories, but not the same.

[...] Des investissements passés peuvent être subsidiés s ils remplissent les conditions.	Naive Bayes →	économie
normalement ce nouveau règlement doit être prêt pour début avril. [...]	MLP →	monde

Table 6.1: Results for both methods by only adding a single word

Since the text talks about money and investments it is not with too much of a suprise that the article is classified in the *economy* category, we cannot say the same thing for the *world* category. To reach other categories we needed to add more than one word. We did the experience to show the amount of distortion we would need to add to misclassify in any target category. These results are shown in Appendix C.
An interesting observation is that the amount of words required to misclassify an article was highly correlated (Pearson coefficient of .86) with the length of the text for Naive Bayes, but there was no correlation of the sort for the MLP. This is quite normal for Naive Bayes since the model is only a weighted sum of the inputs. For the MLP, on the other, hand it goes against the high linearity hypothesis from the previous section.
Another interesting example is the targeted attack on the sports category:

Flandre : des mesures préventives contre les attaques de loups
Le ~~ministre~~ **leader** flamand en charge de la nature et de l'agriculture koen van den heuvel a prévu des mesures **d'attaque** ~~préventives~~ pour des éleveurs dont les animaux pourraient **par malchance,** subir des attaques de loup(s) **solitaires ou en meute**. La **course contre les** ~~prévention des~~ dommages causés par des loups a reçu un **costaud** coup de pouce des autorités flamandes qui vont subventionner une série de mesures préventives. [...] ~~Plusieurs~~ **Le palmarès des** communes ~~flamandes~~ **Limbourgeoises qui** sont à risques **est** : balen beringen bocholt bree dessel dilsen stokkem geel ham hamont **en onzième rang** achel hechtel eksel houthalen helchteren kinrooi lanaken leopoldsburg lommel maaseik maasmechelen meerhout mol oudsbergen peer pelt zonhoven et **en dernier rang** zutendaal .

Table 6.2: Modifications to fool the MNB classifier

Flandre : des mesures préventives contre les attaques de loups
Le ministre flamand [...] début avril. Les éleveurs qui ~~équipent leur terrain d un enclos~~ **s'équipent d'un enclos autour de leur terrain** contre les loups pourront récupérer 80% de leurs investissements. [...]

Table 6.3: Modifications to fool the MLP classifier

While in both cases the confidence score for the sports category was one well under the score for *Belgium*, we see that clever, unnoticeable tricks work with the MLP while Naive Bayes requires a lot of modifications, so much so that when reading the text the reader feels that something is off.
What we take from our experiments with both methods is listed below:

- We expect the number of cases like the example of 6.1 to decrease with the number of data used to train the MNB model, having more articles written by more writers would allow words void of any sense to not be particularly associated with particular categories. That is not guaranteed for the MLP.

- The usual property of adversarial examples, to make drastic changes in the classification confidence by slightly changing all of the inputs, is not applicable with the representation we used. One can imagine that a representation using word embeddings could be attacked using more classical methods, replacing words with very similar words chosen in the direction of the error gradient in the embeddings space.

- It seems that MLPs allow for more "shortcuts" (small modifications of the input that produce big change in the output) than Naive Bayes, for which classification in a target category is not done without using words specific to that category. The MLP was shown to be more easily fooled.

- A big weakness of the classifiers in news recommendation systems, is that an attacker has access to the training data or to data very similar to it on the very website he is targetting. This makes transferrability attacks much easier.

- The attacks against text classifiers are possible, but without a strong generating language model, they will have to be crafted by hand. A good protection would then be to combine mutliple criterion to classify articles, adding more and more constraints to make it humainly impossible to cheat all the systems.

- Since adversarial examples work by making very small changes to many of the input. There is more than ever a need to reduce the dimensions of the input data.

An important note is that the fact that the using another type of model than these two does not mean that it will not be subject to the same flaws: by transferability it is possible to attack another model with examples working on these two [36]. A good argument would be that it does not matter if the article looks weirds when reading it, as long as it makes its way to the targeted users, that is partially true, but what feels weird will probably not be taken as seriously and might be flagged by users if it seems too strange. This is the reason why we did not explore the possibility of inserting typos. Techniques like the insertion of typos or based on the visual similarities of certain characters (for example replacing l's (L) with I's (i)) only work until they are detected and then never work again.

Taking all of that into account, and knowing the stakes of news targetting, we believe that adversarial examples in the text case are practical enough to use them in a real-world attack and would be surprised if such attacks had not been made already.

Chapter 7

Conclusion

We showed a new way to cluster text documents that outperforms other clustering techniques, using intuitive and human-like criteria to perform the clustering. Since it is distance based, it can be used in a supervised fashion as well.

Classification was shown to be breakable for the two best-performing methods on our dataset. Anyone using as argument that they use another model and that, as a consequence, these vulnerabilities do not apply to their case, should first verify that adversarial examples built on other models do not work on theirs. Then, prove that the known adversarial examples generation techniques do not work on their model.

From our experiments, it seems that attacking a probability-based approach like Multinomial Naive Bayes has to mostly be done by using vocabulary specific to the target category. For the MulitLayer Perceptron, attacks seem to be possible without being obviously using words of the target class. We suspect that the MLP, being able to capture more complex inter-words relations, will outperform the Naive Bayes classifier with more training data. The drawback will always be the non explainability of complex Neural Networks. On the subject of adversarial examples, we believe that combining mutliple models would add enough constraints to make adversarial example generation too hard without a generative language model.

Combining a well-chosen misclassification with a profiling of target users, one can use these techniques to spread disinformation or information carrying a political message. If there exists such a thing as a performance-transparency tradeoff and if the design choices are left entirely to software engineers, there is little doubt that better performance will almost always be chosen. Journalists, on the other hand, might want more transparency. The real problem is larger and is more than just a quest for 100% accuracy. The problem is the lack of a consensus on what is expected from these algorithms and what the accepted failure rate under normal and adversarial conditions is. Since salaries and elections might depend on these types of algorithms, transparency or accountability seem vital.

Bibliography

[1] Seth C. Lewis and Oscar Westlund. "The SAGE Handbook of Digital Journalism". In: 55 City Road: SAGE Publications Ltd, Aug. 2019. DOI: 10.4135/9781473957909. URL: http://sk.sagepub.com/reference/the-sage-handbook-of-digital-journalism.

[2] Nicholas Diakopoulos and Michael Koliska. "Algorithmic Transparency in the News Media". In: *Digital Journalism* 5.7 (2017), pp. 809–828. DOI: 10.1080/21670811.2016.1208053. URL: https://doi.org/10.1080/21670811.2016.1208053.

[3] Matthew S. Weber and Allie Kosterich. "Coding the News". In: *Digital Journalism* 6.3 (2018), pp. 310–329. DOI: 10.1080/21670811.2017.1366865. URL: https://doi.org/10.1080/21670811.2017.1366865.

[4] Tetyana Lokot and Nicholas Diakopoulos. "News Bots: Automating news and information dissemination on Twitter". English (US). In: *Digital Journalism* 4.6 (Aug. 2016), pp. 682–699. ISSN: 2167-0811. DOI: 10.1080/21670811.2015.1081822.

[5] Piet Bakker. "AGGREGATION, CONTENT FARMS AND HUFFINIZATION". In: *Journalism Practice* 6.5-6 (2012), pp. 627–637. DOI: 10.1080/17512786.2012.667266. URL: https://doi.org/10.1080/17512786.2012.667266.

[6] Matt Carlson and Nikki Usher. "News Startups as Agents of Innovation". In: *Digital Journalism* 4.5 (2016), pp. 563–581. DOI: 10.1080/21670811.2015.1076344. URL: https://doi.org/10.1080/21670811.2015.1076344.

[7] Cathy O'Neil. *Transcript of "The era of blind faith in big data must end"*. en. URL: https://www.ted.com/talks/cathy_o_neil_the_era_of_blind_faith_in_big_data_must_end/transcript (visited on 08/17/2019).

[8] Julia Angwin Jeff Larson. *How We Analyzed the COMPAS Recidivism Algorithm*. en. text/html. May 2016. URL: https://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm (visited on 08/10/2019).

[9] *Créer un systême de recommandation dans TensorFlow : présentation | Solutions*. fr. URL: https://cloud.google.com/solutions/machine-learning/recommendation-system-tensorflow-overview?hl=fr (visited on 05/16/2019).

[10] *Recommending items to more than a billion people*. en-US. June 2015. URL: https://engineering.fb.com/core-data/recommending-items-to-more-than-a-billion-people/ (visited on 08/17/2019).

[11] Jiahui Liu, Peter Dolan, and Elin RÃžnby Pedersen. "Personalized news recommendation based on click behavior". en. In: *Proceedings of the 15th international conference on Intelligent user interfaces - IUI '10*. Hong Kong, China: ACM Press, 2010, p. 31. ISBN: 978-1-60558-515-4. DOI: 10.1145/1719970.1719976. URL: http://dl.acm.org/citation.cfm?doid=1719970.1719976 (visited on 08/17/2019).

[12] Gabriel de Souza P. Moreira, Felipe Ferreira, and Adilson Marques da Cunha. "News Session-Based Recommendations using Deep Neural Networks". In: *Proceedings of the 3rd Workshop on Deep Learning for Recommender Systems - DLRS 2018* (2018). arXiv: 1808.00076, pp. 15–23. DOI: 10.1145/3270323.3270328. URL: http://arxiv.org/abs/1808.00076 (visited on 08/17/2019).

[13] Alexandre Bovet and Hernán A. Makse. "Influence of fake news in Twitter during the 2016 US presidential election". In: *Nature Communications* 10.1 (2019), p. 7. DOI: 10.1038/s41467-018-07761-2. URL: https://doi.org/10.1038/s41467-018-07761-2.

[14] Nathaniel Persily. "The 2016 US Election: Can democracy survive the internet?" In: *Journal of democracy* 28.2 (2017), pp. 63–76.

[15] Tomas Mikolov et al. "Efficient Estimation of Word Representations in Vector Space". In: (Jan. 2013). arXiv: 1301.3781. URL: http://arxiv.org/abs/1301.3781.

[16] *A full-text visualization of the Iraq War Logs | Jonathan Stray*. en-US. URL: http://jonathanstray.com/a-full-text-visualization-of-the-iraq-war-logs (visited on 08/01/2019).

[17] *TF-IDF*. URL: https://en.wikipedia.org/w/index.php?title=Tf%5C%E2%5C%80%5C%93idf&oldid=908081869 (visited on 08/02/2019).

[18] Stuart P. Lloyd. "Least squares quantization in PCM". In: *IEEE Trans. Information Theory* 28 (1982), pp. 129–136.

[19] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. "Latent Dirichlet Allocation". In: *J. Mach. Learn. Res.* 3 (Mar. 2003), pp. 993–1022. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=944919.944937.

[20] Jordan Boyd-Graber. *Lecture slides on Topic Models*. URL: http://users.umiacs.umd.edu/~jbg/teaching/INST_414/.

[21] Laurent Jacques. *Image processing lecture slides on image Segmentation*.

[22] Ulrike von Luxburg. "A Tutorial on Spectral Clustering". In: *arXiv:0711.0189 [cs]* (Nov. 2007). arXiv: 0711.0189. URL: http://arxiv.org/abs/0711.0189.

[23] Vincent D. Blondel et al. "Fast unfolding of communities in large networks". In: *Journal of Statistical Mechanics: Theory and Experiment* 2008.10 (Oct. 2008). arXiv: 0803.0476, P10008. ISSN: 1742-5468. DOI: 10.1088/1742-5468/2008/10/P10008. URL: http://arxiv.org/abs/0803.0476 (visited on 08/05/2019).

[24] M. E. J. Newman. "Analysis of weighted networks". In: *arXiv:cond-mat/0407503* (July 2004). arXiv: cond-mat/0407503. DOI: 10.1103/PhysRevE.70.056131. URL: http://arxiv.org/abs/cond-mat/0407503 (visited on 08/05/2019).

[25] Leo Breiman. "Random Forests". In: *Mach. Learn.* 45.1 (Oct. 2001), pp. 5–32. ISSN: 0885-6125. DOI: 10.1023/A:1010933404324. URL: https://doi.org/10.1023/A:1010933404324.

[26] Leo Breiman. "Bagging predictors". en. In: *Machine Learning* 24.2 (Aug. 1996), pp. 123–140. ISSN: 0885-6125, 1573-0565. DOI: 10.1007/BF00058655. URL: http://link.springer.com/10.1007/BF00058655 (visited on 08/07/2019).

[27] Fuchun Peng, Dale Schuurmans, and Shaojun Wang. "Augmenting Naive Bayes Classifiers with Statistical Language Models". en. In: *Information Retrieval* 7.3/4 (Sept. 2004), pp. 317–345. ISSN: 1386-4564. DOI: 10.1023/B:INRT.0000011209.19643.e2. URL: http://link.springer.com/10.1023/B:INRT.0000011209.19643.e2 (visited on 08/07/2019).

[28] Christopher Manning, Prabhakar Raghavan, and Hinrich Schuetze. "Introduction to Information Retrieval". en. In: (2009), p. 581. URL: https://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf.

[29] Michel Verleysen. *Nonlinear regression with Multi-Layer Perceptrons, machine learning lecture slides*.

[30] Christian Szegedy et al. "Intriguing properties of neural networks". In: *arXiv:1312.6199 [cs]* (Feb. 2014). arXiv: 1312.6199. URL: http://arxiv.org/abs/1312.6199.

[31] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and Harnessing Adversarial Examples". In: *arXiv:1412.6572 [cs, stat]* (Dec. 2014). arXiv: 1412.6572. URL: http://arxiv.org/abs/1412.6572.

[32] Kevin Eykholt et al. "Robust Physical-World Attacks on Deep Learning Models". In: *arXiv:1707.08945 [cs]* (July 2017). arXiv: 1707.08945. URL: http://arxiv.org/abs/1707.08945.

[33] Stanford University School of Engineering. *Lecture 16 | Adversarial Examples and Adversarial Training*. 2017. URL: https://www.youtube.com/watch?v=CIfsB_EYsVI.

[34] Ian Goodfellow, Patrick McDaniel, and Nicolas Papernot. "Making Machine Learning Robust Against Adversarial Inputs". In: *Commun. ACM* 61.7 (June 2018), pp. 56–66. ISSN: 0001-0782. DOI: 10.1145/3134599. URL: http://doi.acm.org/10.1145/3134599.

[35] Nicolas Papernot et al. "The Limitations of Deep Learning in Adversarial Settings". en. In: *arXiv:1511.07528 [cs, stat]* (Nov. 2015). arXiv: 1511.07528. URL: http://arxiv.org/abs/1511.07528 (visited on 08/14/2019).

[36] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. "Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples". In: *arXiv:1605.07277 [cs]* (May 2016). arXiv: 1605.07277. URL: http://arxiv.org/abs/1605.07277.

[37] Nicholas Carlini and David Wagner. "Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods". In: *arXiv:1705.07263 [cs]* (May 2017). arXiv: 1705.07263. URL: http://arxiv.org/abs/1705.07263.

[38] *Le Soir*. http://lesoir.be. (Visited on 05/01/2019).

[39] *nltk.tokenize package*. https://www.nltk.org/api/nltk.tokenize.html. (Visited on 05/04/2019).

[40] *nltk.stem package: Snowball Stemmer*. https://www.nltk.org/api/nltk.stem.html. (Visited on 05/04/2019).

[41] *Stemming and lemmatization*. URL: https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html (visited on 07/02/2019).

[42] Jean-Philippe Fauconnier. *French Word Embeddings*. 2015. URL: http://fauconnier.github.io.

[43] David M. W. Powers. "Applications and Explanations of Zipf's Law". In: *New Methods in Language Processing and Computational Natural Language Learning*. 1998. URL: https://www.aclweb.org/anthology/W98-1218 (visited on 08/13/2019).

[44] Michel Verleysen and Damien François. "The Curse of Dimensionality in Data Mining and Time Series Prediction". In: *Computational Intelligence and Bioinspired Systems*. Ed. by David Hutchison et al. Vol. 3512. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 758–770. ISBN: 978-3-540-26208-4 978-3-540-32106-4. DOI: 10.1007/11494669_93.

[45] Alexander Spangher. *Building the Next New York Times Recommendation Engine*. en-US. Aug. 2015. URL: https://open.blogs.nytimes.com/2015/08/11/building-the-next-new-york-times-recommendation-engine/ (visited on 06/10/2019).

[46] *sklearn.ensemble.RandomForestClassifier*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html (visited on 06/10/2019).

[47] Alec Radford et al. "Language Models are Unsupervised Multitask Learners". en. In: (), p. 24.

[48] *Flandre: des mesures préventives contre les attaques de loups - Le Soir*. URL: https://www.lesoir.be/214475/article/2019-03-25/flandre-des-mesures-preventives-contre-les-attaques-de-loups.

Appendix A

Clustering Parameter Selection

A.1 k-Means [4.3]

As expected k-Means performs poorly on very-high dimensional data, the cluster on the embeddings works better because of the lower dimension but articles are still only represented by a single word.

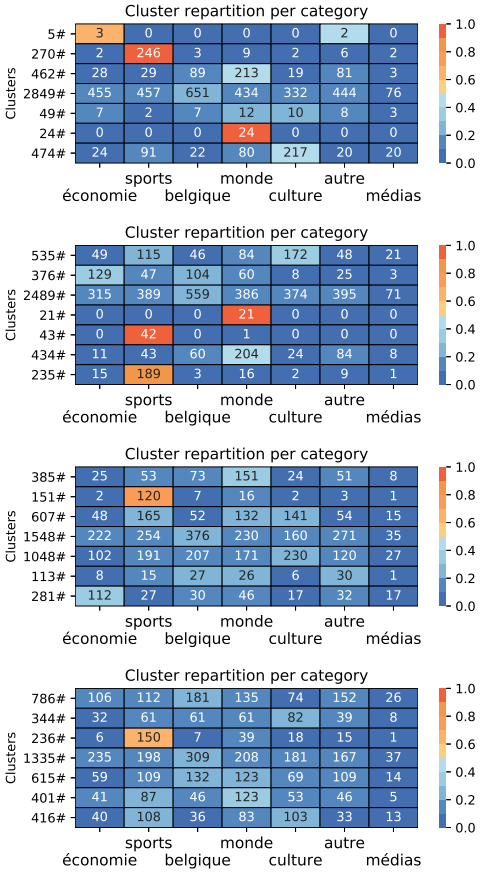

Figure A.1: k-Means on first word embeddings of first word returned by TFIDF parametrization $\{0, 1, 2, 3\}$

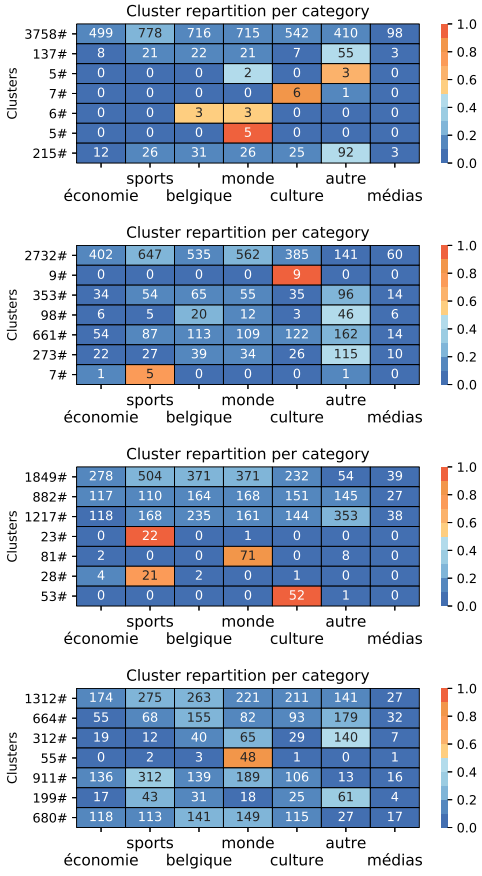

Figure A.2: k-Means on space made out of the $\{1, 3, 5, 10\}$first words returned by TFIDF parametrization 0

A.2 Latent Dirichlet Allocation [4.2]

For LDA, we need to optimize two paramters at the same time so we use this representation. Information about the number of articles in each cluster is lost but it allows to quickly evaluate if certain clusters contain more than one category or if a category is split in multiple clusters. The $\alpha = 5$, $\lambda = 1$ combination seems to be the most ordered with almost no light blue and some lines with only one red box and dark blue ones.

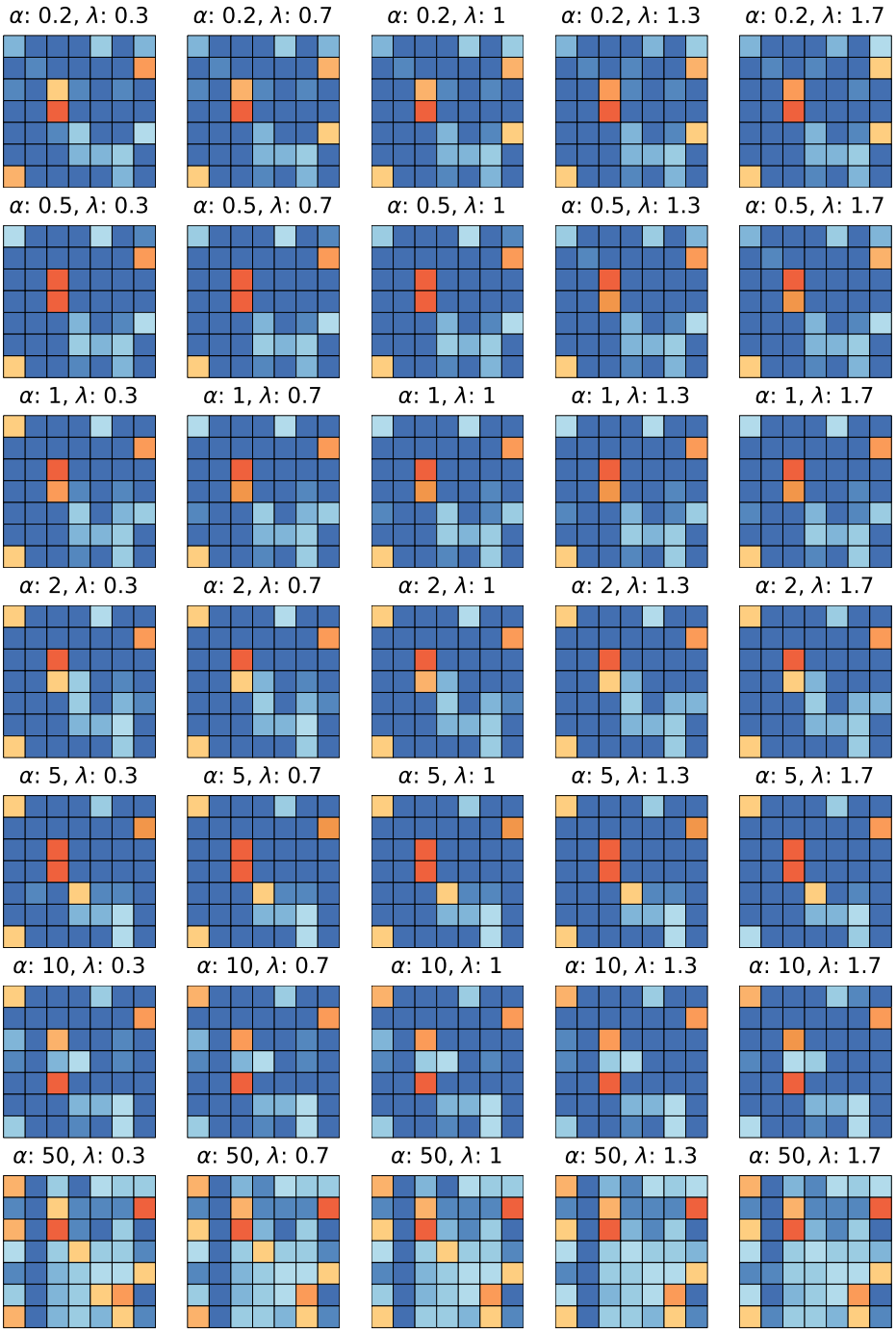

Figure A.3: Latent Dirichlet Allocation parameters influence

A.3 Distances

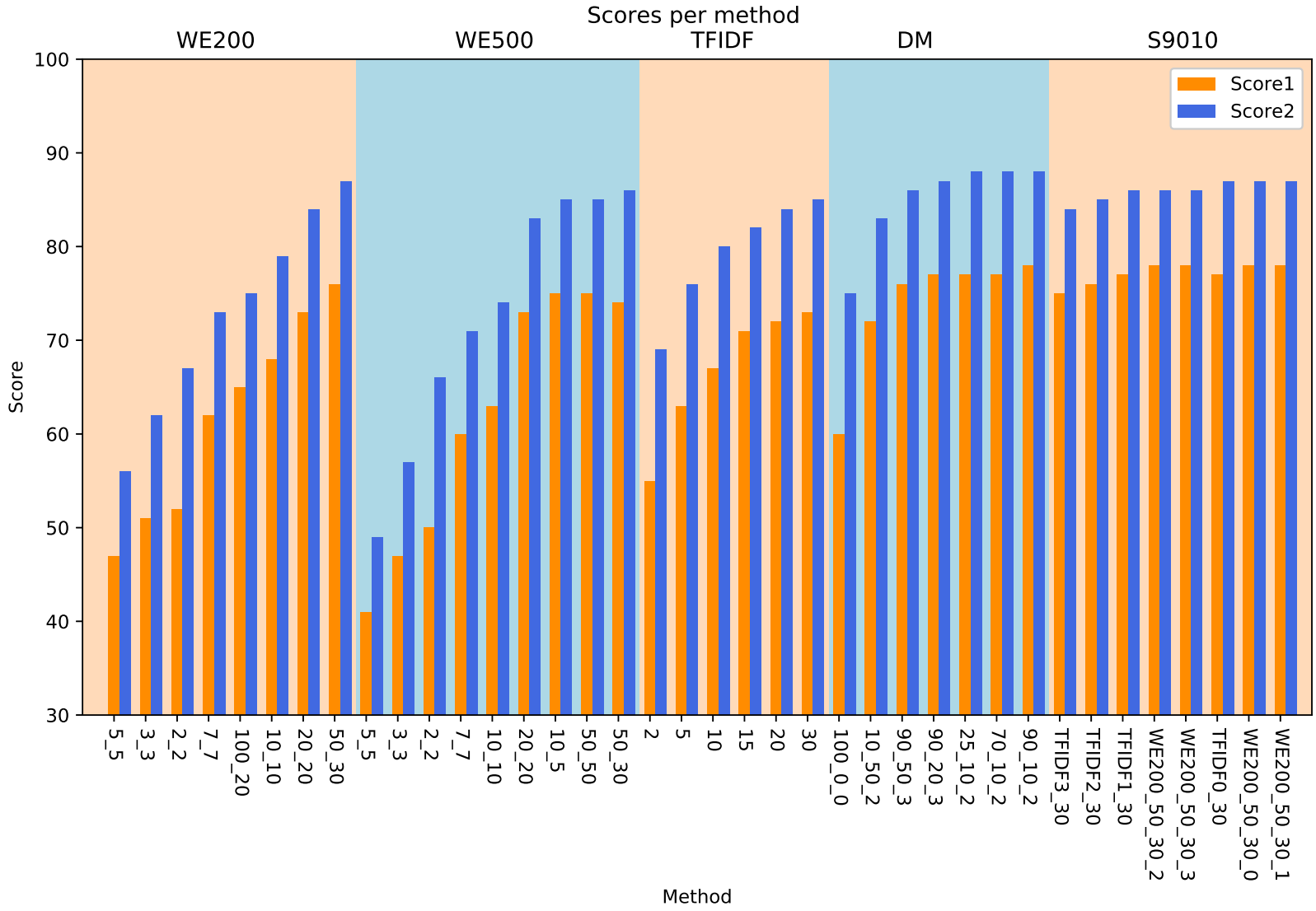

Figure A.4: Score for the distances extracted in Section 4.4

A.4 Spectral Clustering [4.4.1]

Spectral clustering seems to be good at identifying small groups of very close articles, but puts the all the rest in a big cluster for low values of k. For large values of k the repartition is more homogeneous that we would hope to differentiate categories. A value of k=100 seems to work the best.

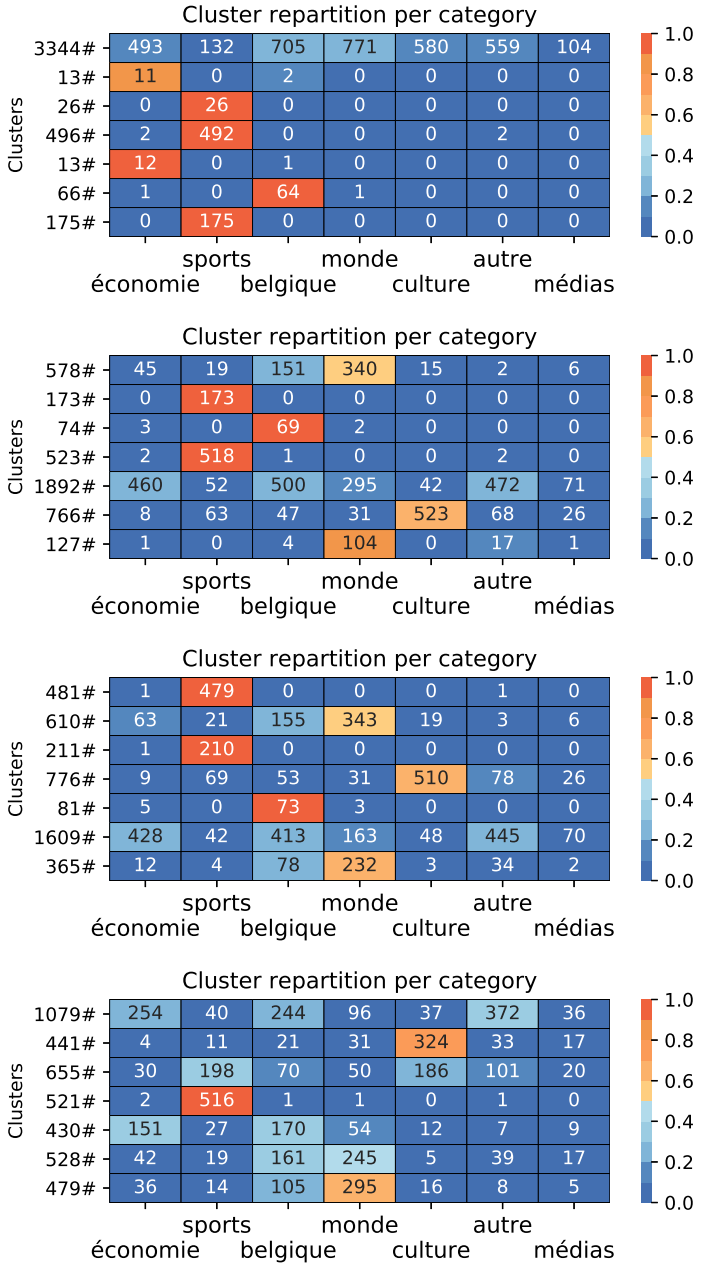

Figure A.5: Spectral Clustering outputs for $k = \{10, 50, 100, 1000\}$

A.5 Louvain [4.4.2]

We vary the number of closest articles we connect to an article in the graph. (We chose $k = 50$ over the similar-performing $k = 20$ because we obtain better results with $k = 50$ when stopping the algorithm one iteration before convergence)

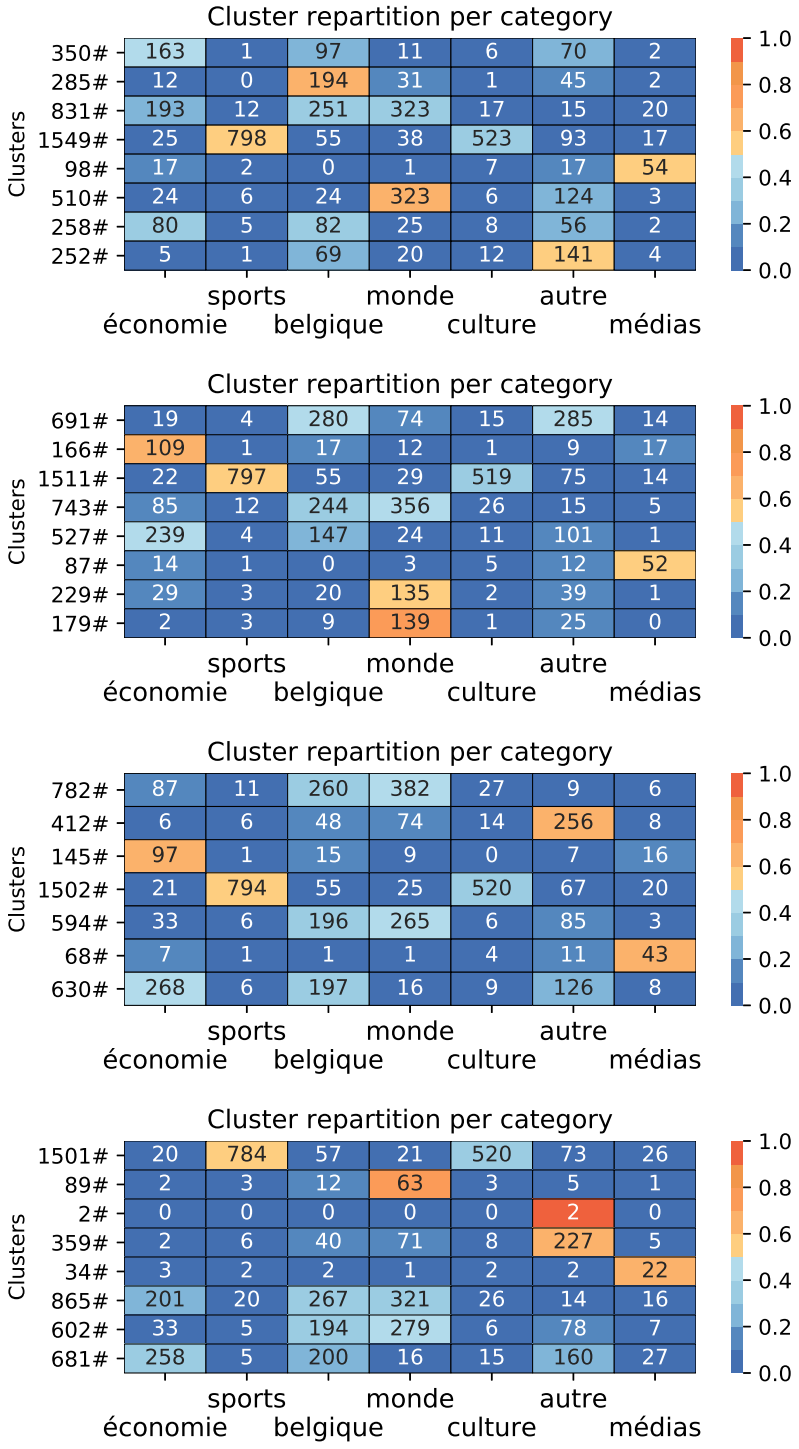

Figure A.6: Louvain outputs for $k = \{10, 20, 50, 100\}$

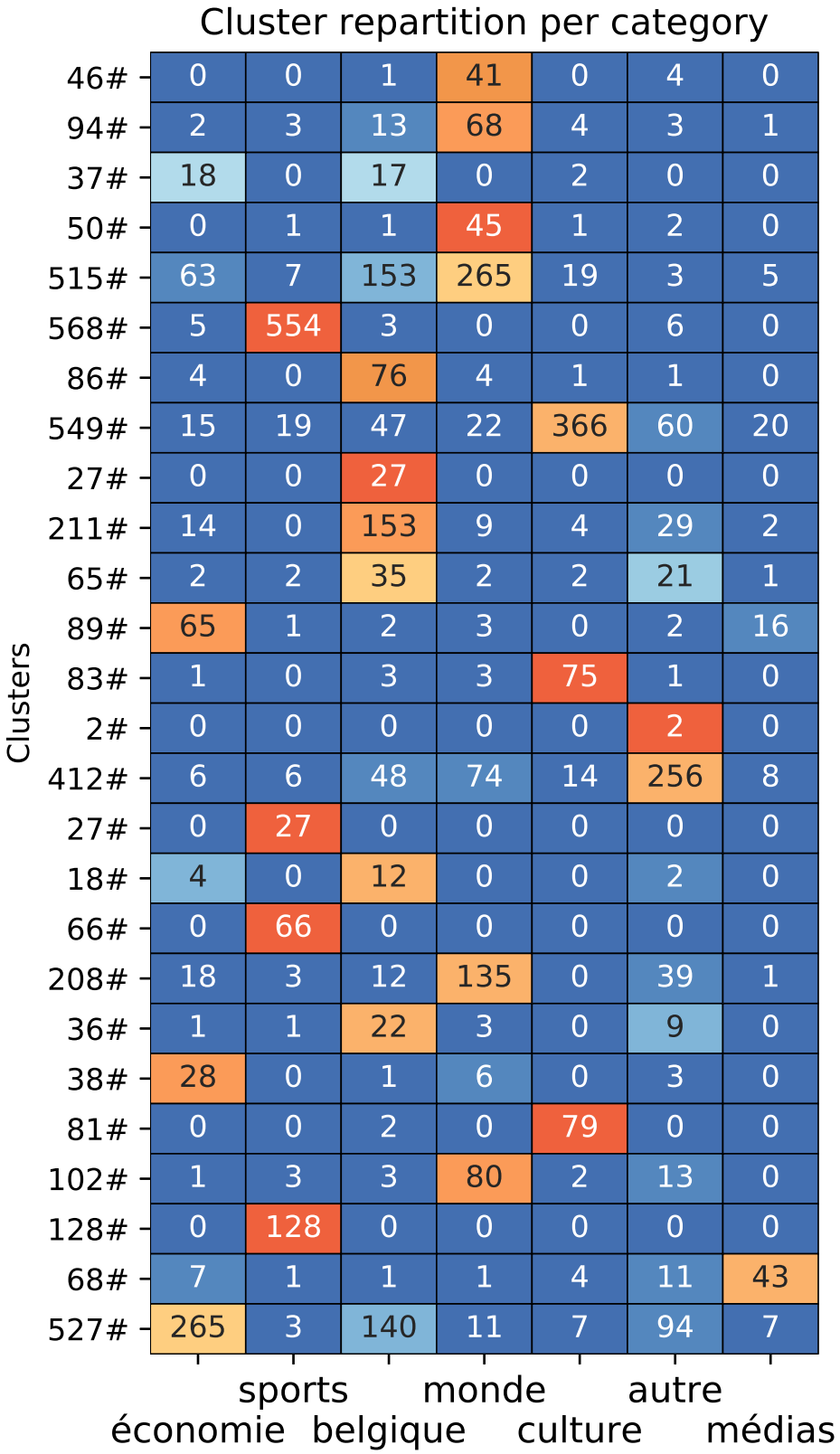

Figure A.7: Louvain, 50-connected graph, algorithm stopped 2 iterations before convergence

Appendix B

Classification

B.1 k-Means parameter selection [5.3.1]

Space	k	μ_p	σ_p
Embeddings	100	0.4558	0.0066
Embeddings	500	0.4848	0.0145
Embeddings	1000	0.4796	0.0113
'title+text'	100	0.6105	0.0163
'title+text'	1000	0.5603	0.0133

Table B.1: Scores of probability of correct classification using 5-fold cross-validation for the Direct method using k-Means

B.2 Random Forest parameter selection [5.3.2]

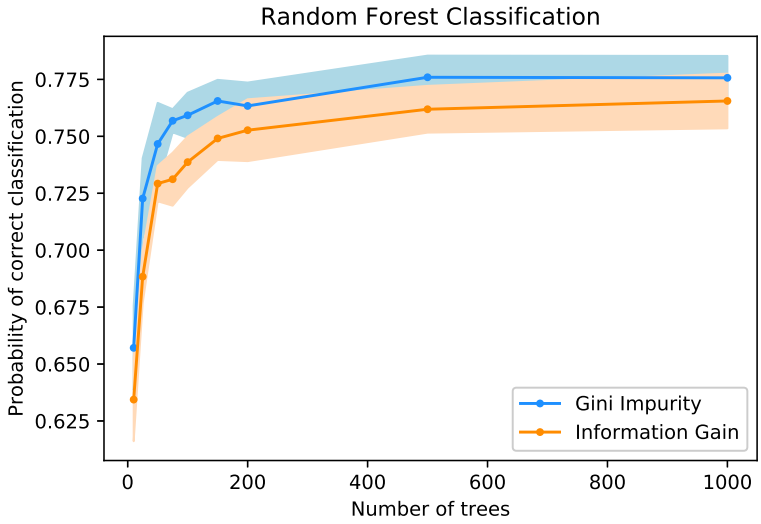

Figure B.1: Influence of the number of trees and the splitting criterion on performance

B.3 Multinomial Naive Bayes parameter selection [5.3.3]

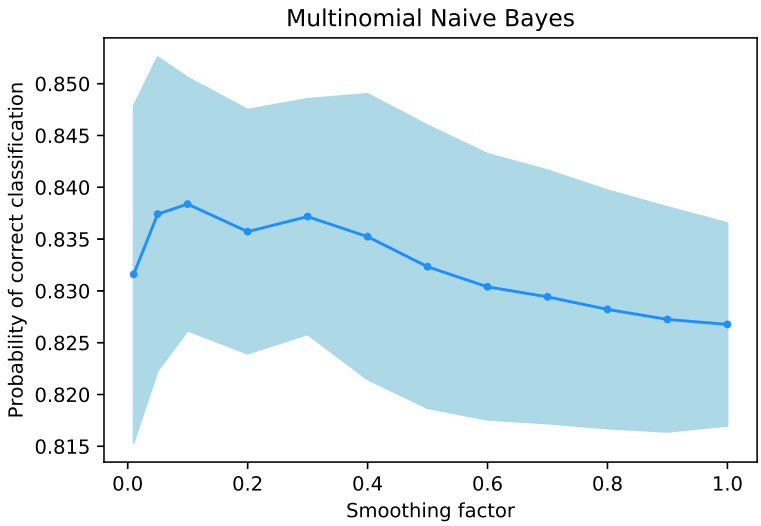

Figure B.2: Influence of the smoothing parameter on the performance

B.4 Neural Networks Model & Parameter Selection [5.3.4]

Layer (type)	Output Shape	Param #
embedding_2 (Embedding)	(None, 100, 200)	4000200
bidirectional_2 (Bidirection	(None, 100, 200)	180600
conv1d_2 (Conv1D)	(None, 98, 5000)	3005000
global_max_pooling1d_2 (Glob	(None, 5000)	0
dense_3 (Dense)	(None, 100)	500100
dropout_2 (Dropout)	(None, 100)	0
dense_4 (Dense)	(None, 7)	707

Total params: 7,686,607
Trainable params: 3,686,407
Non-trainable params: 4,000,200

$$\mu_p = 0.801, \sigma_p = 0.011$$

Layer (type)	Output Shape	Param #
embedding_3 (Embedding)	(None, 100, 200)	4000200
dropout_3 (Dropout)	(None, 100, 200)	0
lstm_1 (LSTM)	(None, 100)	120400
dropout_4 (Dropout)	(None, 100)	0
dense_5 (Dense)	(None, 7)	707

Total params: 4,121,307
Trainable params: 121,107
Non-trainable params: 4,000,200

$$\mu_p = 0.807, \sigma_p = 0.008$$

Layer (type)	Output Shape	Param #
embedding_5 (Embedding)	(None, 100, 200)	4000200
conv1d_3 (Conv1D)	(None, 100, 200)	120200
max_pooling1d_1 (MaxPooling1	(None, 50, 200)	0
lstm_2 (LSTM)	(None, 100)	120400
dense_8 (Dense)	(None, 7)	707

Total params: 4,241,507
Trainable params: 241,307
Non-trainable params: 4,000,200

$$\mu_p = 0.788, \sigma_p = 0.012$$

Layer (type)	Output Shape	Param #
embedding_6 (Embedding)	(None, 100, 200)	4000200
flatten_1 (Flatten)	(None, 20000)	0
dense_9 (Dense)	(None, 10)	200010
dense_10 (Dense)	(None, 7)	77

Total params: 4,200,287
Trainable params: 200,087
Non-trainable params: 4,000,200

$$\mu_p = 0.685, \sigma_p = 0.023$$

Layer (type)	Output Shape	Param #
embedding_7 (Embedding)	(None, 100, 200)	4000200
flatten_2 (Flatten)	(None, 20000)	0
dense_11 (Dense)	(None, 50)	1000050
dense_12 (Dense)	(None, 7)	357

Total params: 5,000,607
Trainable params: 1,000,407
Non-trainable params: 4,000,200

$$\mu_p = 0.765, \sigma_p = 0.015$$

Layer (type)	Output Shape	Param #
embedding_8 (Embedding)	(None, 100, 200)	4000200
flatten_3 (Flatten)	(None, 20000)	0
dense_13 (Dense)	(None, 200)	4000200
dense_14 (Dense)	(None, 7)	1407

Total params: 8,001,807
Trainable params: 4,001,607
Non-trainable params: 4,000,200

$$\mu_p = 0.760, \sigma_p = 0.012$$

Layer (type)	Output Shape	Param #
embedding_9 (Embedding)	(None, 100, 200)	4000200
flatten_4 (Flatten)	(None, 20000)	0
dense_15 (Dense)	(None, 300)	6000300
dense_16 (Dense)	(None, 7)	2107

Total params: 10,002,607
Trainable params: 6,002,407
Non-trainable params: 4,000,200

$$\mu_p = 0.751, \sigma_p = 0.009$$

Layer (type)	Output Shape	Param #
embedding_11 (Embedding)	(None, 100, 200)	4000200
conv1d_4 (Conv1D)	(None, 100, 200)	120200
lstm_3 (LSTM)	(None, 100)	120400
dense_20 (Dense)	(None, 7)	707

Total params: 4,241,507
Trainable params: 241,307
Non-trainable params: 4,000,200

$$\mu_p = 0.787, \sigma_p = 0.010$$

Figure B.3: Mean and standard deviation of the correct classification probability obtained by 5-fold cross validation on the training set

Appendix C

Adversarial Examples

The original article shown below, belongs to the 'Belgium' category. We show the modifications required to move it in all the other categories for the mulinomial Naive Bayes and Multi-Layer Perceptron classifiers.

Flandre : des mesures préventives contre les attaques de loups
Le ministre flamand en charge de la nature et de l agriculture koen van den heuvel a prévu des mesures préventives pour des éleveurs dont les animaux pourraient subir des attaques de loup(s). La prévention des dommages causés par des loups a reçu un coup de pouce des autorités flamandes qui vont subventionner une série de mesures préventives. Des investissements passés peuvent être subsidiés s'ils remplissent les conditions. Ce nouveau règlement doit être prêt pour début avril. Les éleveurs qui équipent leur terrain d un enclos contre les loups pourront récupérer 80% de leurs investissements. Les subsides seront disponibles dans les localités où la présence du loup est avérée. Plusieurs communes flamandes sont à risques : balen beringen bocholt bree dessel dilsen stokkem geel ham hamont achel hechtel eksel houthalen helchteren kinrooi lanaken leopoldsburg lommel maaseik maasmechelen meerhout mol oudsbergen peer pelt zonhoven et zutendaal .

Table C.1: Original Article [48]

In the following examples the ~~scraped~~ text corresponds to words that are removed from the original text, while **bold** text is text added to the original article. All these examples were made with a maximum time of 30 minutes each.

Target Category: 'Monde'

Flandre : des mesures préventives contre les attaques de loups
Le ministre ~~flamand~~ en charge de la nature et de l agriculture koen van den heuvel a prévu des mesures préventives pour des éleveurs ~~dont les~~ **qui redoutent que leurs** animaux ~~pourraient subir des attaques de~~ **subissent le saccage des** loup(s). La prévention ~~des dommages~~ **du carnage** causé~~s~~ par **la tragédie** des loups **assaillants et leurs pillages** a reçu un coup de pouce des autorités flamandes qui vont subventionner une série de mesures préventives. Des investissements passés peuvent être subsidiés s'ils remplissent les conditions. Ce nouveau règlement doit être prêt pour début avril. Les éleveurs qui équipent leur terrain d un enclos contre les loups pourront récupérer 80% de leurs investissements. Les subsides seront disponibles dans les localités où la présence du loup est avérée. **Selon les analystes,** plusieurs communes flamandes sont à risques : balen beringen bocholt bree dessel dilsen stokkem geel ham hamont achel hechtel eksel houthalen helchteren kinrooi lanaken leopoldsburg lommel maaseik maasmechelen meerhout mol oudsbergen peer pelt zonhoven et zutendaal .

Table C.2: Modifications to fool the MNB classifier

Flandre : des mesures préventives contre les attaques de loups
Le ministre flamand en charge de la nature et de l agriculture koen van den heuvel a prévu des mesures préventives pour des éleveurs dont les animaux pourraient subir des attaques de loup(s). La prévention des dommages causés par des loups a reçu un coup de pouce des autorités flamandes qui vont subventionner une série de mesures préventives. Des investissements passés peuvent être subsidiés s'ils remplissent les conditions. **Normalement,** ce nouveau règlement doit être prêt pour début avril. Les éleveurs qui équipent leur terrain d un enclos contre les loups pourront récupérer 80% de leurs investissements. Les subsides seront disponibles dans les localités où la présence du loup est avérée. Plusieurs communes flamandes sont à risques : balen beringen bocholt bree dessel dilsen stokkem geel ham hamont achel hechtel eksel houthalen helchteren kinrooi lanaken leopoldsburg lommel maaseik maasmechelen meerhout mol oudsbergen peer pelt zonhoven et zutendaal .

Table C.3: Modifications to fool the MLP classifier

Target Category: 'Médias'

Flandre : des mesures préventives contre les attaques de loups
Le ministre flamand en charge de la nature et de l agriculture koen van den heuvel ~~a~~ **, interviewé par un journaliste, a déclaré sur les ondes radios avoir** prévu des mesures préventives pour des éleveurs dont les animaux pourraient subir des attaques de loup(s) **guidés par la faim. Pour éviter la paranoia,** la prévention des dommages causés par des loups a reçu un **merveilleux** coup de pouce des autorités flamandes qui vont subventionner une série de mesures préventives. **Selon nos informations,** des investissements passés peuvent être subsidiés s'ils remplissent les conditions. **L'édition de** ce nouveau règlement doit être prêt**e** pour début avril. Les éleveurs qui **préviennet les attaques et** équipent leur terrain d un enclos contre les loups pourront récupérer 80% de leurs investissements. Les subsides seront disponibles dans les localités où la présence du loup est avérée. **Selon les médias , la classification des**~~Plusieurs~~ communes flamandes **qui** sont à risques **est**: balen beringen bocholt bree dessel dilsen stokkem geel ham hamont achel hechtel eksel houthalen helchteren kinrooi lanaken leopoldsburg lommel maaseik maasmechelen meerhout mol oudsbergen peer pelt zonhoven et zutendaal .

Table C.4: Modifications to fool the MNB classifier

Flandre : des mesures préventives contre les attaques de loups
Le ministre flamand en charge de la nature et de l agriculture koen van den heuvel a prévu des mesures préventives pour des éleveurs dont les animaux pourraient subir des attaques de loup(s) **a-t-il annoncé dans une interview à des reporters de la chaine rtl tvi**. **L'enjeu est que si les attaques perdurent cela se répercutera sur la viabilité des exploitations** La prévention des dommages causés par des loups a reçu un coup de pouce des autorités flamandes qui vont subventionner une série de mesures préventives. Des investissements passés peuvent être subsidiés s'ils remplissent les conditions. Ce nouveau règlement doit être prêt pour début avril. Les éleveurs qui équipent leur terrain d un enclos contre les loups pourront récupérer 80% de leurs investissements. Les subsides seront disponibles dans les localités où la présence du loup est avérée. **Selon les rédacteurs de rtl info,** plusieurs communes flamandes sont à risques : balen beringen bocholt bree dessel dilsen stokkem geel ham hamont achel hechtel eksel houthalen helchteren kinrooi lanaken leopoldsburg lommel maaseik maasmechelen meerhout mol oudsbergen peer pelt zonhoven et zutendaal .

Table C.5: Modifications to fool the MLP classifier

Target Category: 'Autre'

Flandre : des mesures préventives contre les attaques de loups
Le ministre flamand en charge de la nature et de l agriculture koen van den heuvel a prévu des mesures préventives **qui interviennent** pour des **paysans** dont les animaux pourraient subir des attaques de loup(s). La prévention des dommages causés par des loups a reçu un coup de pouce des autorités flamandes qui vont subventionner une série de mesures préventives **face à cette tragédie**. Des investissements passés peuvent être subsidiés s'ils remplissent les conditions. Ce nouveau règlement doit être prêt pour début avril. Les éleveurs qui équipent leur terrain d un enclos contre les loups pourront récupérer 80% de leurs investissements. **L'argent** ~~Les subsides~~ seront disponibles dans les localités où la présence du loup est avérée. Plusieurs communes flamandes sont à risques : balen beringen bocholt bree dessel dilsen stokkem geel ham hamont achel hechtel eksel houthalen helchteren kinrooi lanaken leopoldsburg lommel maaseik maasmechelen meerhout mol oudsbergen peer pelt zonhoven et zutendaal .

Table C.6: Modifications to fool the MNB classifier

~~**Flandre :**~~ **Des mesures préventives contre les attaques de loups**
Hier, le ministre flamand en charge de la nature et de l agriculture koen van den heuvel a prévu des mesures préventives pour des éleveurs dont les animaux pourraient subir des attaques de loup(s). La prévention des ~~dommages~~ **maux** causés par des loups a reçu un coup de pouce des autorités flamandes qui vont subventionner une série de mesures préventives. Des investissements **durables** passés peuvent être subsidiés s'ils remplissent les conditions. Ce nouveau règlement doit être prêt pour début avril. Les éleveurs **en péril** qui équipent leur terrain d un enclos **durable** contre les loups pourront récupérer 80% de leurs investissements. Les subsides seront disponibles dans les localités où la présence du loup est avérée. **Depuis hier,** plusieurs communes flamandes sont à risques : balen beringen bocholt bree dessel dilsen stokkem geel ham hamont achel hechtel eksel houthalen helchteren kinrooi lanaken leopoldsburg lommel maaseik maasmechelen meerhout mol oudsbergen peer pelt zonhoven et zutendaal .

Table C.7: Modifications to fool the MLP classifier

Target Category: 'Economie'

Flandre : des mesures préventives contre les attaques de loups
Le ministre flamand en charge de la nature et de l agriculture koen van den heuvel a prévu des mesures préventives pour des éleveurs dont les animaux pourraient subir des attaques de loup(s). La prévention des dommages causés par des loups a reçu un coup de pouce des autorités flamandes qui vont subventionner une série de mesures préventives. Des investissements passés peuvent être subsidiés s'ils remplissent les conditions. **Normalement,** ce nouveau règlement doit être prêt pour début avril. Les éleveurs qui équipent leur terrain d un enclos contre les loups pourront récupérer 80% de leurs investissements. Les subsides seront disponibles dans les localités où la présence du loup est avérée. Plusieurs communes flamandes sont à risques : balen beringen bocholt bree dessel dilsen stokkem geel ham hamont achel hechtel eksel houthalen helchteren kinrooi lanaken leopoldsburg lommel maaseik maasmechelen meerhout mol oudsbergen peer pelt zonhoven et zutendaal .

Table C.8: Modifications to fool the MNB classifier

Flandre : des mesures préventives contre les attaques de loups
Le ministre flamand en charge de la nature et de l agriculture koen van den heuvel a prévu des mesures préventives pour des éleveurs dont les animaux pourraient subir des attaques de loup(s). La prévention des dommages causés par des loups a reçu un coup de pouce des autorités flamandes qui vont subventionner une série de mesures préventives. Des investissements passés peuvent être subsidiés s'ils remplissent les conditions. Ce nouveau règlement doit être prêt pour début avril. Les éleveurs qui ~~équipent~~ **préfèrent équiper** leur terrain d un enclos contre les loups pourront récupérer 80% de leurs investissements. Les subsides seront disponibles **en priorité** dans les localités où la présence du loup est avérée. Plusieurs communes flamandes sont **menacées** ~~à risques~~ : balen beringen bocholt bree dessel dilsen stokkem geel ham hamont achel hechtel eksel houthalen helchteren kinrooi lanaken leopoldsburg lommel maaseik maasmechelen meerhout mol oudsbergen peer pelt zonhoven et zutendaal .

Table C.9: Modifications to fool the MLP classifier

Target Category: 'Culture'

Flandre : des mesures préventives contre les attaques de loups
Le ministre flamand en charge de la nature et de l agriculture koen van den heuvel a ~~prévu~~ **imaginé** des mesures préventives **novatrices qui intéressent les** ~~pour des~~ éleveurs dont les animaux pourraient subir **la tragédie** des attaques de loup(s) **qui surgissent solitaires ou en meute**. La prévention des dommages causés par des loups a reçu un **merveilleux** coup de pouce des autorités flamandes qui vont subventionner une série de mesures préventives. Des investissements passés **pour éviter cette horreur et pour la quiétude des hommes** peuvent être subsidiés s'ils remplissent les conditions. Ce nouveau règlement doit être prêt pour début avril. Les éleveurs qui équipent leur terrain d un enclos contre les loups pourront récupérer 80% de leurs investissements. Les subsides seront disponibles dans les localités où la présence du loup est avérée. Le palmarès des communes flamandes **qui** sont à risques **est**: balen beringen bocholt bree dessel dilsen stokkem geel ham hamont achel hechtel eksel houthalen helchteren kinrooi lanaken leopoldsburg lommel maaseik maasmechelen meerhout mol oudsbergen peer pelt zonhoven et zutendaal .

Table C.10: Modifications to fool the MNB classifier

Flandre : des mesures préventives contre les attaques de loups
Le ministre flamand en charge de la nature et de l agriculture koen van den heuvel a prévu des mesures préventives **subsidiant les** ~~pour des~~ éleveurs dont les animaux pourraient subir des attaques de loup(s) **comme celles qui ont dévasté la Flandre**. La prévention des dommages causés par des loups a reçu un coup de pouce des autorités flamandes qui vont subventionner une série de mesures préventives. Des investissements passés peuvent être subsidiés s'ils remplissent les conditions. Ce nouveau règlement doit être prêt pour début avril. Les éleveurs qui équipent leur terrain d un enclos contre les loups pourront récupérer 80% de leurs investissements **en retour**. Les subsides seront disponibles dans les localités où la présence du loup est avérée. **Les autorités locales reconnaissent la complexité de la définition claire des frontières au sein desquelles les remboursements seront possibles et se concertent encore à ce sujet. Pour l'instant** Plusieurs communes flamandes sont ~~à risques~~ **sont susceptibles d'être candidates**: balen beringen bocholt bree dessel dilsen stokkem geel ham hamont achel hechtel eksel houthalen helchteren kinrooi lanaken leopoldsburg lommel maaseik maasmechelen meerhout mol oudsbergen peer pelt zonhoven et zutendaal .

Table C.11: Modifications to fool the MLP classifier

Target Category: 'Sport'

Flandre : des mesures préventives contre les attaques de loups
Le ~~ministre~~ **leader** flamand en charge de la nature et de l'agriculture koen van den heuvel a prévu des mesures **d'attaque** ~~préventives~~ pour des éleveurs dont les animaux pourraient **par malchance,** subir des attaques de loup(s) **solitaires ou en meute**. La **course contre les** ~~prévention des~~ dommages causés par des loups a reçu un **costaud** coup de pouce des autorités flamandes qui vont subventionner une série de mesures préventives. Des investissements passés peuvent être subsidiés s'ils remplissent les conditions. Ce nouveau règlement doit être prêt pour début avril. Les éleveurs qui équipent leur terrain d un enclos contre les loups pourront récupérer 80% de leurs investissements. Les subsides seront disponibles dans les localités où la présence du loup est avérée. ~~Plusieurs~~ **Le palmarès des** communes ~~flamandes~~ **Limbourgeoises qui** sont à risques **est** : balen beringen bocholt bree dessel dilsen stokkem geel ham hamont **en onzième rang** achel hechtel eksel houthalen helchteren kinrooi lanaken leopoldsburg lommel maaseik maasmechelen meerhout mol oudsbergen peer pelt zonhoven et **en dernier rang** zutendaal .

Table C.12: Modifications to fool the MNB classifier

Flandre : des mesures préventives contre les attaques de loups
Le ministre flamand en charge de la nature et de l agriculture koen van den heuvel a prévu des mesures préventives pour des éleveurs dont les animaux pourraient subir des attaques de loup(s). La prévention des dommages causés par des loups a reçu un coup de pouce des autorités flamandes qui vont subventionner une série de mesures préventives. Des investissements passés peuvent être subsidiés s'ils remplissent les conditions. Ce nouveau règlement doit être prêt pour début avril. Les éleveurs qui ~~équipent leur terrain d un enclos~~ **s'équipent d'un enclos autour de leur terrain** contre les loups pourront récupérer 80% de leurs investissements. Les subsides seront disponibles dans les localités où la présence du loup est avérée. Plusieurs communes flamandes sont à risques : balen beringen bocholt bree dessel dilsen stokkem geel ham hamont achel hechtel eksel houthalen helchteren kinrooi lanaken leopoldsburg lommel maaseik maasmechelen meerhout mol oudsbergen peer pelt zonhoven et zutendaal .

Table C.13: Modifications to fool the MLP classifier

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