Institutional Repository - Research Portal
Dépôt Institutionnel - Portail de la Recherche
researchportal.unamur.be
University of Namur

THESIS / THÈSE

MASTER IN COMPUTER SCIENCE

Algorithms for the Partition and Labelling of Natural Language Document Sets

Puissant, Jean-François

Award date:
2017

Awarding institution:
University of Namur

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 18. Apr. 2024

Université de Namur
Faculty of Computer Science
Academic Year 2016–2017

Algorithms for the Partition and Labelling of Natural Language Document Sets

Jean-François Puissant

Internship mentor: Kevin Françoisse

Supervisor: _____ (Signed for Release Approval - Study Rules art. 40)
Benoît Frenay

A thesis submitted in the partial fulfillment of the requirements
for the degree of Master of Computer Science at the Université of Namur

Abstract

When looking for information online, search engines can return thousands of documents for a single query. Current solutions display results in a list fashion, and face difficulties showing more than a few items without drowning the user in information. Humans tend to only process a few chunks of information at once, but can work faster if the data is organised in a small number of groups. Fortunately, a combination of natural language processing algorithms and graphical representations have the potential to enable users to get a bird's eye view on the results, and help them navigate down to the content they are looking for. We faced two main challenges to build such a system.

First, how to group similar documents together based on their textual content? We propose a new scalable solution for document clustering based on Google's Doc2Vec model and the Louvain algorithm for community detection. This solution shows better performance than competing algorithms such as *Latent Dirichlet Allocation* and *K-Means*. We also discovered techniques to remove the reliance on meta-parameters with minimal performance impact. With the use of the *t-SNE* dimensionality reduction algorithm, visualisation of clusters and documents on a flat screen is possible.

Second, how to label document clusters to help users understand their content easily? We propose two models applicable to cluster and document labelling with promising results.

Preface

Research context

This Master's thesis rests on the work I have done during my internship at Sagacify S.P.R.L., an innovative start-up company located in Etterbeek, Brussels. Founded in 2012 by Kevin Françoisse et François Beuvens, it has grown over the years to be one of the most successful start-up of the ICAB Business Incubator. This organisation rents office spaces and provides a large panel of services to companies, and currently host Sagacify as well as over 10 other companies. This relationship has enabled Sagacify to focus on its business and expand with flexibility.

The company's expertise is quite diverse, focusing on two main sectors of activity. Sagacify has a team of experimented developers dedicated to the "Mobile" branch. This branch is specialised in the development of mobile applications for Android, iOS and Windows Phone platforms. They also have great expertise in Web development, having built many modern Web applications. Recently, the company opened up a new department focused on the development of custom *Machine Learning* software solutions for some select enterprise customers. Those solutions aim at the use of cutting edge *Natural Language Processing* (NLP) and *Big Data* techniques to provide insights on the competitive landscape in a given market.

Acknowledgements

Kevin Françoisse — CEO of Sagacify — oversaw my work and helped me perform the research and development tasks needed. He outlined the problems that Sagacify faced and provided me with access to the large datasets and expertise available in the company. Sagacify manages datasets that hold millions of news articles, in multiple languages. Those articles are retrieved from reputable online news websites by a set of programs developed by Sagacify, much like Google does for the purpose of indexing Web pages to speed-up Google Search. Each article in those datasets is composed of a title and a body of text, among other fields. Given that all the algorithms and models presented later in this document are trained on those datasets, I take this opportunity to greatly thank Sagacify for the invaluable access it provided for my research.

I also greatly thank my supervisor, Benoît Frenay, for the opportunity to work on this project and for the expert help he offered me during this work. His experience led me to try new solutions and approach each problem with the rigour necessary to suceed.

Contents

Preface **1**
Research context . . . 1
Acknowledgements . . . 1

1 Introduction **7**
1.1 Problem introduction . . . 7
1.2 Definition of the research objectives . . . 9
1.2.1 Document clustering . . . 9
1.2.2 Document cluster labelling . . . 9
1.2.3 Document cluster visualisation . . . 10

2 State of the art **13**
2.1 Overview . . . 13
2.2 Representation of words and documents . . . 13
2.2.1 Vector representations of words . . . 13
2.2.2 The Word2Vec algorithm . . . 15
2.2.3 Vector representation of documents . . . 16
2.2.4 The Doc2Vec algorithm . . . 17
2.3 Clustering . . . 18
2.3.1 Latent Dirichlet Allocation . . . 18
2.3.2 K-Means . . . 20
2.3.3 The Louvain algorithm . . . 20
2.4 Visualisation . . . 22
2.4.1 Visualisation of highly dimensional spaces . . . 22
2.4.2 Principal Component Analysis . . . 23
2.4.3 t-Distributed Stochastic Neighbourhood Embedding . . . 23

3 Document Clustering task **25**
3.1 Introduction . . . 25
3.2 Algorithm general description . . . 26
3.3 Data selection and preparation . . . 27
3.3.1 Training Dataset . . . 27
3.3.2 Reference Dataset . . . 30
3.3.3 Dataset transformation . . . 31

3.4 Vector representation of words and documents 33
3.4.1 Training a Doc2Vec Model 33
3.4.2 Inference of new document-vectors 34
3.5 Graph representation of a set of documents 35
3.5.1 Cosine similarity measure 36
3.5.2 Methods for neighbourhood selection 37
3.5.3 Fixed neighbours method 37
3.5.4 Fixed cutoff method 38
3.5.5 Automatic cutoff method 38
3.6 The Louvain algorithm for Clustering 41
3.6.1 Application to the graph representation 41

4 Document Cluster Labelling task 45
4.1 Introduction 45
4.2 Vector relationships with Word2Vec and Doc2Vec 46
4.2.1 Word-vectors relationships 46
4.2.2 Document-vectors and Word-vectors relationships 47
4.2.3 The "Least-angle regression" experiment 48
4.3 Linear models for document labelling 51
4.3.1 Translation model for document labelling 51
4.3.2 Ridge linear model for document labelling 53

5 Evaluation 55
5.1 Introduction 55
5.2 Document clustering task evaluation 55
5.2.1 Datasets and protocol 55
5.2.2 LDA 55
5.2.3 K-Means 62
5.2.4 Proposed algorithm for document clustering 69
5.2.5 Discussion 73
5.3 Document cluster labelling task evaluation 76
5.3.1 Datasets and protocol 76
5.3.2 Translation model 76
5.3.3 Ridge linear model 83
5.3.4 Discussion 89

6 Conclusion & perspectives 91
6.1 Document clustering task 91
6.1.1 Proposed algorithm for document clustering 91
6.1.2 Discussion 92
6.2 Cluster Labelling task 92
6.2.1 Proposed algorithm for document cluster labelling 93
6.2.2 Reflection and criticism 93

Bibliography 98

Appendix **99**
Test dataset articles 99
Politics article n°1 99
Politics article n°2 100
Health article n°1 101
Health article n°2 103
Tech article n°1 104
Tech article n°2 105
Business article n°1 106
Business article n°2 107
Business article n°3 108
Science-environment article n°1 108
Magasine article n°1 110
Magasine article n°2 113

Chapter 1

Introduction

1.1 Problem introduction

Sagacify faces the following challenges: how to give meaning and extract valuable business insights from the very large amounts of textual data available online. The purpose of those datasets is to enable a Google Search-like service that would give companies a way to learn about the competitive landscape in a given market.

Let's say a company X wants to get into the *Internet of Things* (IoT) market. First it needs to find out who are the players in the field, what are they offering and how the market is segmented. This newcomer needs a tool that helps him navigate through all the information available online and visualise how are those companies connected together. This tool would read and interpret the documents available online and provide a way for the user to send queries and elegantly display the relevant documents (figure 1.1).

Figure 1.1: SagaKnow: a Cognitive Entreprise Search engine [1]

A query returns a set of relevant articles; this result set is then processed and displayed to the user. The way the information is displayed is crucial: a query could return a very large amount of documents and would overwhelm the

user. We need a way to group similar documents together and only display the groups (*clusters*), enabling the user to narrow down his/her search to only one cluster if he/she desires. One way to display the documents of a cluster is via a graph representation as shown in figure 1.2, where each document is a node and two similar documents are connected together by an edge.

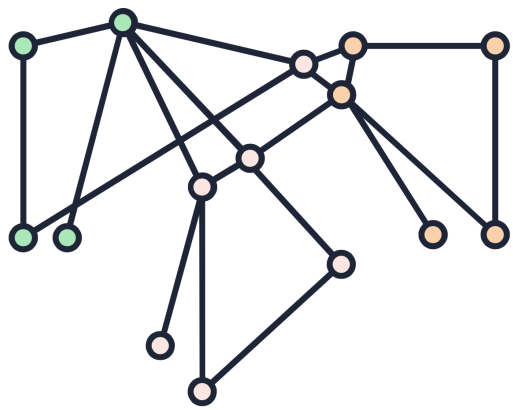

Figure 1.2: Related documents from a query result set [2]

The research at Sagacify is aimed at finding and developing algorithms that can enable a system to process and display a large set of documents in an intuitive way for the user. This problem can be decomposed in a few subtasks. The first task entails **finding clusters of documents** from an input set of documents. The second task entails **assigning a descriptive label to each cluster**, to help the user understand what the cluster represents. In a similar fashion, labels could be assigned to each document. The third task entails **displaying a set of documents as points in a 2D space**. This representation could help the user visualise more easily on a computer screen the sets of documents generated by the query.

Those algorithms could then be implemented in a testing environment in the Sagacify microservice infrastructure. In this environment, Sagacify's current systems would be upgraded to take advantage of the new algorithms without disrupting the production environment — a mission-critical system for Sagacify. If the testing phase goes smoothly and Sagacify is pleased with the improvements, then the new system will move on to the production environment. This way the new algorithms could improve the user's experience while using the *Cognitive Enterprise Search engine*.

1.2 Definition of the research objectives

With the informal description given above in mind, a formal definition of the research objectives that were pursued can be formulated. The three subtasks given in the above section will form the basis of our research objectives.

1.2.1 Document clustering

The first task in the informal description entails **finding clusters of documents** from an input set of documents. More formally, a *document* refers to a string of characters (in a given encoding, typically UTF-8) representing the textual content of an article in a given natural language (English, French, Dutch, German, etc). This string of characters holds the title (if applicable) and the body of the article, separated by a *newline* character.

Similar documents should be grouped together, while dissimilar documents should end up in different clusters. Two documents are similar if they deal with the same issue or talk about the same kind of topics. Finding the best criterion to split documents in groups given their content is up to the algorithm. The algorithm takes as input a set of documents

$$D \subseteq String \tag{1.1}$$

where *String* is the set of all chains of characters, and produces a partition of D into N clusters of documents. Let C be the set of clusters of D, where each document of D appears in one and only one cluster in C, i.e.

$$\forall doc \in D \quad doc \in cluster \Rightarrow \exists! cluster \in C \tag{1.2}$$

The number of clusters N is not assumed to be given as an input and needs to be determined by the algorithm itself. In this manner, the result set of documents generated by a query is the only input needed by the system to perform the clustering task.

1.2.2 Document cluster labelling

Once the document clustering step is done, the result set of documents is organised into a set of nameless clusters. To help the user understand what a given cluster of documents represent, it would be helpful to **assign a descriptive label to each cluster**. A *label* is a string of characters. The textual label of a cluster would be produced given the set of documents in this cluster. It is not limited to words or phrases present in the set of documents; any label is valid as long as it best represents the topic or content of the documents in the cluster.

The algorithm takes as input a set of clusters of documents C and outputs one label for each cluster of documents. Formally, it produces a function

$$\lambda : C \rightarrow String \tag{1.3}$$

mapping each cluster of C to a chain of characters representing its label.

As explained in the informal description, a label may also be given to each document from the result set of documents. This can be considered as a side task and not formally part of the *Document cluster labelling task*. Just as a cluster-wise labelling solution can help a user identify the meaning of each cluster in a group of clusters C, a document-wise labelling can help the user identify the subject of each document inside a cluster of similar documents. To this end, we can envision a system where the result set of a user's query is first organised in clusters of documents. The user then selects a cluster of interest, which will hide the other clusters and display the documents inside the selected cluster. At this point, the document-wise labelling algorithms will help the user identify the documents of interest.

The document-wise labelling algorithm takes as input a set of documents D' and outputs a label for each document. Formally, it produces a function

$$\lambda' : D' \rightarrow String \tag{1.4}$$

mapping each document of D' to a chain of characters representing its label.

1.2.3 Document cluster visualisation

Once documents are arranged into clusters, a way to **display the set of documents as points in a 2D space** is needed. This representation in 2D space enables the system to display the set of documents graphically on the user's display. Each document in the result set $D = \{doc \mid doc \in String\}$ is represented by a node on the screen, and the node's position is determined by this algorithm. Document-nodes that are members of the same cluster — per the algorithm in 1.2.1 — could be drawn in the same colour to illustrate their membership.

The algorithm takes as input the set of documents D and outputs a 2D-coordinate position for each of the documents. Formally, it produces a function

$$\rho : D \rightarrow \Re \times \Re \tag{1.5}$$

mapping each document of D to a pair of real-valued numbers.

In conjunction with a 2D projection for each document, we can define another algorithm for visualisation purposes. If each document is represented by a node on the user's screen, it would also be helpful to draw edges between similar documents. Similarity between documents has already been discussed in section 1.2.1. An algorithm can compare all pairs of documents of D together and yield a binary measure for each pair. If the two documents are found to be similar by the measure, an edge would be draw on the user's screen between the two nodes representing the similar documents.

The similarity algorithm takes as input the set of documents D and outputs a binary value for each of the documents. Formally, it produces a commutative function

$$\sigma : D \times D \rightarrow \{similar,\ dissimilar\} \tag{1.6}$$

$$\sigma(a, b) = \sigma(b, a) \tag{1.7}$$

mapping a pair of documents of D to one of two values. This document similarity algorithm can be used for visualisations purposes as shown in figure 1.3.

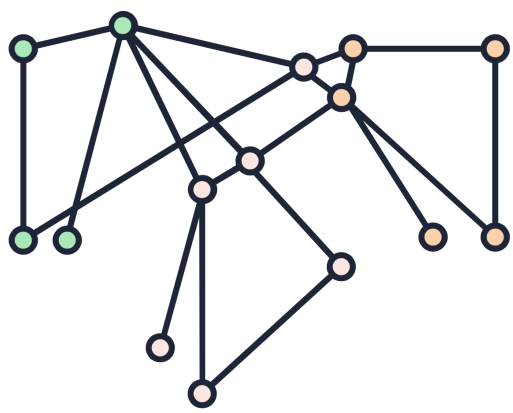

Figure 1.3: Illustration of document-nodes coloured by cluster, positioned in 2D space (equation 1.5) and connected to related documents (equation 1.6). [2]

Chapter 2

State of the art

2.1 Overview

In this chapter, the existing algorithms for the tasks outlined in section 1.2 will be discussed. To minimize the length of this chapter, only a subset of the relevent technology will be presented. Methods and algorithms that could in theory be used to fulfill the requirements outlined in section 1.2 but that are not used in this document will only be discussed briefly. In this manner, this chapter can be used by the reader as a short introduction and overview of algorithms that are put to use in the following chapters.

First, the problems surrounding word and document representations in NLP systems are presented and the relevant technology used by the proposed algorithms is described. Then, algorithms that can perform the clustering step given our desire to avoid meta-parameters are presented. Competing algorithms that will be used during the testing phase in order to quantify and compare the performance of our proposed algorithms are also discussed. Finally, a few well-known algorithms that perform data visualisation are presented.

2.2 Representation of words and documents

All information systems that manipulate words or documents must respect a given numerical representation for those entities in order to enable algorithms to efficiently perform the expected tasks. In this section, different representations available for words and documents will be presented. Innovative representations for words and documents enable cutting-edge performance of natural language processing software.

2.2.1 Vector representations of words

In most NLP systems, a numerical representation for words needs to be chosen given the specifics of the task at hand and the amount of data available. A

simple representation for words would be to assign to each word of a finite immutable vocabulary a unique number between 1 and the size of the vocabulary. Let W be the set of all words in a vocabulary, the function

$$r : W \rightarrow 1..|W| \tag{2.1}$$

returns a unique number for any given word of the vocabulary.

Another approach, often used for simple text classification tasks, is to transform each word into a vector representation. This is called a *feature vector*. Most linear models used for classification, such as *Support Vector Machines* (SVMs) or *Naïve Bayes Classifiers* work natively with those vector representations. A simple and widely used method to produce such representations is the *one-hot* representation [3]. In this system, each word is represented by a unique vector of size $|W|$ where only one component is nonzero. This nonzero component rests in the vector at the index x that corresponds to its number in the vocabulary.

$$\mathbf{V}_x = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_x \\ \vdots \\ v_{|W|} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \tag{2.2}$$

is the vector that represents the x-th word of W. Each word in the vocabulary has one and only one vector representation because the number $r(w)$ is unique. The one hot vector that represents the word w is thus $\mathbf{V}_{r(w)}$.

Infortunately this method has many disadvantages. First, the cardinality of the vectors can be very large, which means that the model that will try to learn from this data will have a very large number of parameters. Such models are more difficult to train and might exhibit worse performance than models with fewer parameters. This problem is compounded when the model has to consider multiple words, the number of parameters increasing rapidly following the growth of the feature space. This phenomenon is called the "Curse of dimensionality" [4].

A better approch is to represent words in a feature space that is as small as possible, i.e. that has a minimum amount of dimensions. One simple approach would be to use dimensionality reduction algorithms such as *PCA* (section 2.4.2) or *t-SNE* (section 2.4.3) on the one-hot representation in order to reduce the cardinality of the vector representations. This introduces a new meta-parameter that has to be tuned, the dimensionality of the vector representation. This meta-parameter is equal to the target number of dimensions of the dimensionality reduction algorithm.

2.2.2 The Word2Vec algorithm

Word2Vec [5, 6] is a recent algorithm that can produce very good vector representations thanks to the use of an innovative artificial neural network. The transformation of words into a vector space is usually called an "embedding" in the literature. The drawback of using this kind of word embedding is that the Word2Vec algorithm has to train a model on a lot of textual data before it becomes good at representing words. Word2Vec models are often used in Big Data and Cloud architectures where the large amounts of textual data are more easily managed.

The Word2Vec algorithm trains itself on unlabelled textual data. The textual data needs to be broken up into individual words, called "tokens". In many cases, even pieces of punctuation are considered individual tokens. The Word2Vec algorithm analyses these lists of tokens to train the word embedding. More precisely, the algorithm analyses windows of a few words at a time and moves these windows by one token position at each iteration. For example, let's consider a window of 5 tokens and a list of 10 tokens extracted from a document. The first window would cover tokens 1 through 5, the next window would cover tokens 2 through 6, etc. In our list of 10 tokens there would be 6 windows in total. In each window, the token at the center is considered the "target word" while the other tokens are the "context words".

Two model architectures are described in the original paper: the "Continuous Skip-gram" model (SG) and the "Continuous Bag-of-Words" model (CBOW) [5]. Both models are implemented with short neural networks with only one hidden layer. The SG model treats the target word as the input and tries to predict the context words on the output. The CBOW model does the opposite: it treats the context words as the input by taking the sum of their respective vector representations and tries to predict the target word. The neural network architecture is shown in figure 2.1.

The vectors fed as input and returned as output by the neural nets are in the form of one-hot vectors. The neural nets are trained by gradient descent and the error is computed as usual from the output of the network. More advanced Word2Vec models also use "hierarchical softmax" and "negative sampling" to boost the speed of the learning procedure and improve the neural net's performance [6]. The $Input \rightarrow Hidden\ layer$ transformation encodes the word embedding that we are trying to train. This transformation is encoded in a big matrix of shape $(size\ of\ the\ vocabulary,\ size\ of\ the\ word\ vectors)$ where the vector representation of each word from the vocabulary is stored. The $Hidden\ layer \rightarrow Output$ transformation is also encoded in a matrix that enables the model to predict the output word(s). The matrices are initialised with random noise and then trained by gradient descent. Once the model is trained, the content of the first matrix are kept as it represents the desired word embedding, while the second matrix is discarded.

The Continuous Skip-gram model generally performs better than the CBOW model, as shown in the original paper [5]. The original author focused on improving this SG model only in the second paper [6]. The Word2Vec algorithm

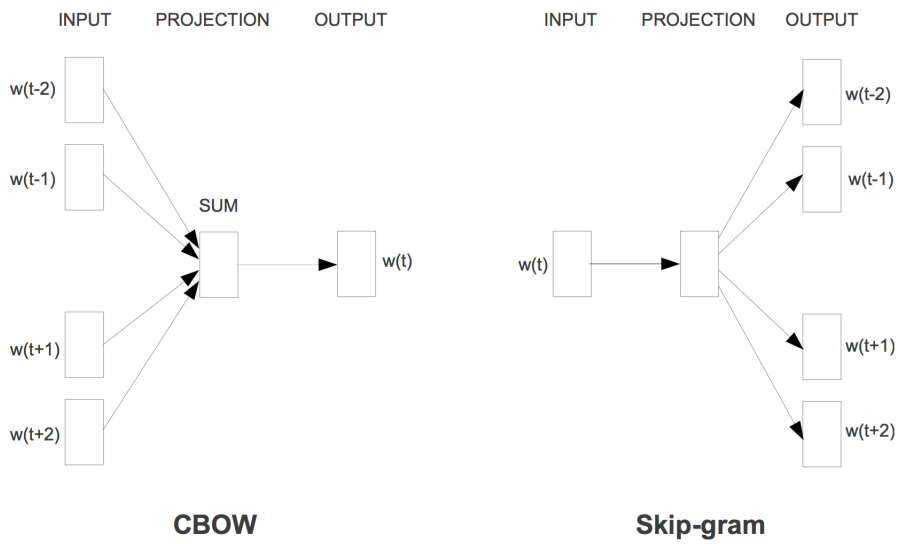

Figure 2.1: Neural net architectures of Word2Vec. The CBOW model predicts the target word based on the context words, and the SG model predicts context words based on the target word. [5]

is an important building block for many NLP applications as it can train on large amounts of data quickly and very often outperforms other algorithms in semantic and syntactic tests [5].

2.2.3 Vector representation of documents

Like words, whole documents can also be represented with feature vectors. A popular algorithm for document representation is the *TF-IDF* algorithm [7], short for "term frequency–inverse document frequency". This method represents documents as a vector of size equal to the vocabulary size, where the i-th vector component represents the frequency of the i-th word from the vocabulary in the document, divided by the frequency of the word in the whole corpus of documents. This last division is the "inverse document frequency" (IDF) weighting. IDF is used to reduce the contribution of frequent words in the document vector because frequent words such as "the", "he" or "this" do not add much information while less frequent words are usually much more informative.

Formally, let's define a document as a list of words

$$doc = word_1 \ word_2 \ ... \ word_n \tag{2.3}$$

and let D be the set of all documents for a given corpus. The TF-IDF vector

for a given document *doc* is

$$tfidf(doc) = tfidf(w_1 \ ... \ w_n) = \sum_{i=1}^{n} \frac{tf(doc, w_i)}{df(D, w_i)} \ \mathbf{V}_{r(w_i)} \tag{2.4}$$

where $tf(doc, w)$ corresponds to the number of occurences of the word w in the document *doc*, and $df(D, w)$ corresponds to the number of documents in D where the word w appears in at least once. The vector $\mathbf{V}_{r(w_i)}$ is the one-hot vector that represents the word w_i. Each one of the indididual vectors in the sum contributes to one component of the TF-IDF document vector. The algorithm is usually applied to a whole corpus D at once and returns a matrix of size $(|D|, \ |W|)$.

2.2.4 The Doc2Vec algorithm

The Doc2Vec algorithm [8] produces vector representations for documents as its name implies. It works in the same manner as Word2Vec, by analysing large amounts of documents via fixed-width windows of tokens. It is a strict evolution of the Word2Vec algorithm, in the sense that a Doc2Vec model has all the features of a Word2Vec model on top of its new functionality. In programming terms, one could say that Doc2Vec "extends" the Word2Vec algorithm while respecting the Liskov susbtitution principle [9].

The Doc2Vec neural network is modified to take into account a new input vector: the "paragraph vector" (also called document vector). This is shown in figure 2.2. The paragraph vector and the word vectors are concatenated together to form the hidden layer of the neural net. The paragraph vector can be thought of as another word for the model [8]. The word embedding matrix from the Word2Vec model is used to transform words into vectors, while the paragraph matrix stores the paragraph vectors of the training documents. The *Hidden layer* $\rightarrow$ *Output* transformation uses the concatenated vectors of the hidden layer to predict the target word. This configuration is called the "Distributed Memory" (PV-DM) model. Both the word embedding and the paragraph vectors are trained by gradient descent and hierarchical softmax [8, 10].

In the Doc2Vec model, both the *Input* $\rightarrow$ *Hidden layer* and *Hidden layer* $\rightarrow$ *Output* transformations matrices are kept after training. A trained model has a Word2Vec word embedding matrix and a paragraph vector matrix that corresponds to the embedding of the training documents. The power of a Doc2Vec model is its ability to "infer" paragraph vectors for new documents. During this inference stage, the word embedding matrix is locked to avoid any modification to the word embedding as we do not want further training. A paragraph vector is initialised with random noise and the neural net starts to perform its usual routine: predicting the target word in the window. As the Doc2Vec model iterates through the windows in the document, the new paragraph vector is updated by gradient descent. Once the vector converges, the algorithm stops and returns

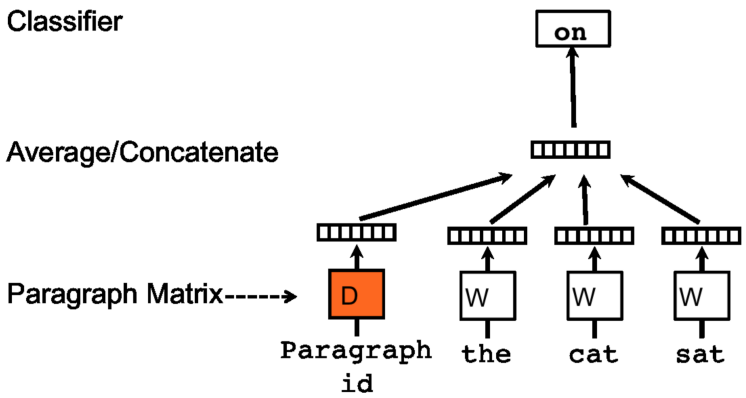

Figure 2.2: Neural net architecture of Doc2Vec. This "Distributed Memory" (PV-DM) model predicts the target word based on the context words in the window and the paragraph vector [8].

the vector. This procedure can be called at any time with the `infer_vector` function from the Gensim implementation of Doc2Vec [10].

2.3 Clustering

Clustering is the task of assigning a set of objects to groups (also called classes or clusters) so that the objects in the same cluster are more similar to each other (according to a predefined property) than to those in other clusters. This is a fundamental problem in many fields, including statistics, data analysis, bioinformatics, and image processing. Some of the classical clustering methods date back to the early 20th century and cover a wide spectrum: connectivity clustering, centroid clustering, density clustering, etc. The result of clustering may be a hierarchy or partition with disjoint or overlapping clusters. Cluster attributes such as count (number of clusters), average size, minimum size, maximum size, etc., are often of interest. [11]

In this section, three methods that relate to clustering will be discussed: *Latent Dirichlet Allocation*, *K-Means*, and the *Louvain algorithm*. The first algorithm is not strictly speaking a clustering algorithm, but can be used for "topic detection" with sets of documents. Those topics are often treated like soft clusters [12].

2.3.1 Latent Dirichlet Allocation

The Latent Dirichlet Allocation (LDA) algorithm is a well known algorithm for topic detection in the NLP field [13]. The original authors describe it as "a generative probabilistic model for collections of discrete data such as text corpora" [13]. The idea is that documents from a corpus D are represented as

a "mixture" of topics. The topics are in turn described by a distribution over words from the corpus D. The topics are said to be "latent" because while we can observe the documents and the words that compose them, the topics are not directly visible.

In the model, each document is represented by a vector of length equal to the number of desired topics k. This vector is a simplex, i.e. a vector where the sum of all its components equal to 1. This represents a mixture of topics. The number of topics k is a meta-parameter that is set before the model is trained. The simplex vector is modeled by the random variable $\theta \sim Dir(\alpha)$ where $Dir(\alpha)$ is the Dirichlet distribution.

Each topic is in turn modeled by a vector of size equal to the size of the word vocabulary. This vector represents a mixture of words that are thought to "generate" the latent topic. In other words, topics are similar to vector representations for documents as they are described in section 2.2.3. Each document from the corpus D of size M is modeled as lists of words

$$doc = w_1 \ w_2 \ ... \ w_N \tag{2.5}$$

where each word w_n is assigned to one of the topics, given by the random variable $z_n \sim Multinomial(\theta)$. The LDA can be represented as a probabilistic graphical model, shown in figure 2.3.

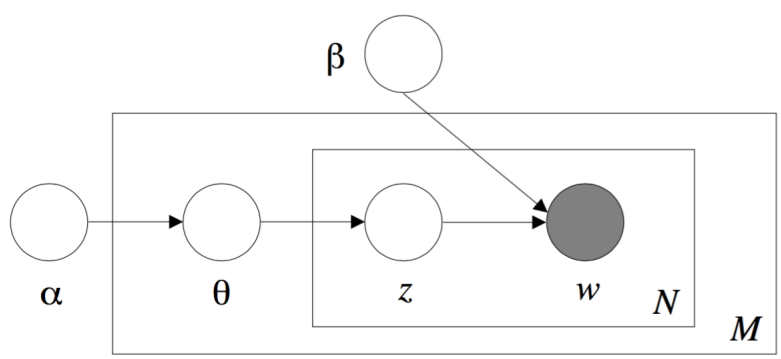

Figure 2.3: Probabilistic graphical model of LDA. The Dirichlet distribution $Dir(\alpha)$ determines the document mixture θ, that in turn determines the word membership to topics z, that in turn predicts the words w of the document given a prior word probabilities matrix β. [13]

The LDA model is trained iteratively with the documents of the corpus D by repeated sampling of the topic vector z for each word encountred. This enables the LDA model to associate documents with multiple topics. Once a LDA model has finished training, two matrices are produced: the "topic-term" matrix and the "document-term" matrix. The topic-term matrix is a matrix of size $(k, |W|)$ that stores the mixture of words for each of the k topics. This

mixture of words can be used to understand what each topic represents. The document-term matrix is a matrix of size $(|D|, k)$ that stores the mixture of topics for each document of D. This matrix can be seen as a soft version of the clustering task because each document does not fully belong to one topic but to a combination of multiple topics.

2.3.2 K-Means

The K-Means algorithm is a simple and popular algorithm for the clustering of vector data [14, 15]. The method aims to partition a set of vector instances into k clusters. Each instance belongs to the cluster with the nearest mean vector, called the "prototype" of the cluster. The problem of finding the optimal partition is very computationally expensive as this problem is NP-Hard [16]. Many efficient heuristic algorithms exist, but they converge to a local optimum. Restarting these heuristic algorithms multiples times is recommended to avoid that the results remain in a local optimum while better solutions are available.

The standard heuristic algorithm (or Lloyd's algorithm [14]) iteratively improves the clustering by the repeated application of two steps: the assignement of instances to clusters and the update of cluster prototypes. At initialisation, k prototypes are randomly selected from the existing instances.

1. During the assignement step, each of the N instances is assigned to the closest prototype by using the squared euclidean distance $||v||^2$. If multiple prototypes are the closest to an instance, it can be assigned to any one of those prototypes.

2. Once the assignement is done, each cluster prototype is updated to the mean of the instances that belong to the cluster. The prototypes are now a good representation for the instances of the cluster.

The assignement and update steps are repeated until the cluster prototypes converges. Usually, the algorithm is stopped when prototypes are moved less than a small distance ϵ between two iterations or if the maximum number of iterations is reached.

Since the k prototypes are chosen at random from instances during the initialisation phase, Lloyd's algorithm is not deterministic. It can benefit from multiple restarting with different prototypes at initialisation in order to produce better results. The quality of the clustering can be assessed by measuring the average distance between the instances and their respective prototypes; lower distances indicate better clusters. After multiple restarts, one can select the clustering that has the least average distance. Another problem is the fact that the k meta-parameter is hard to choose when not explicitly given by external information.

2.3.3 The Louvain algorithm

The Louvain algorithm is a method for community detection in large networks [17]. Those networks (or graphs) are composed of nodes that are linked

together with undirected weighted edges. Unlike K-Means, the Louvain algorithm does not rely on any "number of clusters" meta-parameter. Instead, the structure of the graph itself determines the number of clusters. This flexibility makes this algorithm very powerful in the many applications where determining the number of clusters is hard. However, some meta-parameters may be required for the process of building the graph from raw data instead.

A community in a graph is intuitively defined as a set of well connected nodes that is sparsely connected the other nodes in the graph. More precisely, communities are partitions of nodes that have great connectivity and share high weight edges with other nodes in the partition, and have lower connectivity and share lower weight edges to nodes outside the partition. Those communities can be thought of as the graph equivalent of clusters.

The modularity measures the quality of the division of a graph into communities, and is often used for the detection of those groups by algorithms. In the words of the Louvain algorithm authors, "The modularity of a partition is a scalar value between -1 and 1 that measures the density of links inside communities as compared to links between communities" [17]. In a graph $G = (V, E)$ that contains N nodes, the modularity is

$$Q = \frac{1}{2m} \sum_{i=1}^{N} \sum_{j=1}^{N} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j) \tag{2.6}$$

where A_{ij} represents the weight of the edge between node i and j,

$$k_i = \sum_{j=1}^{N} A_{ij} \tag{2.7}$$

is the sum of the weight of all edges attached to the vertex i, c_i is the community to which the vertex i is assigned, the function $\delta(u, v)$ is the Kronecker delta where

$$u = v \;\Rightarrow\; \delta(u, v) = 1 \tag{2.8}$$

$$u \neq v \;\Rightarrow\; \delta(u, v) = 0 \tag{2.9}$$

and m is the sum of all the edges weights

$$m = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} A_{ij} \tag{2.10}$$

The Louvain algorithm then works in two steps:

1. The algorithm iterates over all the nodes in the graph and assigns each node to a community if the assignement leads to an increase in modularity.

2. The algorithm then creates "super-nodes" out of the communities found the first step. Edges that connect two nodes from different super-nodes now connect the super-nodes. The algorithm goes back to the first step to consider communities of super-nodes.

At each iteration, the modularity is computed with the base graph G. The algorithm stops when no increase of modularity is possible. Those steps are illustrated in figure 2.4.

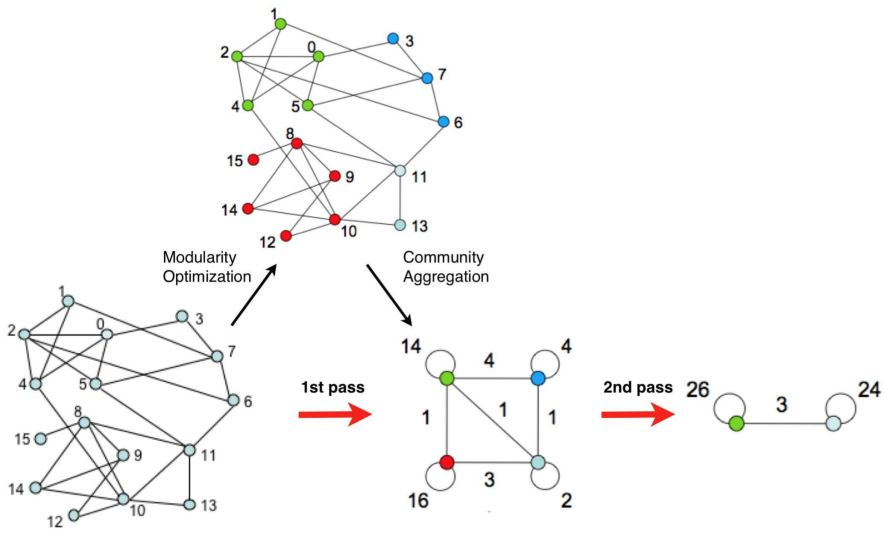

Figure 2.4: Each iteration is made of two phases: one where modularity is optimized by allowing only local changes of communities; one where the found communities are aggregated into super-nodes in order to build a new network of communities. The algorithm repeats those phases until no increase of modularity is possible. [17]

Although the underlying problem is NP-hard [11], the greedy approach of the Louvain algorithm combined with efficient heuristics enables the method to reach an excellent balance of solution quality (as measured by the modularity) and computational complexity. The time complexity of the algorithm seems to scale linearly with the number of edges [11].

2.4 Visualisation

2.4.1 Visualisation of highly dimensional spaces

Once the proposed algorithm for document clustering and the proposed algorithm for document cluster labelling are run on a given document dataset, vector representations for documents are inferred, relationships between documents are determined, and clusters of documents are produced. All this information is difficult for a human to process directly. But the purpose of those algorithms, outlined in section 1.2, is to provide a tool to help users visualise

large amounts of documents. A visualisation system that provides a graphical way of displaying the products of the algorithms is needed.

2.4.2 Principal Component Analysis

Principal Component Analysis (PCA) [18] is a procedure used for reducing the dimensionality of vector data by transforming the possibly correlated vector components into linearly uncorrelated components called the "principal components". The number of principal components is chosen beforehand and corresponds to the desired target number of dimensions after the transformation. All principal components are orthogonal to each other. The transformation is linear, applied by a matrix multiplication on the vector data. The first principal component points in the direction of the highest variance possible, and the succeeding principal components account for a maximum of variance while staying orthogonal to other principal components. For visualisation purposes, the number of principal components is usually 2.

The transformation of the input matrix $\mathbf{X}$ of size (n, p) into the target matrix $\mathbf{T}$ of size (n, m) is done via the matrix $\mathbf{W}$

$$\mathbf{T} = \mathbf{X}\mathbf{W} \tag{2.11}$$

where the columns of the $\mathbf{W}$ matrix are the eigenvectors of $\mathbf{X}^T\mathbf{X}$.

2.4.3 t-Distributed Stochastic Neighbourhood Embedding

The t-distributed stochastic neighbour embedding (t-SNE) [19] is an algorithm for dimensionality reduction of vector data. Unlike PCA, it is nonlinear and instead focuses on modeling the high-dimensional instances by low-dimensional objects in such a way that instances that are close together in the high-dimensional space are transformed into nearby objects in the low-dimensional space, while dissimilar instances are transformed into more distant objects.

More specifically, the algorithm works in three steps:

1. A probability distribution P is constructed over pairs of high-dimensional instances so that similar instances are more likely to be picked.
2. A similar probability distribution Q is constructed over the objects in the low-dimensional space.
3. The algorithm tries to minimise the Kullback–Leibler divergence between the two distributions by moving the objects in the low-dimensional space.

Given a set of high-dimensional instances $\mathbf{x}_1, ..., \mathbf{x}_N$, the algorithm computes the probabilities p_{ij} such that similar pairs $(\mathbf{x}_i, \mathbf{x}_j)$ are more likely

$$p_{j|i} = \frac{exp(-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma_i^2)}{\sum_{k \neq i} exp(-||\mathbf{x}_i - \mathbf{x}_k||^2/2\sigma_i^2)} \tag{2.12}$$

where the bandwidth of the Gaussian kernels σ_i is set in such a way that denser regions in the high-dimensional space have smaller bandwidth to compensate for the higher number of instances.

The pairwise probability p_{ij} is the mean of $p_{j|i}$ and $p_{i|j}$

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N} \tag{2.13}$$

The d-dimensional low-dimensional space is where the lower-dimensions objects are embedded. Those objects are denoted by $\mathbf{y}_1, ..., \mathbf{y}_N$ where $\mathbf{y}_i \in \Re^d$. This map has to be learned by the t-SNE algorithm. To compare the objects with the distribution of the higer-dimensional space, the pairwise similarities q_{ij} corresponding to the pairs $(\mathbf{y}_i, \mathbf{y}_j)$ are computed

$$q_{ij} = \frac{(1 + ||\mathbf{y}_i - \mathbf{y}_j||^2)^{-1}}{\sum_{k \neq m}(1 + ||\mathbf{y}_k - \mathbf{y}_m||^2)^{-1}} \tag{2.14}$$

This equation follows a "tail-heavy" Student-t distribution that is particulary effective at modeling dissimilar instances as far away objects in the d-dimensional space. The actual locations of the points $\mathbf{y}_i$ are, at first, initialized close to the orgin and with small amount of Gaussian noise. Their positions are then optimised by minimizing the Kullback–Leibler divergence of the distribution Q from the distribution P, i.e.

$$KL(P||Q) = \sum_{i \neq j} p_{ij} log \frac{p_{ij}}{q_{ij}} \tag{2.15}$$

Chapter 3

Document Clustering task

3.1 Introduction

The proposed algorithm for document clustering will be described in this chapter. A formal definition of the problem outlined for this task is available in section 1.2.1.

As explained in section 2.3, many clustering algorithms expect a given "number of clusters" meta-parameter. In this task, we do not have a defined number of clusters and would rather let the algorithm figure out the best number of clusters based on the input set of documents. Given those constraints, the algorithms such as LDA or K-Means presented in section 2.3 are not suitable for the task.

A popular clustering algorithm that does not require a preset "number of clusters" meta-parameter is the Louvain algorithm (2.3.3). It is a graph clustering algorithm, meaning that it produces a partition of nodes from an input graph. Its greedy approach to maximizing the modularity measure in the partitions means that it achieves fast execution. Notwithstanding its great speed, it produces good results. Those qualities motivated the rest of the research for a supporting algorithm that can take advantage of the Louvain algorithm.

The Louvain algorithm is designed to work on weighted digraphs. This means we first need to find a way to translate our set of documents in a graph representation. In this manner, each document would become one vertex in the graph and there would be edges between similar documents. The weight of each edge must represent the "similarity" between the two adjacent documents, as described briefly in section 1.2.1. The Louvain algorithm also does not favour complete graphs ; to get the best performance, one must select carefully which vertices are connected together. If dissimilar vertices share an edge, even with very low weight, the algorithm might output less qualitative results. This problem is discussed more in depth in section 2.3.3. This chapter focuses on how the proposed algorithm works with those features.

3.2 Algorithm general description

The proposed algorithm works as follows:

1. Each document from the input dataset $D = \{doc \mid doc \in String\}$ is transformed into a list of words by a *tokenizer*. For this step, any punctuation, symbol or newline character is considered to be a word by itself. The result of this step is a tokenized document dataset $t(D)$.

2. For each tokenized document, a N-dimensional vector representation is inferred by a trained Doc2Vec model. The transformation of a tokenized document into a real-valued vector by the Doc2Vec model is denoted by

$$\epsilon : t(D) \rightarrow \Re^N \tag{3.1}$$

   Note that the tokenized input dataset $t(D)$ and the training dataset for the Doc2Vec model are typically disjoint.

   This document embedding step produces a new representation for the documents of the dataset D. The transformed dataset $emdedding(D)$ is composed of every element of $t(D)$ mapped by the ϵ function

$$embedding(D) = \{\epsilon(doc) \mid doc \in t(D)\} \tag{3.2}$$

3. The vector-space representation $embedding(D)$ is going to be transformed into a graph representation. In this step, we build a weighted undirected complete graph $G = (V, E)$ where each vertex represents a document-vector of $embedding(D)$ and the weight of an edge between two vertices is function of the cosine similarity between the two adjacent document-vectors

$$cosine(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{|\mathbf{A}||\mathbf{B}|} \tag{3.3}$$

$$E = \{(\mathbf{A}, \mathbf{B}, f(cosine(\mathbf{A}, \mathbf{B}))) \mid \mathbf{A}, \mathbf{B} \in embedding(D)\} \tag{3.4}$$

$$V = embedding(D) \tag{3.5}$$

$$d = |V| = |embedding(D)| = |D| \tag{3.6}$$

4. An adjacency matrix A_G is computed such that the distance between two vertices of G is the cosine similarity of the two corresponding document-vectors.

$$w_{ij} = cosine(V_i, V_j) \quad (1 \leq i, j \leq d) \tag{3.7}$$

$$A_G = \begin{bmatrix} 1 & w_{12} & \dots & w_{1d} \\ w_{21} & 1 & \dots & w_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ w_{d1} & w_{d2} & \dots & 1 \end{bmatrix} \tag{3.8}$$

5. As explained in the introduction (3.1), the complete graph G needs to be pruned of unnecessery edges to enable the best performance of the Louvain algorithm. In this manner, the graph will be transformed into $G' = (V, E')$ where only a subset of the edges of G are kept. Different methods will be explored in section 3.5.2 to select the relevant edges and prune the others, based on the values from the adjacency matrix A_G. The set *prune* contains all the unnecessary edges that need to be removed

$$E' = \{(v_i, v_j, w_{ij}) \mid (v_i, v_j, w_{ij}) \in E \wedge (i, j) \notin prune(A_G)\} \tag{3.9}$$

6. We apply the Louvain algorithm on the pruned graph $G' = (V, E')$. As explained in section 2.3.3, the algorithm does not need any other parameter. This step produces a non-empty set of clusters of vertices in the graph G', a result that satisfies the requirements outlined in section 1.2.1.

$$C_{G'} = louvain(G') \subset \{cluster \mid cluster \subseteq V\} \tag{3.10}$$

$$\forall vec \in embedding(D)\ \exists! cluster \in C_{G'} \Rightarrow vec \in cluster \tag{3.11}$$

3.3 Data selection and preparation

3.3.1 Training Dataset

The training datasets are the sets of corpora that are exclusively used for the purpose of training the NLP models necessary for the proposed algorithm.

Sagacify has very large natural language datasets available, in 14 different languages. A subset of those languages is selected, for which there is a large enough amount of textual data available to train a model effectively. Sagacify uses a large ElasticSearch database to store its textual datasets, which enables us to execute queries to retrieve the articles of interest.

Only the datasets in German, English, Spanish, French and Dutch are large enough to train a complex model on. The articles written in those languages were selected for the training dataset, leaving us with no less than **244,706 articles**. This large amount of textual data weighs a couple gigabytes, extracted from Sagacify's ElasticSearch database and saved in the JSON format. A sample article in this JSON file format is displayed in listing 3.1. In this sample, one can see that the article has been annotated with so-called "entities" by the existing NLP pipeline of Sagacify. This additional data is simply ignored.

```
{ "title": "Larson back at Toronto Film Festival with
    shoot-em-up caper 'Free Fire'",
  "entities": {
    "person": [
      { "uri": "Armie_Hammer:dbp",
        "spans": [[187, 199], [1531, 1537]] },
      { "uri": "Cillian_Murphy:dbp",
        "spans": [[201, 215]] },
      { "uri": "Sharlto_Copley:dbp",
        "spans": [[220, 234]] },
      { "uri": "Ben_Wheatley:dbp",
        "spans": [[557, 569]] },
    ],
    "workOfArt": [
      { "uri": "Midnight_Madness_(film):dbp",
        "spans": [[947, 963]] },
      { "uri": "List_of_nocturnal_animals:dbp",
        "spans": [[1658, 1675]] }
    ],
    "facility": [
    { "uri": "2007_Toronto_International_Film_Festival
        :dbp",
      "spans": [[50, 85]] }
    ]
  },
  "date": "2016-09-10T00:23:35.000Z",
  "text": "Larson, who helped debut \"Room\" a year
      ago at the Toronto International Film Festival
      and went on to win best actress at this year's
      Academy Awards, stars alongside an ensemble
      including Armie Hammer, Cillian Murphy and
```

```
    Sharlto Copley in \"Free Fire,\" a dark comedy
    featuring a tense two-hour shoot-out. \"What was
    so cool about it is that we shot completely in
    chronological order aside from a couple small
    things,\" Larson told Reuters on the red carpet
    at the film's premiere. \"It's a very physical
    film,\" she added. Set in Boston, \"Free Fire,\"
    directed by Ben Wheatley, sees two groups of
    Irish, American and South African criminals meet
    at a derelict warehouse to conduct an arms deal
    that quickly goes wrong. A conflict between two
    men escalates into a shootout as everyone fights
    for survival, trying to pick off their enemies
    and get their hands on the suitcase filled with
    money in the middle of the room. The film debuted
     at the festival's Midnight Madness, where a
    rowdy audience whooped and clapped at
    particularly gruesome or comedic moments. As the
    sole woman, Larson's character often makes subtle
     quips about the absurdity of male egos. Sharlto
    Copley, who plays a South African businessman
    with a penchant for tailor-made suits and a
    flirtatious history with Larson's character, said
     he enjoyed sparring with the actress. \"We have
    like a little dynamic between our two characters,
    \" he said. \"I think being the only woman in the
     film, like she was - she had a lot to deal with,
     with a lot of testosterone around her.\" Hammer,
     who has two other films debuting at the film
    festival - historical slavery drama \"The Birth
    of a Nation\" and thriller \"Nocturnal Animals\"
    - said \"Free Fire\" had \"such a sense of
    camaraderie to it.\" \"I've never really felt
    that on any other project,\" he said. (Reporting
    by Piya Sinha-Roy; Editing by Cynthia Osterman)",
  "url": "http://uk.reuters.com/article/us-
    filmfestival-tiff-freefire-idUKKCN11G003?feedType
    =RSS&feedName=entertainmentNews",
  "locale": "en"
}
```


Listing 3.1: A sample JSON article from the training dataset

Given that Sagacify's datasets grow every day, it is important to select and save the training dataset outside of the production servers so that it does not

change over time. This way, it can be used to train different generations of the algorithm ; those generations can then be compared together during development to assess if a modification improves the algorithm's performance. Also, a dataset of a couple gigabytes can still be stored in a simple laptop hard drive and does not require access to a remote database server and the development of specialized scripts to enable "online" training. Finally, storing the dataset locally greatly improves the training speed and enables quick iterations while the algorithm is still in development.

3.3.2 Reference Dataset

For the purpose of testing the proposed algorithm for document clustering, a **reference dataset** has been selected. This reference dataset is disjoint from the datasets used to train the other models, such as Doc2Vec. This independence gives us confidence that the results produced during the testing phase are representative of the performance of the proposed algorithm on new, unseen data.

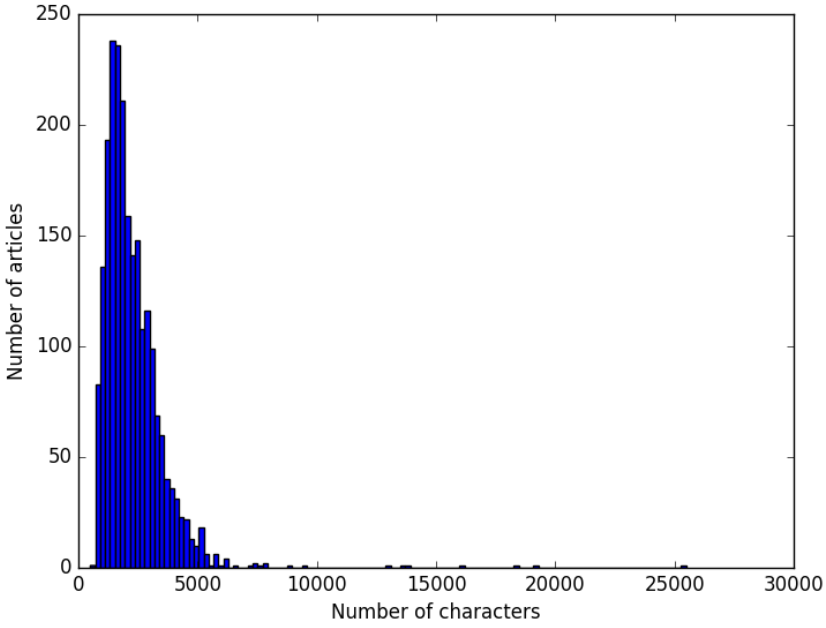

Figure 3.1: Histogram of the number of characters in the BBC dataset articles

We selected as reference dataset the "**BBC Dataset**", a collection of 2225 news articles written in English and published by the BBC on its online press portal during 2004 and 2005. This dataset is segmented in five topical areas : "*business*", "*entertainment*", "*politics*", "*sport*" and "*tech*". There are 510, 386,

417, 511 and 401 articles in each category respectively. Articles typically range from one to five thousands characters, as seen in figure 3.1.

3.3.3 Dataset transformation

The next step is to transform each article from a given dataset into a list of words, a.k.a. the *tokenisation* step. For this purpose, we use the "*Pattern*" library written in Python 2 by the Computational Linguistics Research Group (CLiPS) of the University of Antwerp (Belgium). This library features many useful tools for the processing of natural language, such as a multi-lingual tokenizer that currently supports English, French, Italian, Spanish, Dutch and German. This multi-lingual support is the reason this library has been selected instead of more well-known ones, such as Standford's *NLTK* library, that only supports English. A Python 3.0+ compatible version of the library is used, ported by the author at Sagacify.

The tokenization step is computationally expensive to execute on large datasets, such as the training dataset of section 3.3.1. To accomodate this, a **multiprocessing** approach is used to speed up the execution of this step. In Python, multithreading is not effective because all code executed by the CPython interpreter share the *GIL* (Global Interpreter Lock). This means that running multiple threads of execution in a single CPython process will not improve performance, as only one thread can effectively execute Python code at any given time. To take advantage of multiple hardware threads of execution, the Python code must run in multiple CPython processes simultaneously. This can be achieved with the "`multiprocessing`" package available in the Python standard library. A limitation of this system is that you cannot share data directly between multiple processes; one must use the "`Queue`" system instead that allows you to send messages between the processes. This message-passing paradigm helps you build scalable programs that fully benefit from the vast hardware ressources available in server clusters. The software architecture of the concurrent tokenizer is shown in figure 3.2.

The purpose of this concurrent tokenizer is to transform the training dataset into a "token file". The training dataset is a large JSON file, potentially too large to be loaded in one go into main memory. A better strategy is to read the file incrementally and extract each JSON article one by one, yielding it into the "article message queue" (fig. 3.2). The queues are bounded, only accepting a fixed amount of messages before blocking. The "parser" processes read from the first queue, parse one article and put the resulting list of tokens into the second queue. A simple Python example for this procedure is displayed in listing 3.2. The "token file writer" process then reads from this queue and writes the tokens into an output "token file". The format of this output file is very simple and meant for efficient extraction. Each list of tokens is written as one line in the output file, where tokens are separated by the *whitespace* character (0x20). Subsequent parsed documents are then separated by the *newline* character (0x0A). Another simple Python example for this procedure is displayed in listing 3.3.

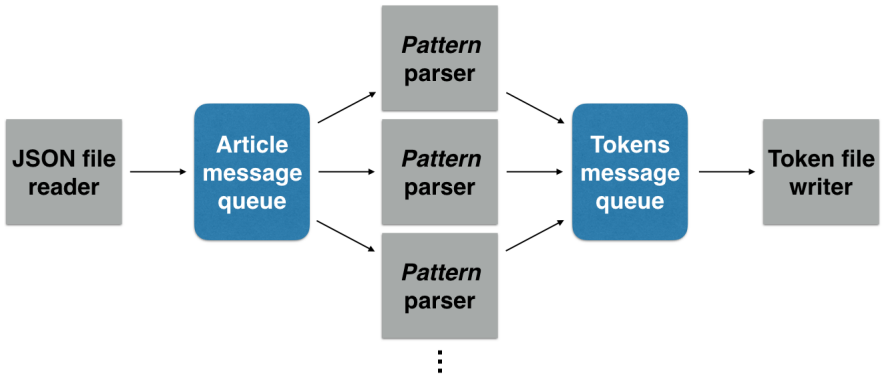

Figure 3.2: N+2 Processes pipeline for tokenization. Grey squares are processes, blue boxes are messages queues and arrows denotes the flow of data.

```
from pattern.text import parse

def json_article_parser(article_queue, token_queue):
    article = article_queue.get()
    while(article):
        text = article['title'] +'\n' +article['text']
        tokens = parse(text, lang=article['locale'])
        token_queue.put(tokens)
        article = article_queue.get()
    token_queue.put(None)
```

Listing 3.2: Example code for the "Pattern parser" process

```
def token_file_writer(token_queue, token_file):
    f_out = open(token_file, 'w')
    tokens = tokens_queue.get()
    while(tokens):
        line = ' '.join(tokens) +'\n'
        f_out.write(line)
        tokens = tokens_queue.get()
    f_out.close()
```

Listing 3.3: Example code for the "token file writer" process

The purpose of producing an intermediary representation for the dataset (the tokenfile) is to enable multiple traversals of the prepared dataset efficiently. As

the prepared dataset is computationally expensive to produce and potentially too large to fit into main memory, it makes sense to save it to disk and then iterate as many times as necessery over its contents. This is useful for training the Doc2Vec model, as discussed in section 3.4.1.

3.4 Vector representation of words and documents

3.4.1 Training a Doc2Vec Model

The Doc2Vec model, presented in section 2.2.4, serves two purposes at the same time: providing a vector representation for words and a vector representation for documents. This is possible as any Doc2Vec model shares all the functionality of a Word2Vec model and adds more capabilities regarding document representations. In the object-oriented programming paradigm, you can consider that the Doc2Vec class *extends* the Word2Vec class and follows the Liskov substitution principle [9]. This means that any Doc2Vec model is also a Word2Vec model, and provides the same functionality.

We used the Doc2Vec implementation offered by the *Gensim* Python library [10] that follows the paper by Quoc Le and Tomas Mikolov [8]. This library also offers a Word2Vec implementation [20], which the Doc2Vec implementation extends to offer more functionality. To create a new Doc2Vec model, the documentation specifies that a new "Doc2Vec" object needs to be created and that some basic parameters need to be set :

```
from gensim.models.doc2vec import Doc2Vec

model = Doc2Vec(documents, size=300, window=8,
                min_count=5, iter=10, dm=1,
                dm_mean=0, dm_concat=0)
```

Listing 3.4: How to initialize and train a new Doc2Vec model [10]

The `documents` variable is an iterable of documents, i.e. a generator function that reads parsed documents from the tokenfile (of section 3.3.3) and yields each document to the Doc2Vec model. The `size` parameter defines the dimensionality of the desired vector representation for both words and documents. The `window` and `min_count` parameters usage is explained in section 2.2.4. In our case, we set those parameters to **300**, **8** and **5** respectively following the results given by the "Empirical evaluation of doc2vec..." paper by Jey Han Lau and Timothy Baldwin [21]. The other parameters, `dm`, `dm_mean` and `dm_concat` will configure the model to use the "Distributed Memory" model with addition of vectors (`dm_mean=0`) instead of computing the mean of vectors or concatenating them (`dm_concat=0`). If this last parameter is set to 1 instead of 0, the resulting model will be much larger and more complex. This configuration would be better suited if we had much more data available for training [8]. At the end of

the day, the parameters diplayed in listing 3.4 offer great general purpose performance and can still be tuned by cross-validation when more data becomes available if we are willing to consume the vast computational ressources that are required. For the purpose of our research however, we can demonstrate the performance of the proposed algorithms without requiring costly cross-validation of meta-parameters.

This Doc2Vec model was trained with the training dataset of section 3.3.1. This dataset contains 250k documents and 150M tokens. Compared to the datasets used in [21], our training dataset is quite modest in size. Nonetheless, the Doc2Vec model took a couple hours to train on a dedicated machine powered by a Core i7-3615QM (3.3 GHz) and 16GB of DDR3. The `workers` variable has been set to **8** given the specifications of this processor. Training on a larger dataset would of course require more time and system memory. Once the model is trained, it contains both a trained Word2Vec and Doc2Vec model. We can check that they both work by executing a simple task :

```
>>> model.most_similar(positive=['woman', 'king'],
    negative=['man'])
[('queen', 0.4628202915191), ...]

>>> model.infer_vector(['a', 'sample', 'document', '
    for', 'testing', '.'])
array([ -5.37366718e-02,  -3.64980032e-03,
        ...
        -1.76466368e-02], dtype=float32)
```

Listing 3.5: Quick testing of a trained Doc2Vec model

The trained model is then saved to disk to be used in subsequent experiments. We use the well-known `pickle` module to serialize our `model` object :

```
import pickle

with open('doc2vec_model_multi.pickle', 'wb') as file:
     pickle.dump(model, file)
```

Listing 3.6: Saving a trained Doc2Vec model to disk

3.4.2 Inference of new document-vectors

The purpose of the Doc2Vec model is the inference of vector representations for new, unseen documents. As demonstrated in listing 3.5, the model offers the `infer_vector` method to produce new document-vectors. To transform the reference dataset of section 3.3.2 we simply read each article from disk, parse the

contents, and infer a new document-vector with the model. Given that articles are arranged by category, we store the resulting document-vectors in a Python map data structure — a `dict` — where each category name is mapped to a list of document-vectors corresponding to the articles in this category :

```
import os
from pattern.text import parse

def infer_vectors_from_files(dataset_path
                             ='./bbc_dataset/'):
    vectors = {}
    for category in os.listdir(dataset_path):
        # e.g. category == 'business'
        vectors[category] = []
        path = dataset_path +category +'/'
        for file in os.listdir(path):
            with open(path+file) as file:
                text = file.read()
            tokens = parse(text, lang='en')
            vector = model.infer_vector(tokens)
            vectors[category].append(vector)
    return vectors
```

Listing 3.7: Inference of new document-vectors from the reference dataset

In reference to the formalism shown in the algorithm general description of section 3.2; if we consider the set of documents from the reference dataset as D, this `vectors` data structure can thought of as the projection of the dataset $embedding(D)$ where ϵ denotes the application of the `infer_vector` method of the Doc2Vec model :

$$embedding(D) = \{\epsilon(doc) \mid doc \in t(D)\} \tag{3.12}$$

3.5 Graph representation of a set of documents

Given the document-vectors produced in the section 3.4.2 above, we will now build an equivalent graph representation as outlined in step (3) of the algorithm general description of section 3.2. To do so, each document-vector inferred from the reference dataset will be mapped to one unique vertex in the weighted complete graph $G = (V, E)$. For simplicity we assume that the vertex set V and $embedding(D)$ are equivalent. The weight of the edge between

two adjacent document-vectors will be equivalent to a function of the *cosine similarity* measure of those two vectors (3.5.1).

3.5.1 Cosine similarity measure

The cosine similarity measures the similarity in direction between two vectors of the same cardinality by calculating the cosine of the angle between those vectors. The cosine is equal to the dot product of the normalized vectors.

$$cosine(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{|\mathbf{A}||\mathbf{B}|} = \frac{\sum_{i=1}^{N} A_i B_i}{\sqrt{\sum_{i=1}^{N} A_i^2}\sqrt{\sum_{i=1}^{N} B_i^2}} \tag{3.13}$$

The cosine similarity can take values in the interval [-1, 1]. If two vectors have different magnitudes but share the same direction, for example the vectors $\mathbf{v}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ and $\mathbf{v}_2 = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$, the cosine similarity measure of those two vectors will be 1. If the cosine is equal to -1, the vectors have inverse directions. If the cosine is equal to 0, the vectors are orthogonal ot each other. Thanks to the use of normalized vectors, the magnitude of the vectors becomes irrelevent.

This measure is very useful to compare word-vectors and document-vectors and has seen extensive use in the literature [6, 21, 22, 23]. When comparing word-vectors together for example, a cosine of 1 means that the words are identical in meaning, and a cosine of 0 means that the words do not share meaning. In the same manner, documents that have similar semantic content will have high cosine similarities compared to other pairs of documents [21]. Authors cite that given the high dimensionality involved with word and document embeddings, empirically the cosine similary performs well nonetheless and has established itself as the go-to method for this task. Given its popularity and effectiveness, we chose to use it as well.

In a weighted graph, it is expected that the weights are nonnegative. As the cosine similarity has codomain [-1, 1], we chose to shift those values linearly to the range [0, 1] with the help of the f function. We then use $f \circ cosine$ as the edge weighting method.

$$f(x) = \frac{1+x}{2} \tag{3.14}$$

$$E = \{(\mathbf{A}, \mathbf{B}, f(cosine(\mathbf{A}, \mathbf{B}))) \mid \mathbf{A}, \mathbf{B} \in embedding(D)\} \tag{3.15}$$

We can then easily build the adjacency matrix A_G.

$$w_{ij} = cosine(V_i, V_j) \quad (1 \leq i, j \leq d) \tag{3.16}$$

$$A_G = \begin{bmatrix} 1 & w_{12} & \dots & w_{1d} \\ w_{21} & 1 & \dots & w_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ w_{d1} & w_{d2} & \dots & 1 \end{bmatrix} \tag{3.17}$$

3.5.2 Methods for neighbourhood selection

Once the graph G is built, we need to prune the unnecessary edges off to enable the Louvain algorithm to do its work. A cluster (or community) in a graph is intuitively defined as a set of densely connected vertices that is sparsely connected to other clusters in the graph. In a complete graph, all vertices are connected to each other which means there are no natural communities, unless we consider the weight of the edges and perform an exhaustive search. The Louvain algorithm is not meant to be used on complete graphs — even if they are weighted — because the greedy nature of the algorithm leads to the consideration of only a subset of edges. This greedy approach will cause the algorithm to underperform on complete graphs. In the original paper, the authors benchmark the performance of the algorithm on different graphs, none of them complete. The ratio between the number of vertices and the number of edges ranges from 0.0498 for the "Web uk-2005" dataset to 0.4416 for the small "Karate" dataset [17]. We need to prune our graph G to approximatively reach this range of ratios. In doing so, we create a new graph $G' = (V, E')$ where only a subset of edges remain.

$$E' = \{(v_i, v_j, w) \mid (v_i, v_j, w) \in E \wedge (i, j) \notin prune(A_G)\} \tag{3.18}$$

The prune-set *prune* is built with one of the methods for neighbourhood selection described below. For simplicity, we assume that E, E', and *prune* are commutative.

$$(v_i, v_j, w) \in E \Rightarrow (v_j, v_i, w) \in E \tag{3.19}$$

$$(v_i, v_j, w) \in E' \Rightarrow (v_j, v_i, w) \in E' \tag{3.20}$$

$$(i, j) \in prune \Rightarrow (j, i) \in prune \tag{3.21}$$

3.5.3 Fixed neighbours method

This first method is the simplest. For every vertex of G, we sort the edges connected to it by weight in descending order. We then add to *prune* any edge that is not in the top m elements of the sorted list. The number m is a new meta-parameter, that needs to be tuned to enable the best performance. This method performed very well on the reference dataset, but the presence of a

meta-parameter is not ideal if the algorithm is expected to perform optimally with new data.

$$E_i = \{(v_i, v_j, w) \mid (v_i, v_j, w) \in E\} \tag{3.22}$$

$$sort(E_i) = [e_1, e_2, ...e_n] \mid e_x, e_{x+1} \in E_i \wedge e_x.weight \geq e_{x+1}.weight \tag{3.23}$$

$$\forall x \in 1..d \;\; (v_x, v_j, w) \in sort(E_x)[m:] \Rightarrow (x, j) \in prune \tag{3.24}$$

3.5.4 Fixed cutoff method

In theory the *Fixed neighbours method* does not promote connectivity between vertices in dense regions in comparison to vertices in sparse regions (outliers) as every vertex will be connected to around m others regardless. Higher connectivity in denser regions might be beneficial to the performance of the Louvain algorithm. To explore this possibility, we tested another meta-parameter-based method based on the "minimum weight" cutoff parameter c. The idea is to prune any edge that has an associated weight lower than c.

$$E' = \{(v_i, v_j, w) \mid (v_i, v_j, w) \in E \wedge w \geq c\} \tag{3.25}$$

3.5.5 Automatic cutoff method

This third method is an extension of the *Fixed cutoff method* where the cutoff parameter is chosen dynamically given the specific weight distribution in the edges of the complete graph $G = (V, E)$. This weight distribution varies given the document dataset used to produce the graph G with the steps (1) to (4) described in section 3.2. We studied the distribution of the cosine similarity between document-vectors of the reference dataset and tried to find out to which distribution family it belongs. The distribution is shown in figure 3.3. Upon visual inspection, we can see that this distribution looks like it belongs to the Gamma family. We fitted a Gamma model to this data with a maximum likelihood algorithm in order to infer the three parameters of the potential Gamma distribution: $\Gamma(\alpha = k,\ \theta = scale,\ bias)$. The power density function of the fitted Gamma distribution is overlayed over an histogram of a sample of all the cosine similarity measures in figure 3.3.

Fitting a Gamma distribution to the cosine similarity distribution gives us the opportunity to use a dynamic cutoff that depends on the specific distribution of a given dataset. This way, no meta-parameter needs to be tuned for the algorithm to have good performance. Of course, the *Fixed neighbours method* or *Fixed cutoff method* with manual tuning might display better performance than this method. In an environment where the input dataset to the proposed algorithm is variable, a general purpose method for neighbourhood selection

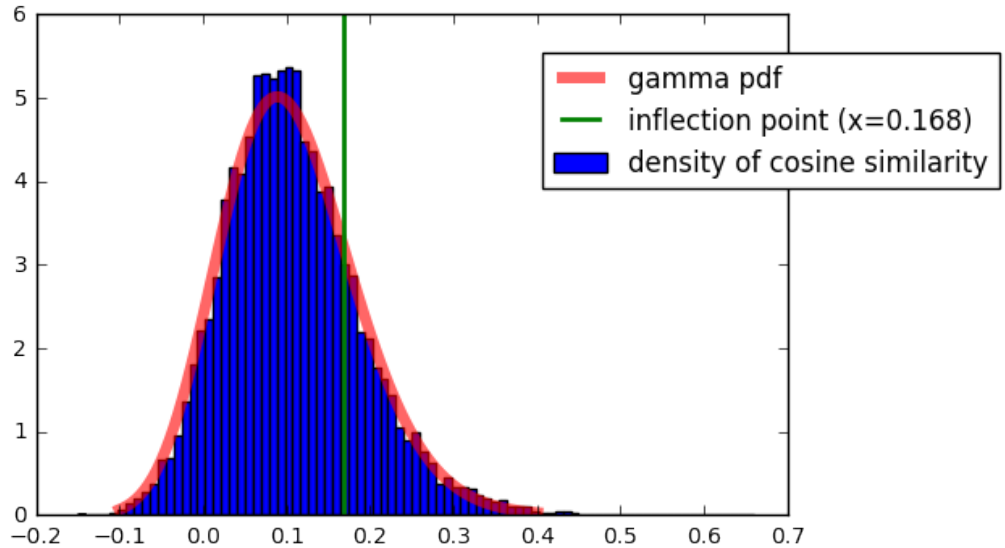

Figure 3.3: Histogram of the values of the cosine similarity between a sample population of document-vectors and their neighbours in the reference dataset. The curves shown are the PDF of the fitted Gamma distribution of parameters (k = 22.2759, bias = -0.2726, scale = 0.017) and the inflection point of the PDF. The vertical axis is not to scale.

that does not require tuning is preferred. In this manner, the *Automatic cut-off method* enables the proposed algorithm to fulfill the product requirements outlined in section 1.1.

Once the parameters of the Gamma distribution are estimated, we can compute the expression of the first, second, and third order derivatives of the power density function. The expression of the first derivative can be found easily, but the other expressions are much more difficult to come across. For reference, the expression of the power density function f and its derivatives are shown in equations 3.26 to 3.29 inclusive.

$$f(x \mid k, \theta) = \frac{x^{k-1}\ e^{-\frac{x}{\theta}}}{\theta^k\ \Gamma(k)} \tag{3.26}$$

$$\frac{f(x \mid k, \theta)}{dx} = -\frac{x^k\ e^{-\frac{x}{\theta}}}{\theta^{k+1}\ \Gamma(k)} + \frac{x^{k-2}\ e^{-\frac{x}{\theta}}}{\theta^k\ \Gamma(k)}(k-1) \tag{3.27}$$

$$\frac{f(x \mid k, \theta)}{dx^2} = -\frac{kx^{k-1} - \frac{x^{k+1}}{\theta}}{\theta^{k+1}\ \Gamma(k)\ e^{\frac{x}{\theta}}} + \frac{(k-2)x^{k-3} - \frac{x^{k-1}}{\theta}}{\theta^k\ \Gamma(k)\ e^{\frac{x}{\theta}}}(k-1) \tag{3.28}$$

$$\frac{f(x \mid k,\theta)}{dx^3} = (k-1)\frac{(k-2)(x^{k-4}(k-3) - \frac{x^{k-3}}{\theta}) - (k-2)\frac{x^{k-3}}{\theta} + \frac{x^{k-2}}{\theta^2}}{\theta^k\ \Gamma(k)\ e^{\frac{x}{\theta}}} - \frac{(k-1)(x^{k-3}(k-2) - \frac{x^{k-2}}{\theta}) - (k-1)x^{k-2} + \frac{x^{k-1}}{\theta^2}}{\theta^{k+1}\ \Gamma(k)\ e^{\frac{x}{\theta}}} \quad (3.29)$$

We provide the figure 3.4 to help the reader visualise the shape of those functions. Only the second and third order derivatives are shown, on top of the power density function and the corresponding histogram of cosine similarities. The inflection point lies in fact at the abcissa where the second order derivative is zero.

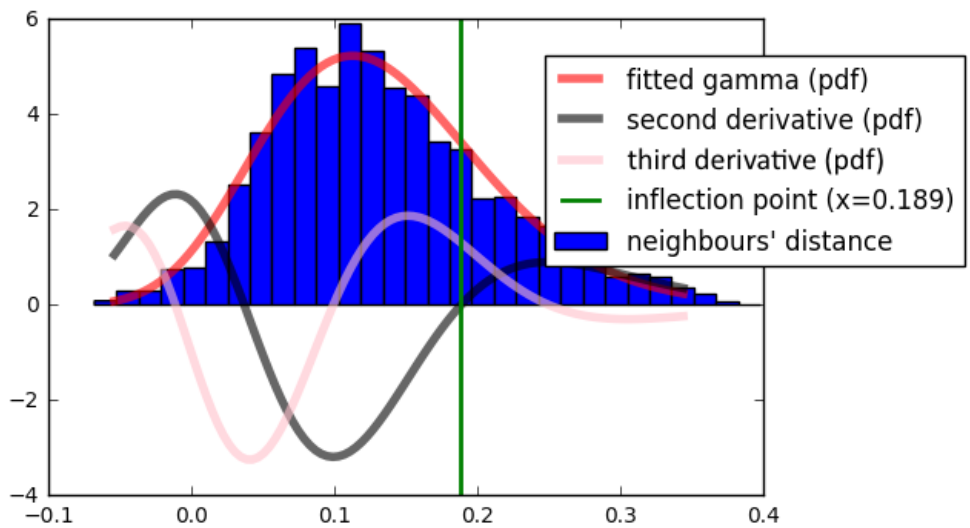

Figure 3.4: Histogram of the values of the cosine similarity between a sample document-vector and its neighbours in the reference dataset. The curves shown are the PDF of the fitted Gamma distribution; the second order derivative of the PDF; the third order derivative of the PDF; and the inflection point of the PDF. The vertical axis is not to scale.

Now that we have the expression of the derivatives, we can choose from a number of candidates cutoff abcissa that will follow the shape of the Gamma distribution. The performance of those candidates will be tested in section 5.2.4. The candidates that were studied are:

1. The inflection point(s) of the power density function. If $1 < k \leq 2$, the function is concave downward and then upward, with inflection point

at $x = k - 1 + \sqrt{k-1}$. If $k > 2$, the function is concave upward, then downward, then upward again, with inflection points at $x = k-1 \pm \sqrt{k-1}$ [24]. We empirically found that when there are two inflection points, the larger one always performs better as the cutoff parameter. We use the inflection point at $x = k - 1 + \sqrt{k-1}$ in the algorithm implementation.

$$candidate_1 = k - 1 + \sqrt{k-1} \tag{3.30}$$

2. The local maxima of the second order derivative. More specifically, the abcissa of the rightmost local maximum of the second order derivative. You can spot this point on figure 3.4. This point is close to the fixed cutoff point of the *Fixed cutoff method* that was determined by meta-parameter optimisation on the reference dataset. To compute it, we need to find the roots of the third derivative and select the rightmost abcissa :

$$roots(k, \theta, b) = \{x \mid \frac{f(x - b \mid k, \theta)}{dx^3} = 0\} \tag{3.31}$$

$$candidate_2 = max(roots(k, \theta, b)) \tag{3.32}$$

3.6 The Louvain algorithm for Clustering

The Louvain algorithm is a popular algorithm for community detection [17] and is presented in section 2.3.3. In the field of community detection, a community or cluster in a graph is intuitively defined as a set of densely connected nodes that is sparsely connected to other clusters in the graph. In the case of the graph G, nodes that belong to the same cluster should have higher cosine similarity than nodes that belong to other clusters. The Louvain algorithm is a great fit for the problem at hand because it offers great execution speed, high quality clustering and does not need a fixed number of clusters parameter like other clustering algorithms like the K-Means algorithm (section 2.3.2).

The Louvain algorithm performs poorly on complete graphs, as explained in section 3.5.2. To enable good performance, the graph G is pruned of edges with low cosine similarity weights. Those edges often connect nodes that should belong to different clusters and slow down the algorithm by introducing bad connections. Considering the greedy nature of the algorithm, bad connections may decrease the quality of the community detection.

3.6.1 Application to the graph representation

The application of the Louvain algorithm to the G' graph representation is very simple. The implementation used for this work is provided by Thomas Aynaud in the `community` Python module [25]. This implementation relies on the `NetworkX` Python module for graph manipulation [26]. The `NetworkX` graph

library is used throughout the proposed algorithm implementation for the representation of the graphs G and G', and for the pruning operations described in section 3.5.2. Example Python code for the creation of the graph representation is shown in listing 3.8. Once the graph representations using `NetworkX` are produced, the Louvain algorithm can be applied as shown in listing 3.9.

```
import networkx as nx
import numpy as np

def make_trio_list(docindex, neighbours_indexes, A):
    l = []
    for i in neighbours_indexes:
        l.append((docindex, i, (1+A[docindex][i])/2))
    return l

def create_graph(A, N):
    G_prime = nx.Graph()
    for i in range(A.shape[0]):
        G_prime.add_node(i)
        top_n_neighbours = np.argpartition(A[i], -N)[-N:-1]
        G_prime.add_weighted_edges_from(make_trio_list(i, top_n_neighbours, A))
    return G_prime
```

Listing 3.8: Example code for the creation of the pruned graph G' given the adjacency matrix A and the fixed neighbours method of section 3.5.3 with parameter N.

```
import community
import networkx as nx

def find_communities(G_prime):
    partition = community.best_partition(G_prime)
    n_communities = len(set(partition.values()))
    return (partition, n_communities)
```

Listing 3.9: Example code for application of the Louvain algorithm on the pruned graph G'

The `create_graph` function builds the graph G' in one go by selecting the appropriate edges from the adjacency matrix A. Nodes are identified by their index in the matrix A. For each node, the top N neighbours are selected by taking the indexes of the top N values of the i-th line of the adjacency matrix. This `A[i]` vector represents the values of the cosine similarity metric between

the i-th vector in the transformed dataset and all other vectors. Once the `top_n_neighbours` list is created, all the relevant edges are added to the graph G' in one step by the `NetworkX` function `add_weighted_edges_from`. This function expects a list of 3-tuples that hold, in order, the index of the first node and the index of the second node that need to be connected together and the weight value. The weight value between node i and j is defined as

$$w(i,j) = \frac{A_{ij} + 1}{2} \tag{3.33}$$

to respect the equation 3.14 of section 3.5.1. This way, edge weights are always positive and in the range $[0, 1]$.

The `find_communities` function takes as input this pruned graph and produces a partition of nodes thanks to the Louvain algorithm. The weights between nodes need to be positive [17], which is why the cosine similarity metric is not used directly.

Chapter 4

Document Cluster Labelling task

4.1 Introduction

Once a set of documents is divided into clusters, one important task is the production of labels for each cluster to help end users understand what the cluster represents. Without labels, users must instead rely on less effectives methods. For example, one might pick some instances in a given cluster and analyse them quickly in order to infer what the cluster is about. If this is repeated for each cluster, the user might lose interest and think this is something the machine ought to do for him. Users will appreciate having a simple label for each partition if it frees them from having to dig through some examples in order to undersand on a superficial level why the data was divided into those partitions. A labelling system that produces one word per cluster is the simplest to implement.

When working with clusters of documents, labels that can summarise a given cluster of documents and convey a high-level picture of its content are necessary. When designing the proposed algorithm, we settled on a system that can produce a one-word-label for each cluster. A short label like this is quick and easy to read, but sill has the potential to give a lot of information. There is no requirement that the one-word-label needs to appear in the cluster's documents. This freedom is important because the best label it not garanteed to be part of the cluster's content. For example, consider a set of documents about forecasts for the dividends publicly-traded companies will pay in the next quarter. A good label such as "finance" might not appear in any of the documents.

The proposed algorithm relies on the use of word-vectors and document-vectors produced by a single Doc2Vec model. Given the similarities in the Word2Vec and Doc2Vec algorithms, there might be relationships between the word and document representations. In section 1.2.2 of the first chapter, a formal definition of this task is outlined.

4.2 Vector relationships with Word2Vec and Doc2Vec

The Word2Vec and Doc2Vec algorithms that are used to produce the word and document representations share a lot of similarities. The Doc2Vec algorithm is a strict extension of the Word2Vec algorithm; the structure of the neural net and the training objective are the same. In both cases the training objective is the maximisation of the performance of the prediction of a target word given its sourrounding words, or the prediction of the surrounding words given the target word, for each window of w words in a training document [5, 6, 8, 27]. In the case of the Doc2Vec algorithm, the neural net is given another input vector in supplement of the "target word" or "context words": the *paragraph-vector* [8]. This additionnal information drives the hidden layer to build a vector representation for documents in order to fulfill the training objective. In the neural net's point of view, the paragraph-vector is like another word in the window. Of course the real purpose of the algorithm is not the training objective itself but rather the byproduct of the trained hidden layer, i.e. the embedding of documents.

The document embedding is trained alongside the word embedding, and is largely dependent on it in the case of a pre-trained word embedding [21]. This dependence means that there might be relationships between the word-vectors and document-vectors that belong to each embedding. In practice, the dimensionality of the word and document embeddings produced by the Doc2Vec model are identical, 300 to 1000 typically. This opens the way for the use of vector algebra that mixes the word-vectors and document-vectors.

4.2.1 Word-vectors relationships

First of all, inter word-vector relationships have already been studied quite extensively in the original papers and subsequently by other researchers [6, 20, 28]. Those emergent relationships can be quite surprising, such as the "country-capital" example shown in figure 4.1 or the "male-female" example shown in figure 4.2. This demonstrates that a Word2Vec model is able to encode complex relationships between words thanks to linear relationships in the embedding space. Those linear relationships enables us to use word-vector algebra as a way to query the model for relationships. The equation 4.1 is often used as an example to demonstrate this behaviour [29]. An interpretation of this equality is that the vector in equation 4.2 is the linear translation between the word-vector for "woman" and "man". This translation in the vector space encodes the transformation in meaning from a masculine to a feminine word. If we appy this transformation in the vector space to the word "king", we expect to get a vector that is very similar the vector representing the word "queen". This similarity between vectors is typically quantified by the cosine similarity measure. The use of this measure for vector-space comparisons is discussed in section 3.5.1.

$$\epsilon('woman') - \epsilon('man') + \epsilon('king') \approx \epsilon('queen') \tag{4.1}$$

$$\overrightarrow{to_the_feminine} = \epsilon('woman') - \epsilon('man') \tag{4.2}$$

$$\epsilon('king') + \overrightarrow{to_the_feminine} \approx \epsilon('queen') \tag{4.3}$$

The Word2Vec implementation used in this document [20] has a method designed for such inferences. This shows the popularity of vector-space algebra with the Word2Vec model.

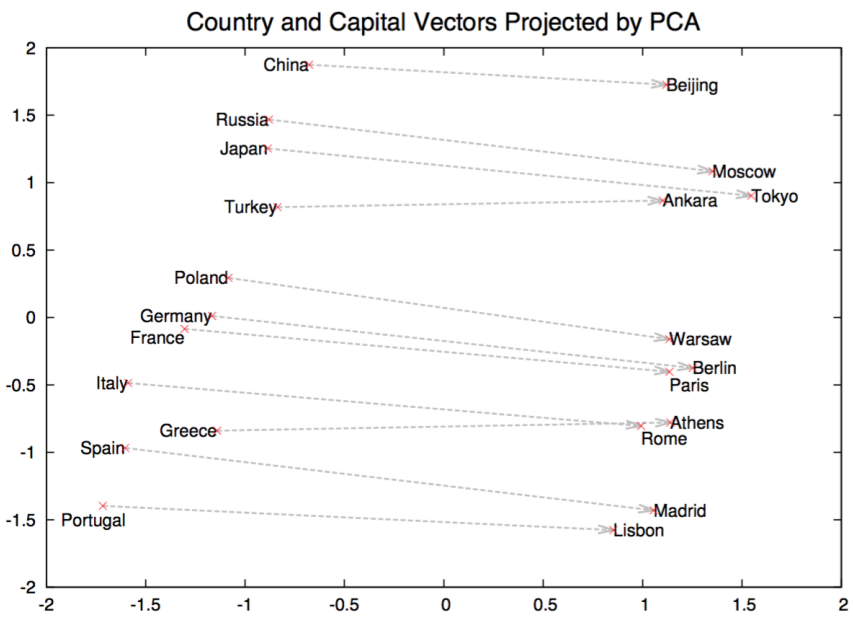

Figure 4.1: Original caption from the paper: "Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities. The figure illustrates ability of the model to automatically organize concepts and learn implicitly the relationships between them, as during the training we did not provide any supervised information about what a capital city means." [6]

4.2.2 Document-vectors and Word-vectors relationships

Firstly, meaningful word-vector relationships with Word2Vec are possible given that they have been demonstrated multiple times [6, 27, 29]. Secondly, the Doc2Vec model treats the paragraph-vector like another word in the window [8]. Also, the additionnal information provided by this vector improves the

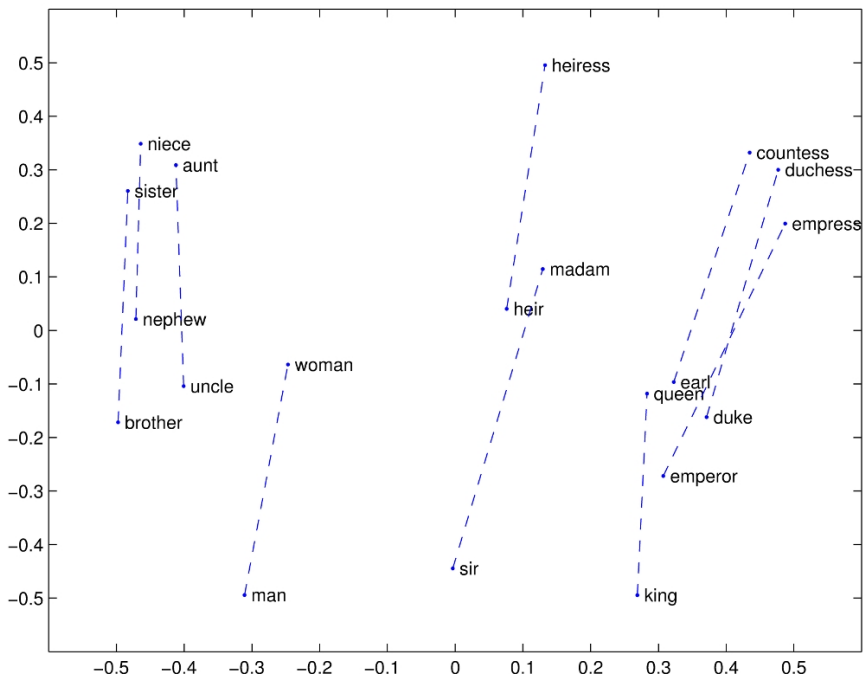

Figure 4.2: Similar to figure 4.1, "A two-dimensional projection of the word-vectors for a set of masculine and feminine grammatical gender forms of words. During the training we did not provide any supervised information about the gender of the words" [29]

algorithm performance [21] which means that this vector does provide relevant information to the model given the document's content. Training a new Doc2Vec model with an existing word embedding is possible and results in a model with a working document embedding [21]. In this case, the neural net inside the Doc2Vec model must have fitted a document embedding given the restrictions of the fixed word embedding. Since the untrained neural net treats the paragraph-vector like another word, two Doc2Vec models with the same inital random seeding for the paragraph-vectors but with two different word embeddings will produce different document embedding spaces. This means that, in general, the document embedding should be linked to the word embedding.

With this dependence in mind, we formulate the hypothesis that **there are relationships between document-vectors and word-vectors** even though they belong to different embedding spaces.

4.2.3 The "Least-angle regression" experiment

A first experiment was set out to get information about the validity of this hypothesis. It has already been shown in section 3.4.2 that, given a trained

Doc2Vec model, the inferred paragraph-vectors is a function of the word-vectors of the input document. The nature of this relationship can be explored with the following hypothesis: **an inferred document-vector is a linear combination of the word-vectors present in the original document**.

In order to test this statement, the experiment is set up as follows. Formally, we start with a document corpus D and a transform this corpus into the embedding corpus of document-vectors

$$embedding(D) = \{\epsilon(doc) \mid doc \in t(D)\} \tag{4.4}$$

We want to find a linear combination of word-vectors that has a minimal reconstruction error for a given target document-vector. Let $\hat{y}$ be the estimated target document-vector and $\mathbf{V}$ be the matrix of all word-vectors in the vocabulary of size s of the Doc2Vec model. This matrix has size (s, N) where N is the size of the word-vectors and doc-vectors. The expression of $\hat{y}$ is

$$\hat{y} = c_1\mathbf{V}_1 + c_2\mathbf{V}_2 + \cdots + c_s\mathbf{V}_s \tag{4.5}$$

where the estimator $\hat{y}$ is a linear combination of word-vectors and c is the coefficient vector. Note that there is no intercept in this equation.

To find the values of the coefficient vector, the "Least-angle regression" (LARS) [30] algorithm has been chosen. This regression algorithm is quite effective in dealing with highly dimensional data and has been suggested by B. Frenay as a solution for solving our linear combination problem.

The implementation offered by the Scikit-learn library has been used to solve the problem [31, 32]. This implementation offers two important Python methods: `fit` and `predict`. The first method will execute the training procedure of LARS with the provided instances matrix X of size `(n_samples, n_features)` and the target vector y of size `(n_samples,)`. Once the procedure is finished, the model provides a vector of coeficients `coef_` of size `(n_features,)`. The size of this vector corresponds to the size of the vocabulary detailed above. The intercept is equal to zero and ignored because the model has been set to not include any intercept in the formula. The second method, `predict`, takes as input the matrix X of shape `(n_samples, n_features)` and returns a vector of predicted values C of shape `(n_samples,)`. This vector is equivalent to the estimator $\hat{y}$ defined in equation 4.5 and should not be mistaken for the coefficient vector c.

In this case, we can see that $s =$ `n_features` and thus `n_samples` must be equal to N, the size of the word-vectors and document-vectors. Given that the vocabulary matrix V is of size $(s, N) =$ `(n_features, n_samples)`, it needs to be transposed to respect the shape required by the instance matrix X. All those operations are detailed in the Python code example in listing 4.1.

```
from gensim.models.doc2vec import Doc2Vec
from sklearn.linear_model import Lars

def lars_experiment(docvec, N_words):
    lars = Lars(n_nonzero_coefs=N_words, fit_intercept
        =False)
    X = model.syn0.transpose()
    y = docvec
    lars.fit(X, y)
    y_hat = lars.predict(X)
    return (y_hat, lars.coef_)
```

Listing 4.1: Example code for the LARS experiment

When this code is ran, the function `lars_experiment` return the estimated vector $\hat{y}$ for a given input document-vector `docvec` and the coefficient vector c. This last vector represents the participation of each word-vector of the vocabulary $\mathbf{V}$ in the given document-vector. Also, the LARS algorithm expects a fixed number of non-zero coeficients as meta-parameter. If the number is higher, the model will use more word-vectors to produce the estimator $\hat{y}$ but can also run into numerical instabilities.

During the experiments, if the number is set above 162 non-zero coefficients, numeric stability warnings will show up. In those cases, the model will not fit the data properly. The relationship between the meta-parameter and the performance of the prediction is shown graphically in figure 4.3. You can see that there is a nice progression in performance as the meta-parameter grows, but after a certain amount the performance drops abruptly. Two methods to measure performance are shown: the first is the cosine similarity between y and $\hat{y}$, the second is the `score` metric offered by the LARS implementation. The function is specified to "return the coefficient of determination R^2 of the prediction" [33]. Furthermore, "The coefficient R^2 is defined as $(1 - u/v)$, where u is the regression sum of squares $((y_true - y_pred)^2).sum()$ and v is the residual sum of squares $((y_true - y_true.mean())^2).sum()$" [33].

With this sample document, our metrics tend to agree on the optimal number of non-zero coefficients meta-parameter. It seems that the optimal value is between 144 and 148 as shown in the figure 4.3.

Even with this optimal value, the vectors y and $\hat{y}$ are not very similar. A score of 0.42445 out of a maximum of 1.0 is not very good. The cosine similarity is not great either: a similarity of 0.67243 means that the vectors do not really point in the same direction; it's very underwhelming given the high complexity of our model. The data does not seem to favor the hypothesis that the document-vector is a linear combination of word-vectors present in the vocabulary.

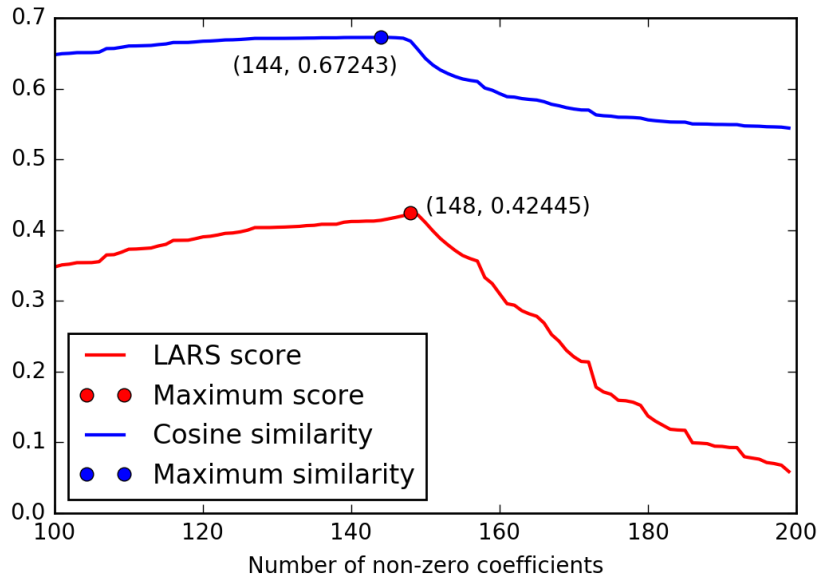

Figure 4.3: Relationship between the performance of the LARS algorithm for a range of non-zero coefficients values given a sample document-vector from the "business" category.

4.3 Linear models for document labelling

Since the data at hand does not support the hypothesis of section 4.2.3, other venues for automatic labelling need to explored. By drawing inspiration from the vector algebra presented in section 4.2.1, there might be similar relationships between word-vectors and document-vectors as described in section 4.2.2.

To explore this possibility, two models are proposed below. The reference dataset will be treated as a training dataset for these experiments. The documents content will be the training data, and the topical category name of a given document will be the target value. In other words, both models use supervised training unlike the models presented before in this document.

4.3.1 Translation model for document labelling

This model relies on the "linear relationship assumption". This assumption states that the vector representation of a document can be used to get the vector representation of the word corresponding to the desired label for the document by means of vector addition with a vector d. This assumption is not proven and will only be tested empirically in section 5.3.2. The equation

$$\epsilon(t(doc)) + d \approx \epsilon(label) \tag{4.6}$$

captures "the linear relationship assumption" where doc is a document, $\epsilon(t(doc))$ is the associated document-vector and $\epsilon(label)$ is the word-vector associated with the correct label for the document. This model can therefore assign a label to a document, or to a group of documents if the mean of their document-vectors is given to the model. If D_a is an annotated dataset of documents, i.e. a set of couples $(doc, label)$, we have the formula

$$\forall (doc, label) \in D_a \Rightarrow \epsilon(t(doc)) + d \approx \epsilon(label) \tag{4.7}$$

The reference dataset can be considered as dataset D_a of the formula 4.7 because every document belongs to a given topical category. This category will be the document's label. In that case, the formula can be rearranged

$$\forall (doc, label) \in D_a \Rightarrow d \approx \epsilon(label) - \epsilon(t(doc)) \tag{4.8}$$

and be used to provide an estimator for the vector d

$$\hat{d} = \frac{\sum_{(doc,label)\in D_a} \epsilon(label) - \epsilon(t(doc))}{|D_a|} \tag{4.9}$$

The formula 4.9 forms the basis for training the translation model for document labelling. Example Python code for training the vector $\bar{d}$ on the reference dataset is shown in listing 4.2. The variable `acc_f64` is the accumulator for the sum of the individual differences $\epsilon(label) - \epsilon(t(doc))$. The accumulator is divided by the number of documents $|D_a|$ at the end of the function and returned in the same `float32` format the Doc2Vec model uses.

```
import numpy as np
from gensim.models.doc2vec import Doc2Vec

def train_simple_model(doc2vec, labelled_dataset)
    n = 0
    acc_f64 = np.zeros(shape=(300,), dtype=np.float64)
    for doc, category in labelled_dataset:
        n += 1
        label_wordvec = doc2vec[category]
        docvec = doc2vec.infer_vector(parse(doc, lang
            ='en'))
        acc_f64 += (label_wordvec - docvec)

    acc_f64 /= n
    return acc_f64.astype(dtype=np.float32)
```

Listing 4.2: Example code for training the Translation model

4.3.2 Ridge linear model for document labelling

The Ridge regression model for document labelling is an extension of the translation model of section 4.3.1. This time, a more complex Ridge model is trained with the same reference dataset as before. The "linear relationship assumption" is extended such that the relationship between a given document-vector and the word-vector associated with its label is now

$$A\ \epsilon(t(doc))^T + b \approx \epsilon(label) \tag{4.10}$$

instead of the relationship of equation 4.6. The parameter A is a square matrix of size (N, N) where N is the dimensionnality of the word- and document-vectors. The parameter b is the intercept like the vector d of equation 4.6. The relationship of equation 4.10 is a strict extention of equation 4.6 because both are equivalent when A is the identity matrix. This model can therefore assign a label to a document, or to a group of documents if the mean of their document-vectors is given to the model, in the same manner as the translation model. This is fulfills the requirements outlined in section 1.2.2.

The Ridge regression solves the linear least squares problem with L_2 regularization [34]. The L_2 regularization is very important in this case because the Ridge model could easily overfit the data given that there are N^2 parameters in the A matrix. The α meta-parameter controls how much the model is penalised by the size of the parameters of A. There is no penalty for the size of the parameters of b. The α value needs to be set before training. To help with this, the Scikit-learn emplementation of the Ridge model [34] offers a cross-validation system to determine the best value for α. The `RidgeCV` implementation [35] is used for this purpose in the code. Example Python code for training the Ridge model on the reference dataset with cross-validation is shown in listing 4.3.

```
import numpy as np
from gensim.models.doc2vec import Doc2Vec
from sklearn.linear_model import RidgeCV

def train_ridge_model(doc2vec, labelled_dataset)
    i = 0
    X = np.zeros(shape=(len(labelled_dataset), 300),
        dtype=np.float32)
    Y = np.zeros(shape=(len(labelled_dataset), 300),
        dtype=np.float32)
    for doc, category in labelled_dataset:
        X[i] = doc2vec.infer_vector(parse(doc, lang='
            en'))
        Y[i] = doc2vec[category]
        i += 1
    ridge = RidgeCV(alphas=np.logspace(-2, 2, num=100,
        base=10), fit_intercept=True).fit(X, Y)
```

```
    return ridge
```

Listing 4.3: Example code for training the Ridge linear model with cross-validation for the α meta-parameter

The `alpha` meta-parameter is chosen in a range of 100 values evenly log-spaced between 10^{-2} and 10^{2}. In this configuration and with the reference dataset as input, the best alpha meta-parameter is 11.768119. Given that the generalisation performance of the algorithm is very important, a high value for α is desirable since it will lead to high regularization. With higher regularization, the contribution of the b parameter is higher in the sense that the matrix A will be closer to the identity matrix, which has the minimal penality for the model. In that case, the equation 4.10 would be equivalent to the corresponding equation for the translation model of section 4.3.1. This observation confirms that L_2 regularization does reduce the overfitting risk in theory.

Chapter 5

Evaluation

5.1 Introduction

In this chapter, the performance of the proposed algorithm for document clustering and the performance of the proposed algorithm for document cluster labelling will be evaluated. The performance of the first proposed algorithm will be tested in section 5.2.4 and compared to other algorithms for document clustering. The performance of the second proposed algorithm will tested in section 5.3.2 and 5.3.3. The quality of the labels produced will be tested qualitatively. Let's keep in mind that while the document clustering and cluster labelling tasks are separated in two different algorithms in our case, other algorithms produce similar results that fulfill the requirement of both section 1.2.1 and section 1.2.2 in one go.

5.2 Document clustering task evaluation

5.2.1 Datasets and protocol

To provide an even playing field, the different algorithms presented in the section 5.2 will have their performance evaluated against the same testing dataset. This testing dataset will be the "reference dataset" that is outlined in section 3.3.2. For algorithms that require pre-training, such as the proposed algorithm in regards to the Doc2Vec model, such pre-training will be done with the independent training dataset outlined in section 3.3.1.

5.2.2 LDA

Algorithm description

The Latent Dirichlet Allocation (LDA) algorithm is a well known algorithm for topic detection in the NLP field [13]. This algorithm treats each document from the dataset as a bag-of-words. Typically, a TF-IDF representation of the

dataset is computed and passed to the LDA algorithm. This matrix representation of the dataset encodes documents as vectors of word frequencies (term frequency, TF) scaled by the inverse frequency of this word in the whole corpus (inverse document frequency, IDF). This scaling decreases the value of the TF-IDF frequency for words that appear often in the corpus, and enables less frequent but more informative words to stand out. The TF-IDF algorithm is described in more detail in section 2.2.3.

The LDA algorithm assigns a vector to each document that describes the document's membership to each of the T topics. Those vectors are stored in the "document-term" matrix. Each topic is in turn described by a vector that quantifies the amount to which each word in the vocabulary contributes to the topic. Those vectors are stored in the "topic-term" matrix. The topics are said to be "latent" because while we can observe the documents and the words that compose them, the topics are invisible. A mathematical model describes how the words in the documents that belong to a given topic are distribued from the words that contribute to the topic. The membership of words to topics and documents to topics are quantified by the aforementioned vectors. A more detailed description of the algorithm is available in section 2.3.1.

Protocol

The evaluation protocol is the following. Firstly, each document from the reference dataset is preprocessed and transformed into a list of tokens (words). Words that appear in only one document or words that appear in more than 100 documents are removed from the tokens. This threshold method is recommended for building a dynamic stop-word list that is tailored to the dataset at hand [36], instead of using predefined lists. The tokenized documents are then transformed into the TF-IDF matrix X where each line represents a document's word frequencies. To preform this step, we use the `TfidfVectorizer` class from the Scikit-learn library [31, 37]. The code is shown in listing 5.1.

```
from sklearn.feature_extraction.text import
    TfidfVectorizer

def document_generator(labelled_dataset):
    for doc, category in labelled_dataset:
        yield doc

def transform_dataset(labelled_dataset):
    tfidf_vect = TfidfVectorizer(max_df=100, min_df=2)
    gen = document_generator(labelled_dataset)
    X = tfidf_vect.fit_transform(gen)
    return X
```

Listing 5.1: Transformation of the reference dataset with TF-IDF

Secondly, a LDA model is fitted to the data, which produces the "topic-term" matrix. The model is then used to produce the "document-term" matrix with the same data **X**. Those two steps are performed in one go by a single function for efficiency. The LDA implementation used is again provided by the Scikit-learn library [31, 38]. The LDA algorithm expects a fixed "number of topics" meta-parameter. This is a real drawback for real-world applications as explained in section 1.2.1. The reference dataset is known to have 5 natural categories. The algorithm will thus be tested with 5 topics. This puts the LDA algorithm at an advantage compared to other algorithms that do not benefit from this information, and does not represent real-world performance. The code for this step is shown in listing 5.2.

```
from sklearn.decomposition import
    LatentDirichletAllocation as LDA

def lda_topic_detection(X, n_topics=5):
    lda = LDA(n_topics=n_topics, learning_method='
        batch')
    doc_term_matrix = lda.fit_transform(X)
    topic_term_matrix = lda.components_
    return (doc_term_matrix, topic_term_matrix)
```

Listing 5.2: Execution of the LDA algorithm on the reference dataset. The learning method is set to "batch" in order to provide compatibility with future versions of Scikit-learn.

Results with 5 topics

When the code is run, the function of listing 5.2 returns the document-term and the topic-term matrices. The first matrix can be used to determine to which of the 5 topics each document has the most membership. This relationship means that every document "most belongs to" a single topic. This fulfills the requirements of section 1.2.1. The number of documents that most belong to each topic is shown in figure 5.1. In this figure, the original document categories are also shown with different colours. The first topics (index 0) has 299 "politics" articles, 94 "sport" articles, 82 "business" articles, 38 "tech" articles and only 3 "entertainment" articles.

Given that 57.9% of articles that most belong to the first topic comes from the "politics" category, we could say that the first topic represents — with questionable accuracy — the latent "politics" category. We can explore the topic-term matrix for more information. In this matrix, each line encodes the contribution of each word in the vocabulary to a given topic. The 10 most "contributive" words for the first topic are shown in listing 5.3. Those words are "lib", "MPs", "Lord", "Kennedy", etc. Given that the articles come from a U.K.

source, most of those 10 words do make sense in a "politics" context.

```
get_top_words(topic_terms_matrix[0], 10)
> [(10.297443872359713, 'lib'),
>  (9.9781147672493997, 'mps'),
>  (9.8618977809101498, 'lord'),
>  (8.408209398239757, 'kennedy'),
>  (8.1916831125865297, 'iraq'),
>  (7.349273391691419, 'immigration'),
>  (7.3335315333048969, 'olympic'),
>  (6.9796884869626039, 'blunkett'),
>  (6.9721545398480496, 'lords'),
>  (6.9494796839568078, 'taxes')]
```

Listing 5.3: Extraction of the top-10 words that contribute the most to the first topic (index 0). The words are lowercase because of the pre-processing applied by the `TfidfVectorizer`. The number shown to the left of each word is the Posterior Dirichlet parameter for the topic [13].

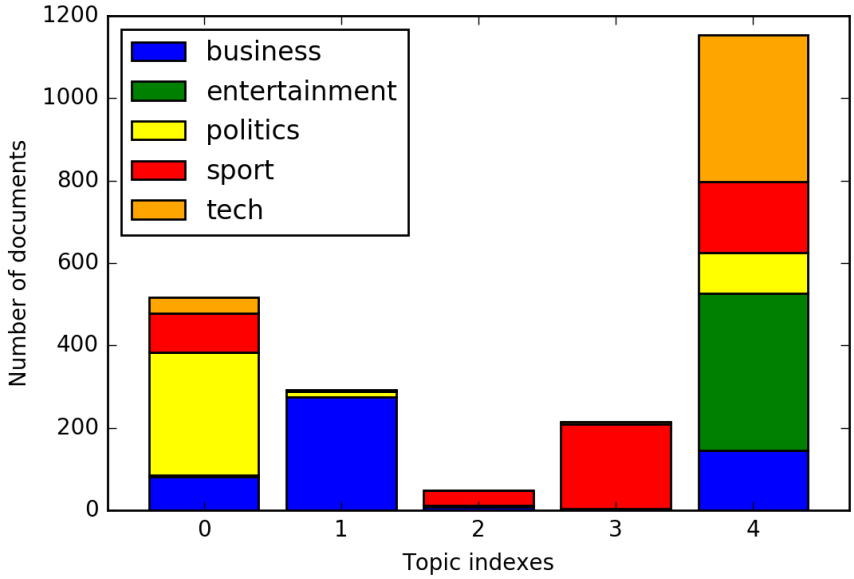

Figure 5.1: Number of documents that "most belong to" each topic, as determined by the document-term matrix produced by the LDA algorithm. The bars are stacked by colour, corresponding to the original document categories.

The second topic is composed of 274 “business” articles and 15 “politics” articles out of a total of 292 articles. The topic is composed of 93.8% “business” articles. The top-5 words for this topic are shown in listing 5.4. In those words we find “Yukos”, which is a Russian oil and gas company. The other words make great sense given the predominence of “business” articles.

```
get_top_words(topic_terms_matrix[1], 5)
> [(10.720504186904439, 'yukos'),
>  (9.5224975444604496, 'china'),
>  (7.7482739477288574, 'profits'),
>  (7.2729073152612322, 'stock'),
>  (7.1168972078219639, 'euros')]
```

Listing 5.4: Extraction of the top-5 words that contribute the most to the second topic (index 1).

Skipping over the topics 2 and 3 for sake of briefness, the topic at index 4 looks more divided. It seems to be composed of nearly all “tech” and “entertainment” articles, on top of a sizeable chunk of articles from the three other categories. In decreasing order, 381 “entertainment” articles, 357 “tech” articles, 172 “sport” articles, 145 “business” articles and 99 “politics” articles most belong to the last topic out of a total of 1154 documents. The last topic is thus composed of 33.0% “entertainment” articles and 30.9% “tech” articles.

The words that most contribute to this topic reflect this, as shown in listing 5.5. Some words looks like they come from a “technology” context (“Broadband”, “phones”, “Sony”) while other seem to be from a “music” context (“band”, “album”, “festival”). This is consistent with the distribution of categories shown in figure 5.1.

```
get_top_words(topic_terms_matrix[4], 10)
> [(12.249997406423272, 'broadband'),
>  (11.782225638452054, 'band'),
>  (11.045212088838147, 'eu'),
>  (10.384209261110666, 'album'),
>  (10.161976001370403, 'festival'),
>  (10.131851811709227, 'search'),
>  (10.124918196087643, 'phones'),
>  (9.2657010448874377, 'sony'),
>  (9.0739300977059756, 'apple'),
>  (8.753595128379791, 'oscar')]
```

Listing 5.5: Extraction of the top-10 words that contribute the most to the last topic (index 4).

Accuracy

In order to provide a metric that quantifies the performance of the LDA algorithm on this task and that can be compared with the other algorithms, the accuracy of the prediction will be used. To follow the research problem outlined in section 1.2.1, the LDA algorithm must cluster the documents with the same label together in one topic with the "most belong to" relationship. Each topic will have a category label assigned to it, given the most represented document label for the category. The "politics" label will be assigned to the topic at index 0; the "business" label will be assigned to the topic at index 1, the "sport" label will be assigned to the topics at indexes 2 and 3, and the "entertainment" label will be assigned to the last topic. Given this repartition, the accuracy of the model is **53.84 %**. The metric takes into account the fact that the "entertainment" and "tech" documents were not properly separated into different topics (clusters). But the fact that the "sport" category is split into two different topics is not penalized.

Results with other numbers of topics

The same experiment can be run with a different number of topics. By default, the LDA identifies 10 latent topics in the dataset [38]. This number is a good baseline number of topics that is often used for data exploration. The results of the LDA algorithm with 10 topics are shown in figure 5.2.

The same highly unbalanced distribution of documents in each topic can be seen again. The model tends to lumps a disproportionate amount of documents in the topic at index 4, regardless of category, while leaving other topics nearly empty. While some topics are able to specialize in a given category, like the topics at indexes 5 through 9, no topic seems to be able to retain most of the article mass from one category. The topic at index 4 holds 1193 documents out of the 2225 total from the reference dataset, namely 53.6% of all articles. This topic does not seem to specialize in a category as all 5 of them are significantly represented. The top-10 words that contribute to the topic at index 4 are shown in listing 5.6.

```
get_top_words(topic_terms_matrix[4], 10)
> [(13.199806726116796, 'eu'),
>  (12.220992777035168, 'lord'),
>  (10.198946486588833, 'lib'),
>  (9.9203686683295391, 'mps'),
>  (9.4069120330247813, 'iraq'),
>  (8.8305779686645227, 'book'),
>  (8.6431951051801938, 'debt'),
>  (8.6302803126040804, 'kennedy'),
>  (8.1768716517491153, 'taxes'),
>  (8.1694153078700662, 'trial')]
```

Listing 5.6: Extraction of the top-10 words that contribute the most to the topic at index 4.

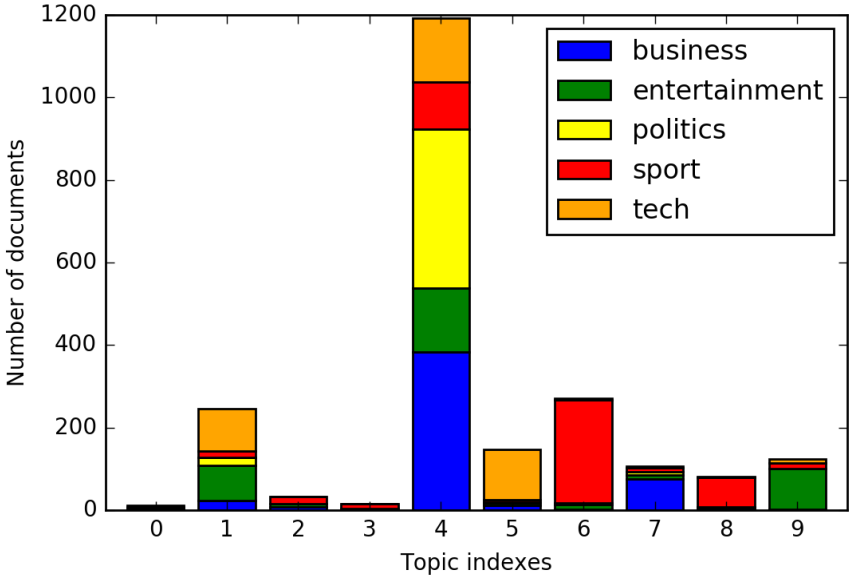

Figure 5.2: Number of documents that "most belong to" each of the 10 topics, as determined by the document-term matrix produced by the LDA algorithm. The bars are stacked by colour, corresponding to the original document categories.

The words in this topic seem to be from a political context. It is surprising given that only 32.3% of the articles that most belong to this topic are from the "politics" category. We can note that 92.4% of all "politics" articles most belong to the topic though.

The phenomenon of unbalanced distribution of documents between topics increases with the number of topics. The document distribution between 20 topics by the LDA algorithm is shown in figure 5.3. The topics continue to be less specialized in regards to the original categories, and the categories do not seem to concentrate in one or two topics. Only the "business" category concentrates in the topic at index 17, where 73.9% of all articles from this category most belong to. This topic is nonetheless non-specialized because the majority of documents that most belong to it are not from the "business" category.

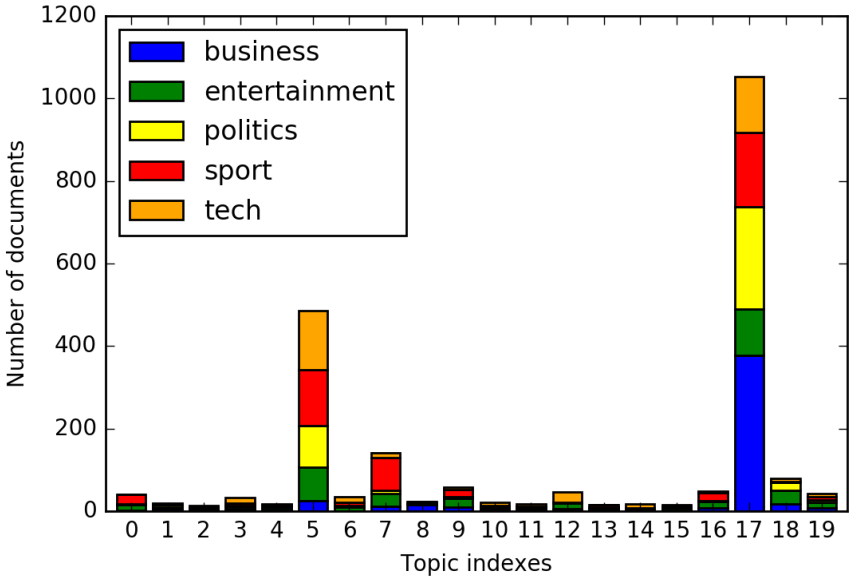

Figure 5.3: Number of documents that "most belong to" each of the 20 topics, as determined by the document-term matrix produced by the LDA algorithm. The bars are stacked by colour, corresponding to the original document categories.

5.2.3 K-Means

Algorithm description

A popular clustering algorithm for vector data is the K-Means algorithm. This algorithm partitions the data into clusters given the euclidean distance between vector instances. The first step is to produce K random prototypes that are similar to existing instances. For each existing instance, the algorithm determines which of the K prototypes is the closest given their euclidean distance. Each instance is said to belong to the Kth cluster if the Kth prototype is the closest prototype. In the second step, the prototypes' positions are updated to the mean vector vector value of the vector instances that belong to the cluster. Those two steps are repeated until convergence of the prototypes' positions. The K-Means algorithm then returns the cluster index in the range $[0, K-1]$ for each of the data instances. A more detailed explaination is available in section 2.3.2.

Protocol

Each document from the reference dataset is transformed into a feature vector by the Doc2Vec algorithm. This procedure is identical to the procedure

used by the proposed algorithm for document clustering, described in section 3.4.2. The Doc2Vec model used for inference of new document-vectors is the same throughout this document.

As explained in the section above, the K-Means algorithm uses the number of clusters K meta-parameter. In a real-world setting, the number of clusters is not known in advance. To illustrate this, the performance of the K-Means algorithm will be tested on a range of K values. Another variation that needs to be tested is whether the application of dimensionality redution techniques such as PCA (section 2.4.2) or t-SNE (section 2.4.3) on the testing data improves the K-Means clustering performance. The values 3, 5, 7, and 10 for K will be tested. Also, the application of PCA to 50 dimensions, and the application of PCA and t-SNE to 2 dimensions will be tested.

Results

The accuracy metric used in this section is the same metric used for the LDA evaluation, described in section 5.2.2.

First, different values for K will be tested. The model achieves 61.71% accuracy for $K = 3$. The figure 5.4 shows the repartition of categories in the different clusters. The "sport" category is well differentiated in the cluster at index 2, while the other categories are grouped by two. The "business" category is grouped with "politics" while "tech" is grouped with "entertainment"; this makes senses in regards to the content of each category.

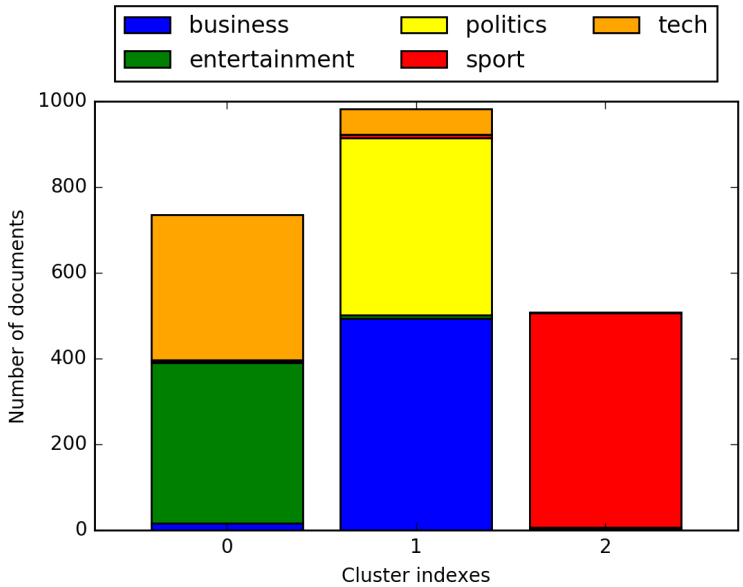

Figure 5.4: Number of documents in each of the 3 clusters produced by K-Means. No dimensionality reduction is applied.

With $K = 5$, the model reaches 94.74% accuracy. The figure 5.5 shows that each cluster captures one category very well, with only minimal overlapping. For example, the cluster at index 3 holds 482 "business" articles and 23 "politics" articles out of 520 total documents. Thus 92.69% of documents in this cluster belong to the correct "business" category. Overall, the clustering has excellent quality.

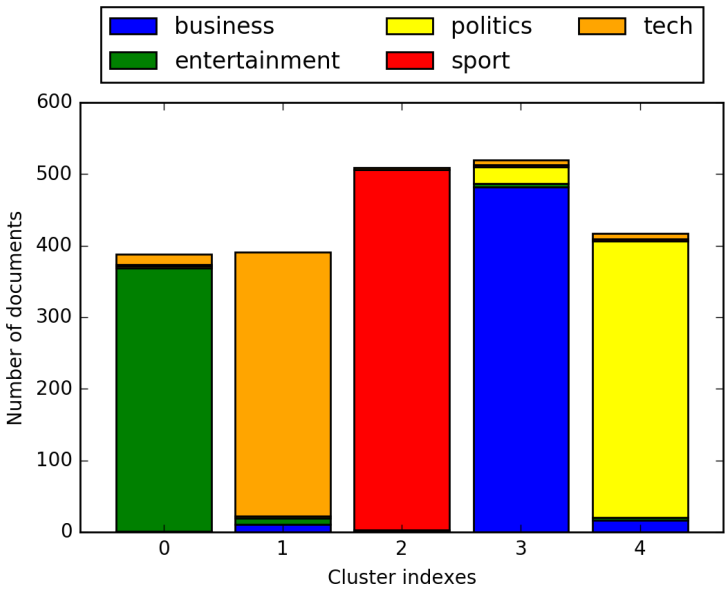

Figure 5.5: Number of documents in each of the 5 clusters produced by K-Means. No dimensionality reduction is applied.

Moving on to $K = 7$, a sharp decrease in quality is expected given that the number of clusters do not match with the number of categories. But the accuracy of the model is actually 93.98%, still very good compared to $K = 5$. The repartition in clusters is shown in figure 5.6. We can see that the "sport" category and the "business" category are split into different clusters. We have already seen that the algorithm segragates the "sport" category with $K = 3$. This reinforces the belief that the algorithm finds "sport" articles to be the most different compared to other categories. The fact that the algorithm splits "business" articles into two different clusters is interesting. As explained in section 5.2.2, the accuracy is not penalised if the model splits a category into multiple clusters, as long as those clusters are specialised in the category.

The last value to be tested is $K = 10$. The accuracy only drops to 92.49%, again thanks to the algorithm splitting categories into specialised clusters. The "sport" articles are split further into three well specialised clusters. The cluster at index 7 is the least specialised of the 10, holding 150 "politics" articles out of 236 documents in the cluster. Further investigation of the articles' content

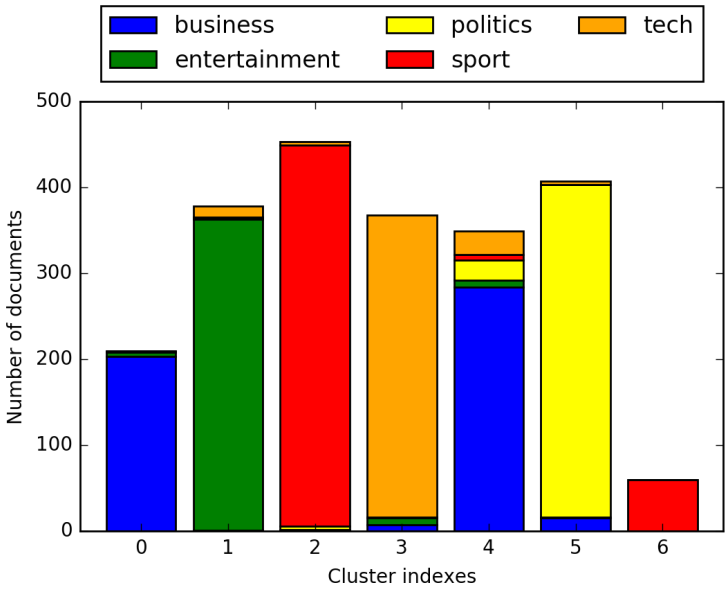

Figure 5.6: Number of documents in each of the 7 clusters produced by K-Means. No dimensionality reduction is applied.

could shed light as to why this cluster is much more varied than the others. One explaination could be that the K-Means prototype of this cluster ended up in a less specialized region of the vector space by simple chance, a fact compounded by the high number of other prototypes.

The performance of the K-Means clustering algorithm with the Doc2Vec-produced document-vectors is very good, even with less optimal values for K. The evolution of the accuracy of the K-Means model with increasing values for K is shown in figure 5.8. We can see that the model does not lose much accuracy as the number of clusters increases. At $K = 30$, the accuracy of the model is still 92.40%. At this point, the categories are split into more and more clusters, but they seem to stay well specialized.

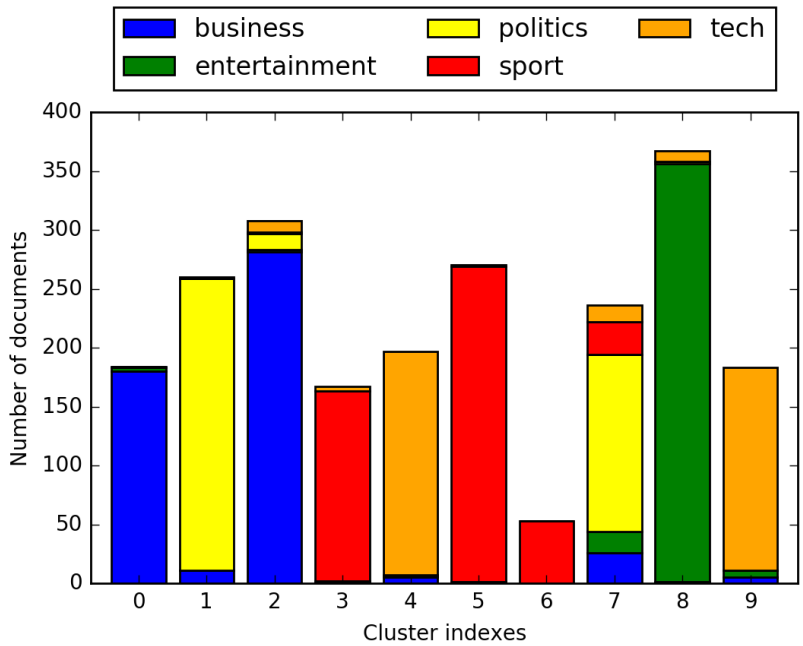

Figure 5.7: Number of documents in each of the 10 clusters produced by K-Means. No dimensionality reduction is applied.

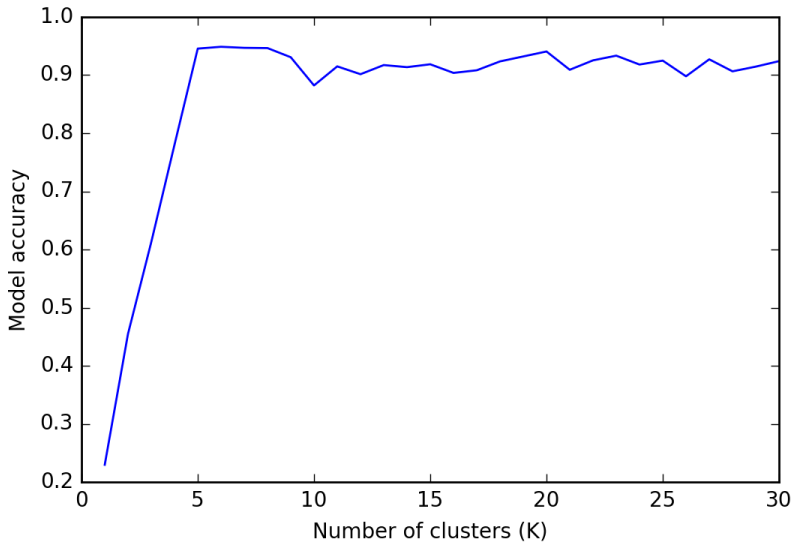

Figure 5.8: Evolution of the accuracy of K-Means models with an increasing number of clusters K. No dimensionality reduction is applied.

Does the performance of the K-Means model decreases when dimensionality reduction algorithms are applied to the input vectors ? The input vectors have a dimensionality of 300, since they are provided by the Doc2Vec model of section 3.4.1. The first dimensionality reduction algorithm that will be tested is PCA (section 2.4.2). It will reduce the size of the input vectors from 300 to 50 dimensions. The second algorithm is t-SNE (section 2.4.3). It will start from the reduced vectors produced by PCA and reduce the vector size further down to 2. The results of the K-Means clustering with $K = 5$ with those dimensionality reductions is shown in figure 5.9.

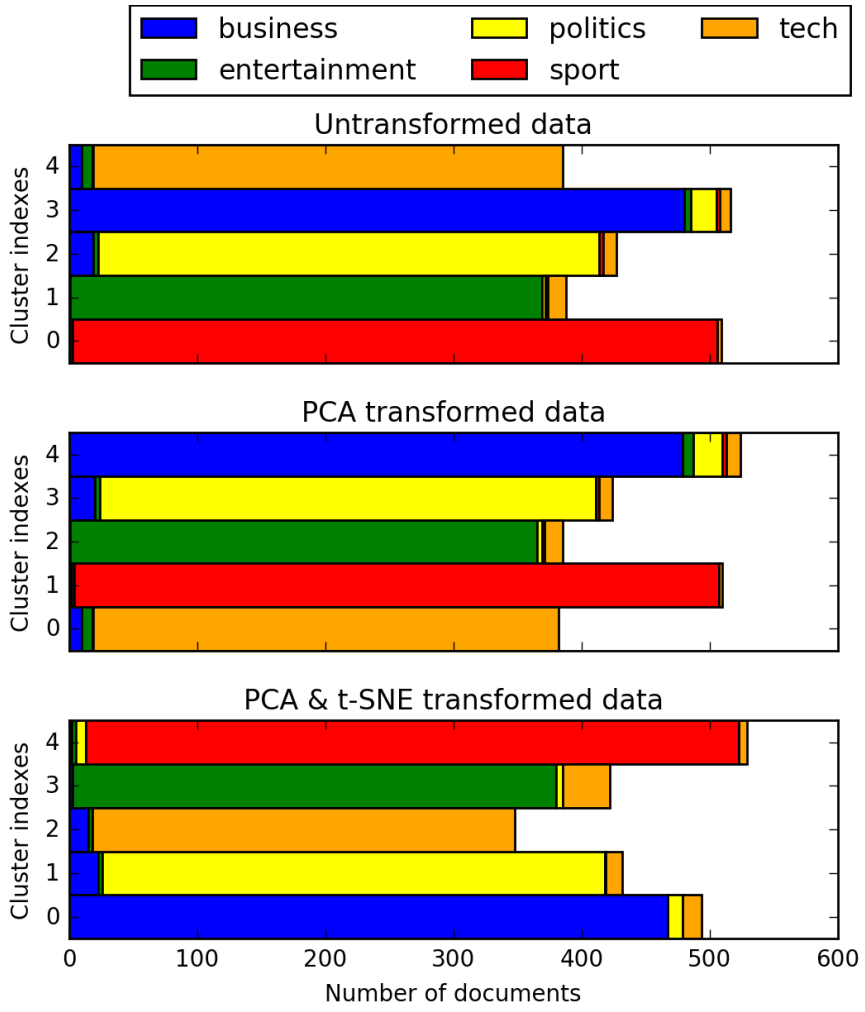

Figure 5.9: Number of documents in each of the 5 clusters produced by K-Means. Different levels of dimensionality reduction are applied.

We can see that the clusters are not significantly changed from one version to the other. The clusters look like they are shuffled between each chart because each model randomly generates different prototypes on initialisation. This shuffling can be safely ignored. The respective accuracies for the untransformed data, PCA, and t-SNE versions are 94.74%, 94.20% and 93.30%. This confirms that most of the information for cluster identification persists through dimensionality reduction and that K-Means is able to capture this information to provide almost equivalent clustering quality. The fact that the K-Means algorithm performs very well on 2D data also means that the clusters should be visible when projected on a screen. This projection is useful as it helps us visualise the labels directly on the projected data.

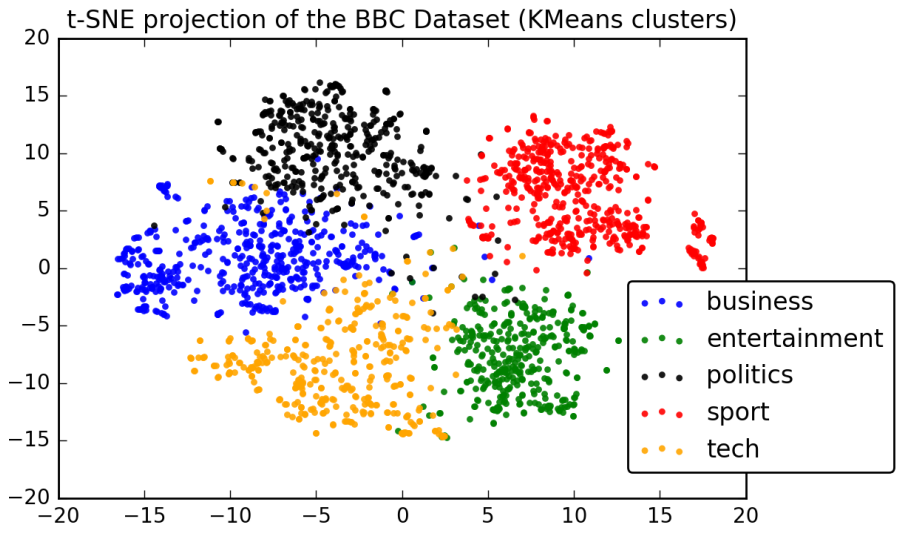

Figure 5.10: 2D projection of the reference dataset with t-SNE. Each dot is a document, coloured by the clusters produced by K-Means ($K = 5$) with no dimensionality reduction applied. The label for each cluster is the most represented category in the cluster.

Thanks to the t-SNE projection of the document-vectors from the reference dataset, the clusters can be displayed on a 2D screen. In figure 5.10, the document-vectors are projected on a 2D plane by the t-SNE algorithm and coloured by cluster. Those clusters are produced by the K-Means algorithm with $K = 5$ and with no dimensionality reduction applied to the input vectors. Each cluster is well visible, even without the colours. The "entertainment" or "sport" clusters look compact and dense compared to their surroundings. The "tech" cluster is very scattered in comparison, with instances even slipping between the "business" and "politics" clusters. Given that the K-Means algorithm considers that those documents belong to the "tech" cluster, this "slipping in between"

might simply be an artifact of the 2D projection. Nonetheless, the clusters look like they make sense to our eyes. The 94.74% accuracy performance of the algorithm on this data also contributes to the belief that the algorithm works very well in our case. This visualisation of the clustering performance will be used again in the following sections for other algorithms. Comparisons could be made visually on the projections to evaluate the relative performance of the algorithms.

5.2.4 Proposed algorithm for document clustering

The performance of the proposed algorithm for document clustering described in section 3.2 will now be evaluated. Compared to the K-Means evaluation of section 5.2.3, the proposed algorithm is quite similar. It also relies on the document-vectors inferred by the Doc2Vec model, but uses the Louvain algorithm (described in section 2.3.3) instead of K-Means to produce the clusters of documents.

Protocol

The protocol used to execute the proposed algorithm is described in detail in section 3.2. Each document from the reference dataset is transformed into a feature vector by the Doc2Vec algorithm. This procedure is described in section 3.4.2. The Doc2Vec model used for inference of new document-vectors is the same throughout this document. This step is identical for the K-Means evaluation above (section 5.2.3). This vector space representation is transformed into a graph representation, as described in section 3.5. The three solutions for neighbourhood selection of section 3.5.2 will be tested. Finally, the Louvain algorithm will be applied to the pruned graph and the accuracy metric will be computed.

Results - Fixed neighbours method

First, the *Fixed neighbours method* of section 3.5.3 is tested. This method only keeps the edges in the graph that are in the top-f in weight for each node. The results are displayed in figure 5.11. With low values of f, the Louvain algorithm creates a maximum number of clusters. With $f = 1$, there are 2225 clusters of 1 document each. The accuracy is trivially maximized in this case because each cluster of one document is composed of 100% of documents with the most represented label. The sheer number of clusters render this partition completely useless. With larger values of f, the accuracy drops then climbs back to around 95% with more manageable numbers of clusters. The number of clusters stagnates at 5 starting from $f = 46$. There is a curious drop of accuracy starting from $f = 101$ to $f = 104$. This phenomenon cannot be explained directly and is most likely an artefact of the Louvain algorithm with this specific dataset and meta-parameters. Overall, the performance of the algorithm is very good in the range from $f = 46$ to $f = 100$. In this range,

the average accuracy is 94.89% and peaks at 95.42% at $f = 50$. Outside this range and if we ignore $f = 1$, the best accuracy is 95.91% with $f = 20$ where the algorithm produces 7 clusters. The performance of the algorithm at $f = 20$ and $f = 50$ is very promising for the proposed algorithm.

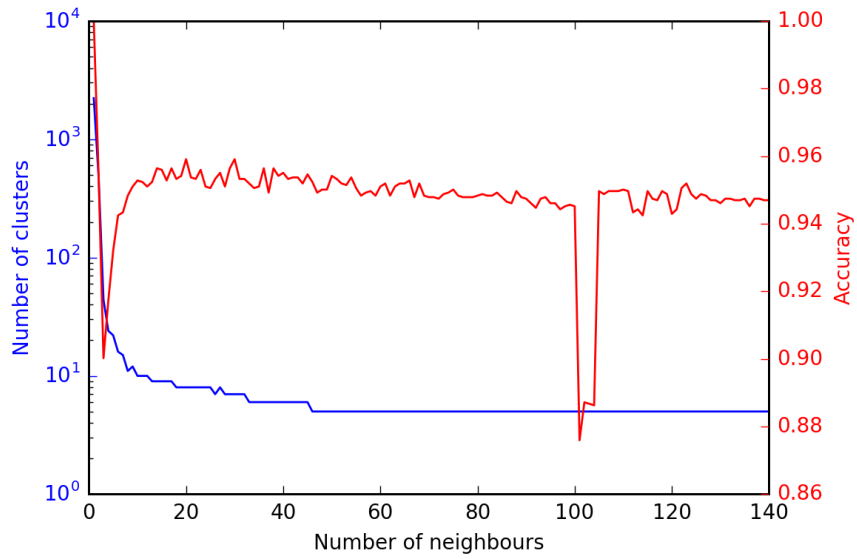

Figure 5.11: Evolution of the accuracy and the number of clusters produced by the Louvain algorithm with the "fixed neighbours method", for a range of numbers of neighbours (in steps of 1). Higher number of neighbours implies less pruning.

Results - Fixed cutoff method

Next, the *Fixed cutoff method* of section 3.5.4 is tested. This method is similar to the fixed neighbours method, as it uses a meta-parameter c. Any edge that has an associated cosine similarity measure stricly lower than c is pruned from the graph. The results are displayed in figure 5.12. With low values of c, the number of edges pruned is low; the graph is well-connected. The Louvain algorithm faces difficulties with large amount of edges, as we can see that the accuracy is quite low (around 60%) and the number of clusters is 3. With cutoff values from $c = 0.15$ through $c = 0.22$, the performance is excellent with an average accuracy of 94.22% and a number of clusters equal to 5. At $c = 0.25$, the accuracy of the model with 5 clusters reaches its best performance so far at **95.73%**. Above $c = 0.25$, the number of clusters rapidly climbs as the fixed cutoff method prunes more and more edges and leads the Louvain algorithm to identify more clusters because of the decrease in connectivity.

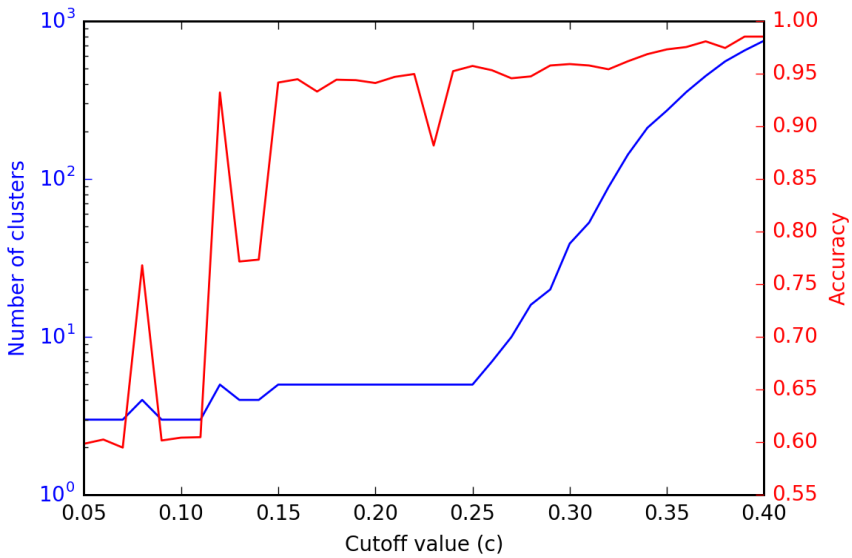

Figure 5.12: Evolution of the accuracy and the number of clusters produced by the Louvain algorithm with the "fixed cutoff method", for a range of cutoff values (in steps of 0.01). Higher values of c implies more pruning.

Results - Automatic cutoff method

Finally, the *Automatic cutoff method* of section 3.5.5 is tested. This method works like the *Fixed cutoff method* but chooses values for the c parameter with information from the distribution of cosine similarities in the graph. In section 3.5.5, multiple candidate values of c are proposed: $candidate_1$ and $candidate_2$. The expression of those candidate values is the following, given the (k, θ, b) parameters of the Gamma distribution

$$candidate_1 = k - 1 + \sqrt{k - 1} \tag{5.1}$$

$$roots(k, \theta, b) = \{x \mid \frac{f(x - b \mid k, \theta)}{dx^3} = 0\} \tag{5.2}$$

$$candidate_2 = max(roots(k, \theta, b)) \tag{5.3}$$

Those two candidate values will be tested individually. The first candidate is the higher inflection point of the Gamma distribution. For our reference dataset, this value is $candidate_1 = 0.166534823475$. With this value for c, the Louvain algorithm produces 5 clusters and has an accuracy of 94.52%. The results are

shown in figure 5.13. We can see that the first candidate value leads to a very good accuracy. Given that the range $c = 0.15$ to $c = 0.22$ has shown good results above in this document (figure 5.12), this is not surprising.

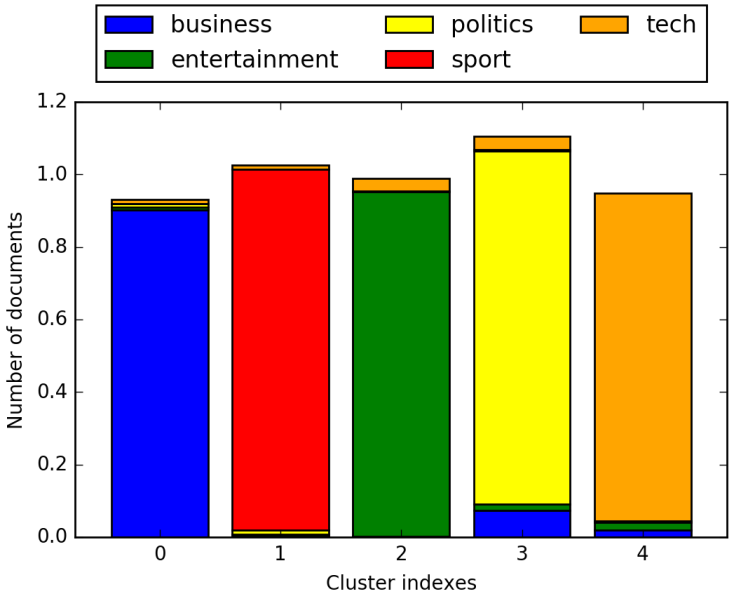

Figure 5.13: Number of documents in each of the 5 clusters produced by the Louvain algorithm with the Automatic cutoff method ($c = 0.16653$).

The second candidate is the highest root of the third derivative of the Gamma distribution PDF. In our case, $candidate_2 = 0.231571459531$. With this value of c, the Louvain algorithm still produces 5 clusters and has an accuracy of 88.04%. The results are shown in figure 5.14. We can see why the accuracy is significantly lower than for the other candidate. The "business" category seems to be split between the clusters at indexes 0 and 4. This unfortunate splitting can be reproduced with c values between $c = 0.225$ and $c = 0.233$, and corresponds with the "dip" in accuracy observed with those values in figure 5.12. If the second candidate was above 0.235, the Louvain algorithm would identify the "business" articles correctly and the performance would be much better (around 95% accuracy). Unfortunately, the cutoff value ended up right in a local minima for the accuracy. The figure 5.15 shows a 2D projection of the documents and the colours corresponding the clusters produced by the algorithm. We can visually see the mislabelling of a group of document in black, at the far left of the plot. Those documents belong to the "business" category, but are clustered with the other "politics" documents. For comparison purposes, the same 2D projection of the reference dataset with the original categories is shown in figure 5.16. This figure can be compared to the figures 5.10 and 5.15.

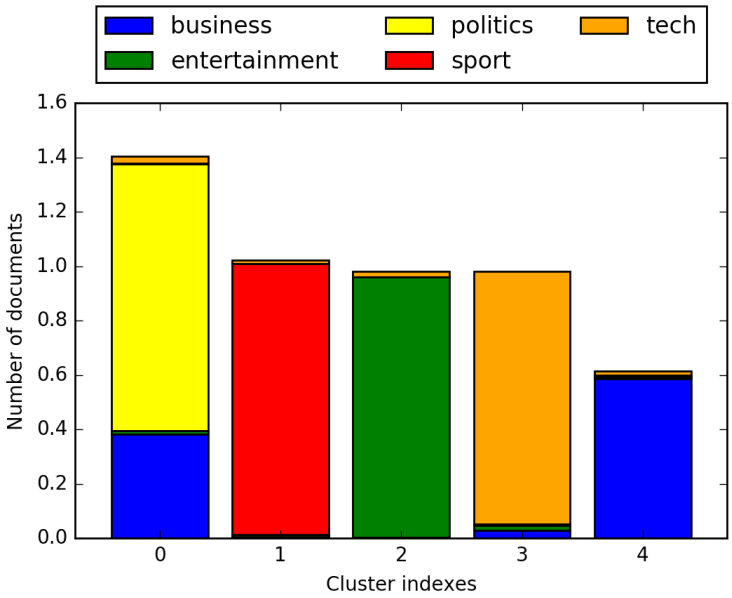

Figure 5.14: Number of documents in each of the 5 clusters produced by the Louvain algorithm with the Automatic cutoff method ($c = 0.23157$).

5.2.5 Discussion

The performance of the LDA algorithm from section 5.2.2 is quite subpar, but maybe not unexpectedly so given that it is not considered as a document clustering algorithm by the original authors [13]. The performance of the algorithm with the number of topics equal to 5 is only **53.84%**. It is better than guessing but not great. There might be ways to use LDA more effectively for the clustering task, but this research would fall out of the scope of this document.

The performance of the K-Means algorithm with Doc2Vec is excellent given the simplicity of the clustering algorithm. One could say that most of the work is done by the Doc2Vec algorithm and that K-Means simply reaps the rewards from the quality document embedding provided by the former. With $K = 5$, the model achieves **94.74%** accuracy. The accuracy does not decrease much as the number of clusters K increases (see figure 5.8), a sign that K-Means performs well even with non-optimal meta-parameter values.

The performance of the proposed algorithm is very similar to the performance of K-Means. The *Fixed neighbours method* performed well with meta-parameter values starting from $f = 46$ to $f = 100$. In this range, the model identified 5 clusters and the average accuracy is 94.89%, peaking at 95.42% for $f = 50$. Slightly better accuracy is possible with $f = 20$ at 95.91%, but the number of identified clusters climbs to 7. Next, the *Fixed cutoff method* performed well overall. It identifies 5 clusters and has an average accuracy of

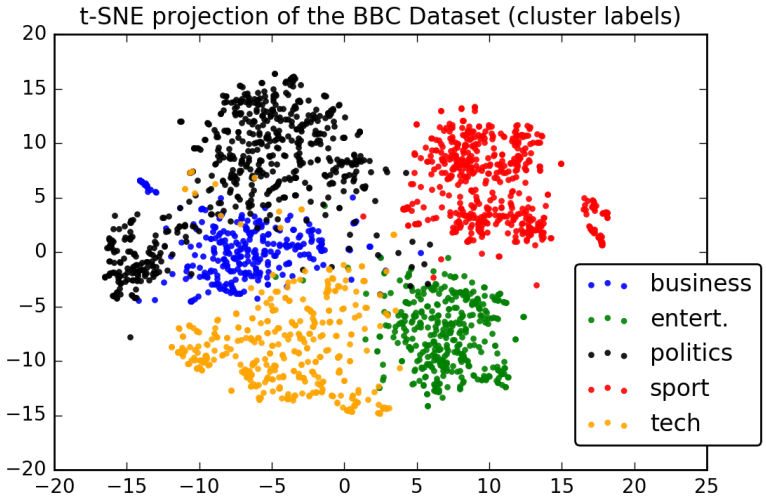

Figure 5.15: 2D projection of the reference dataset with t-SNE. Each dot is a document, coloured by the clusters produced by the Louvain algorithm with the Automatic cutoff ($c = 0.23157$). The label for each cluster is the most represented category in the cluster.

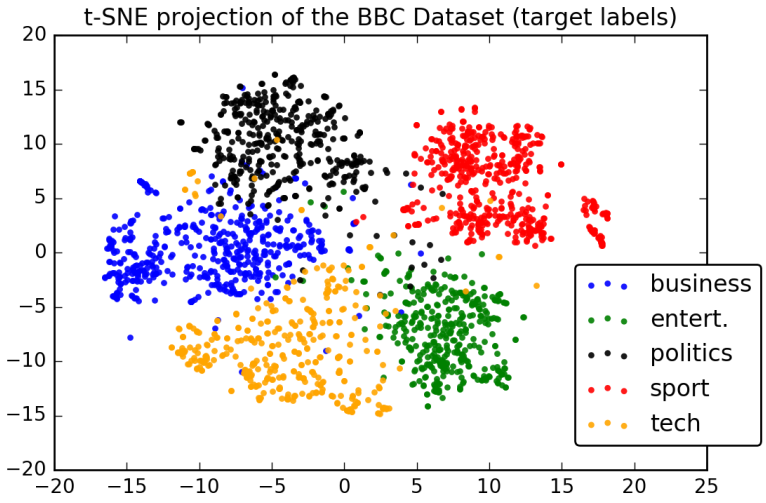

Figure 5.16: 2D projection of the reference dataset with t-SNE. Each dot is a document, coloured by category as they appear in the original dataset.

94.22% in the range from $c = 0.15$ to $c = 0.22$. The model peaks at **95.73%** accuracy with $c = 0.25$. Finally, the *Automatic cutoff method* performed very well too. The first candidate produced an accuracy of 94.52% with 5 clusters identified. The second candidate performed worse, with an accuracy of only 88.04%. The results for this configuration are shown in figure 5.14. Given our testing, the best meta-parameter based method for neighbourhood selection is the *Fixed cutoff method*. The *Fixed neighbours method* performed almost as well and seems more forgiving when run with less optimal values of f. For a method that tries to select a reasonable value for meta-parameters automatically, the first candidate described in section 3.5.5 performed best with 1.21 percentage point of lost accuracy compared to the *Fixed cutoff method*.

In conclusion, the data shows that Doc2Vec-based clustering algorithms perform significantly better than methods like LDA. This might not be a fair comparaison, as LDA is meant for topic detection and not document clustering. This algorithm is nonetheless often used for this purpose given its popularity in NLP systems [12]. When comparing K-Means and the Louvain algorithm in the proposed algorithm, both solutions perform within less than a percentage point of difference in accuracy. The proposed algorithm performs better, arguably at the cost of more meta-parameter tuning. The *Automatic cutoff method* with its first candidate value performed within 0.22 percentage points of accuracy compared to the K-Means algorithm. This very small difference gives us confidence that in situations where meta-parameter tuning is not desirable, the *Automatic cutoff method* is will perform almost as well as the K-Means solution.

5.3 Document cluster labelling task evaluation

5.3.1 Datasets and protocol

The performance of the proposed models for cluster labelling will be evaluated in this section. The two models are capable of labelling both clusters of documents and unique documents. This covers the requirements outlined in section 1.2.2. As described in section 4.3.1 and 4.3.2, the labels produced are meant to help users understand what the topic of a document or a group of documents is. The performance of the models is subjective as it is difficult to provide a metric that can tell objectively if a label is better than another based of the document(s) content.

To evaluate the quality of the labels produced, the protocol will be the following. A set of 12 articles is collected from the BBC UK website, disjoint from the reference dataset. Those articles are similar to the ones from the reference dataset, and are representative of articles available online currently. The articles have been selected at random from the BBC Website main page. All articles were published in the last quarter of 2016. Each document from this "test dataset" will be analysed by the two proposed models and given a one word label. The quality of the labels will be discussed in each section, and the models will be compared together in section 5.3.4. The full test dataset is available in the appendix. For simplicity, only the title and the URL of each article will be shown in this document, next to the model results.

5.3.2 Translation model

Protocol

The first model is the Translation model of section 4.3.1. Example Python code that can be used to produce labels from an article stored on disk is shown in listing 5.7.

```
from pattern.text import parse

def get_labels(doc2vec, filepath, enc='utf-8'):
    with open(filepath, 'r', encoding=enc) as file:
         body = file.read()
    tokens = parse(body, lang='en')
    docvec = doc2vec.infer_vector(tokens)
    label_wordvec = docvec + delta_vec
    return doc2vec.most_similar([label_wordvec])
```

Listing 5.7: Example code for the "Translation model" procedure. The `delta_vec` variable is the d vector of section 4.3.1. The function returns a list of labels and cosine similarities, sorted by similarity in decreasing order.

For each article, the code produces a list of tuples containing one label and its associated cosine similiarity to the `label_wordvec` vector. The list is sorted to show the best labels first, in decreasing order of cosine similarity. The label that would be produced by the algorithm is the first label of the list. But for this evaluation, all the possible labels are shown to help the reader see what is the output of the model.

Results

For each article, the model results are presented in a listing and are followed by a commentary of the performance. The first article is shown below in listing 5.8.

```
get_labels(doc2vec, 'test-dataset/politics1.txt')

> [('politics', 0.41274237632751465), ('sahel', 0.3933
  6004853248596), ('yemen', 0.39059895277023315), ('
  iraq', 0.3899107575416565), ('singapore', 0.3894087
  076187134), ('seoul', 0.3790009915828705), ('
  thailand', 0.3759443759918213), ('nepal', 0.3747716
  546058655), ('japan', 0.3716893196105957), ('
  entrepreneurship', 0.3702220320701599)]
```

Listing 5.8: Test article n°1, a "politics" article titled *Boris Johnson's Saudi 'proxy wars' comment 'not UK's view'* (http://www.bbc.com/news/uk-politics-38248316)

The model correctly gives the "politics" label to this article. The other labels quote some countries from the Middle East, but not Saudi Arabia which was the concerned party in the article. The model seems to have captured the fact that Eastern countries are concerned but failed to identify the correct one.

```
get_labels(doc2vec, 'test-dataset/politics2.txt')

> [('politics', 0.40667617321014404), ('democracy', 0.
  3964253067970276), ('sport', 0.3685542941093445),
  ('business', 0.358774334192276), ('life', 0.3525587
  9163742065), ('brand', 0.3483536243438720 7), ('
  cinema', 0.34806352853775024), ('florida', 0.347510
  30802726746), ('philosophy', 0.3429262042045593),
  ('practice', 0.3427788019180298)]
```

Listing 5.9: Test article n°2, a "politics" article titled *Marine Le Pen: No free school for illegal migrants in France* (http://www.bbc.com/news/world-europe-38249570)

The model also gives the correct "politics" label to this article. The other labels such as "democracy" are also interesting given that the article talks about Marine Le Pen's policies in France relative to migrant rights. The other labels are not good. Nonetheless, the first two labels are very good.

```
get_labels(doc2vec, 'test-dataset/health1.txt')

> [('viruses', 0.3468813896179199), ('norovirus', 0.34
  443336725234985), ('genus', 0.32989341020584106),
  ('salmon', 0.32080113887786865), ('scenery', 0.3198
  687732219696), ('insects', 0.3194999694824219), ('
  fish', 0.31441938877105713), ('outdoors', 0.3125343
  322753906), ('tuberculosis', 0.3124887943267822),
  ('epidemics', 0.30826932191848755)]
```

Listing 5.10: Test article n°3, a "health" article titled *Zika outbreak: The mosquito menace* (http://www.bbc.com/news/health-35427491)

The model has not been trained to produce the label "health", so it's not surprising that it does not output it in this case. The first labels are "viruses" and "norovirus". This is very good given that the article talks about the new threat of the Zika virus, passed on by mosquitoes. The "viruses" label is of high quality and adds confidence to the belief that the model can produce good labels for new, unseen documents.

```
get_labels(doc2vec, 'test-dataset/health2.txt')

> [('reassignment', 0.33455559611320496), ('clothing',
   0.3326927423477173), ('physics', 0.320884197950363
  16), ('time-tested', 0.31476008892059326), ('
  genital', 0.3053966760635376), ('therapy', 0.303605
  19886016846), ('laundry', 0.29849502444267273), ('
  objectification', 0.2984828054904938), ('deceitful'
  , 0.296955943107605), ('cults', 0.2956291139125824)
  ]
```

Listing 5.11: Test article n°4, a "health" article titled *Australian court approves intersex child's surgery* (http://www.bbc.com/news/world-australia-38218115)

Again with a "health" article, the model does not produce the "health" label. Nonetheless, the first two labels are very intresting: "reassignment" and "clothing". The first label probably refers to the procedure name, *reassignment surgery*. It is quite interesting that the model would assign such an insightful label to the article, given that this word does not even appear in the document content. The next label is less interesting, as the word "clothes" appears only

once in the document. This word might relate to a symbol for gender in the word embedding, but this is very speculative. The words“genital” and “therapy” also appears in the label list, but only in the 5th and 6th place. They would have not made terrible labels, but their place on the list indicates that the model does not find them as good as “physics” or “time-tested”, which are bad labels.

```
get_labels(doc2vec, 'test-dataset/tech1.txt')

> [('entertainment', 0.4098912477493286), ('4k', 0.390
  6046450138092), ('business', 0.371237576007843), ('
  technology', 0.36098745465278625), ('lasers', 0.360
  98742485046387), ('video', 0.3594483733177185), ('
  camera', 0.35685962438583374), ('audio', 0.35622224
  21169281), ('cameras', 0.34889301657676697), ('
  television', 0.34297287464141846)]
```

Listing 5.12: Test article n°5, a “tech” article titled *BBC tests 4K Planet Earth II in HDR on iPlayer* (http://www.bbc.com/news/technology-38242187)

This time, the model is given an article from a known category. Unfortunately, it does not returns the label “tech” as expected. The first labels are “entertainment” and “4k”. Given that the article talks about the rollout of new broadcasting technologies for the Internet, it’s not unreasonable for the model to generate such labels. This technology is meant for entertainment use. The second label, “4k”, is very good considering that this word appears 10 times in the article and describes the new technology well. The correct label is in the 4th place, a not very good result. The subsequent labels all relate to the field of video production and diffusion, not a bad guess for the model.

```
get_labels(doc2vec, 'test-dataset/tech2.txt')

> [('entertainment', 0.40472647547721863), ('hardware'
  , 0.40079760551452637), ('architecture', 0.33150649
  070739746), ('skyscrapers', 0.33115434646606445),
  ('soundtracks', 0.32402488589286804), ('networking'
  , 0.323863685131073), ('entrepreneurship', 0.322033
  22649002075), ('communications', 0.3219808936119079
  6), ('bionics', 0.3167329430580139), ('furniture',
  0.31451117992401123)]
```

Listing 5.13: Test article n°6, a “tech” article titled *Fan-made version of ’classic’ World of Warcraft returns* (http://www.bbc.com/news/technology-38250351)

With another “tech” article, the model still does not produce the correct label. The first two labels are “entertainment” and “hardware”. The second label

is quite bad, as the article is about the lauch of new independent game servers for the online game "World Of Warcraft", not about new hardware. The first label is better, given that the article arguably does not focus on the technological side of the new developments but on the "entertainment" implications of the events. Online video games are meant for entertainment after all. The next labels are of low quality unfortunately.

```
get_labels(doc2vec, 'test-dataset/business1.txt')

> [('tech', 0.3924986720085144), ('banking', 0.3854607
  045650482), ('technology', 0.3802368044853210 4), ('
  entertainment', 0.3790784180164337), ('tourism', 0.
  372322142124176), ('innovation', 0.3596175014972687
  ), ('business', 0.35428622364997864), ('laos', 0.35
  351771116256714), ('infrastructure', 0.348271548748
  01636), ('robotics', 0.34543293714523315)]
```

Listing 5.14: Test article n°7, a "business" article titled *Japan's third-quarter growth rate revised down sharply* (http://www.bbc.com/news/world-asia-38246195)

This article talks about new growth projections for the Japanese economy. In this case the model produces very poor labels. The correct "business" label is the 7th down the list, even below 2 of the other labels from the reference dataset. The "tech" label produced by the model does not really relate to any of the article's content. Overall, the model shows very poor performance.

```
get_labels(doc2vec, 'test-dataset/business2.txt')

> [('cheating', 0.39005202054977417), ('accounting', 0
  .36106598377227783), ('daimler', 0.3586364984512329
  ), ('audi', 0.3546139895915985), ('racing', 0.35309
  54122543335), ('telecom', 0.3528476357460022), ('
  politics', 0.3476407527923584), ('japan', 0.3460657
  298564911), ('fraud', 0.34535008668899536), ('
  healthcare', 0.3422929644584656)]
```

Listing 5.15: Test article n°8, a "business" article titled *Volkswagen emissions: UK and six other nations face legal action* (http://www.bbc.com/news/business-38247779)

This time again, the model does not produce the expected "business" label. But surprisingly, it returns the words "cheating" and "accounting". The article talks about the legal implications of VW's diesel cheating scandal. The word "cheating" appears two times in the article, while "accounting" does not appear

at all. This first label is very interesting and of very good quality given the tone and the content of the article. Some of the other labels are relative to the automotive world, but are not good labels. Overall, "cheating" is a great label even though the model does not produce the expected word.

```
get_labels(doc2vec, 'test-dataset/business3.txt')

> [('entertainment', 0.38709139823913574), ('business'
  , 0.3768441677093506), ('democracy', 0.363245248794
  55566), ('mathematics', 0.34541869163513184), ('
  peacebuilding', 0.34427011013031006), ('wings', 0.3
  4351736307144165), ('politics', 0.3358584940433502)
  , ('ict', 0.33474376797676086), ('taiwan', 0.318081
  5577507019), ('foshan', 0.31782642006874084)]
```

Listing 5.16: Test article n°9, a "business" article titled *Michael Jordan wins trademark case in China's top court* (http://www.bbc.com/news/world-asia-38246196)

The third business article gives the following results: "entertainment" and "business". The second label is the one that was expected. It's difficult to understand why the model would produces the label "entertainment" for this article, apart from the fact that "Michael Jordan" might appear more often is those kinds of articles. This explaination is not very convicing however, given the limited amount of data that the model was trained on. The other labels are all of low quality. Overall, the model misses the mark on the first label but is correct on the second try. Given that this article would be labelled with "entertainment" by the model in the real world, this counts as poor performance.

```
get_labels(doc2vec, 'test-dataset/science-environment1
  .txt')

> [('sustainability', 0.3873894214630127), ('
  innovation', 0.3541393280029297), ('conservation',
  0.3470483124256134), ('politics', 0.346402406692504
  9), ('badgering', 0.34528154134750366), ('business'
  , 0.341096460819244 4), ('genetics', 0.3386029005050
  659), ('technology', 0.3372189998626709), ('physics
  ', 0.3362698554992676), ('chemistry', 0.33058571815
  49072)]
```

Listing 5.17: Test article n°10, a "science-environment" article titled *Trump nominee to rekindle climate battle?* (http://www.bbc.com/news/science-environment-38249208)

The model is presented with an article with a new label, “science-environment”. It is not expected that the model will be able to produce such a label. The first labels are “sustainability”, “innovation”, “conservation” and “politics”. All those labels are good labels, considering that the article relates to incoming reforms to the US environmental regulations with the Trump administration. The best label is arguably “politics”, but the other labels referring to environmental policies in general are good too. The other labels are not great, as usual. The performance of the model on this article is good overall.

```
get_labels(doc2vec, 'test-dataset/magasine1.txt')

> [('format', 0.37098658084869385), ('furniture', 0.35
  794728994369507), ('cupboard', 0.3453362286090851),
   ('currency', 0.34013062715530396), ('fragrance', 0
  .33735328912734985), ('artwork', 0.3357024788856506
  3), ('refrigerator', 0.33356431126594543), ('
  basements', 0.3318791389465332), ('premises', 0.331
  20444416999817), ('sport', 0.3257705271244049)]
```

Listing 5.18: Test article n°11, a “magasine” article titled *The mystery of the stolen Klimt* (http://www.bbc.com/news/magazine-38242917)

This article comes from the “magasine” category of the BBC. Again, the model is not expected to be able to produce this label. This time, the model seems to have chosen labels of very poor quality, not at all representing what is inside the article. Only the 6th label, “artwork”, is descriptive of the document’s content. The article tells the tale of a stolen painting that is expected to be found soon. The model performed very badly on this particular instance.

```
get_labels(doc2vec, 'test-dataset/magasine2.txt')

> [('corpses', 0.341932475566864), ('archipelago', 0.3
  405414819717407), ('oahe', 0.3151187300682068), ('
  rods', 0.31241151690483093), ('downtrodden', 0.3120
  9391355514526), ('flowers', 0.31076931953430176),
  ('after-effects', 0.3093026578426361), ('manhandled
  ', 0.30520564317703247), ('gold', 0.305075168609619
  14), ('shrew', 0.30240678787231445)]
```

Listing 5.19: Test article n°12, a “magasine” article titled *Skeleton Lake of Roopkund, India* (http://www.atlasobscura.com/places/the-skeleton-lake-of-roopkund-india)

The last article is not from the BBC but was featured as a link to similar “magasine” articles on the BBC website at the time of visit. The article talks

about an Indian lake littered by bones and supposedly cursed if you believe the local tales. The model returned the label "corpses". This first label is quite good, probably in reference to the bodies found in the lake. The word "corpses" does not appear in the article. The next labels are all of poor quality howerver. Overall, the performance of the algorithm is good on this article given that it is different from the usual BBC articles.

Judging the performance of the labelling task is quite subjective. Nonetheless, through this limited testing the model seems to perform well, sometimes producing new labels, and other times producing nonsense. It is clear that more work is necessary and that the model does not reach a high enough level of quality in a sustained manner.

5.3.3 Ridge linear model

Protocol

The Ridge model for document labelling will be tested in this section. The model is tested with the same articles used in section 5.3.2 to enable side-by-side comparisons. For each article, the model results are presented in a listing and are followed by a commentary of the performance. Example Python code for the labelling procedure with the Ridge linear model is shown in listing 5.20.

```
from pattern.text import parse
from gensim.models.doc2vec import Doc2Vec
from sklearn.linear_model import RidgeCV

def get_labels_ridge(doc2vec, filepath, enc='utf-8'):
    with open(filepath, 'r', encoding=enc) as file:
         body = file.read()
    tokens = parse(body, lang='en')
    docvec = doc2vec.infer_vector(tokens)
    label_wordvec = ridgecv.predict([docvec])[0]
    return doc2vec.most_similar([label_wordvec])
```

Listing 5.20: Example code for the "Ridge linear model" procedure. The `ridgecv` model is trained on the reference dataset beforehand (see section 4.3.2). The function returns a list of labels and cosine similarities, sorted by similarity in decreasing order.

Like in section 5.3.2, for each article the code produces a list of tuples containing one label and its associated cosine similiarity to the `label_wordvec` vector. The list is sorted to show the best labels first, in decreasing order of cosine similarity. The label that would be produced by the algorithm is the first label of the list. The model results are presented in a listing and are followed by a commentary of the performance. The first article is shown below in listing 5.21.

Results

```
get_labels_ridge(doc2vec, 'test-dataset/politics1.txt')

> [('politics', 0.8946842551231384), ('entertainment', 0.7601702213287354), ('democracy', 0.7013632655143738), ('innovation', 0.694045901298523), ('japan', 0.6481683254241943), ('entrepreneurship', 0.6335444450378418), ('india', 0.6292811632156372), ('photography', 0.6224591732025146), ('germany', 0.6201233863830566), ('robotics', 0.6138731241226196)]
```

Listing 5.21: Test article n°1, a "politics" article titled *Boris Johnson's Saudi 'proxy wars' comment 'not UK's view'* (http://www.bbc.com/news/uk-politics-38248316)

The model return the correct "politics" label for this article. The other labels are of very bad quality. For example, the labels "japan" or "photography" have nothing to do with the article's content. The algorithm performs well here thanks to the first label.

```
get_labels_ridge(doc2vec, 'test-dataset/politics2.txt')

> [('politics', 0.9311158061027527), ('democracy', 0.7258799076080322), ('innovation', 0.6955740451812744), ('japan', 0.6568806171417236), ('india', 0.6410096883773804), ('entertainment', 0.6364849805831909), ('life', 0.6328984498977661), ('history', 0.6265016794204712), ('germany', 0.6238709688186646), ('singapore', 0.6197148561477661)]
```

Listing 5.22: Test article n°2, a "politics" article titled *Marine Le Pen: No free school for illegal migrants in France* (http://www.bbc.com/news/world-europe-38249570)

The model also gives the correct "politics" label to this article. The second label, "democracy", is good. The translation model produced the same first two labels, so both algorithms perform the same on this article. The cosine similarity is much higher in this case though, because of the greater power of the Ridge model thanks to the A matrix transformation.

```
get_labels_ridge(doc2vec, 'test-dataset/health1.txt')
```

```

> [('tech', 0.7379768490791321), ('entertainment', 0.6
  804618835449219), ('technology', 0.676891028881073)
  , ('politics', 0.6763032078742981), ('business', 0.
  6716618537902832), ('innovation', 0.628604412078857
  4), ('democracy', 0.5982980728149414), ('tourism',
  0.5835195779800415), ('robotics', 0.574208855628967
  3), ('japan', 0.5643472075462341)]
```

Listing 5.23: Test article n°3, a "health" article titled *Zika outbreak: The mosquito menace* (http://www.bbc.com/news/health-35427491)

Like the translation model, the Ridge model has not been trained to produce the label "health", so it's not surprising that it does not output it either in this case. The model predicts the "tech" label instead. This is not the label we wanted, the algorithm does not perform well in this case. We can already see that the Ridge model is more rigid than the translation model, since the 5 first labels are all the categories from the reference dataset. The Ridge model does not produce new labels like "viruses". Overall, the performance of Ridge is quite underwhelming.

```
get_labels_ridge(doc2vec, 'test-dataset/health2.txt')

> [('politics', 0.885235607624054), ('entertainment',
  0.7540264129638672), ('democracy', 0.70180803537368
  77), ('innovation', 0.6975721716880798), ('japan',
  0.6548116207122803), ('india', 0.6348474025726318),
   ('entrepreneurship', 0.6306245923042297), ('
  robotics', 0.6261179447174072), ('photography', 0.6
  235710382461548), ('germany', 0.6230024099349976)]
```

Listing 5.24: Test article n°4, a "health" article titled *Australian court approves intersex child's surgery* (http://www.bbc.com/news/world-australia-38218115)

Like the translation model, the Ridge model does not produce the "health" label. The model returns "politics", which is not wrong given that the ruling in favor of reassignement surgeries relates to the new policy of Australia. It seems that the Ridge model focuses on the training labels and does not output new, unexpected labels.

```
get_labels_ridge(doc2vec, 'test-dataset/tech1.txt')

> [('tech', 0.9389554262161255), ('entertainment', 0.6
  18045449256897), ('pharmaceutical', 0.6127828359603
  882), ('technology', 0.6036483645439148), ('tourism
```

```
    ', 0.545702338218689), ('telecoms', 0.5364497303962
    708), ('retail', 0.5346923470497131), ('automotive'
    , 0.5172970294952393), ('healthcare', 0.51677918434
    14307), ('gaming', 0.5112901329994202)]
```

Listing 5.25: Test article n°5, a "tech" article titled *BBC tests 4K Planet Earth II in HDR on iPlayer* (http://www.bbc.com/news/technology-38242187)

The Ridge model is given an article from a known category and outputs the correct "tech" label. The "entertainment" label that was outputted by the translation model is second in the list here. The other labels such as "pharmaceutical" or "tourism" are very bad labels though. In this case the Ridge model works very well.

```
1 get_labels_ridge(doc2vec, 'test-dataset/tech2.txt')
2
3 > [('tech', 0.8421440124511719), ('entertainment', 0.6
    402087211608887), ('technology', 0.6199949979782104
    ), ('sport', 0.5928340554237366), ('pharmaceutical'
    , 0.5827624797821045), ('tourism', 0.56366789340972
    9), ('retail', 0.5413546562194824), ('automotive',
    0.5364145040512085), ('telecoms', 0.534510254859924
    3), ('gaming', 0.5305140018463135)]
```

Listing 5.26: Test article n°6, a "tech" article titled *Fan-made version of 'classic' World of Warcraft returns* (http://www.bbc.com/news/technology-38250351)

Once again the Ridge models outputs the correct label: "tech". A pattern is emerging, where the Ridge model performs very well on articles close to the reference dataset. The second label is good given that the article is about video game technology. Apart from "technology", the other labels are not good. The Ridge model performs well again here overall.

```
1 get_labels_ridge(doc2vec, 'test-dataset/business1.txt
    ')
2
3 > [('business', 0.9724071025848389), ('technology', 0.
    561241626739502), ('industry', 0.5309603214263916),
     ('brand', 0.5246981382369995), ('innovation', 0.51
    23313665390015), ('staff', 0.5103206634521484), ('
    economy', 0.49762609601020813), ('workplace', 0.491
    87999963760376), ('research', 0.48951956629753113),
     ('democracy', 0.4881765842437744)]
```

Listing 5.27: Test article n°7, a "business" article titled *Japan's third-*

quarter growth rate revised down sharply (http://www.bbc.com/news/world-asia-38246195)

The Ridge model returns the correct label again: "business". While the "industry" or "economy" labels are acceptable too, the other words are bad labels. The Ridge model performs well thanks to its correct first guess.

```
get_labels_ridge(doc2vec, 'test-dataset/business2.txt')

> [('business', 0.9042314887046814), ('politics', 0.597512602806091 3), ('sport', 0.5807574987411499), ('technology', 0.5665467381477356), ('democracy', 0.559263706207275 4), ('economy', 0.5492373108863831), ('innovation', 0.5472148656845093), ('workplace', 0.5306861400604248), ('brand', 0.5286573767662048), ('industry', 0.5235980749130249)]
```

Listing 5.28: Test article n°8, a "business" article titled *Volkswagen emissions: UK and six other nations face legal action* (http://www.bbc.com/news/business-38247779)

Like clockwork the Ridge model predicts the correct label. But the interesting "cheating" label of the translation model is nowhere to be seen. The second label "politics" is acceptable given that the article is about the EU and its member states. The Ridge model performs well again.

```
get_labels_ridge(doc2vec, 'test-dataset/business3.txt')

> [('sport', 0.7625449895858765), ('business', 0.7330538034439087), ('politics', 0.666789174079895), ('technology', 0.6294835805892944), ('innovation', 0.6243115663528442), ('democracy', 0.6214156150817871), ('entertainment', 0.606941819190979), ('culture', 0.5708436965942383), ('architecture', 0.5680924654006958), ('singapore', 0.561517596244812)]
```

Listing 5.29: Test article n°9, a "business" article titled *Michael Jordan wins trademark case in China's top court* (http://www.bbc.com/news/world-asia-38246196)

The Ridge model gets it wrong by predicting the label "sport". This label is nonetheless quite good given that the article talks about the famous sports

star Michael Jordan. By comparison, the translation model predicted "entertainment" which is a bad label. In the end the Ridge model performed quite well on this article.

```
get_labels_ridge(doc2vec, 'test-dataset/science-environment1.txt')

> [('politics', 0.8223012685775757), ('business', 0.7395147085189819), ('democracy', 0.686608612537384), ('innovation', 0.6548107266426086), ('sport', 0.6495604515075684), ('history', 0.6060526371002197), ('technology', 0.6037402153015137), ('culture', 0.593707263469696), ('unity', 0.5928846597671509), ('japan', 0.5925574898719788)]
```

Listing 5.30: Test article n°10, a "science-environment" article titled *Trump nominee to rekindle climate battle?* (http://www.bbc.com/news/science-environment-38249208)

This article has a new label, "science-environment". Like the translation model, it is not expected that the algorithm will be able to produce the label. The Ridge model returns "politics", which is a great label given that the topic is about Trump's new EPA lead. The Ridge model performed arguably better than the translation model in this test.

```
get_labels_ridge(doc2vec, 'test-dataset/magasine1.txt')

> [('entertainment', 0.8571096062660217), ('technology', 0.6172723770141602), ('tech', 0.6127650737762451), ('tourism', 0.6099857091903687), ('innovation', 0.6017702221870422), ('sport', 0.5966168642044067), ('gaming', 0.5946000218391418), ('politics', 0.5887945890426636), ('transportation', 0.5821294188499451), ('education', 0.5635738372802734)]
```

Listing 5.31: Test article n°11, a "magasine" article titled *The mystery of the stolen Klimt* (http://www.bbc.com/news/magazine-38242917)

This article comes from the "magasine" category of the BBC. The Ridge model returns "entertainment", which is the best label if we only consider the 5 categories from the reference dataset. We cannot reasonably expect the model to produce "magasine", but it's disappointing to see that the model does not produce new labels for new articles like the translation model can. In the end

the Ridge model performed well but shows more rigidity.

```
get_labels_ridge(doc2vec, 'test-dataset/magasine2.txt')

> [('entertainment', 0.8470234870910645), ('politics', 0.7817751169204712), ('innovation', 0.6752939820289612), ('democracy', 0.6552845239639282), ('japan', 0.6292685270309448), ('robotics', 0.6144338846206665), ('photography', 0.6139617562294006), ('technology', 0.6138921976089478), ('transportation', 0.6137055158615112), ('singapore', 0.6120977401733398)]
```

Listing 5.32: Test article n°12, a "magasine" article titled *Skeleton Lake of Roopkund, India* (http://www.atlasobscura.com/places/the-skeleton-lake-of-roopkund-india)

On the final article, the Ridge model returned the same "entertainment" label. It's not a wrong label given that the content of the article take the form of a kind of story. Once again, this label is the best of the 5 training categories. The performance of the Ridge model is good overall, but shows that it has great difficulties going beyond the 5 aforementioned categories.

5.3.4 Discussion

When the results of the two algorithms for cluster labelling are compared, it is clear that the translation model produces more diverse labels than the Ridge model. On the other hand, the translation model sometimes performs poorly even on articles similar to the reference dataset. For example, the article in listing 5.14 relative to the Japanese economy was labelled "tech" by this model, even though it clearly belongs to the "business" category. Another example is the article in listing 5.16 that talks about Michael Jordan's trademark case. The translation model also performed poorly by producing the label "entertainment" while the document in clearly in the "business" category. The Ridge model performed objectively better on the whole dataset, but never came out with new labels like the first model. On the VW article for example (listing 5.15), the translation model predicted the very interesting label "cheating". The Ridge model returned the more rigid but correct "business" label instead.

In conclusion, the cluster labelling task results show great potential, but the proposed algorithms fall short of the expectations set in the sections 1.1 and 1.2.2. More work is necessary to get a cluster labelling algorithm based on Doc2Vec that meets those expectations. Ideally, an improved algorithm would have both the robustness of the Ridge model, and the informative and diverse labels of the translation model.

Chapter 6

Conclusion & perspectives

6.1 Document clustering task

The clustering task entails the partition of a set of documents written in a natural language into a number of clusters. Similar documents should be grouped together while dissimilar documents should end up in different clusters automatically. This task plays a role in many natural language processing (NLP) systems and has seen extensive research in the literature [13, 36, 12]. Recent breakthroughs based on artificial neural networks have created new opportunities for research on this task. Those new algorithms are Word2Vec [5, 6], Doc2Vec [8, 21] and other similar algorithms [27] based on those innovative artificial neural networks. These systems enable the production of high quality word and document embeddings that can improve the performance of many NLP algorithms. In this document, a new algorithm for document clustering based on Doc2Vec is presented.

6.1.1 Proposed algorithm for document clustering

The research was focused on a new algorithm for document clustering that uses the Doc2Vec algorithm for document embedding and the Louvain algorithm for community detection in graphs. The Doc2Vec model is responsible for the production of vector representations for documents, while the Louvain algorithm uses a graph representation based on the Doc2Vec embedding to identify "communities", i.e. clusters of similar documents. The Louvain algorithm is different from typical clustering algorithms such as K-Means [14, 15] because it does not rely on a preset *number of clusters* meta-parameter. It detects the clusters based on the features of the document graph alone. Yet the meta-parameters are merely moved outside the clustering algorithm itself, and into the parameters of the graph building algorithm.

Three methods are proposed in this document for the creation of a document graph from a set of document-vectors produced by the Doc2Vec algorithm. Those methods are called the *Fixed neighbours method*, the *Fixed cutoff method*,

and the *Automatic cutoff method*. While the first two methods are parametric, the last method removes the need for parameters at the cost of some accuracy for the resulting model. This tradeoff is valuable for systems similar to the one described in section 1.1.

6.1.2 Discussion

The performance of the proposed algorithm met most expectations. It reached 95% accuracy in our testing, largely beating the Latent Dirichlet Allocation algorithm [13] (section 5.2.2) and even slightly outperforming K-Means (section 5.2.3). The *Automatic cutoff method* showed excellent performance too, nearly matching the performance of K-Means (section 5.2.4). The fact that the third method reached this kind of performance without meta-parameter tuning is promising.

As explained in section 3.3.1 and 3.4.1, the Doc2Vec model used in the proposed algorithm is not trained on a lot of data (250k documents, 100M tokens) compared to other similar models such as GloVe (5B tokens) [27, 29]. The performance shown here can probably be improved with the use of a Doc2Vec model trained on more data and using greater vector sizes such as 500 or 1000 dimensions.

In conlusion, more testing still needs to be done to confirm that the results presented in this document are representative of the algorithm's general performance. Also, the current implementation is not very efficient in its use of computing ressources. The runtime performance and memory usage can probably be improved since most of the implementation is written in Python. Finally, the proposed algorithm should be tested with a Doc2Vec model trained on more data.

6.2 Cluster Labelling task

Once a dataset is partitioned into different clusters of instances, it can be helpful if each cluster is annotated with a descriptive label. This label is meant for human consumption, as a way to help him or her understand what the cluster represents. Some clustering algorithms such as K-Means produce pseudo-labels automatically when executed on a dataset. In the case of K-Means, a centroid is associated to each cluster. This centroid is an average of the cluster instances, a sort of prototype instance that represents the algorithm's idea of what a typical cluster instance looks like. This is a pseudo-label because it can't be used for human consumption directly. Other models suchs as LDA (section 2.3.1) associate a list of words to each detected topic, which can be used as a rudimentary label for the set of documents that "most belong to" the topic.

In the field of natural language processing, automatically labelling document clusters is notoriously difficult [39, 40]. Some research has been done on the topic of document cluster labelling with the Word2Vec and Doc2Vec algorithms [41], with good results. We wanted to explore this possibility in this research.

6.2.1 Proposed algorithm for document cluster labelling

The proposed algorithm for document cluster labelling is described in section 4.3. Two models are proposed, a simple translational model and a Ridge linear model. Both models try to model the relationship "label of" in the word and document vector space produced by Doc2Vec. The models were tested qualitatively on a set of 12 articles similar to the reference dataset. Sadly the results did not meet our expectations. The translation model performed well on some articles, producing new and very descriptive labels, but failed on others. The Ridge model performed better overall but only produced labels that were part of the reference dataset on which it was trained on. A model that can both produce new and insightful labels and also works reliably might be possible.

6.2.2 Reflection and criticism

More work needs to be done on the document cluster labelling algorithms. The idea of linear relationships between word and document vectors seems to work only to a limited extent in our research. The limitations of the two proposed models might be due to the fact that the Doc2Vec model used for the experiments is not trained on a lot of data. Document and word embeddings of higher quality are probably required for the proposed models to work reliably. Embeddings of higher dimensions and a Doc2Vec model trained on much more data might provide the vector quality needed for the cluster labelling task. Training a Doc2Vec model with pre-initialised word emdedding, in a similar manner than what is described in [21] might also provide better embeddings.

There are also great challenges in the testing of the labelling results since the quality of a label depends on human assessment. Qualitative studies can show the performance of a given model on a few examples but are difficult to use for the development and validation of new models and algorithms. Developing good quality metrics for document labelling and document cluster labelling might be as difficult as the development of the labelling algorithms themselves.

Bibliography

[1] Kevin Françoisse and François Beuvens. Sagaknow product image. https://sagacify.com/img/dev/sagaknow-products-color.svg.

[2] Kevin Françoisse and François Beuvens. Kosmio abstract graph. http://kosm.io/images/datapp.svg.

[3] Richard Socher. Cs224d deep learning for natural language processing — Lecture 2: Word vectors. https://cs224d.stanford.edu/lectures/CS224d-Lecture2.pdf.

[4] Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition, 1957.

[5] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[6] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[7] Gerard Salton and Michael J McGill. Introduction to modern information retrieval. 1986.

[8] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In Tony Jebara and Eric P. Xing, editors, *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196. JMLR Workshop and Conference Proceedings, 2014.

[9] Barbara H. Liskov and Jeannette M. Wing. A behavioral notion of subtyping. *ACM Trans. Program. Lang. Syst.*, 16(6):1811–1841, November 1994.

[10] Radim Řehůřek. Deep learning with paragraph2vec. https://radimrehurek.com/gensim/models/doc2vec.html.

[11] Scott Emmons, Stephen Kobourov, Mike Gallant, and Katy Börner. Analysis of network clustering algorithms and cluster quality metrics at scale. *PLOS ONE*, 11(7):1–18, 07 2016.

[12] Daniel Ramage, Paul Heymann, Christopher D. Manning, and Hector Garcia-Molina. Clustering the tagged web. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, WSDM '09, pages 54–63, New York, NY, USA, 2009. ACM.

[13] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[14] S. Lloyd. Least squares quantization in pcm. *IEEE Trans. Inf. Theor.*, 28(2):129–137, September 2006.

[15] J. Macqueen. Some methods for classification and analysis of multivariate observations. In *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[16] Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. The planar k-means problem is np-hard. In *Proceedings of the 3rd International Workshop on Algorithms and Computation*, WALCOM '09, pages 274–285, Berlin, Heidelberg, 2009. Springer-Verlag.

[17] Vincent D Blondel, Jean loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks, 2008.

[18] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2:559–572, 1901.

[19] L.J.P. van der Maaten and G.E. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[20] Radim Řehůřek. Deep learning with word2vec. `https://radimrehurek.com/gensim/models/word2vec.html`.

[21] Jey Han Lau and Timothy Baldwin. An empirical evaluation of doc2vec with practical insights into document embedding generation. *CoRR*, abs/1607.05368, 2016.

[22] Adriaan M. J. Schakel and Benjamin J. Wilson. Measuring word significance using distributed representations of words. *CoRR*, abs/1508.02297, 2015.

[23] Michal Campr and Karel Ježek. *Comparing Semantic Models for Evaluating Automatic Document Summarization*, pages 252–260. Springer International Publishing, Cham, 2015.

[24] Kyle Siegrist. The gamma distribution. `http://www.math.uah.edu/stat/special/Gamma.html`.

[25] Thomas Aynaud. Community detection for networkx's documentation. http://perso.crans.org/aynaud/communities/.

[26] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.

[27] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[28] Omer Levy and Yoav Goldberg. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning, CoNLL 2014, Baltimore, Maryland, USA, June 26-27, 2014*, pages 171–180, 2014.

[29] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. https://nlp.stanford.edu/projects/glove/.

[30] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.

[31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 1.1. generalized linear models. http://scikit-learn.org/stable/modules/linear_model.html#least-angle-regression.

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. LARS — scikit-learn 0.18.1 documentation. http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lars.html#sklearn.linear_model.Lars.score.

[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Ridge — scikit-learn 0.18.1 documentation. http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html.

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Ridgecv — scikit-learn 0.18.1 documentation. `http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.RidgeCV.html`.

[36] Hassan Saif, Miriam Fernández, Yulan He, and Harith Alani. On stopwords, filtering and data sparsity for sentiment analysis of twitter. In *LREC 2014, Ninth International Conference on Language Resources and Evaluation. Proceedings.*, pages 810–817, 2014.

[37] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Tfidfvectorizer — scikit-learn 0.18.1 documentation. `http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html`.

[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Latentdirichletallocation — scikit-learn 0.18.1 documentation. `http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.LatentDirichletAllocation.html`.

[39] Alexandrin Popescul and Lyle H. Ungar. Automatic labeling of document clusters, 2000.

[40] Jey Han Lau, Karl Grieser, David Newman, and Timothy Baldwin. Automatic labelling of topic models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1536–1545, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[41] Han Kyul Kim, Hyunjoong Kim, and Sungzoon Cho. Bag-of-concepts: Comprehending document representation through clustering words in distributed representation, 2015.

Appendix

Test dataset articles

This section presents the test dataset articles used in sections 5.3.2 and 5.3.3, in the same order in which they were presented in those sections.

Politics article n°1

(http://www.bbc.com/news/uk-politics-38248316)

Boris Johnson's Saudi 'proxy wars' comment 'not UK's view'

Downing Street has said Foreign Secretary Boris Johnson's comments on Saudi Arabia do not represent "the government's position".

Footage has emerged from an event last week at which Mr Johnson said UK ally Saudi Arabia was engaging in "proxy wars" in the Middle East. The PM's spokeswoman said these were the foreign secretary's personal views. She said a forthcoming visit to the region would give him a chance to set out the UK's position on Saudi Arabia. Mr Johnson's comments were made at a conference in Rome last week but only emerged after the The Guardian newspaper published footage of the event. In it the foreign secretary said: "There are politicians who are twisting and abusing religion and different strains of the same religion in order to further their own political objectives. "That's one of the biggest political problems in the whole region. And the tragedy for me - and that's why you have these proxy wars being fought the whole time in that area - is that there is not strong enough leadership in the countries themselves."

'Awkward comments'

Mr Johnson told the Med 2 conference: "There are not enough big characters, big people, men or women, who are willing to reach out beyond their Sunni or Shia or whatever group to the other side and bring people together and to develop a national story again. "That is what's lacking. And that's the tragedy," he said, adding that "visionary leadership" was needed in the region. He went on: "That's why you've got the Saudis, Iran, everybody, moving in and puppeteering and playing proxy wars." The BBC's diplomatic correspondent James Landale said the emergence of the comments would be "awkward

if not embarrassing for the foreign secretary". "Once again Mr Johnson's use of language is causing headlines that his diplomats will need to explain," our correspondent said.

Downing Street's comment came as Prime Minister Theresa May returned from a visit to the Gulf where she had dinner with the leaders of Saudi Arabia, Kuwait, the United Arab Emirates, Qatar, Bahrain and Oman. Her spokeswoman said that Mrs May wanted to strengthen the relationship with Saudi Arabia, saying, "we are supporting the Saudi-led coalition in support of the legitimate government in Yemen against Houthi rebels". She said: "Those are the prime minister's views - the foreign secretary's views are not the government's position on, for example, Saudi Arabia and its role in the region."

Robert Lacey, a historian and author of the Kingdom and the House of Saud, said that while he agreed with Mr Johnson's comments, he questioned whether he should be saying them about an ally. He told BBC Radio 4's Today programme that he believed it was a gaffe and that Mr Johnson was acting more like a journalist.

Politics article n°2

(http://www.bbc.com/news/world-europe-38249570)

Marine Le Pen: No free school for illegal migrants in France

French far-right leader Marine Le Pen has called for an end to free education for the children of illegal immigrants. In a speech in Paris, the National Front's candidate in the 2017 presidential race next spring said she had nothing against foreigners. "But I tell them: if you come to our country don't expect to be taken care of, to be looked after, that your children will be educated without charge," she said. "No more playtime," she added. Ms Le Pen is widely expected to be one of two candidates to reach the second and final round of the election in May. Who will be France's president in 2017? France's centre-right radical candidate Latest opinion polls suggest she is running neck-and-neck in polling for the first round with the centre-right candidate Francois Fillon, who was selected in national primaries at the end of November. The Socialists will hold their own primaries next month. However, around two-thirds of voters would choose Mr Fillon over Ms Le Pen in the deciding round.

Ms Le Pen had been widely reported as saying the education ban should apply to foreigners, but after her speech she clarified to the AFP that she was talking about illegal immigrants. However, she also said that foreigners using the state education system must be living in France legally and contributing to the tax system. The FN's new direction Cutting immigration has long been a bedrock policy of the FN, and Ms Le Pen has campaigned hard to cut legal immigration dramatically to 10,000 people a year, and to halt the right to family reunions. Education has also been a central theme for the far-right leader. In 2014, she spoke out against providing substitute meals for pork dinners in schools for Muslims and Jews. However under French law dating back to 1881,

free primary education is a basic right for all children. The right to education is also a fundamental tenet of the European Convention on Human Rights. French Education Secretary Najat Belkacem said in a blog entry that Ms Le Pen had not only demonstrated total indifference to the terrible plight of many migrant children, she showed the same level of ignorance of fundamental French values and laws. Since she took over the running of the FN from her father almost six years ago, Marine Le Pen has worked hard to detoxify the party's extremist image and attract mainstream voters. Her presidential campaign bears no trace of the party name and features a blue rose, borrowing imagery from both left and right.

But Ms Le Pen has had to respond to Mr Fillon's election as candidate by setting out distinctive policies. On Wednesday, she accused him of being in hock to France's business organisation Medef, the European Commission, the banks and German Finance Minister Wolfgang Schaeuble. In a live interview on Wednesday on one of France's main TV channels, TF1, she suggested she would put France back together in five years. "I want to sort out the schools, France's diplomacy and the economy" she said, adding that she wanted a referendum on France's membership of the European Union.

Health article n°1

(http://www.bbc.com/news/health-35427491)

Zika outbreak: The mosquito menace

This blood sucker might not be your best friend, but it loves you. The mosquito, Aedes aegypti, is causing widespread fear in Brazil where it is spreading the Zika virus that has been linked to thousands of babies being born with birth defects. So what do we know about it?

It loves our cities

This is not some jungle-dwelling insect that rarely comes into contact with people. It is one of those animals, like cockroaches, pigeons and urban foxes, that thrives in built-up areas. It does not need natural water sources to breed as it can lay eggs in the small and plentiful pools of stagnant water, such as gutters or flower pots, found in cities . Cities like Singapore have big problems with the mosquito. Prof Uriel Kitron, from Emory University, said: "Aegypti is really adapted well for urban areas. "It is becoming more and more of an urbanised world and aegypti thrives - Brazil is an extreme example going from 20There's also plenty of food - us.

They harbour a bunch of diseases

Zika is currently getting all the attention, but Aedes aegypti has long been known as the yellow fever mosquito. Yellow fever is a viral disease that can cause jaundice, bleeding and multiple organ failure in some cases. And 390 million people are infected with the dengue virus each year, largely as a result of bites from Aedes aegypti. Chikungunya, another virus causing alarm as it

expands around the world, is also spread by the insects. They do it by biting someone who is infected and then biting someone else.

Only the females bite and they love human blood

Both the male and female mosquitoes feed off plant nectar. However, the females need a blood meal in order to produce eggs. Two traits make them especially good at spreading disease. First they feed almost exclusively on humans, so if they pick up a disease from one person, they are likely to pass it on with their next bite. Other mosquitoes are less efficient spreaders as they feed off a variety of animals. These particular blood suckers also drink little and often. Rather than gorge their entire meal from one person and running the risk of being swatted they graze from multiple people making it easier to spread disease. They are also active during the day so bed nets are less effective.

It's a lazy hitch-hiker

"They're amazing hitch-hikers, but they're lousy flyers," argues Prof Kitron. The mosquitoes that carry malaria can fly for miles, but Aedes aegypti is pretty lazy by comparison. It stays close to the area where it was born and 100m is a long way for it to fly. Prof Kitron says: "But they're good at catching rides, their whole life cycle can take place in the water in a boat." So despite its short flying distances it has managed to conquer the world.

It has a cousin that could bring Zika to Europe

Aedes aegypti lives in tropical and subtropical climates - it is why experts are worried Zika could spread across the Americas, a huge swathe of Africa as well as India and China. However, the similar Aedes albopictus - or Asian Tiger mosquito - likes cooler more temperate areas.

It has been linked to outbreaks of chikungunya virus in Italy and France. There are suspicions that it could also spread Zika virus, but it is not yet clear.

Out of Africa

Aedes aegypti exists in two forms or sub-species. Aedes aegypti formosus is the original type found in Africa, which does not often come into contact with people and is not a major spreader of disease. Aedes aegypti aegypti is the form that has spread around the world. It probably arrived in the Americas from Africa after being transported around the ocean in the ships of European colonisers. "Yellow fever outbreaks were known in Africa from around 1400 and the first confirmed cases, from the Yucatan, were documented in 1648," Dr Michael Bonsall from Oxford University said. It is now found throughout tropical and subtropical regions around the globe.

Killed off once before

Eradication campaigns were successful in the Americas with 18 countries getting rid of the insect by 1962. But insecticide resistance plus a lack of political will led to the mosquito rebounding. But now far more people live in its preferred urban home and a similar achievement would require a huge effort. Follow James on Twitter.

Health article n°2

(http://www.bbc.com/news/world-australia-38218115)

Australian court approves intersex child's surgery

A five-year-old Australian child born genetically male will grow up as a sterilised female after a court agreed to her having surgery. The child, known only as Carla, identifies as a girl but has no female reproductive organs, Family Court documents show. The court approved a request by Carla's parents to surgically remove male gonads inside her body. People with a combination of sex characteristics are called intersex. 'Stereotypically female' behaviour When Carla turned five, her parents wanted to clarify if they needed court permission for the complex and irreversible surgery. The Family Court heard Carla was born with female-appearing genitalia and exhibited "stereotypically female" behaviour, which included never wanting to be referred to as a male and a preference for "female toys, clothes and activities".

Court documents seen by the BBC show medical experts testified that surgery would remove the risk of Carla developing tumours and that she had no certainty of future fertility. The surgery should happen before puberty, they said. What is intersex? If you are born with a mix of male and female sexual characteristics, this means you have a disorder or difference of sex development (DSD), also known as being intersex. There are numerous different conditions which come under this umbrella term. Taken together, they are more common than you might think - experts say perhaps one in 2,000 babies are born with some kind of sex development difference. These conditions occur when the reproductive organs and genitals do not develop as expected. As a result, you might have female sex chromosomes but your reproductive organs and genitals are male - or the opposite way round. Or you may have a mixture of male and female organs and genitals, or some that are neither clearly male nor female. This occurs because of how your particular genes respond to the sex hormones in your body. DSDs can be treated with hormone therapy, psychological support and - sometimes - surgery.

'At 12 I grew a beard and had a period'

The court ruled the parents did not need permission to arrange surgery. The ruling was made in January but it was not immediately made available to the public, The Australian newspaper said. "I consider the proposed medical treatment 'therapeutic' as being necessary to appropriately and proportionately treat a genetic bodily malfunction that, untreated, poses real and not insubstantial risks to the child's physical and emotional health," Family Court Judge Colin Forrest said in making his ruling.

Campaigners question surgery

Some intersex campaigners have challenged the ethical basis of irreversible surgery, arguing that gender identity is complex. One advocate, Morgan Carpenter, told the BBC that children should decide their identity for themselves when they are older. "Gender assignment is always appropriate," he said.

"What is not appropriate is surgically enforced gender assignment." Mr Carpenter said he believed medical and legal professionals often wrongly approached variations in sex development as disorders in need of correction. "We need clinicians to consult the community to develop non-surgical options," he said.

Some terms used to discuss gender identity

Intersex: Applies to a person with a combination of sex characteristics - chromosomes, genitals or reproductive organs - neither solely male nor female.

Non-binary: Applies to a person who does not identify as "male" or "female".

Genderqueer: Similar to non-binary, sometimes shortened to "queer", an ambiguous word that can also be used to describe a person's sexual orientation, eg lesbian, gay or bisexual.

Transgender: Applies to a person whose gender is different from their "assigned" sex at birth, often shortened to "trans".

See more: A guide to transgender terms

Tech article n°1

(http://www.bbc.com/news/technology-38242187)

BBC tests 4K Planet Earth II in HDR on iPlayer

The BBC has begun a trial of 4K high dynamic range (HDR) video on its iPlayer streaming platform. The test involves four minutes of footage from its Planet Earth II series, which promise to reveal more detail and present more vibrant colours than was possible before. It is part of efforts to develop technologies that will make live broadcasts in the new formats possible. But only a minority of TVs can screen the footage at this stage. "One of the clips is a frog on a leaf with lots of rain, and the reason this is so interesting is that the redness of the frog is a really deep Ferrari red that you would never get in broadcast television at the moment," explained Phil Layton, head of broadcast and connected systems at BBC Research Development. "We want to show that this is how the BBC could make ultra-high definition HDR material available to iPlayer. "And we want to use this as a trigger to work with manufacturers to get their products updated so there's a pathway there for future on-demand BBC content."

As the trial went live, only Panasonic's latest screens supported the test's underlying technology - known as hybrid log-gamma (HLG). Although recent models from other manufacturers can also be updated to add the facility, it is unclear whether the firms will do so. Holding back support for HLG would give the companies an opportunity to market future models as being the first to support the format closer to when it launches. More pixels, better pixels From the consumer's point-of-view, there are two key improvements. A 4K picture - also referred to as ultra-high definition (UHD) - means that four times as many pixels are used than in 1080p HD. This makes it possible to reveal details - such as the texture of an animal's fur or the faces of the crowd at a football match - that would have appeared more blurry otherwise.

The main caveats are that the TV needs to be suitably large and the viewers close enough to it, otherwise they will not appreciate the difference. HDR takes advantage of the fact displays can go brighter than before to allow an image to be graded with more levels of brightness between black and white. This is not about everything looking brighter, but rather using the greater contrast range to allow shadows to appear less murky and highlights - such as sunlight gleaming off water or stars twinkling at night - to be better defined. In addition, HDR uses a wider colour space - meaning it is capable of showing shades of colours that could not be transmitted in traditional broadcasts, helping footage appear more lifelike.

Many people who have experienced both, say that HDR has a greater impact on picture quality than 4K. "It gives more nuance to the picture, which if you are talking about Planet Earth is going to be amazing," said Stuart Miles, founder of the Pocket-lint news site. "The best way to describe it is it's like when you add a subwoofer to a sound system. "Until you've had it you don't realise you need it, but once you've added it you ask how you could have lived without it." Higher costs The BBC will make the footage available via the different smart TV versions of its iPlayer app as soon as the relevant models support it. But it will be some time before it starts offering scheduled broadcasts in 4K and HDR.

Mr Layton said the technological challenges could be resolved within the next 18 months. But programmes will cost more to make if they take advantage of the innovations. So, the improved quality will have to be weighed against the fact the majority of viewers will be unlikely to have TVs that support the new technologies for some time to come. In the meantime, Amazon and Netflix both offer some pre-recorded shows and movies in HDR and 4K. And BT and Sky both offer movies and sport in 4K but not HDR.

Tech article n°2

(http://www.bbc.com/news/technology-38250351)

Fan-made version of 'classic' World of Warcraft returns

Fans of World of Warcraft are set to launch a server that lets people play the game as it was 10 years ago. In early 2016, the same fans faced legal action from WoW's creator, Blizzard, which caused the shutdown of a similar service. An outcry over the shutdown led Blizzard to halt legal action and start talks about running legacy servers. But the WoW fans are starting their own server rather than waiting for Blizzard's own legacy support system. Shared bond WoW fans will be able to sign up for an account and start playing on the Elysium server on 17 December. It uses code initially developed for the Nostalrius server that, at its peak, had more than 150,000 active players. Since it debuted in 2004, World of Warcraft has changed through the addition of many expansions. Legacy servers such as Nostalrius seek to preserve the game in its "classic" form before most expansions were released. World of Warcraft is an online multi-player game in which players explore the vast landscape of Azeroth, complete quests and

interact with other gamers. In April, Blizzard threatened legal action against Nostalrius, saying that letting the server run would undermine its claim to the intellectual property used in the official game. More than 250,000 people then signed a petition calling for Nostalrius to be resurrected. But rather than allow that, Blizzard met the developers for Nostalrius and signalled it was willing to talk about ways to preserve earlier versions of the game. Nostalrius's developers halted plans to release the code for their legacy server while Blizzard worked on its own system. But with no sign of that official legacy service appearing, the Nostalrius code, and many of its developers, has been transferred to the Elysium project, which will run the server. Anyone who used to have an account on Nostalrius will be able to move it to Elysium. "We all share a single bond that unites us; a love for vanilla World of Warcraft," wrote the Elysium project on its web page. Blizzard has not commented on the decision to create the Elysium server.

Business article n°1

(http://www.bbc.com/news/world-asia-38246195)

Japan's third-quarter growth rate revised down sharply

Japan's economy grew much slower than initially estimated in the third quarter of the year, figures have revealed, as business investment fell. The Cabinet Office said the economy grew 1.3% in the three months to the end of September, compared to the same period a year ago. However, that was sharply lower than the previous estimate of 2.2%. The new data indicated that investment by companies in the quarter had been weaker than initially estimated.

Capital expenditure fell 0.4% in the quarter, as steel and real estate companies reduced investment. However, consumer spending was revised up, while separate data showed improving sentiment in the services sector.

Currency impact

Just over a quarter of the growth came from net exports. Economists are hopeful that exports will pick up following the rise in the value of the dollar since Donald Trump's election as US president. "With a stronger dollar and potentially higher demand in the US, companies are returning to their investment planning boards, which would fill the missing link in Japan's current recovery," said Martin Schulze from the Fujitsu Research Institute in Tokyo.

Japan's economy has been struggling for several years, raising questions about Prime Minister Shinzo Abe's strategy to revive the Japanese economy. Mr Abe took office in late 2012, launching a growth plan that included three main elements - pumping more money into the economy, boosting government spending and cutting red tape. But some analysts believe it is still too early to write off his policies, which have been dubbed Abenomics. "The revised economic growth numbers are disappointing, but a new US president, a higher US dollar and a lower yen could be all it needs to tip a finely balanced Japanese

economy into growth," said David Kuo, chief executive of the Motley Fool Singapore.

Business article n°2

(http://www.bbc.com/news/business-38247779)

Volkswagen emissions: UK and six other nations face legal action

The European Union (EU) has started legal action against seven nations. Four, including the UK and Germany, are under fire for failing to take action against Volkswagen for cheating emission tests. Member states have two months to respond. The German car giant has had huge fines in the US over its use of "defeat devices" used to hide true levels of emissions. More than one million cars in the UK are involved. Spain and Luxembourg are the other two nations who the EU says have not taken action against the company. Another three countries - the Czech Republic, Lithuania and Greece - are being hauled up for not even including the possibility of fining carmakers over potential violations. On top of this, the European Commission has also called Germany and the UK to account for refusing to share details of breaches of EU emissions laws they discovered through their own investigations earlier this year. Industry commissioner Elzbieta Bienkowska said in a statement: "National authorities across the EU must ensure that car manufacturers actually comply with the law." Analysis, BBC business reporter Theo Leggett The European Commission is showing distinct signs of frustration over the apparent unwillingness of European governments to take action against Volkswagen over its use of defeat devices - or even to keep it informed about their investigations. In the United States, where the company was actually found to be cheating emissions tests, it has already reached a 15bn civil settlement with the authorities; it could face criminal fines as well. There has been nothing of the kind in Europe. European law does state that carmakers which break the law should face penalties - but it's up to individual governments to enforce the rules and hand out punishments. It's worth remembering though that VW still claims that the software fitted to its cars was not actually illegal in Europe. What the Commission really wants is the power to punish errant carmakers itself. That would, for example, prevent the governments of countries with powerful motor industries from being too soft on the manufacturers they would rather not upset. There is already a proposal on the table to give it those powers, but it needs support from member states. This legal action - which could drag on for years - may just be a move to bring governments around to the Commission's way of thinking. Fines and claims Under EU law, member nations are responsible for overseeing whether cars meet standards or not. Investigations in the UK, Germany, France and Italy have uncovered the use of cheat devices, but no action has been taken against VW, which employs 120,000 people in Germany with almost 500,000 employed elsewhere around the world. The company has agreed to pay 15bn in a settlement with US authorities and owners of about 500,000 vehicles after the

software cheat was exposed, but so far European nations have taken no such action against the company. About 11 million cars worldwide have the software. As well as fines, Volkswagen is facing €8.2bn (9.1bn; £7bn) in damages claims from 1,400 German investors over its emissions scandal, a state court has said. Australia also launched legal action against the carmaker and asset manager Blackrock and a group of institutional shareholders said they would sue VW for €2bn. The claims relate to the drop in Volkswagen's share price after the scandal broke. The VW group comprises 12 brands from seven European countries: Volkswagen passenger cars, Audi, Seat, Skoda, Bentley, Bugatti, Lamborghini, Porsche, Ducati, Volkswagen Commercial Vehicles, Scania and Man.

Business article n°3

(http://www.bbc.com/news/world-asia-38246196)

Michael Jordan wins trademark case in China's top court

China's supreme court has ruled in favour of US basketball legend Michael Jordan in a trademark dispute. The People's Supreme Court ruled a Chinese sportswear company must stop using the characters for Jordan's name, read as Qiaodan in Chinese. Qiaodan Sports registered the name more than a decade ago but Jordan's lawyers said it built its business around his Chinese name without his permission. Jordan has welcomed the decision which overturns previous rulings against him. "I am happy that the Supreme People's Court has recognized the right to protect my name through its ruling in the trademark cases," he said in a statement sent to the BBC. "Chinese consumers deserve to know that Qiaodan Sports and its products have no connection to me." "Nothing is more important than protecting your own name, and today's decision shows the importance of that principle." The basketball star first started legal action against Qiaodan Sports in 2012. His team argued that Qiaodan's trademarks had damaged his legal rights to use his name and asked the court to invalidate more than 60 trademarks used by the company. Partial victory The court agreed Qiaodan Sports had violated trademark law and its registration of the name should be revoked. But his win is only a partial victory. While the company cannot use Jordan's Chinese name, the court upheld a ruling allowing it to use the Romanized version of Qiaodan, pronounced "Chee-ow-dahn." A Shanghai court is yet to hear a separate naming rights case.

Science-environment article n°1

(http://www.bbc.com/news/science-environment-38249208)

Trump nominee to rekindle climate battle?

The nomination of Oklahoma attorney general Scott Pruitt to be the next head of the US Environmental Protection Agency (EPA) has two important

ramifications. The first is a clear signal from the incoming Trump administration that environmental regulations, especially as they apply to the production of energy, are set for fundamental reform. The second implication of Mr Pruitt's nomination is that the Trump camp is not willing to accept that many aspects of the science of climate change are now settled. Leading the charge In his official biography Mr Pruitt revels in his role as a "leading advocate against the EPA's activist agenda". Elected as attorney general of Oklahoma in 2010, Mr Pruitt has engaged in a legal fight with the Federal government on a number of issues including Obamacare. But it is in fighting the EPA and President Obama's climate regulations that he has really made an impact. As the chief law officer of a major energy producing state, Mr Pruitt has taken a lead role in the 28-state challenge to the President's Clean Power Plan. He has also secured an injunction blocking EPA's "Waters of the US" rule, which expands the scope of the Clean Water Act. "He understands the regulatory stranglehold that the EPA has had on industry during the Obama administration," Harold Hamm, a top energy adviser to Mr Trump, told the Wall Street Journal.

"I believe that he will work to unleash prosperity in America through the proper use of regulations and adhering to the rule of law," said Mr Hamm, who is an ally of Mr Pruitt, and also CEO of Continental Resources, a major independent oil producer. Republican politicians also welcomed the move. "Pruitt is excellent choice for EPA," said Texas governor Greg Abbott via Twitter. "He I teamed up on many lawsuits against the EPA. He'll bring needed change." The appointment though has sent shivers through the environmental community, many of whom believe it is akin to putting the fox in charge of the hen house. Sam Adams from the World Resources Institute said it raised "some deeply troubling questions". "Americans depend on EPA to promote human health and protect families and future generations. Its ability to protect air, water, and the climate for all people must continue," he said in a statement. Mr Pruitt is sure to face a tricky nomination process. Much will be made of his links to oil and gas companies. He has had funding from the political-action committee of Charles and David Koch and from Devon Energy, another Oklahoma based oil and gas company. The relationship with Devon Energy backfired somewhat when the New York Times revealed in 2014 that Mr Pruitt had sent letters of complaint to the EPA that were actually written by Devon's lawyers. Climate disagreement What gives environmentalists greater concern, though, are Mr Pruitt's views on climate change. Writing in the National Review earlier this year, Mr Pruitt made it clear that he does not accept the widely-held scientific views about the scale of temperature rises that are being caused by human emissions of carbon dioxide and that the impacts of this warming will be devastating for many parts of the world. "That debate is far from settled. Scientists continue to disagree about the degree and extent of global warming and its connection to the actions of mankind," he wrote. "That debate should be encouraged - in classrooms, public forums, and the halls of Congress. It should not be silenced with threats of prosecution. Dissent is not a crime." Senior scientists say the evidence on the causes of climate change and the human role in them is very clear. "Can we detect and attribute a signal in the warming that

we observe that is connected to CO2 emissions and concentrations, then yes the science is settled to a very high extent," said Prof Arthur Petersen, from UCL, and a former chief scientist with the Netherlands Environmental Assessment Agency.

"If the question is the global average temperature and its causes - yes, it is settled beyond any reasonable doubt." But researchers do acknowledge that there are several outstanding questions on the amount of carbon that can be emitted in the future and the likely response of the climate system to those emissions. "Aspects of the debate are far from settled - if you ask me how much carbon can we afford to dump in the atmosphere and keep temperatures well below two degrees, I would say there is an uncertainty in that number of a factor of three," Prof Myles Allen from the University of Oxford, told BBC News. He says that he believes the Trump administration is being "smart" and cautions researchers against engaging in a battle over fundamental issues. The legitimate argument, he says, is about how much climate change are we as a world are prepared to take and what do we do to limit it. "Part of that is how much do we care about small island states versus the interests of American industry. These are ethical and political questions," he said. "If the Trump administration wants to come out and say we are good with four degrees (of warming), they should open that argument. "If on the the other hand they agree with the Paris goal of stabilising temps well below two degrees and nothing they've said suggests they disagree with that goal, they just think it will be easier than many people fear- then we can have a very different conversations, which are about how we achieve that." Follow Matt on Twitter and on Facebook

Magasine article n°1

(http://www.bbc.com/news/magazine-38242917)

The mystery of the stolen Klimt

Nearly 20 years ago a valuable portrait was stolen, in bizarre circumstances, from a gallery in the northern Italian city of Piacenza. Until recently there appeared to be little prospect of it ever being recovered - but then police received some perplexing new information, and they now think it will be back in the city within weeks or months. Carabiniere Sgt Maj Salvatore Cavallaro was on a ladder looking out on to the roof of Piacenza's Ricci-Oddi gallery through a partially open skylight. "It doesn't fit," he shouted to his colleagues below, as he compared the size of a heavy gilded frame on the roof beside the skylight with the narrow opening. "No way the thief could have fished the painting from up here." You could just about imagine that a thief on the roof had hooked the frame on a line, and pulled it up to the ceiling - but that would have been no use. This was clearly not the exit through which the portrait by the Viennese turn-of-the-century artist Gustav Klimt had left the building. So why, on 22 February 1997, was the frame on the roof?

Ten months earlier the Portrait of a Lady had been involved in a drama of

a different kind, thanks to a sharp-eyed 18-year-old art student, Claudia Maga. While flipping through The Complete works of Gustav Klimt she had noticed a strong resemblance between The Lady and another Klimt painting, Portrait of a Young Lady, that had not been seen since 1912. "The Young Lady had a scarf and a hat but they both had in common the same glance over the left shoulder, the same smile and the same beauty spot on the left cheek," Maga says. She had photocopied and enlarged the two small photographs in the book, had drawn the profile of the Young Lady on tracing paper and put it on top of The Lady. "And that was it," she says. "The Lady was concealing another portrait beneath it, the only double portrait Klimt has ever painted."

Maga got the gallery's former director, Ferdinando Arisi, interested in her theory. A few weeks later he picked her up from art school, drove her to the gallery, and removed the portrait from its frame. Wrapping it in brown paper they headed for the local hospital, where sure enough a series of X-rays revealed the dim shadow of the earlier work beneath the surface. The story behind the painting was the next surprise. Klimt had fallen madly in love with a young girl from Vienna, it was said, who had quickly become his muse. Then, when she suddenly died, he painted over her portrait to forget the pain of his loss.

Piacenza was in ecstasy. A special exhibition was arranged to show off the painting in a new location close to the city hall. At the same time, the gallery was to be renovated, and many paintings started to be packed and moved into storage. Workers were coming and going. And when the Klimt itself went missing, no-one immediately realised. The gallery staff assumed it had been removed deliberately as part of the preparations for the new exhibition. The painting looked convincing, but the smell of the oil paint was fresh... "Then we received a phone call from the gallery," Sergeant Cavallaro says. "They muttered, 'We can't find The Lady.'" When he arrived on the scene, he says, "The doors of the gallery were open, people were going in and out, and the security system was switched off." Faced with the mysterious theft and no leads, the police were stumped. They asked a notorious local art thief for advice, but even he was unable to put them on the right track. Little progress had been made when, on 1 April 1997, border police intercepted a package on the Italian/French frontier at Ventimiglia. It was addressed to the former Italian PM Bettino Craxi, who was at the time hiding from the law in Hammamet, Tunisia. When they opened it, they found a Klimt. Stefano Fugazza the then director of the Ricci-Oddi gallery thought it was an April Fool's joke, but Arisi, his predecessor, was optimistic and keen to get to Ventimiglia straight away. "We drove madly to Ventimiglia," Fugazza writes in his diaries, "but the only thing we came back with was a speeding ticket." The painting looked convincing, but the smell of the oil paint was fresh. It wasn't the original, it was a high-quality forgery.

There is another strange paragraph in Fugazza's diary. Days before the painting disappeared, he writes, he had contemplated the idea of talking to the carabinieri and, with their permission, pretending that the portrait had been stolen, in order to draw more attention to the forthcoming exhibition. "But now The Lady has gone for good," he adds, "and damned be the day I even thought

of such a foolish and childish thing." Soon afterwards Cavallaro was assigned to other work and the case was closed.

It remained closed until 2013, when the carabinieri made a fresh attempt to identify a fragment of a fingerprint found on the frame - in vain, as it turned out. But last summer a local journalist arranged a meeting between the new carabiniere investigator, Col Luca Pietranera, and an art thief he had got to know in one of Piacenza's many bars. And the thief turned out to be a mine of information. The thief told the colonel that he was the man the original investigators had asked for advice, back in February 1997, as they searched for the perpetrator of the theft. He then confessed that he was, in fact, the man who had carried out the theft and left the gilded frame on the roof as a coup de theatre. Then he explained that what he had stolen that day was in fact a copy.

So what had happened to the real painting? "Oh well, I stole it months before anybody had noticed," the now elderly thief proudly tells me, when we meet in a cafe in Piacenza. A few months after the discovery of the double Klimt, around November 1996, he just walked into the gallery and replaced the original with a copy, he says. "Nobody blinked, nobody noticed. It was an easy and carefully planned inside job." He is now helping the carabinieri with their inquiries into a number of crimes, in return for immunity from prosecution. But why was it necessary to steal the copy, I ask the chain-smoking thief. This was done, he says, to hide the fact that it was a copy. The special exhibition would doubtless have brought Klimt experts from far and wide, one of whom would surely have spotted it was a fake - and this might have been disastrous for the gallery insider who had assisted with the theft.

But the thief's most startling claim is that the painting will be returned by the 20th anniversary of the theft (or more accurately, the theft of the copy). In other words, by February next year. How he can know this is unclear. After all, by his own account, the painting was long ago sold by a dealer for a large quantity of cash and cocaine. But he confidently makes this prediction - and the carabinieri say he may well be correct. They are currently in contact with police in another European country where the painting is thought to be held in a private collection. The only thing they are not completely sure of is whether the painting they are now on the trail of is the original, or a copy. This, they say, they will only be able to confirm when they have it in their hands. Where are the copies? The forgery that hung, unnoticed, in the gallery for three months is lost - it is described by the art thief as "high-quality" and is believed to have had two layers, like the original, with the Lady painted over the Young Lady Carabinieri assume the forgery found by border police in Ventimiglia is somewhere in the basement of the Ricci-Oddi gallery - it is also thought to be a double portrait When the gallery held a news conference to announce the results of the fruitless Ventimiglia escapade, they displayed a poor-quality copy of the portrait, which is now held by police in Bologna Another poor-quality replica surfaced in a second-hand shop in Piacenza

Magasine article n°2

(http://www.atlasobscura.com/places/the-skeleton-lake-of-roopkund-india)

Skeleton Lake of Roopkund, India

A lake with hundreds of ancient skeletons surrounding it. The surprise is what killed them...

In 1942 a British forest guard in Roopkund, India made an alarming discovery. Some 16,000 feet above sea level, at the bottom of a small valley, was a frozen lake absolutely full of skeletons. That summer, the ice melting revealed even more skeletal remains, floating in the water and lying haphazardly around the lake's edges. Something horrible had happened here.

The immediate assumption (it being war time) was that these were the remains of Japanese soldiers who had died of exposure while sneaking through India. The British government, terrified of a Japanese land invasion, sent a team of investigators to determine if this was true. However upon examination they realized these bones were not from Japanese soldiers—they weren't fresh enough.

It was evident that the bones were quite old indeed. Flesh, hair, and the bones themselves had been preserved by the dry, cold air, but no one could properly determine exactly when they were from. More than that, they had no idea what had killed over 200 people in this small valley. Many theories were put forth including an epidemic, landslide, and ritual suicide. For decades, no one was able to shed light on the mystery of Skeleton Lake.

However, a 2004 expedition to the site seems to have finally revealed the mystery of what caused those people's deaths. The answer was stranger than anyone had guessed.

As it turns out, all the bodies date to around 850 AD. DNA evidence indicates that there were two distinct groups of people, one a family or tribe of closely related individuals, and a second smaller, shorter group of locals, likely hired as porters and guides. Rings, spears, leather shoes, and bamboo staves were found, leading experts to believe that the group was comprised of pilgrims heading through the valley with the help of the locals.

All the bodies had died in a similar way, from blows to the head. However, the short deep cracks in the skulls appeared to be the result not of weapons, but rather of something rounded. The bodies also only had wounds on their heads, and shoulders as if the blows had all come from directly above. What had killed them all, porter and pilgrim alike?

Among Himalayan women there is an ancient and traditional folk song. The lyrics describe a goddess so enraged at outsiders who defiled her mountain sanctuary that she rained death upon them by flinging hailstones "hard as iron." After much research and consideration, the 2004 expedition came to the same conclusion. All 200 people died from a sudden and severe hailstorm.

Trapped in the valley with nowhere to hide or seek shelter, the "hard as iron" cricket ball-sized [about 23 centimeter/9 inches diameter] hailstones came

by the thousands, resulting in the travelers' bizarre sudden death. The remains lay in the lake for 1,200 years until their discovery.

Know Before You Go There are no roads to this place yet, so one has to undertake a 3-4 day trek to reach the skeleton lake starting from Gwaldum in Chamoli district. The skeleton lake is covered with ice for most of the time during the year.