Birkbeck College University of London
Department of Computer Science and Information Systems

PROJECT REPORT

COMPARATIVE ANALYSIS OF TEXT SUPPORT METRICS

Author: **Simona Stavarache**
ID: **13174625**
Supervisor: **Stelios Sotiriadis**

SEPTEMBER 2020

A report submitted in partial fulfilment of the requirements for the MSc in Data Analytics (Advanced Computing Technologies)

ACADEMIC DECLARATION

"This report is substantially the result of my own work except where explicitly indicated in the text. I give my permission for it to be submitted to the TURNITIN Plagiarism Detection Service. I have read and understood the sections on plagiarism in the Programme Handbook and the College website.

The report may be freely copied and distributed provided the source is explicitly acknowledged."

TABLE OF CONTENTS

1. Abstract

The problem of determining the semantic similarity of two texts has been well studied in the natural language processing literature over the years. Because of two main concerns, namely the increased propagation of fake news in online environments and the rise of digital marketing and analytics, the need for a metric or method for determining textual supportivity has increased. This paper presents a comparative analysis of algorithms used previously for determining text similarity and their adaptations to the new task, performed on a database of tweets recorded in July 2020 that can be considered representative for the actual social media context. Our results reveal the advantages of WordMover's Distance and Glove embeddings, the optimisations but also challenges raised by the use of clustering and sentiment analysis, and set up adventurous research ideas for future work.

2. Introduction

2.1. Problem & Motivation

The technological advances from the past few decades have undeniably changed the world and how humans interpret or understand it, creating a new, high demand industry also known as Data Science. It started with the IT infrastructure development that pushed towards having increasingly more storage capabilities at a constantly decreasing price rate, while the size of computers became smaller, but also with the impressive work in the automation area, an ideal that has always been a humankind priority. Before this data revolution, gathering information - opinions, data artefacts, relationships, news, etc. - was difficult, mostly through questionnaires, census or single point of truth which were done manually, for smaller parts of the population and took years and years to collect. Due to high improvements in these areas, new applications that generate insane amounts of data have developed - Facebook, Twitter, IoT devices - offering the right context for a new type of business consultancy to emerge - based on automated data analysis services. Today's business management industry is in the process of not relying only on their "gut" as it used to, but according to MIT Sloan School of Management [4], the leaders started to embrace a "data-based analytics" approach in their day to day decisions. But what makes data to be this high demand resource in today's economy? According to Professor Erik Brynjolfsson from MIT, data should be looked at as a currency that is interchangeable and has great value if used accordingly. Individual businesses that actively invested in their data analysis teams and adopted a computer-controlled decision-making process have noticed an increase in profitability of up to 5% more than other types of businesses that relied on historic practices. Besides increasing the earnings, analytics can reveal existent issues or ideologies that do not conform with today's mentalities/trends. However, there is still a gap between the real-time available data -- only in the last 2 years, the collected amount reached an order of magnitude of trillion gigabytes (about 90% of the data gathered until now) [5] -- and the efficient usage of it. Precisely because of this, a generally believed idea is that data collection without a proper way of usage makes the process of understanding its information more complex, rather than simplifying it. Furthermore, another struggle in the data science field consists of the various types of structured, semi-structured and unstructured data available - short texts which are prone to mistakes, differences in dialect and emoticons, scientific papers which are long with specific jargon, numerical quantitative and qualitative data, different distributions, machine data, genomic data.

All types of data can offer useful information, but previous techniques tackled only a structured input, excluding an unlimited flux of information: language. When considering the amount of unstructured data produced it is critical to research new ways, more complex and efficient to fully automate the analysis process. From medical records to social media, all observations might be crucial when trying to deal with unexpected events like today's COVID 19 pandemic - looking for solutions and predictions - or natural disasters - Walmart plan for the 2004 hurricane, Frances, that used a predictive model based on the last hurricane data history to shape its catalogue [6] - or to understand people's habits to offer tailored services which will benefit both the provider and the consumer - custom medicine, systemic interactions, psychological understanding, marketing, election polls, political overview etc.

To be able to provide these services and many more that will be discussed in 2.3, an important step is to analyse unaltered human interactions. We need to take into consideration that people change their actions/behaviour in different situations to follow some societal unwritten rules or to strive above others. A good source for behavioural analytics is social media as it has the biggest and widest collection of messages, statuses or other useful information that can optimize the model such as one's influence over the network. In addition to that, social media provides the advantage of people not being so constrained by norms as much as in the real world, because accountability online is perceived as very low so they are more likely to express their beliefs.

The problems described above can be engaged using a different approach that we suggest as the main subject for this paper: we look for techniques and metrics that can offer some kind of order between texts by finding paragraphs - messages, posts - that support each other in theme and polarity, thus offering a clear structure to our data.
This research looks for most accurate techniques that show the relatedness between texts by comparing several metrics combined with ML and NLP.

2.2. Environment - Social Network

With the Internet's development, first used in the US Army for data exchange, "hyperlinks link content"[7], the system evolved rapidly as smaller communities were benefiting from shared access to data. This success guaranteed the expansion of the Internet as we know it today, individuals from built-up communities - forums - being able to exchange any type of information within seconds. Moreover, the need for a digital social environment combined with this, yet unexplored, medium got the people's interest. Starting around the 2000s, MySpace became the first social media application, but it rapidly declined in just eleven years, being overthrown by the two most used applications of today: Facebook and YouTube. However, the social media environment evolved in remarkably diverse types of platforms - messages, photos, news, short stories, videos, conferences - that made correspondence extremely accessible and for everyone's taste, with profiles becoming "self-portrayals"[8] and means of opinion-forming and expression. As people started to dedicate approximately 6 hours per day on these platforms, sharing their data about their desires, cravings, dreams etc. from the new medium got the attention of big corporations as the potential of knowing exactly what people want and what they would be willing to spend their money on was alluring. Furthermore, the majority of people today take their news and information through social media which can lead to major problems due to fake news propagation and the ease of online manipulation.

However, the magnitude of social data gathered, due to the evolution of the transmitter-receiver model, has not only marketing value, but scientifical as well. Researchers now have the resources, at least data related, to deal with global issues such as illnesses or global warming. As today's technology evolved and is still evolving towards new techniques of processing unstructured data, we would like to focus our attention on a 14 years old platform: Twitter.

Twitter was developed in 2006, reaching by 2018 around 329 million users with an impressive of 500 million tweets per day written. This platform was produced in North America as a microblogging service based on short public posts, consisting of only 280 characters. But why Twitter? Compared to other social media platforms, Twitter offers the largest publicly available data archive, being the go-to environment for data analytics. Besides the availability, Twitter messages are unique due to the limitations imposed as they must be concise causing vagueness, but simple in nature - most of the posts present a challenge due to syntaxes like grammatical errors, emoticons and links, not necessarily the depth of the message - the classical "covfefe" example being excluded from the argument. Moreover, this platform offers the closest (available) real-life conversations covering a wide range of subjects as politics - lately, politicians started to share news/comments through Twitter -, economics - people share their reports about market stocks or financial related topics -, news - all major news companies maintain and share their findings through their profile -, sports, fashion, etc. Another advantage of choosing Twitter is the scientific challenge that the short, ambiguous texts provide and offer researchers the possibility of reinventing/redesigning the already existing approaches. Due to the high level of sarcasm present in the colloquial talking, it became extremely difficult to find the exact meaning. As tweets contain such a limited number of characters, finding their polarity through context is problematic.

We will further discuss some real scenarios and explain how our solution can be applied.

2.3. Scenarios

The scope of this paper is to present a comparative analysis of different techniques for text similarity, not only syntactic but also semantically, as our goal is to deliver a framework type of application that can be used in further scenarios described below but not limited to. Nevertheless, our goal is not to implement these scenarios but to offer the main functionality that can help achieve these goals.

In the next lines we present a definition for our desired algorithm and then try to contextualize it by providing some examples of possible use cases:

Let there be an algorithm A taking as input two text T1 and T2 and deciding whether or not there exists a supportive relationship between the two.

1. Fake news spreading

Also known as alternative facts, this technique is used to produce news that might reflect only part or no part of the truth, to deliberately misinform society to influence its overall policies and undermine its internal authority. These pseudo-news tend to be narrated in a sensationalist fashion to attract readers, keeping them entertained, while the absence of accountability of what people share or write on social media encourages them to spread/share fake news. Today, as the technology evolved and false information became easier to spread without the existence of any real consequences, it is of utter importance to discover and lower the impact of the clickbait stories.

Our solution could be used in such use case as follows: assuming there exists a database D containing an extremely large number of texts that have been proven as fake news or coming from blacklisted accounts known for producing such texts before, then by using A, it would be possible to determine if a new text can be considered as such by performing an iterative parse of D and running A for each of the elements with the new text. In this case, it would be extremely important to update D if a new fake news text is found. However, this method is heavily dependent on the method chosen for initially constructing D, but datasets with a high degree of reliability can already be found online, such as the one in [9] or [10].

2. *Identifying biases*

Biases are often ideas that tend to be close-minded and detrimental towards individuals, societies or convictions. This disruptive behaviour can lead to false assumptions that can slow down the evolution of socio-scientific progress. Humans tend to form biases as an evolutionary technique to deal with the new, the unknown.

This scenario is similar to the one described above, however, the focus, in this case, should be on the distribution of ideas as it is proven that the number of similar ideas can affect people's decisions both in negative and positive ways. For this case, we are focusing on the biases that can lead to negative results, as their prior identification and mitigation discovery could be crucial in many situations. Assuming that we have a database D with known biases and a database D' with test data, using our solution A we can find similarities and cluster the data from D' into the already known biases. However, there might be large sets that do not fit with D, which might lead to the discovery of new biases.

3. *Profiling*

Psychological profiling defines the process of analysing individuals behaviour and their characteristics usually to predict their actions.

Following a similar approach as above, an interesting application could be related to politics, by connecting individuals with similar philosophical views in this area. Parties can be formed automatically, based on their real vision and interests, thus creating more homogenous unions. This application can also be generalized towards creating smaller communities that have people with similar beliefs, concepts or attitudes, the end goal being a more targeted socialization structure. This time the focus should be on the individuals. We assume having a database D, that contains a large number of various individuals posts, tweets, messages, etc. Using our chosen solution A, we can find semantic similarities and group/cluster people's messages by concept similarity. Further, we can find and put in contact individuals that have in common more ideas or developed in a similar way along the years.

4. *Digital marketing*

The Internet's appearance has completely changed the way marketing was done, by adding a channel that facilitates communication around the globe in just a few seconds. This evolution stood at the bases of a new branch in the advertisement industry, also known as digital marketing - promotion of products or services through the digital environment. However, this industry does not limit itself to the delivery of the ads but has also influenced the scientific path in data analytics.

Our solution, A, can also contribute to the digital marketing evolution by directly analysing trends and people that follow them. Assuming that we have a vast database D with tweets and reviews, A can find how many people share their opinions about the inquired product/service, if anyone, considered an influencer, is talking about it and the polarity of the reviews. Such results can be extremely useful when analysing the size or engagement of the specific market.

5. *Security threats detection*

By taking advantage of different vulnerabilities, such as - in our case - the need of humanity to believe in attractive scenarios that can also give a psychological advantage over others, malicious entities can polarize a society. Such attacks are difficult to deal with as they rely on the basic right of freedom of speech and belief, being extremely complicated to distinguish between genuine faiths and opinions inspired by propaganda campaigns. We propose a mitigation method in the following lines:

Let there be a database D of extended size with data collected from Twitter or other microblogging based social media, including a large number of different users. Assuming that we have a database D' containing known/categorized phrases or slangs usually used by extremists, scammers or terrorists to deliver hate speeches, gather recruits or induce aggression, we can use our proposed solution A to find users that have relatable perceptions of the world or use similar speeches to persuade others.

2.4. Other work

In the following lines, we will discuss some similar applications to the one we propose for the sake of a better understanding of the general problem space and the capabilities of the technology involved.

There are extensive works in the area of similarity between documents, but little development of literature regarding short texts or sentences due to the difficulties that this analysis presents. However, a major part of today's data consists of small texts - usually discussions -, thus the need of comparing and combining techniques to achieve a good translation human - machine at semantic level with various applications in information retrieval. A good starting point is in Jackendoff's [11] paper where he stipulates that judgements of likeness or contrasts form the main criterion for the construction of semantic relationships. More exactly, he categorized linguistic concepts into two classes - synonymy, redundancy or paraphrasing for the former, while antonymy, inconsistency and contradiction drive from the latter.
Furthermore, looking into papers that tackle a quite similar dataset format as ours - sentence similarity - we can find three dominant approaches: vector-based, corpus-based and hybrid methods as described in [12].

In vector-based approaches, a document/text is computed as a vector of words, relying on the pattern matching method to find any correlation among data. However, this method has much more applicability over longer documents as this technique takes into consideration the number of words shared between them, but not necessarily impossible as described in [13]. Huerta uses over sentences a quite novel system that extends the pure vector topology with the addition of precomputed discriminative weights. These are being computed by using a maximum entropy criterion whereas the overall variance of a feature set of weights is used for determining the semantic importance of that feature. Furthermore, towards finding if a sentence X is the rephrasing of Y, Huerta uses Generalised Singular Value Decomposition and describes both the query and the document in a concept space using k singular values.

The corpus-based method uses large datasets of natural language that aggregates all meanings or contextual appearances of multiple words achieving a collective set of constraints used in finding semantic similarities. An important technique when considering the corpus-based methods is the Latent Semantic Analysis(LSA), described in [15]. Its main focus is on the contexts in which the input words appear, but also in those where they don't, as these contexts seem to form a relevant set of constraints regarding the words' semantics. In a more colloquial manner, 2 words can be considered similar if they share close enough sets of context constraints. For practical purposes, singular value decompositions are used for determining the semantic representations based on statistical relationships between words. However, this also introduces a computational limit to the size of the context to be evaluated, but nonetheless, the technique has proven to be extremely valuable in practice. For example, relevant results have been achieved in the evaluation of standard vocabulary and subject matter tests [16], with the interesting twist of closely mimicking human word sorting and categorical judgements.

A combination of corpus-based techniques and knowledge-based measures defines the third method: hybrid. This approach is illustrated in [14] where besides the use of corpus statistics - Brown corpus - for adaptability in various domains, it also uses a semantic net - WordNet - as a source of human logic. In very broad terms, the similarity of two sentences is calculated as a combination of syntactic - in the sense of order - and semantic - using a cosine similarity approach - information vectors. The outputs of this approach were compared with the average graded similarity computed by 32 humans on the same inputs, having a Pearson correlation of 0.816 with 0.01 significance.

2.5. Structure of the thesis

In this section we would like to present this paper's structure as follows:

Chapter 3 continues with the fundamental knowledge needed for applying different types of similarities or machine learning techniques. It builds in-depth descriptions of traditional NLP techniques that will be used in the preprocessing part and chosen similarities for initial tests.

Chapter 4 is reserved for the detailed explanation of our model's architecture and how we will conduct the comparison between the metrics.

Chapter 5 provides the reader with information about the databases and the collection techniques used.

Chapter 6 is reserved for the tools, programming languages, workflows, class diagrams and frameworks used for implementation and code analysis.

Chapter 7 is split into two main objectives: results from preliminary tests over a smaller dataset offering us enough information to establish the similarities that apply better to our case and for each similarity we will present a mathematical illustrative example of how the similarities work to have an overall understanding of the processes involved in the code.

Chapter 8 will contain the final results over the entire database of the selected similarities, while in **Chapter 9** a thorough analysis will be conducted over the results.

Chapter 10 presents the author's review of this research with some ideas for future work and new paths in the research area that could be explored.

3. Background

3.1. Preprocessing techniques and word embeddings

In this section, we will briefly discuss some preprocessing techniques needed in our research as Lowercasing, Lemmatization, Stopword Removal, Normalization and Noise Removal, followed by some text enrichment techniques/embeddings that got our attention - PoS tagging, Word2Vec, GloVe and BERT. We will just give a short overview and references for further individual research as the details of these techniques are not the purpose of this paper.

The first preprocessing technique is quite self-explanatory, as it lowercases all the text data to avoid mismatches of the same word, written differently. Lemmatization resumes to the reduction of words to their root - using either a dictionary-like WordNet or special rules - to avoid text sparsity. The stopwords removal method discards all words that are not essential - lack of useful information - to the model, in order to reduce the number of features and keep the dataset at a decent size. A quite important part of text preprocessing mostly in the case of the online writing, where different illiterate or colloquial forms of writings are in place is normalization. This method transforms these unusual forms of text such as “2moro”, “b4” or “:)” into a canonical form - “tomorrow”, “before”, “smile”. However, there is no standard algorithm for normalization but dictionary mappings, statistical machine translation and spelling-correction are some approaches used in practice. Lastly, noise removal is essential in natural language analysis as it removes characters or digits that might reduce the quality of the analysis.

PoS Tagging
Also known as Part-of-Speech tagging, is defined as the process of designating a corresponding part of speech to a word given the context, thus obtaining a contextual understanding of the word's relationship in a sentence or phrase. POS tagging returns a tree-based structure, also used in determining named entities. There are four techniques: *lexical based* - based on the training corpus -, *rule-based* - create rules that mimic the language specificity, for example, words that end in “ed” or “ing” should be considered verbs -, *probabilistic* - computes the probability of a particular sequence of parts of speech occurring - and deep learning methods - using RNN (Recurrent Neural Networks). More details can be found in [49].

Word2Vec
It is a two-layer neural network used to project text inputs into a multidimensional vector space by assigning a vector to each unique word from the input. Words that share some kind of a context are located in vicinity to each other. We can describe two methods: *Continuous Bag-of-Words* or CBOW and *Skip-Gram*. The two models have opposite approaches as the former makes predictions using the context words, while the latter uses the target words to predict the surrounding context. A wider explanation can be found in [50].

GloVe
It obtains vector representations of words, being trained over a global word - word cooccurrences matrix returning a linear structure of a word vector space. By collecting statistics from a corpus, it can learn the frequency with which words co-occur. More information can be found in [51].

BERT
With its complete name: Bidirectional Encoder Representations from Transformers is a pre-trained model over unlabelled data such as the entire Wikipedia created by Google and considered “state-of-the-art” technique. BERT can be described as a two-step process: learning unsupervised or semi-supervised data bidirectionally - learns from both left and right sides - and tuning the model to be usable for others as a knowledge BlackBox - supervised. An in-depth analysis of BERT’s process is in [52].

3.2. Jaccard Similarity

A quite basic approach defines the similarity between two finite sets (sentences) as the division between the size of the intersection and the size of the reunion. This approach is known as Jaccard Similarity and was developed by Paul Jaccard in 1901. The Jaccard Similarity Coefficient can be mathematically defined as follows:

$$Jaccard\_Similarity\_Coefficient = \frac{|A \cap B|}{|A \cup B|}$$

Eq. 1

In other terms, if the two sets have the exact same elements, in our case words, the expected result would be 1, while if no words are the same the returned value is 0. All other values are in the [0,1] interval.

This simple approach is beneficial but in natural language, where almost every word has several meanings this concept is prone to fail. We can illustrate that by taking the two marginal cases when Jaccard coefficient is 1 or 0. For the former case, the compared sentences should be identical in order to obtain that result. However, in a linguistic context, sentences that use the same words might not always have the same meaning which contradicts with the Jaccard Coefficient. For example, we have the next two sentences:

S_1: Only he told her the truth.
S_2: He only told her the truth.

In this case, even though all the words are the same, a small change in their order can fully affect the overall meaning, while the Jaccard result declares them identical.
The same issue can arise for the latter case when two statements have a similar meaning but different words as in our next example:

S_1: The President of the US speaks in front of the press.
S_2: Trump delivers his speech to the journalists.

These sentences have the exact same meaning but when Jaccard Coefficient is calculated the result tends to be extremely close to 0, thus concluding that S_1 and S_2 have nothing in common, conclusion easily dismissed by a human that has the contextual knowledge. Taking the two examples presented before where Jaccard index fails, we are left with the question if this is enough to rule out this similarity. In the next section, we will discuss it further.

Before focusing on natural language, it is necessary to realise that the Jaccard similarity metric can be used on any two instances of a structured language - in the sense that they follow some predefined rules for example areas such as mathematics, biology, botany and many more - as long as the intersection and union processes have a meaning.

A good starting point for facilitating the understanding of this similarity metric, its useability and limitations in natural language is demonstrated by Jure Leskovec et. al. In [1], the addressed problems for this coefficient are “character-level similarity” problems for example in plagiarism - finding portions of text that are shared between two or more documents -; another application of Jaccard index is in search engines for detecting mirror pages - concept used in web development in order to share the load between hosts by duplicating pages, however, these pages might have some small differences as the associated host information - and avoid returning similar results; Google News tries to capture all pages that share the same article - accustomed in newspaper associations -, with little to heavy alterations, in order to retrieve only one for the customer. It is important to emphasize that Jaccard Similarity by its own cannot interpret the semantic meaning of words but offers meaningful insights for a wide range of applications as Leskovec pinpoints in his paper.

Moreover, this similarity has been used by scientists in several applications for document similarity in combination with other techniques like Genetic Algorithms in [2] or after clustering in [3]. In [2], the Jaccard coefficient has been used as the fitness function following a keyword approach. More exactly, each document was represented as a binary sequence of 0 and 1, where 1 denotes that a certain keyword is in the document while 0 defines the opposite. The fitness value was computed using the simple Jaccard formulation described earlier in Eq. 1 indicating a custom weight for each document - higher value leads to higher similarity. Furthermore, an interesting approach is described in [3] using the Jaccard Coefficient as a polarity metric for Roman and English words after applying a clustering technique. This process, also known as JIBCA, has three major parts: training set - E-commerce reviews previously-stored -, testing set - obtained by extracting opinion words from the reviews (both roman, translated into English and English words) - and the Jaccard index - used to match the training set with the testing one -. The results have been compared with other existing approaches and it scored an impressive 7% increase in its accuracy.

After further analysis, we concluded that this similarity metric still has potential and can offer extensive information when compared to other metrics. We will follow its implementation and preliminary results in the Testing section.

3.3. K-means

K-means is an unsupervised machine learning technique - self-conducted analysis that might reveal various patterns - that finds relations among data by clustering - grouping relatable elements. Starting with randomly or arbitrarily chosen centroids - points in data -, it performs sequential calculations for centroid optimization.

We can define the procedure for K-means as follows [18]:

1. We have an input of:
   k - representing the number of clusters that should result
   D - our database
2. We randomly or arbitrarily choose k initial centroids

   REPEAT
3. We assign each point to a cluster using a proximity measure such as euclidean distance or cosine similarity.

$$Euclidean\_D = || x_i^{(j)} - c_j ||^2 \quad \text{OR} \quad Cosine\_Similarity = \frac{A * B}{||A|| * ||B||}$$

Eq. 2

4. We update the cluster mean and recalculate distances
   UNTIL no more changes to the clusters

5. We have as output k clusters

In our case, natural language, we need to map the space of words into a multidimensional vector space where distance calculations can be considered meaningful. To achieve the sentence - vector translation we can apply classical NLP techniques as BoW, TF, TF-IDF or Word Embeddings, thoroughly described in 3.1 section. However the advantage of word embedding is that they identify context-related information, so we will focus more on this approach. Another relevant aspect to take into account is the “Curse of Dimensionality” - namely the fact that large corpora like texts converted in multiple dimensional vectors can end up suffering from data sparsity issues and therefore lose statistical significance. Another disadvantage to this method is its tendency to end up in local optimum, which requires several runs to find the best solution. Nonetheless, this clustering method still presents interest to this paper as it is easily adaptable to new observations, is fast and can be used as an optimization method in our search for semantic knowledge.

A good application of this algorithm over textual materials can be seen in [19], where Huan et. al. use KL divergence - Kullback-Leibler/ relative entropy - to calculate the similarity between centroids - firstly chosen to have a broad distribution - and the sample data - translated into the classical vector space using TF-IDF for weights. The results of this approach have shown improvements not only of the precision - from around 0.6 to over 0.8 -, recall - slightly increased over 0.7 - and F-measure - from less than 0.7 to 0.8 -, but also the time of clustering and the number of iterations needed. To conduct these experiments a dataset of 4250 TanCorp texts and 1200 ChineseNewsCorp texts was used.

3.4. Cosine Similarity

Cosine similarity calculates, as suggested, the cosine between two vectors. In contrast to other vector-based techniques, cosine takes into account the vector's orientation, not relying only on their magnitude. This similarity can be defined mathematically as follows:

$$Cosine\_Similarity = \frac{A * B}{||A|| * ||B||}, \; where$$

$$A*B = \sum_{i=1}^{n} A_i * B_i \qquad and \qquad ||A|| * ||B|| = \sqrt{\sum_{i=1}^{n} A_i^{\,2} \; \sum_{i=1}^{n} B_i^{\,2}}$$

Eq. 3

In other words, cosine similarity is the division between the scalar product of the two vectors and the product of the vector's module, both defined in **Eq. 3** and its outcome is bounded in the [0,1] interval. In our case, cosine can be applied after carefully choosing a technique/embedding for computing the vectors for our texts, as discussed in the previous section and described in section 3.1. The performance of this metric is highly correlated to how the input words are translated into the numerical values.

Moreover, [21] proposes four operations that can be performed before actually computing the cosine similarity to enhance the produced outputs. For example, both vectors should be normalized to the same number of dimensions, in the case of discovering two synonyms or a hypernym-hyponym relation differing in syntax, using WordNet, only one of them should be used as dimension name in all cases. Another improvement is the recalibration of the dimensions as a product of initial similarity value and the hypernym-hyponym similarity value. In [21] these operations enhanced the resulting similarity from 0.25 to 0.875 in the case of "rodent" and "angora". Although the paper does not present a wide enough range of experiments for statistical significance, these methods should be taken into consideration for further testing.

3.5. LSI

In this section, we will present another clustering method, also known as Latent Semantic Indexing, that uses principal component analysis to lower the dimensionality of document vectors. By assuming that the underlying corpus information is contained in inferior dimensionalities, LSI accomplishes its topic retrieval goal as follows:

1. We first construct the matrix A that contains the occurrences of all unique words from the documents - texts from the database - and the query - the analysed document/text. Each document is represented by a column, each row is a unique term and each cell a_{ij} is calculated as the number of j-word occurrences in the i-document, resulting in a sparse and high dimensional matrix.
2. SVD is applied to decompose the matrix following the formula that A is equal to the product of three matrices U, S and V^T. The algorithm for SVD computation can be found in [22]

$$A=USV^T$$

Eq. 4

3. An approximation of the resulted matrices is done depending on the chosen/suggested number of dimensions (k), by keeping the k columns from U and V, and k columns and rows from S
4. Each row from the V matrix contains the new coordinates for each individual document originally in the matrix A, also known as eigenvectors.
5. To be able to apply a similarity metric between the new document vectors and the query, we need to compute the new query coordinates in k dimension. To achieve that, the new query, new_q, is the product of the transposed initial query, q^T, the matrix U in k dimensions, U_k and the inverse of the matrix S in k dimensions, S_k^{-1}

$$new\_q = q^T U_k S_k^{-1}$$

Eq. 5

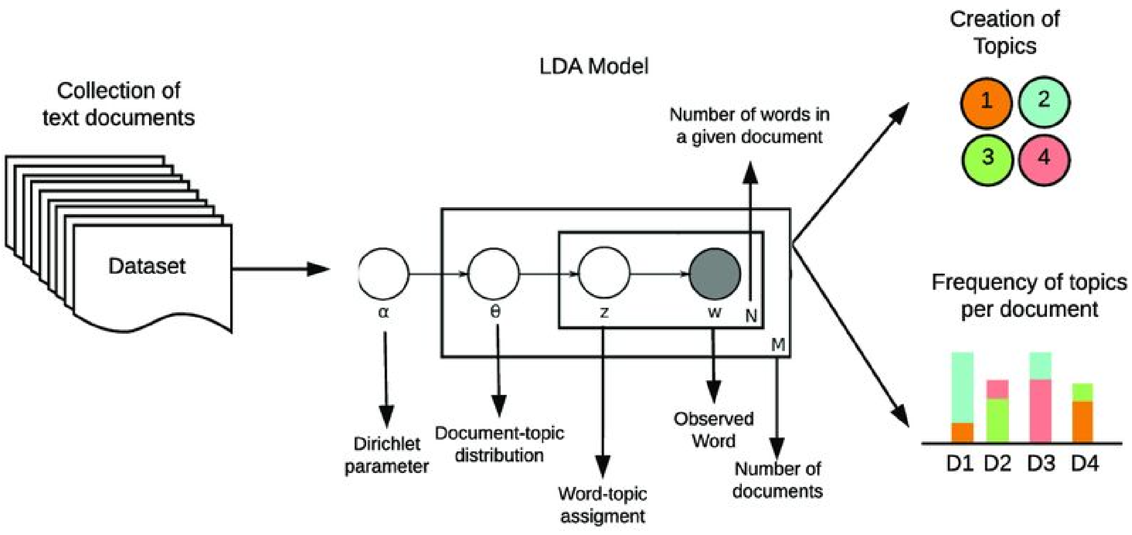
6. Now we can apply any vector-based similarity to compute the similarity between each document and the query

Following naturally from the description above, the most relevant aspects of this approach is that it is simple to explain and implement, offers better results than other vector space models and is faster compared to other dimensionality reduction methods as it involves only term matrix decomposition. Yet, LSI can only return a number of topics less or equal to the rank of the matrix. Moreover, depending on the dataset, LSI might not capture all meanings of a word.

By further analysing this method, we can find different approaches studied and applied in the past. In [23], LSI is combined with TF-IDF technique in order to mutually bypass some of their limitations when tackling the problem of finding relevant user questions from the governmental QA service using the LSI corpus to discover similar topics with the ones in the queries. Their results showed noticeable improvement from using only TF-IDF or LSI to using the combination of both. Another method is explored in [24] to reduce feature dimension in search engines for Arabic languages by using the cosine similarity measure to establish the term to document relationship instead of word occurrences functions when creating the matrix A. Fawaz et. all results have concluded that this method outperformed with 5.83% when compared to the classical LSI method even when used with highly mixed content documents as medical records. However, this method's efficiency relies on smaller datasets, as the cosine similarity used in creating A has a cost of $O(n^2)$ where n represents the number of documents included in the analysis. Tackling the same issue of time cost, [25] presents a parallel approach by standard multi-core computers. It uses multiple threads to conduct matrix computations using the Fork-Join method - terms are extracted and taken to the root form to exclude any duplications, while stop words and special characters are removed as they don't present any semantic meaning. After finishing all documents the threads are joined together in creating the matrix A.

3.6. LDA with Jensen-Shannon distance

LDA, known as Latent Dirichlet Allocation is an unsupervised three-layer Bayesian generative model that, instead of assigning topics similarly to our previous model, LSI, it constructs two dependent distributions: one concerning document's topics, following a Dirichlet prior and the other, the words concerning the topics with the polynomial distribution. The primary assumption behind this technique is the fact that in a singular document, multiple topics can coexist and in addition to that, the words within the document can contribute to multiple topics, some of them even at the same time. As an operational consideration, the number of topics to be searched for must be specified, but detailed topic mappings or categorisation is not necessary.



[26] Fig. 1 Probability generation LDA

Fig. 1 closely describes the operations conducted by LDA, by assuming that each document, denoted as M, has in its composition N number of words represented as a Dirichlet distribution with ɑ as topic weight in relation to documents. Z is the assignment of a word to a topic, while W is defined as the word taken under observation in relation to document M. Moreover, a document-topic distribution, Θ, is also taken into consideration, thus ɑ should be chosen carefully as if selected too high can lead to a homogeneous distribution of topics or the opposite can prevent the inference process. Parameters are usually chosen using a probabilistic sampling such as Gibbs sampling described in [27]. LDA can be computed as follow:

$$p(\theta, z, w \mid \alpha, \beta) = p(\theta \mid \alpha) \prod_{n=1}^{N} p(z_n \mid \theta)\, p(w_n \mid z_n , \beta)$$

where β is the probability distribution of all hidden topics

Eq. 6

However, our goal is to find similar documents in our statistical corpus - topic distribution. To achieve that, we need to use a symmetric metric, as commutativity is required in our case - the same similarity between A - B and B - A. Jensen-Shannon (JS) computes the distance between two documents by comparing the divergence of the distributions. Its formula is based on KL divergence described in [19]. Jensen-Shannon distance can be computed as follows:

$$D_{JS}(P||Q) = \frac{1}{2} D_{KL}(P||M) + \frac{1}{2} D_{KL}(Q||M)$$

where P, Q are discrete distributions and D_{KL} is the KL divergence

Eq. 7

An improvement of this technique can be observed in [27] where M. Shao and L. Qin introduce word co-occurrences analysis to deal with the disadvantage of JS distance as it cannot differentiate semantic relations among text topics. By comparing classical LDA with JS and LDA with JS plus word co-occurrence, results have shown a slight improvement for the latter in all three metrics: precision, recall and F-metric. Even though the improvements are in the order of decimals, this method effectively reduces problems such as synonymy or polysemy, thus making it a valid optimization.

3.7. Word Mover's Distance

The novelty in Word Mover's distance, known as WMD, is related to its approach of leveraging the results of advanced embeddings like word2vec or Glove, described in section 3.1, thus generating a more qualitative word - vector translation. This method deals with both syntactic and semantic problems by treating the area resulting from the embedded document as a weighted cloud and the distance between two texts is defined as the minimum amount of distance needed to "travel" from embedded text A to embedded text B [28]. Moreover, WMD can be formulated as a problem of the transportation problem, Earth Mover's Distance (EMD), explained in [29]. We can formulate WMD as following the EMD formula - minimum amount of work needed to convert P to Q:

$$\mathrm{EMD(P, Q)} = \min \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} GD(p_i, q_j)}{\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij}}$$

where GD(p, q) is the distance between p and q,
F = {f_{ij}} is an acceptable flow and P,
Q have different weights

Eq. 8

WMD has developed from an optimisation problem of EMD, by trying to solve a concept known as **flow**. Each word receives a flow of 1/N, where N is the number of words from a sentence. In a more precise way, the aim of this technique is to minimise the difference between average flows - distributions - of two sentences: the minimum of the sum of products of all the possible permutations of flows and their Euclidean distance. This can be mathematically written as follows:

$$\min \sum_{i,j=1}^{n} T_{ij} * c(i, j), \textit{ where } T \geq 0$$

$$\textit{Subject to: } \sum_{i=1}^{n} T_{ij} = d_j \textit{ and } \sum_{j=1}^{n} T_{ij} = d_i$$

$$\textit{with } \mathrm{d}_i = \frac{c_i}{\sum_{j=1}^{n} c_j}, \textit{ where } c_i \textit{ frequency of word i in a document}$$

Eq. 9

The main advantages of this approach are that it does not require hyperparameter tuning, therefore having a greater degree of understandability and has access to the encoded in the techniques like word2vec or Glove. From a result based perspective, WMD outperforms other consecrated task classification distances, as it is the first to link high-quality embeddings with the EMD problem space.

Introduced recently, this approach has rapidly gained popularity, as enough research and optimisations have already been proposed. In [30], WMD was used to detect contextual emotion, a task extensively explored in NLP and sentiment analysis. To achieve this task, Alshahrani et. all adapt both word2vec and WMD methods to calculate the distance between headlines and words from WordNet-Affect dictionary. The minimum cumulative distance from the words in the headline to matching the emotion words is chosen as the final distance. The results are optimal for all categories of emotions. We should keep this approach in mind as a solution to our sentiment analysis. Another approach that tries to enhance the impact of syntactic information by using Part-of-Speech technique can be found in [31]. The main idea of the paper is that if the same words occur in two sentences, but incorporating different syntactic meaning, the distance should take it into account. Their improved model has been tested with four different datasets and all results showed an improvement from original WMD.

Finally, the WMD method seems to have a lot of potential for our research, given its so advertised results.

3.8. Siamese Manhattan LSTM (MaLSTM)

The Siamese approach differs from other network-based techniques by the fact that it uses two identical subnetwork structures. This method has previously successfully been used to perform forged signatures recognition, sentence semantic similarity, image recognition, object tracking and many more. The Manhattan part from the name defines the distance that through observations along the years has outperformed others as Euclidean or Cosine distances. Siamese LSTM is an adaptation of the Long

Short-Term Memory network, RNN used for labelled data in deep learning. LSTM is used to capture long-term dependencies, sequential data modelling and to prevent the vanishing gradient problem - limitation of classical RNNs.

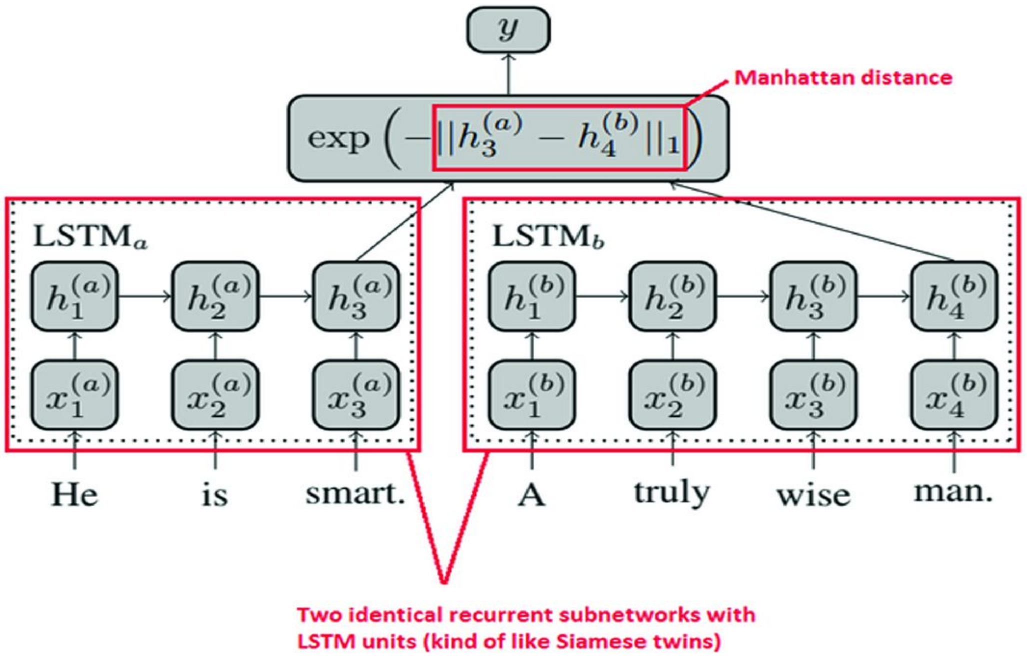
The Manhattan similarity function can be mathematically described as follows:

$$Manhattan\_Similarity = exp(-\| h^{(a)} - h^{(b)} \|_1)$$

where, in our case, $h^{(a)}$ and $h^{(b)}$ are two word-vector representations -hidden states - that contain the underlying meaning of a sentence, each from one subnetwork of the siamese architecture

Eq. 10

Fig. 2 offers an illustrative example of the Siamese Manhattan LSTM architecture. The two sentences are each assigned to one of the two recurrent subnetworks from the siamese model, data is projected into the hidden layers, as similar items are grouped together while the divergent ones are dispersed. The similarity measure is done using the Manhattan distance, described in **Eq. 10**, between the final hidden states of the LSTM layers.



[36] Fig. 2 MaLSTM model architecture

This method has been successfully used in different scenarios like in [37] for question retrieval both in Arabic and English languages using real-world Yahoo! answers dataset, [36] for student tests grading and sentence similarity [38].

3.9. Knowledge Based

By using sentiment networks like WordNet - for English words -, knowledge measures establish the semantic similarity between texts. WordNet - for English words - is constructed as a large graph with syntactic groups - verbs, adverbs, adjectives etc - of distinct cognitive synsets, which offers great advantages for computational processes, thus being the most popular. In this area, several metrics have developed:

1. Leacock & Chodorow [39]:

$$Leacock\_Chodorow\_Similarity = -\log \frac{length}{2 * D}$$

where length is the shortest path between two concept - nodes - and D maximum depth of the taxonomy

Eq. 11

2. Lesk - algorithm explained in [40]
3. Wu & Palmer [41]:

$$Wu\_Palmer\_Similarity = \frac{2 * depth(least{-}common{-}subsumer)}{depth(concept_1) + depth(concept_2)}$$

Eq. 12

4. Resnik [42]:

Resnik = IC(least-common-subsumer)

where IC(concept) = -log P(concept)

where P(c) - probability of finding a concept in a corpus and IC - information content

Eq. 13

5. Lin [43]:

$$Lin\_Similarity = \frac{2 * IC(least{-}common{-}subsumer)}{IC(concept_1) + IC(concept_2)}$$

Eq. 14

6. Jiang & Conrath [44]:

$$Jiang\_Conrath\_Similarity = \frac{1}{IC(concept_1) + IC(concept_2) - 2 * IC(least{-}common{-}subsumer)}$$

Eq. 15

Mihalcea has compared the above knowledge-based measures and other corpus-based measures - Pointwise Mutual Information and LSA - in [45] and the results suggested that the best approach, the one that offers the best accuracy at 70.3% and F-measure of 81.3%, is when the knowledge-based metrics are actually combined together using **Eq. 15**, with the note that the comparison is done word-to-word, only in the same classes of speech and the result is averaged. Moreover, compared to vector-based approaches, the inquired metrics have outperformed considerably in finding similarities.

$$\text{combined\_sim}(T_1, T_2) = \frac{1}{2}\left(\frac{\sum_{w \in \{T_1\}} (maxSim(w, T_2) * idf(w))}{\sum_{w \in \{T_1\}} idf(w)} + \frac{\sum_{w \in \{T_2\}} (maxSim(w, T_1) * idf(w))}{\sum_{w \in \{T_2\}} idf(w)}\right)$$

where T_1 and T_2 are the input segments
and maxSim is calculated by finding the maximum similarity result
from the described measures

Eq. 15

4. Specifications, Design & Implementation

This section presents the architecture, our class diagrams and the implementation of our solution, taking into account our main goal of this paper: a comparative analysis.

4.1. Contributions

This part contains our main contributions for this paper:

1. Data gathered from a personal Twitter account following a list of political and economical entities
2. Analysis of JSON packets received and feature selection
3. Implementation of a script to clean the necessary data and collect it into one dataset
4. Implementation of scripts for specific data preparation and tokenization for the learning phase: removal of unwanted characters, lowercasing, noise removal like punctuation, stop words removal, normalization, lemmatization and tokenization
5. Implementation of word vectorization methods/embeddings: term frequency (TF), term frequency-inverse document frequency (TF-IDF), word2vec with convolutional bag of words (CBOW) and skip-gram (SG), bidirectional encoder representations from transformers (BERT) and global vectors for word representation (GloVe)
6. Implementation of clustering methods: k-means, latent semantic indexing (LSI), latent Dirichlet allocation (LDA)
7. Implementation of neural network-based model: siamese network architecture with long short term memory networks (LSTM)
8. Implementation of Support Vector Machine for sentiment analysis using Amazon review sentiment dataset
9. Implementation of distances to compute the similarity: Jaccard distance, cosine similarity, Manhattan distance, Jensen Shannon distance, word mover’s distance and combined distance between Leacock & Chodorow, Lesk, Lin, Wu & Palmer, Jiang & Conrath
10. Component integration, testing and analysis over our gathered data
11. Performance testing
12. Identification of improvements arose from the problems experienced during the implementation

4.2. Architecture

This subchapter proposes an activity diagram with all the operational phases for our system. This architecture can be seen in Fig. 3.

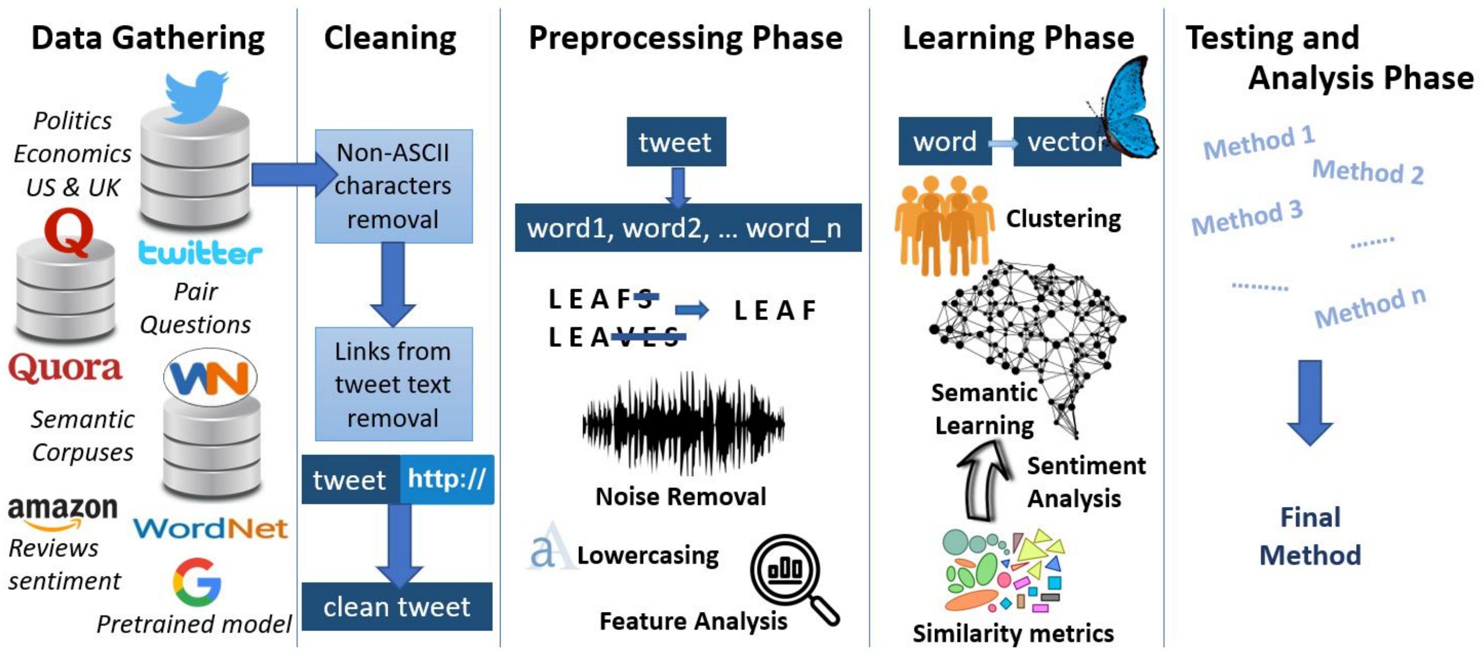


Fig. 3 Operational diagram

We can split our system into five main phases: data gathering, cleaning, preprocessing, learning and analysis. We will further discuss each individual phase focusing on their specifications and objectives.

A. Data Gathering Phase
This phase's main goal is to deliver the required datasets for our analysis, suited for this paper's target - to study direct speech communication. It is extremely important that the collected datasets can offer a glimpse in the colloquial conversations consisting in challenging unstructured data - slangs, misspelling, abbreviations, typos, technical, emoticons etc. To achieve this, Twitter became the best option for the main dataset from all social media platforms, as for its construction as a microblogging environment. As this paper's main target is to find similarities in this medium we reserved a full chapter to describe the implementation and presentation of this dataset and can be found further in this paper in section 5.

However, a complete translation human-computer is difficult to be achieved, requiring extensively annotated datasets for model training and for semantic significance. With this intent, our system uses Quora pair questions, WordNet dataset with Brown and SemCor corpora, Amazon and Google vectors datasets. The former can be found on Kaggle [46] and consists of 404,290 question pairs similar in the meaning used for training models. The questions were collected from Quora's Q&A platform and involved in a Kaggle competition to predict similar questions in order to group questions that might have the same answer. Each sample has the following fields: id - unique id for each pair -, qid1 and qid2 - the ids for each question -, question1 and question2 - the text of each question - and is_duplicate field that offers information about the similarity between the pair questions. As this dataset is constructed over, to an extent, similar environment as twitter we can use it in our system as a training set - learning how to recognise similar blocks of texts - for our neural network model, which requires a strict structure of input - features and predictor -, part of which our twitter dataset misses - the predictor.

Furthermore, we use the WordNet [47] corpus to add semantic meaning to our tweets as words that have nothing in common syntactically can have the same sense and vice versa. WordNet contains 155,327 English words organised as synsets - semantic relationships as synonyms, hyponyms, meronyms, antonyms etc. - and it has a hierarchical structure under the form of "IS A". Moreover, it uses part of speech (POS) to group words under four sub-nets: nouns, verbs, adjectives and adverbs with interchangeable relations. Our system also uses two statistical corpora: SemCor - semantically annotated texts by mapping WordNet 3.0 - and Brown - a collection of American text samples varied in genres.

Amazon dataset has 1000 positive and 1000 negative tagged product reviews with a variable length of four sentences. Even though they are slightly different from a regular tweet we consider them suitable for training our SVM sentiment classification model because of the similar length and writing style.

Lastly, the Google dataset contains a pre-trained corpus with more than 3 billion of 300-dimension vectors of commonly used English words in the news industry. The pre-trained dataset offers an underlying level of knowledge that might become useful in semantic support.

B. Cleaning Phase

The objective of this phase is to return a dataset containing only interpretable words in English. Even though we realise that textual artefacts such as emojis, links and non-ASCII characters can provide us with additional information regarding the underlying text, the process of interpretation of such artefacts require additional metrics that are not the purpose of this paper. A relevant example of understanding such complex textual artefacts by means of image recognition, topic extraction and clustering can be seen in [48]. However, such approaches haven't been fully integrated into traditional text processing techniques. Thus, we decided to remove all non-ASCII characters and links found in the text, using regex, and constructed a smaller dataset containing only the id and the text for further analysis, while still keeping other information such as user, likes, followers etc. in the original 84 feature dataset - constructed by combining all JSON structures received via the API into a relational structure type and saved as CSV - for both presentational and analytical reasons that can be done in the future.

C. Preprocessing Phase

This phase continues the previous part as it further prepares the data for analytical purposes as each model/distance requires different types of inputs. In the initial analysis, we decided to remove most of the metadata received via Twitter API as each tweet came with information and links towards resources used for both the post and the user. Even though this data can be extremely useful, this initial research focuses only on the textual part. We kept the complete primary dataset intact as features such as likes, shares and followers can offer underlying information about the impact of a tweet over the community and analyse its spreadability. Further, we have implemented some traditional NLP techniques for preprocessing as *lowercasing* - machines make a distinction between the upper casing and lower casing thus, identical words represent different encodings; this behaviour constraints us in converting all letters to only one type of casing, in our case and usually used, lowercasing; *noise removal* - as punctuation doesn't present any valuable or extra information we implemented a function to remove any such characters; *lemmatization* - as per our example from the diagram, we can observe that words that might have a different suffix can have the same base, thus meaning and by using it we can also recognise misspellings and reduce the words to their root form; we used WordNet to achieve lemmatization; *tokenization* - splitting each sentence into a list of words; this method is not always necessary and will be used only when required by a model/distance.

D. Learning Phase

This phase is split into four main parts and combinations of these will be further analysed and compared in the next section. We have firstly constructed six possible word embeddings - TF, TF-IDF, word2vec with CBOW and SG, BERT and GloVe - required as input for model and distance applications later described. The next part contains three clustering or topic modelling algorithms - k-means, LSI and LDA - for optimization purposes by further reducing the search space - looking for similar texts only in the predicted cluster/theme. The third main part builds two identical networks trained by using google pre-trained word2vec dataset to enhance the semantic sense by adopting a technique called siamese with LSTM as networks. The fourth and last part of the learning phase is represented by the actual distances used to compare either representation of vectors or directly the texts. A more detailed explanation can be found in the Implementation subchapter (4.3).

E. Analysis phase

This phase consists in the comparison of combinations of preprocessing, clustering, neural networks and distances techniques described earlier to find the most suited scheme for tweets analysis. It is of utmost importance to set up 2 comparison criteria: retrieval accuracy and computing time given the size of the input database. More information about this can be found in Chapter 8.

4.3. Implementation

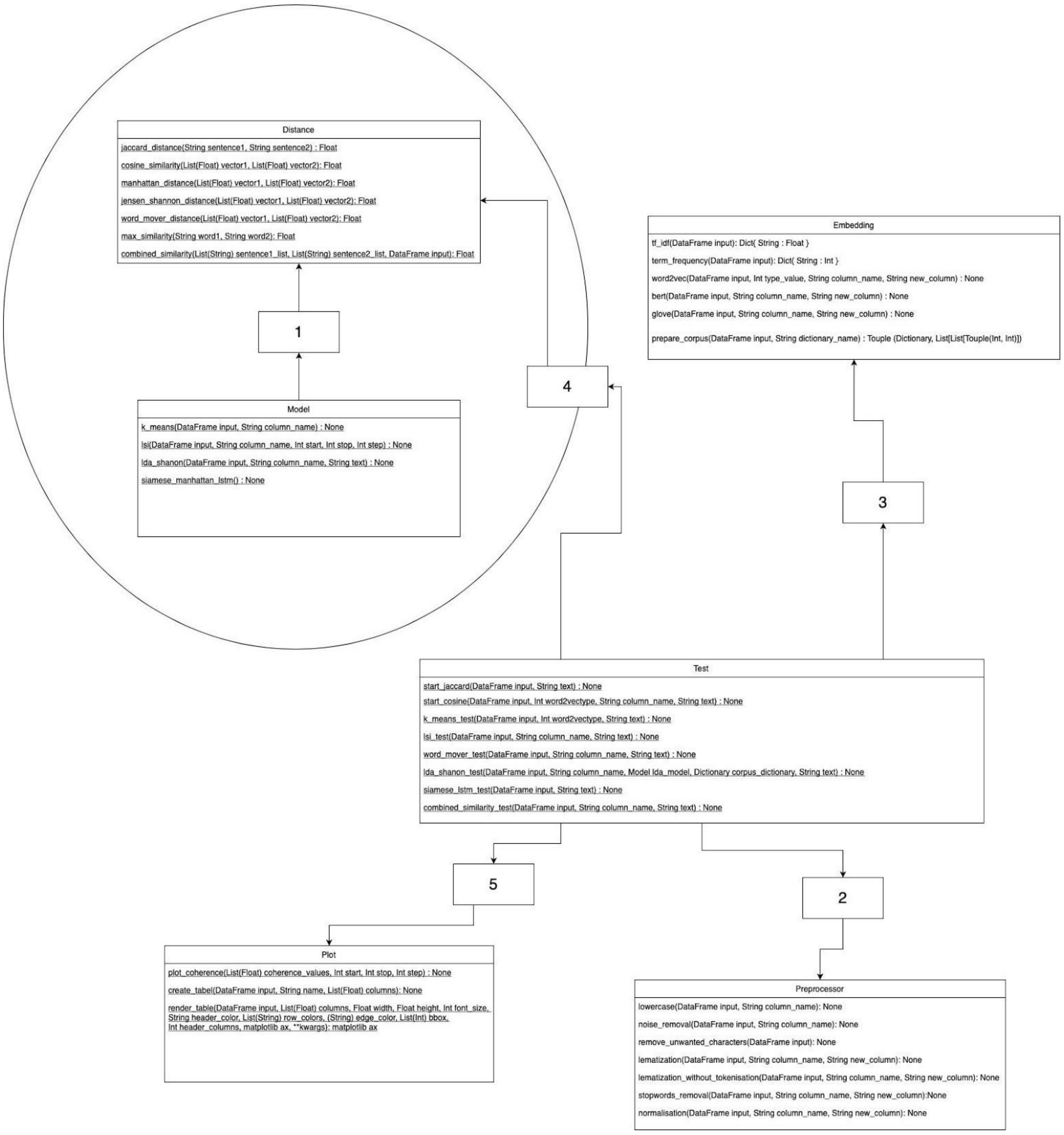

Distance
jaccard_distance(String sentence1, String sentence2) : Float
cosine_similarity(List(Float) vector1, List(Float) vector2): Float
manhattan_distance(List(Float) vector1, List(Float) vector2): Float
jensen_shannon_distance(List(Float) vector1, List(Float) vector2): Float
word_mover_distance(List(Float) vector1, List(Float) vector2): Float
max_similarity(String word1, String word2): Float
combined_similarity(List(String) sentence1_list, List(String) sentence2_list, DataFrame input): Float
1
Model
k_means(DataFrame input, String column_name) : None
lsi(DataFrame input, String column_name, Int start, Int stop, Int step) : None
lda_shanon(DataFrame input, String column_name, String text) : None
siamese_manhattan_lstm() : None
4
Embedding
tf_idf(DataFrame input): Dict{ String : Float }
term_frequency(DataFrame input): Dict{ String : Int }
word2vec(DataFrame input, Int type_value, String column_name, String new_column) : None
bert(DataFrame input, String column_name, String new_column) : None
glove(DataFrame input, String column_name, String new_column) : None
prepare_corpus(DataFrame input, String dictionary_name) : Touple (Dictionary, List[List[Touple(Int, Int)]])
3
Test
start_jaccard(DataFrame input, String text) : None
start_cosine(DataFrame input, Int word2vectype, String column_name, String text) : None
k_means_test(DataFrame input, Int word2vectype, String text) : None
lsi_test(DataFrame input, String column_name, String text) : None
word_mover_test(DataFrame input, String column_name, String text) : None
lda_shanon_test(DataFrame input, String column_name, Model lda_model, Dictionary corpus_dictionary, String text) : None
siamese_lstm_test(DataFrame input, String text) : None
combined_similarity_test(DataFrame input, String column_name, String text) : None
5
2
Plot
plot_coherence(List(Float) coherence_values, Int start, Int stop, Int step) : None
create_tabel(DataFrame input, String name, List(Float) columns): None
render_table(DataFrame input, List(Float) columns, Float width, Float height, Int font_size, String header_color, List(String) row_colors, (String) edge_color, List(Int) bbox, Int header_columns, matplotlib ax, **kwargs): matplotlib ax
Preprocessor
lowercase(DataFrame input, String column_name): None
noise_removal(DataFrame input, String column_name): None
remove_unwanted_characters(DataFrame input): None
lematization(DataFrame input, String column_name, String new_column): None
lematization_without_tokenisation(DataFrame input, String column_name, String new_column): None
stopwords_removal(DataFrame input, String column_name, String new_column):None
normalisation(DataFrame input, String column_name, String new_column): None

Fig. 4 Workflow diagram

Before starting this subchapter we would like to define our solution skeleton as follows and in each section, we will discuss the technicalities for each procedure:

```
INPUT DATA: type
        extra corpuses
        preprocessing
        embeddings
        machine-learning model
        distance
OUTPUT + meaning
```

Fig. 4 presents the workflow described earlier from the implementation point of view and in this subchapter we will focus on some technical specifications, the actual implementation and data structures used. As our intention is to be able to use each preprocessing, embedding, distance with any machine learning model, combine them and determine the most suited form for our dataset, we opted for a categorical organization of the procedures and functions used. However, in practice, the final options can be reorganized in a class-oriented approach. For the implementation of the system, we used Python 3.7. and chose to work on pandas DataFrame structure as it is extremely efficient when working on large datasets but also for analysis tasks as it has a simplistic data representation which is essential in data analysis for better results. Moreover, it has built-in functions that facilitate different necessary operations over the data like filtering, segmenting, segregating or even customizing it. All computations from our system are done over the DataFrame, by constructing or editing existing columns.

A. Distance: this script contains the compared metrics that compute the level of similarity between two sentences: Jaccard, Cosine, Manhattan, Jensen Shannon, Word Mover's Distance and the combined similarity between five similarity metrics described in section **3.10**.
It is important to note that in the following two tables the distances are bounded to such small intervals because of the normalization process that previously happened.

	Jaccard distance	*Cosine distance*	*Word Mover's distance*
Interval results	[0, 1]	[-1, 1]	[0, 1]
Similarity value	higher value => higher similarity	higher value => higher similarity	lower value => higher similarity
Extra corpuses	None	None	google word2vec pre-trained model

Libraries used	None	NumPy	None

	Manhattan distance	*Combined similarities*	*Jensen Shannon*
Interval results	[0, 1]	[0, 1]	[0, ∞]
Similarity value	higher value => higher similarity	higher value => higher similarity	higher value => higher similarity
Extra corpuses	None	WordNet with Brown and Semcor corpuses	None
Libraries used	Keras	WordNet	SciPy

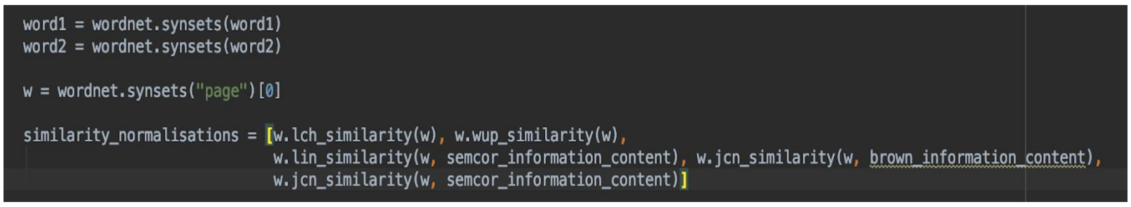
Fig. 5 Distances table

As some of these metrics implementation is trivial and already explained mathematically in **chapter 3**, we will focus on the more intriguing ones.

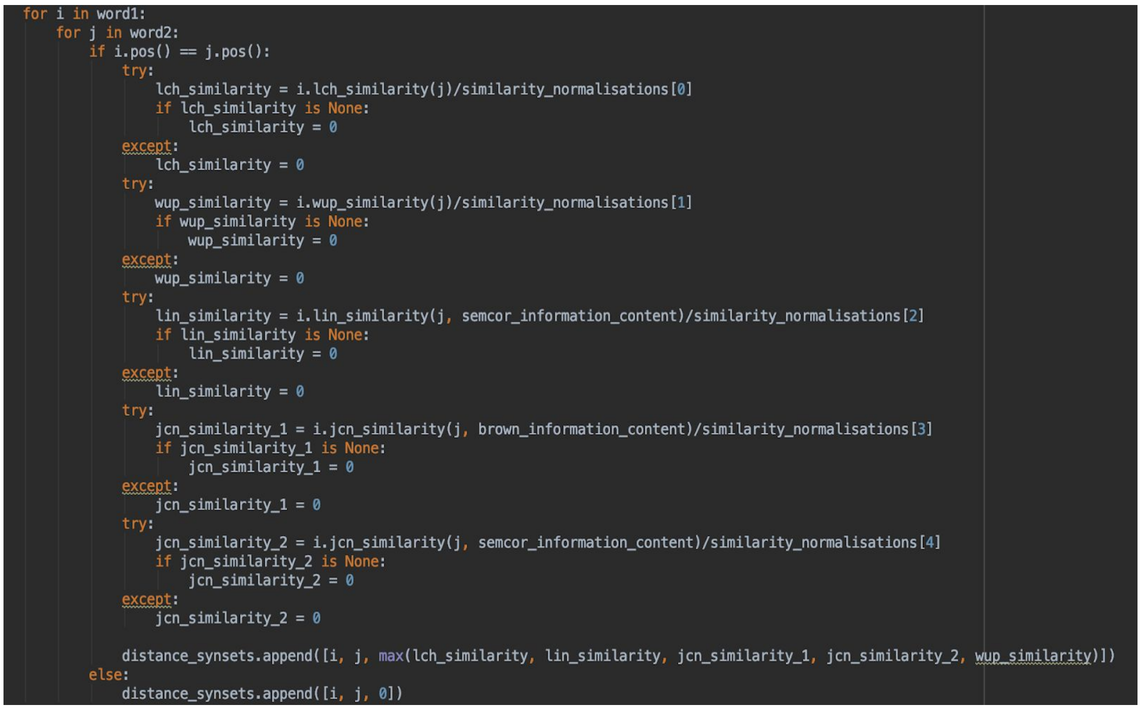
The *Word Mover's* function receives as parameters two sentences and we used a pre-trained model from google - word2vec-google-news-300" with its built-in command - wmdistance - to compute the spatial distance. This approach allowed us to meaningfully compare documents even though they don't share any common words, by finding the minimum distance from a point A to B, where the points are represented by words from each sentence. Lower the distance, the more similar the sentences are. For better results, we normalized the vectors from the corpus to ensure that they have the same length.

A quite interesting method follows a greedy approach of choosing the maximum similarity between the six metrics described previously in section **3.10.** This method was firstly described by Mihalcea [45] in 2006 by comparing the six metrics and additional techniques with each other. The results showed increased accuracy for the combination of the six metrics. We decided to follow Mihalcea's solution as follows:

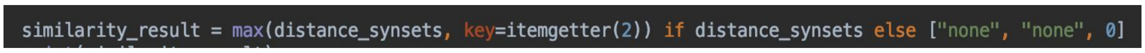
- ➢ it is a pairwise comparison of all the possible synsets - in the WordNet sense - of the linear combination of the words in the two texts;
- ➢ in *max_similarity function* [Fig. 4] we compute for each "word - word" combination, all measures and choose the maximum of them; we should note that words can have different parts of speech but keep the meaning, thus we decided to compare all possible synsets of a word to their correspondent part of speech from the second;



```
word1 = wordnet.synsets(word1)
word2 = wordnet.synsets(word2)

w = wordnet.synsets("page")[0]

similarity_normalisations = [w.lch_similarity(w), w.wup_similarity(w),
                             w.lin_similarity(w, semcor_information_content), w.jcn_similarity(w, brown_information_content),
                             w.jcn_similarity(w, semcor_information_content)]
```



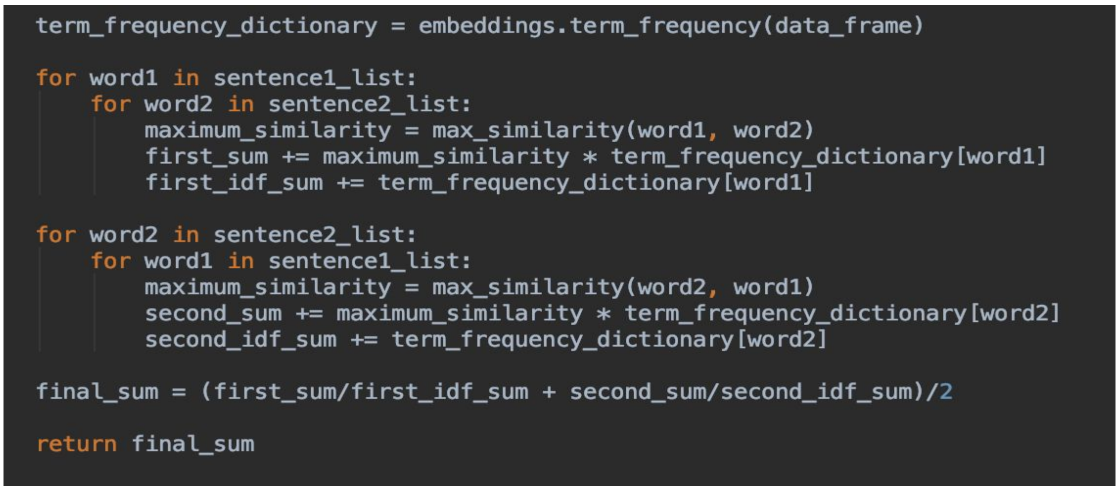
```
for i in word1:
    for j in word2:
        if i.pos() == j.pos():
            try:
                lch_similarity = i.lch_similarity(j)/similarity_normalisations[0]
                if lch_similarity is None:
                    lch_similarity = 0
            except:
                lch_similarity = 0
            try:
                wup_similarity = i.wup_similarity(j)/similarity_normalisations[1]
                if wup_similarity is None:
                    wup_similarity = 0
            except:
                wup_similarity = 0
            try:
                lin_similarity = i.lin_similarity(j, semcor_information_content)/similarity_normalisations[2]
                if lin_similarity is None:
                    lin_similarity = 0
            except:
                lin_similarity = 0
            try:
                jcn_similarity_1 = i.jcn_similarity(j, brown_information_content)/similarity_normalisations[3]
                if jcn_similarity_1 is None:
                    jcn_similarity_1 = 0
            except:
                jcn_similarity_1 = 0
            try:
                jcn_similarity_2 = i.jcn_similarity(j, semcor_information_content)/similarity_normalisations[4]
                if jcn_similarity_2 is None:
                    jcn_similarity_2 = 0
            except:
                jcn_similarity_2 = 0

            distance_synsets.append([i, j, max(lch_similarity, lin_similarity, jcn_similarity_1, jcn_similarity_2, wup_similarity)])
        else:
            distance_synsets.append([i, j, 0])
```



```
similarity_result = max(distance_synsets, key=itemgetter(2)) if distance_synsets else ["none", "none", 0]
```

Fig. 6 Max Similarity Function

- ➢ in *combined similarity function* [Fig. 6] we build the final result that measures the degree of similarity between the two texts by summing the sum of the maximum similarity of each word from the first text - calculated in *max_similarity function* - multiplied by the word's IDF score and divided by the sum of the IDF scores with the sum of the maximum similarity of each word from the second text - calculated in *max_similarity function* - multiplied by the word's IDF score and divided by the sum of the IDF scores; in the end we divide the sum with 2 and retrieve the final value [Fig, 7]. This equation is mathematically described in **Eq. 15**
- ➢ due to normalization issues, we had to relinquish the Resnik measure.



```
term_frequency_dictionary = embeddings.term_frequency(data_frame)

for word1 in sentence1_list:
    for word2 in sentence2_list:
        maximum_similarity = max_similarity(word1, word2)
        first_sum += maximum_similarity * term_frequency_dictionary[word1]
        first_idf_sum += term_frequency_dictionary[word1]

for word2 in sentence2_list:
    for word1 in sentence1_list:
        maximum_similarity = max_similarity(word2, word1)
        second_sum += maximum_similarity * term_frequency_dictionary[word2]
        second_idf_sum += term_frequency_dictionary[word2]

final_sum = (first_sum/first_idf_sum + second_sum/second_idf_sum)/2

return final_sum
```

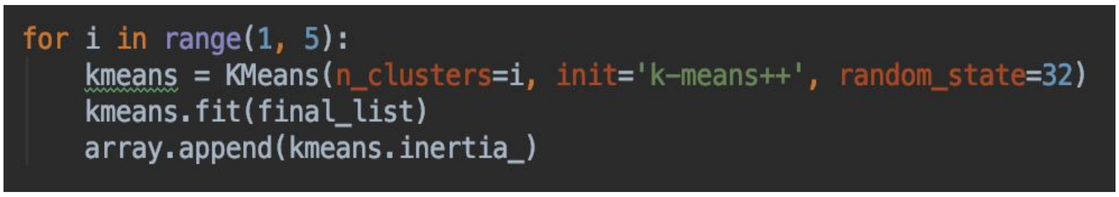
Fig. 7 Combined similarly

B. Model: the Model section consists of three clustering and topic formation algorithms, a sentiment approach via SVM and one neural network.

	K-means	LSI	LDA	Siamese LSTM	SVM
ML Type	clustering	topic modelling	topic modelling	neural network	classification
Extra corpuses	None	None	None	Quora Question, GoogleNews	Amazon reviews sentiment
Libraries	Sklearn	Gensim	Gensim	TensorFlow, Keras	Sklearn
INPUT type	vectors	tokenized text	tokenized text	text	text

Fig. 8 Models table

K-means starts from a previously given number of clusters k and initializes its centroids by shuffling the dataset and choosing randomly the k data points. Further, it computes the sum of the squared distance using Euclidean distance between the data points and all centroids for data reassignation until no changes are necessary. A more detailed explanation can be found in section **3.3**. It is critical for k-means that the number of clusters used for training is chosen optimally, thus we have used the "elbow" technique in our implementation to find the best k for the given dataset [Fig 9]. This method is based on compiling the model with various numbers of clusters and then deciding the optimal one by choosing the inflection point of the fitting curve as at that point the model fits the best. This is shown in Fig. 10 - the point is projected on the 2 clusters band.



```
for i in range(1, 5):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=32)
    kmeans.fit(final_list)
    array.append(kmeans.inertia_)
```

Fig. 9 K-means tuning

This works in practice because the inflection point determines a change of curvature sign: initially, it's clear that by choosing small numbers of clusters, the curve will go in a definite direction; it is in the moment that the direction changes that the model starts losing accuracy. Moreover, given the fact that k-means tends to converge to a local minimum, it is extremely important to choose the first k centroids in a more reliable way than randomly, thus we used in our implementation a scikit-learn initialization scheme called k-means++ to choose the first k clusters to be relatively far away from each other. This has proved to offer better results than a random approach. Additionally, to ensure that the results are reproducible and the result from the "elbow" technique will match the final training we set up the seed used in the random number generator as a RandomState instance. Finally, the model was saved in the "Models" to be directly loaded when needed.

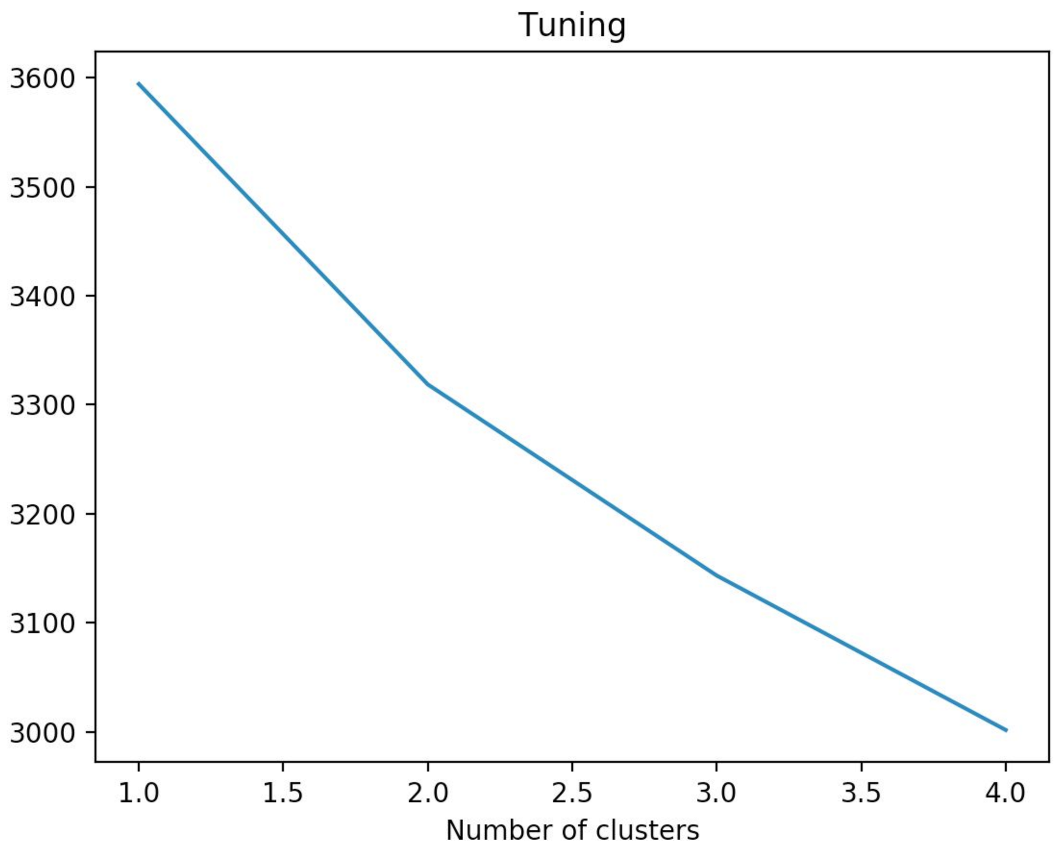
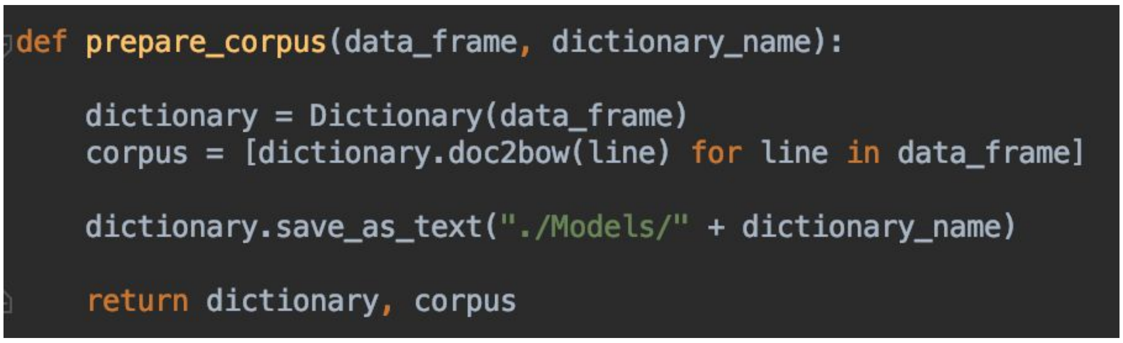


Fig. 10 Tuning plot with Elbow Technique

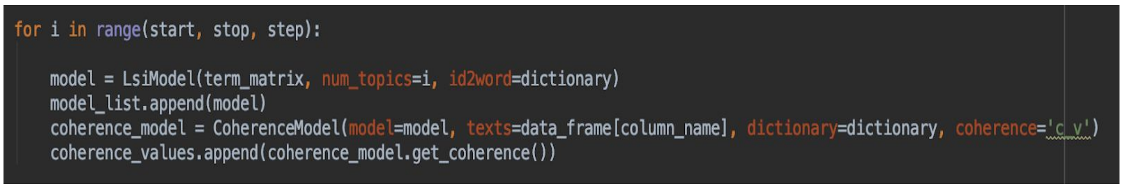
LSI, also known as ***LSA***, reduces the dimensionality of the dataset by applying SVD - singular value decomposition - and choosing the most significant t dimensions, thus discovering hidden themes between documents/texts. After, similarity measures can be applied to compute distances between blocks of texts in the new dimension space. The first step in implementing this method was to generate two essential parts of the model: the document topic matrix and the term-document matrix - using Bag of Words technique so that the rows would represent the terms and the columns, the documents. The two matrices are created in the *prepare_corpus function* from *embeddings* using our tokenized tweets dataset [Fig. 11].



```python
def prepare_corpus(data_frame, dictionary_name):

    dictionary = Dictionary(data_frame)
    corpus = [dictionary.doc2bow(line) for line in data_frame]

    dictionary.save_as_text("./Models/" + dictionary_name)

    return dictionary, corpus
```

Fig. 11 Matrix preparation

Similar to k-means, the exact number of topics needs to be given and to choose the optimal one we have built a coherence model that averages the pairwise word similarity in a topic - the highest score is considered to be optimal where the underlying model fits best the dataset [Fig. 12].



```python
for i in range(start, stop, step):

    model = LsiModel(term_matrix, num_topics=i, id2word=dictionary)
    model_list.append(model)
    coherence_model = CoherenceModel(model=model, texts=data_frame[column_name], dictionary=dictionary, coherence='c_v')
    coherence_values.append(coherence_model.get_coherence())
```

Fig. 12 Tuning with Coherence Matrix

For the coherence metric we have used *C_V* - essentially, c_v extracts changes in the frequency of occurrence of words in a movable word interval and then constructs vectors of NPMI (normalized pointwise mutual information) measures of the top words relating to each other producing the final coherence value as an arithmetic means of cosine similarities between each of these vectors and the whole text. The result can be seen in Fig. 13, concluding on 7 topics. Finally, we train and save the model for later use, in the "Models" folder.

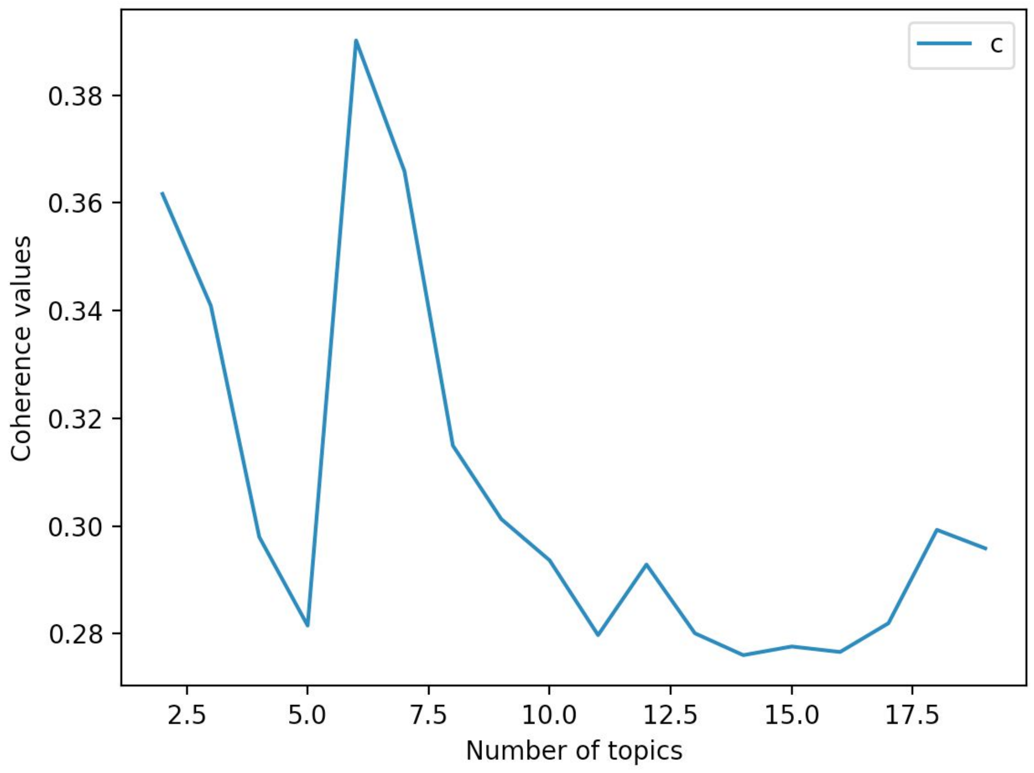
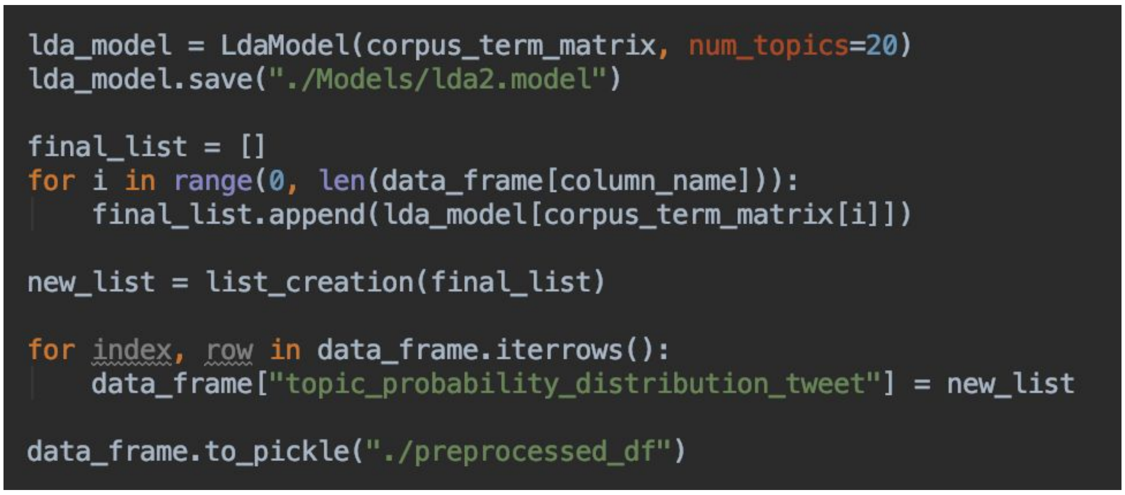


Fig. 13 Coherence plot

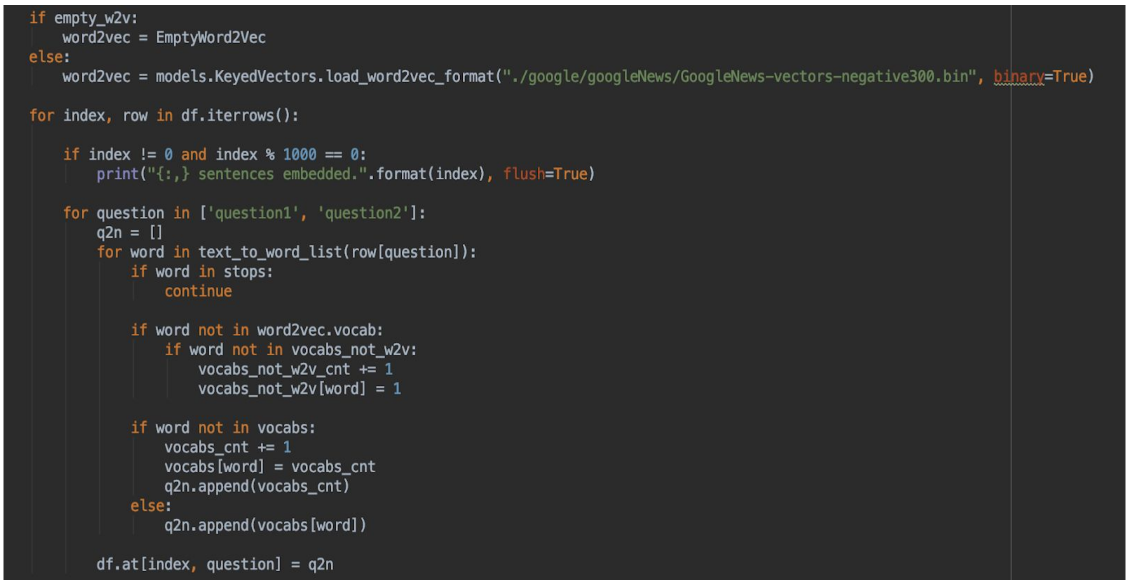
LDA uses a similar input data as LSI: a document topic matrix and a term-document matrix that were built with the same function - *prepare_corpus* and the same Bag of Words technique. The main goal of the LDA algorithm is to improve the two matrices topic distribution by means of sampling techniques. More exactly, LDA tries to adjust the topic for each word in a document with a probability. More details can be found in section **3.6.** After training, our dataset has assigned topic probabilities for each block of text which will be used to find the similarity with a new document using Jensen Shannon distance that compares the divergence of the distributions. Finally, we have predicted our dataset's topic distributions [Fig. 14] and the model was also saved in the "Models" folder.



```
lda_model = LdaModel(corpus_term_matrix, num_topics=20)
lda_model.save("./Models/lda2.model")

final_list = []
for i in range(0, len(data_frame[column_name])):
    final_list.append(lda_model[corpus_term_matrix[i]])

new_list = list_creation(final_list)

for index, row in data_frame.iterrows():
    data_frame["topic_probability_distribution_tweet"] = new_list

data_frame.to_pickle("./preprocessed_df")
```

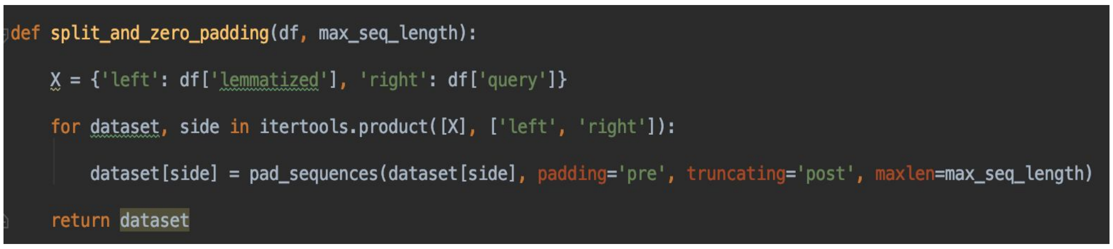
Fig. 14 Topic distribution

Next, we have created a **Siamese Network Architecture** by using two identical **LSTM networks** - LSTM has been proven to not suffer from the usual RNN vanishing gradient problem and handles variable amounts of data - and trained using Quora pair questions. The training set is organised as a set of similar questions or not which suited the best to the siamese architecture. The GoogleNews dataset was used in the word to vector embedding discussed in **4.2.** phase **A**. Before initializing the neural networks we firstly need to clean the Quora dataset and create for each sentence a list of indexes - each word receives the corresponding index from the vocabulary - using the pre-trained Google News Vectors for semantic meaning [Fig. 14]. The resulting lists need to be padded in order to have the same length [Fig. 15].



```
if empty_w2v:
    word2vec = EmptyWord2Vec
else:
    word2vec = models.KeyedVectors.load_word2vec_format("./google/googleNews/GoogleNews-vectors-negative300.bin", binary=True)

for index, row in df.iterrows():

    if index != 0 and index % 1000 == 0:
        print("{:,} sentences embedded.".format(index), flush=True)

    for question in ['question1', 'question2']:
        q2n = []
        for word in text_to_word_list(row[question]):
            if word in stops:
                continue

            if word not in word2vec.vocab:
                if word not in vocabs_not_w2v:
                    vocabs_not_w2v_cnt += 1
                    vocabs_not_w2v[word] = 1

            if word not in vocabs:
                vocabs_cnt += 1
                vocabs[word] = vocabs_cnt
                q2n.append(vocabs_cnt)
            else:
                q2n.append(vocabs[word])

        df.at[index, question] = q2n
```

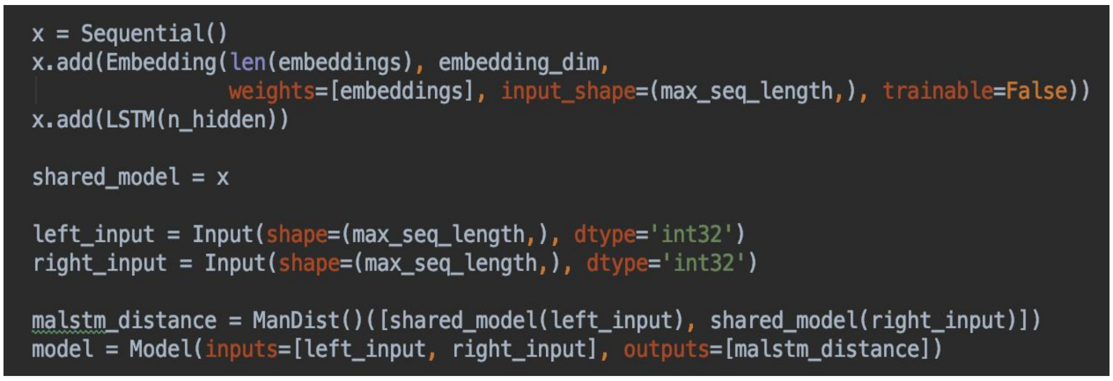
Fig. 14 Word to vector for MaLSTM



```
def split_and_zero_padding(df, max_seq_length):

    X = {'left': df['lemmatized'], 'right': df['query']}

    for dataset, side in itertools.product([X], ['left', 'right']):

        dataset[side] = pad_sequences(dataset[side], padding='pre', truncating='post', maxlen=max_seq_length)

    return dataset
```

Fig. 15 Vector Padding

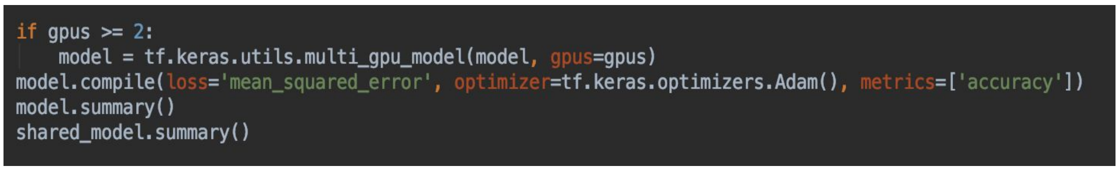
Next, we initialize 2 distinct networks and feed them with the vector representation of 2 sentences and produce hidden state encoding semantic meanings that are then compared using the Manhattan Distance [Fig. 16].



```
x = Sequential()
x.add(Embedding(len(embeddings), embedding_dim,
                weights=[embeddings], input_shape=(max_seq_length,), trainable=False))
x.add(LSTM(n_hidden))

shared_model = x

left_input = Input(shape=(max_seq_length,), dtype='int32')
right_input = Input(shape=(max_seq_length,), dtype='int32')

malstm_distance = ManDist()([shared_model(left_input), shared_model(right_input)])
model = Model(inputs=[left_input, right_input], outputs=[malstm_distance])
```

Fig. 16 Siamese Neural Network Construction

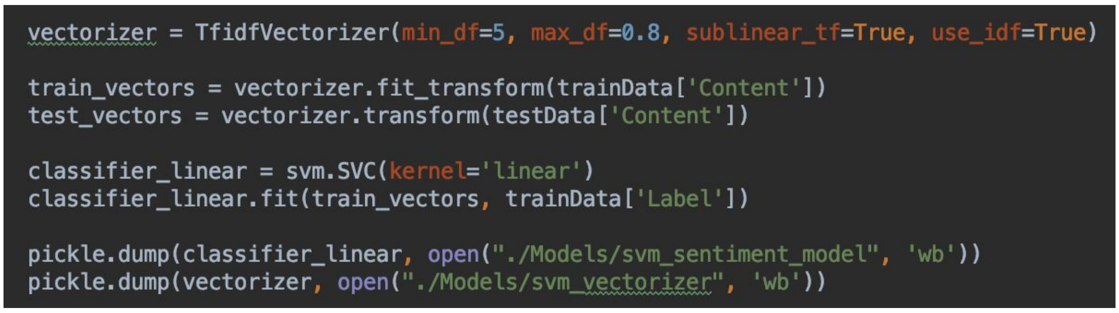
Because of the separation of networks, the model is suitable for parallelism, but the Keras library produces errors when trying to save the model if more than 2 GPUs have been used during training. However, this was not a practical impediment, since our testing machine only had 1 CPU with some integrated graphics processing units, but for platform independence, we have restricted the number of GPUs to 2 [Fig. 17]. Nevertheless, a comparison with respect to suitability to parallelism could make the objective of future work. Finally, we saved the model in the "Models" folder.



```
if gpus >= 2:
    model = tf.keras.utils.multi_gpu_model(model, gpus=gpus)
model.compile(loss='mean_squared_error', optimizer=tf.keras.optimizers.Adam(), metrics=['accuracy'])
model.summary()
shared_model.summary()
```

Fig. 17 GPU restriction

Lastly, but surely not the least, we have chosen a support vector machine model for our sentiment analysis task. In this part, we train a linear **SVM** with Amazon reviews dataset to classify each tweet from our dataset as positive and negative and the query itself. By finding the polarity of texts and their similarity with a query, we can significantly reduce the number of comparisons by focusing only on the texts that belong in the same sentiment class as our query. With respect to the actual implementation, we are using a TF-IDF embedding on the raw training and testing texts and then we build a SVM with a linear kernel using Sklearn library [Fig. 18]. Lastly, the model is saved in the “Models” folder.



```
vectorizer = TfidfVectorizer(min_df=5, max_df=0.8, sublinear_tf=True, use_idf=True)

train_vectors = vectorizer.fit_transform(trainData['Content'])
test_vectors = vectorizer.transform(testData['Content'])

classifier_linear = svm.SVC(kernel='linear')
classifier_linear.fit(train_vectors, trainData['Label'])

pickle.dump(classifier_linear, open("./Models/svm_sentiment_model", 'wb'))
pickle.dump(vectorizer, open("./Models/svm_vectorizer", 'wb'))
```

Fig. 18 SVM training and model saving

C. Preprocessing: The preprocessing techniques that were previously discussed both in the **3.1.** and **4.2.** section **C** chapters and in this section we will only highlight some important aspects of the procedures’ implementation. For the *lemmatization function*, we used WordNet to be able to reduce a certain word to its root while for the *stopwords_removal* we used *stopwords corpus* from the nltk package. For both *noise_removal* and *remove_unwanted_characters*, we used regex for characters recognition and replaced them with white spaces.

D. Embeddings: Like in the previous section, we will just highlight some important aspects of our implementation as these were largely explained in the **3.1.** chapter.

	Term Frequency	TF-IDF	word2vec
Libraries	Collections	gensim	gensim
Extra Corpuses	None	None	None
INPUT type	tokenized text	tokenized text	tokenized

	word2vec_secondary	BERT	GloVe
Libraries	Collections	sent2vec	spacy
Extra Corpuses	GoogleNews	None	en_core_web_sim
INPUT type	text	text	text

Fig. 19

To make the transition sentence-vector we firstly transformed every word from each sentence in vectors in the *word2vec function*, next we computed the term frequency-inverse document frequency for each word in the *TF-IDF function*. Finally, we used the TF-IDF value to add weights for each word-vector from each sentence and then calculated the average between the vectors multiplied by their weights to compute the whole sentence's vector - in the *word2vec_sentence function*. BERT uses a vectorizer to convert sentences into vectors while GloVe uses en_core_web_sm corpus. Both of them are developed by Google for its search engine and both can offer statistical information about the words and add value to the similarity calculations.

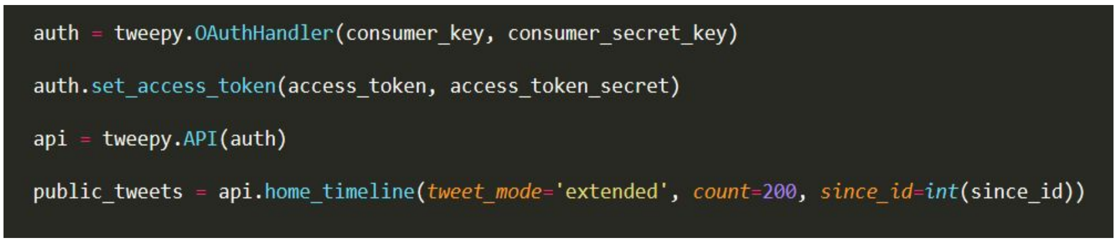
E. Plots: these functions are used to present the visual representation of the results.

F. Tests: this part has access to all the classes presented earlier and can be used to produce combinations of systems.

5. Data Collection

Given our underlying goal -- namely to study and determine the underneath meaning of short texts, as in direct communication, by finding hidden connections that show some kind of relatedness and support between short paragraphs-- we chose Twitter posts as the main data source for this research. Besides the fact that it is suitable for our criteria of short, direct speech messages, Twitter's advantage resides in the company's policy of providing openly accessible data streams by means of a well-documented API. The access to this resource is enabled after submitting a request and receiving approval - credentials - on Twitter's Developer Platform, but no clear requirements are specified for being granted access. There are two methods of retrieving data from Twitter: using a streaming API - retrieves all live data as per request - or the rest API - makes requests of data based on given parameters. Both can be used with python's tweepy library.

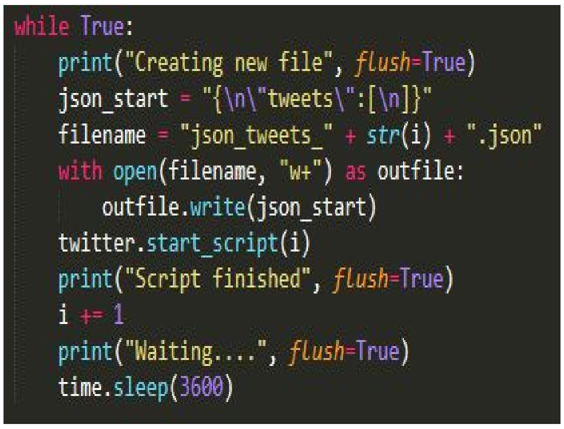
For our research, we used python 3.7 and tweepy with the RESTful API as we wanted to obtain older tweets and have more control over the data collected and the timeslots. We used the "home_timeline" method with "tweet_mode" parameter set on extended - to receive the complete text from a tweet -, "count" on 200 - from our observations in an hour interval, no more than 200 tweets were new - and we added a since_id to avoid retrieving duplicates. Data was saved in JSON format. By using the "home_timeline" command we ensured that data was retrieved only from the Twitter accounts that we follow [Fig. 20].



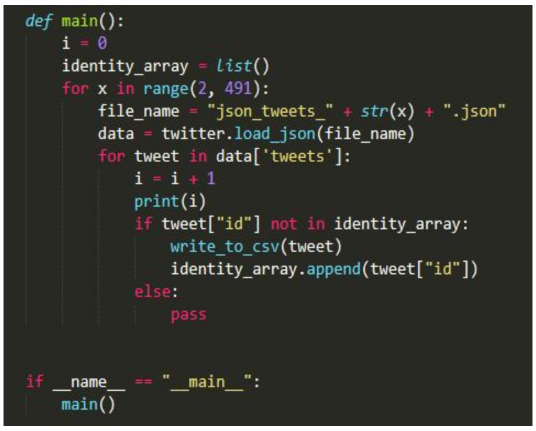
```
auth = tweepy.OAuthHandler(consumer_key, consumer_secret_key)

auth.set_access_token(access_token, access_token_secret)

api = tweepy.API(auth)

public_tweets = api.home_timeline(tweet_mode='extended', count=200, since_id=int(since_id))
```

Fig. 20 Twitter API call

We decided to create a daemon to run the script once every hour in accordance with our observations that the maximum of new tweets per hour is around 200. Interestingly enough, setting the requests hourly proved useful for both gathering information without multiple duplicates, but at the same time avoiding a ban for too many requests in a small period of time. Finally, to be sure that no duplicates were retrieved, we have created another python script to remove any duplicates encountered and composed together a final dataset with CSV format - comma-separated values to ease the analysis -, more compact, supported by scientific applications, records are ordered and always in the same place and it can contain a header.



```python
while True:
    print("Creating new file", flush=True)
    json_start = "{\n\"tweets\":[\n]}"
    filename = "json_tweets_" + str(i) + ".json"
    with open(filename, "w+") as outfile:
        outfile.write(json_start)
    twitter.start_script(i)
    print("Script finished", flush=True)
    i += 1
    print("Waiting....", flush=True)
    time.sleep(3600)
```

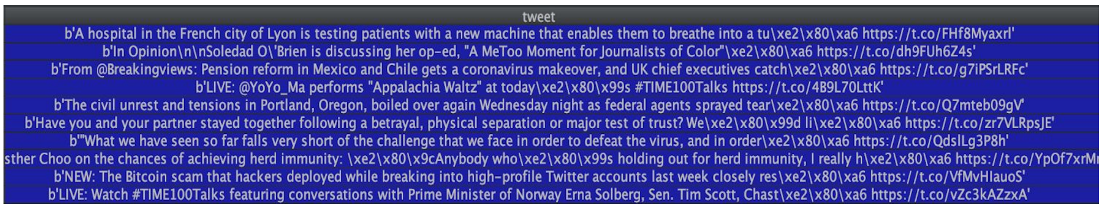


```python
def main():
    i = 0
    identity_array = list()
    for x in range(2, 491):
        file_name = "json_tweets_" + str(x) + ".json"
        data = twitter.load_json(file_name)
        for tweet in data['tweets']:
            i = i + 1
            print(i)
            if tweet["id"] not in identity_array:
                write_to_csv(tweet)
                identity_array.append(tweet["id"])
            else:
                pass


if __name__ == "__main__":
    main()
```

Fig. 21 Daemon and CSV creation

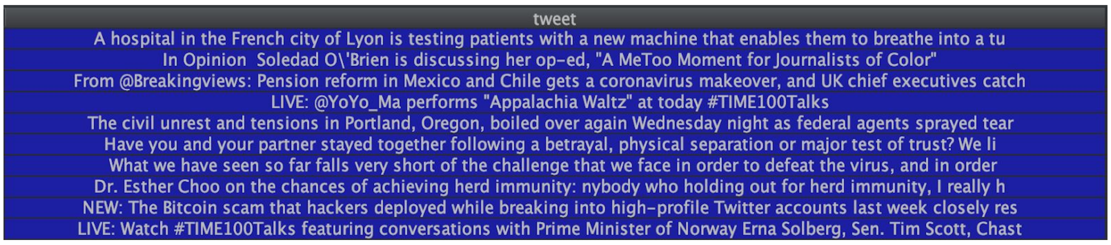
Data were collected from 50 different news, economics and political entities from the UK and the US adding up to a total of 45,835 unique tweets with 84 features - containing, besides the tweet's text, additional information about the user's profile, followers, likes and resources used. At a first glance, we can already notice some critical issues: the existence of non-ASCII characters and links in the actual text. Thoroughly cleaning was done before any analysis. Fig. 22 presents the changes in the dataset after applying the preprocessing functions described in the implementation chapter.

Initial dataset:



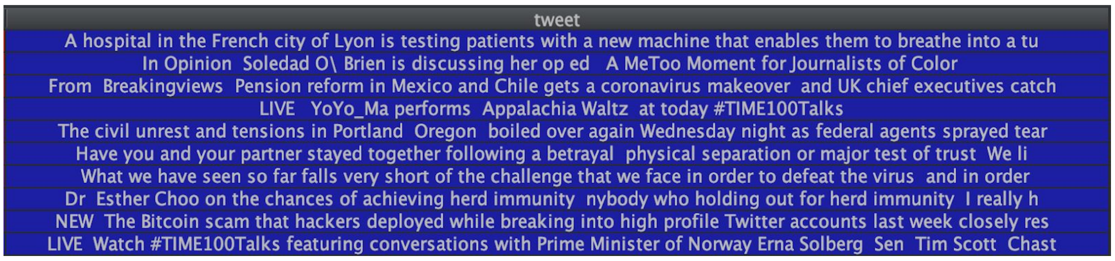
tweet
b'A hospital in the French city of Lyon is testing patients with a new machine that enables them to breathe into a tu\xe2\x80\xa6 https://t.co/FHf8Myaxrl'
b'In Opinion\n\nSoledad O\'Brien is discussing her op-ed, "A MeToo Moment for Journalists of Color"\xe2\x80\xa6 https://t.co/dh9FUh6Z4s'
b'From @Breakingviews: Pension reform in Mexico and Chile gets a coronavirus makeover, and UK chief executives catch\xe2\x80\xa6 https://t.co/g7iPSrLRFc'
b'LIVE: @YoYo_Ma performs "Appalachia Waltz" at today\xe2\x80\x99s #TIME100Talks https://t.co/4B9L70LttK'
b'The civil unrest and tensions in Portland, Oregon, boiled over again Wednesday night as federal agents sprayed tear\xe2\x80\xa6 https://t.co/Q7mteb09gV'
b'Have you and your partner stayed together following a betrayal, physical separation or major test of trust? We\xe2\x80\x99d li\xe2\x80\xa6 https://t.co/zr7VLRpsJE'
b'"What we have seen so far falls very short of the challenge that we face in order to defeat the virus, and in order\xe2\x80\xa6 https://t.co/QdsILg3P8h'
sther Choo on the chances of achieving herd immunity: \xe2\x80\x9cAnybody who\xe2\x80\x99s holding out for herd immunity, I really h\xe2\x80\xa6 https://t.co/YpOf7xrM
b'NEW: The Bitcoin scam that hackers deployed while breaking into high-profile Twitter accounts last week closely res\xe2\x80\xa6 https://t.co/VfMvHIauoS'
b'LIVE: Watch #TIME100Talks featuring conversations with Prime Minister of Norway Erna Solberg, Sen. Tim Scott, Chast\xe2\x80\xa6 https://t.co/vZc3kAZzxA'

Unwanted Characters Removed:



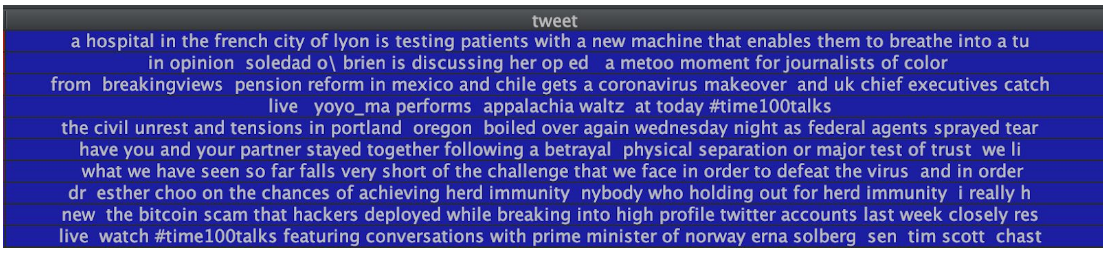
tweet
A hospital in the French city of Lyon is testing patients with a new machine that enables them to breathe into a tu
In Opinion Soledad O\'Brien is discussing her op-ed, "A MeToo Moment for Journalists of Color"
From @Breakingviews: Pension reform in Mexico and Chile gets a coronavirus makeover, and UK chief executives catch
LIVE: @YoYo_Ma performs "Appalachia Waltz" at today #TIME100Talks
The civil unrest and tensions in Portland, Oregon, boiled over again Wednesday night as federal agents sprayed tear
Have you and your partner stayed together following a betrayal, physical separation or major test of trust? We li
What we have seen so far falls very short of the challenge that we face in order to defeat the virus, and in order
Dr. Esther Choo on the chances of achieving herd immunity: nybody who holding out for herd immunity, I really h
NEW: The Bitcoin scam that hackers deployed while breaking into high-profile Twitter accounts last week closely res
LIVE: Watch #TIME100Talks featuring conversations with Prime Minister of Norway Erna Solberg, Sen. Tim Scott, Chast

Noise Removed:



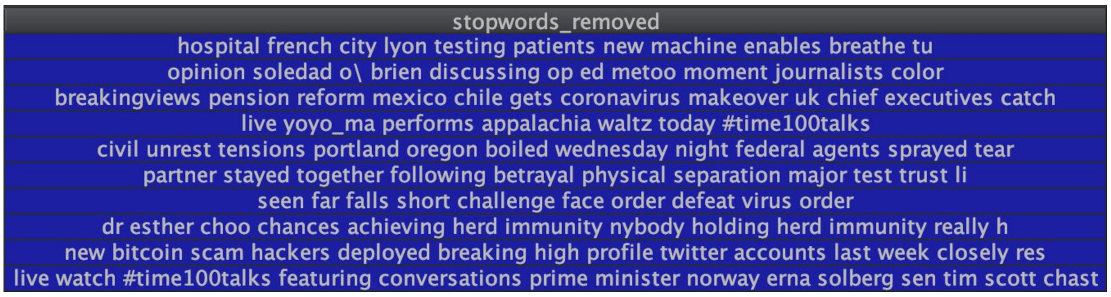
tweet
A hospital in the French city of Lyon is testing patients with a new machine that enables them to breathe into a tu
In Opinion Soledad O\ Brien is discussing her op ed   A MeToo Moment for Journalists of Color
From Breakingviews Pension reform in Mexico and Chile gets a coronavirus makeover and UK chief executives catch
LIVE   YoYo_Ma performs Appalachia Waltz at today #TIME100Talks
The civil unrest and tensions in Portland Oregon boiled over again Wednesday night as federal agents sprayed tear
Have you and your partner stayed together following a betrayal physical separation or major test of trust We li
What we have seen so far falls very short of the challenge that we face in order to defeat the virus and in order
Dr Esther Choo on the chances of achieving herd immunity nybody who holding out for herd immunity I really h
NEW The Bitcoin scam that hackers deployed while breaking into high profile Twitter accounts last week closely res
LIVE Watch #TIME100Talks featuring conversations with Prime Minister of Norway Erna Solberg Sen Tim Scott Chast

Lowercasing:



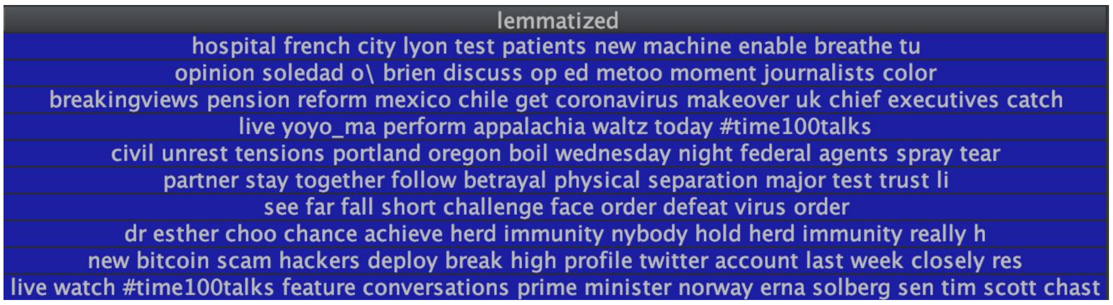
tweet
a hospital in the french city of lyon is testing patients with a new machine that enables them to breathe into a tu
in opinion soledad o\ brien is discussing her op ed   a metoo moment for journalists of color
from breakingviews pension reform in mexico and chile gets a coronavirus makeover and uk chief executives catch
live   yoyo_ma performs appalachia waltz at today #time100talks
the civil unrest and tensions in portland oregon boiled over again wednesday night as federal agents sprayed tear
have you and your partner stayed together following a betrayal physical separation or major test of trust we li
what we have seen so far falls very short of the challenge that we face in order to defeat the virus and in order
dr esther choo on the chances of achieving herd immunity nybody who holding out for herd immunity i really h
new the bitcoin scam that hackers deployed while breaking into high profile twitter accounts last week closely res
live watch #time100talks featuring conversations with prime minister of norway erna solberg sen tim scott chast

Stopwords Removed:



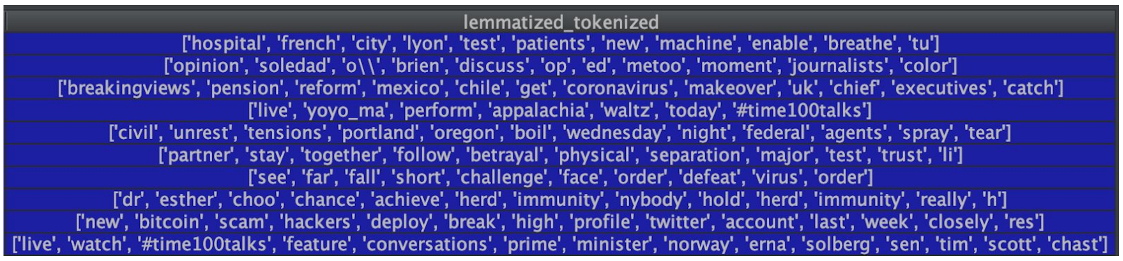
stopwords_removed
hospital french city lyon testing patients new machine enables breathe tu
opinion soledad o\ brien discussing op ed metoo moment journalists color
breakingviews pension reform mexico chile gets coronavirus makeover uk chief executives catch
live yoyo_ma performs appalachia waltz today #time100talks
civil unrest tensions portland oregon boiled wednesday night federal agents sprayed tear
partner stayed together following betrayal physical separation major test trust li
seen far falls short challenge face order defeat virus order
dr esther choo chances achieving herd immunity nybody holding herd immunity really h
new bitcoin scam hackers deployed breaking high profile twitter accounts last week closely res
live watch #time100talks featuring conversations prime minister norway erna solberg sen tim scott chast

Lemmatized:



lemmatized
hospital french city lyon test patients new machine enable breathe tu
opinion soledad o\ brien discuss op ed metoo moment journalists color
breakingviews pension reform mexico chile get coronavirus makeover uk chief executives catch
live yoyo_ma perform appalachia waltz today #time100talks
civil unrest tensions portland oregon boil wednesday night federal agents spray tear
partner stay together follow betrayal physical separation major test trust li
see far fall short challenge face order defeat virus order
dr esther choo chance achieve herd immunity nybody hold herd immunity really h
new bitcoin scam hackers deploy break high profile twitter account last week closely res
live watch #time100talks feature conversations prime minister norway erna solberg sen tim scott chast

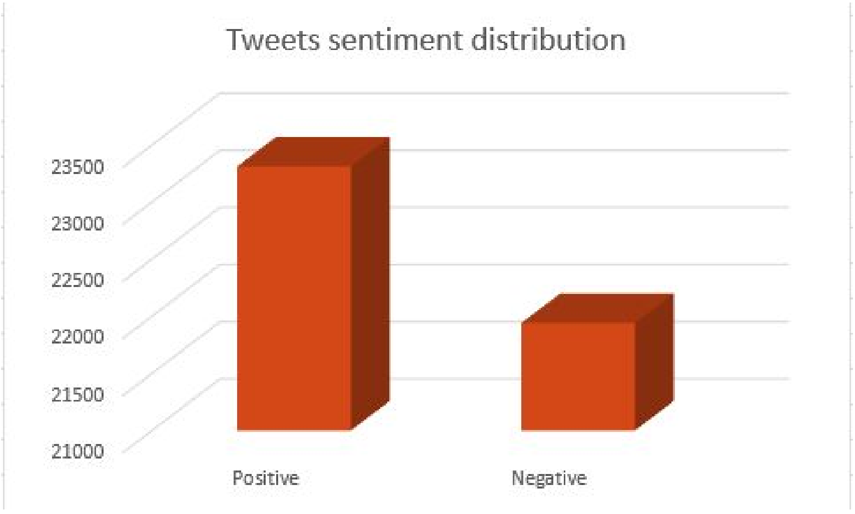
Tokenized:



lemmatized_tokenized
['hospital', 'french', 'city', 'lyon', 'test', 'patients', 'new', 'machine', 'enable', 'breathe', 'tu']
['opinion', 'soledad', 'o\\', 'brien', 'discuss', 'op', 'ed', 'metoo', 'moment', 'journalists', 'color']
['breakingviews', 'pension', 'reform', 'mexico', 'chile', 'get', 'coronavirus', 'makeover', 'uk', 'chief', 'executives', 'catch']
['live', 'yoyo_ma', 'perform', 'appalachia', 'waltz', 'today', '#time100talks']
['civil', 'unrest', 'tensions', 'portland', 'oregon', 'boil', 'wednesday', 'night', 'federal', 'agents', 'spray', 'tear']
['partner', 'stay', 'together', 'follow', 'betrayal', 'physical', 'separation', 'major', 'test', 'trust', 'li']
['see', 'far', 'fall', 'short', 'challenge', 'face', 'order', 'defeat', 'virus', 'order']
['dr', 'esther', 'choo', 'chance', 'achieve', 'herd', 'immunity', 'nybody', 'hold', 'herd', 'immunity', 'really', 'h']
['new', 'bitcoin', 'scam', 'hackers', 'deploy', 'break', 'high', 'profile', 'twitter', 'account', 'last', 'week', 'closely', 'res']
['live', 'watch', '#time100talks', 'feature', 'conversations', 'prime', 'minister', 'norway', 'erna', 'solberg', 'sen', 'tim', 'scott', 'chast']

Fig. 22 Preprocessing Results

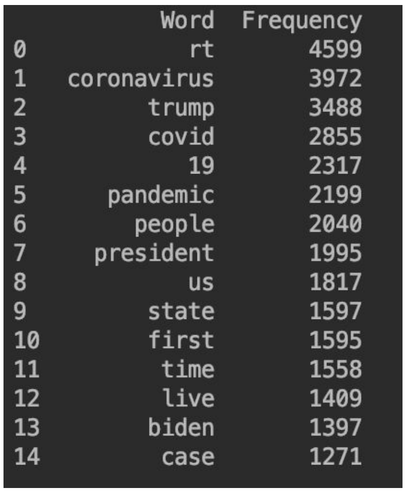
We can further analyse the data by comparing its polarity sparsity by using Amazon reviews sentiment and SVM for predicting each tweet polarity. The results are shown in Fig. 22 and we can conclude that our dataset is balanced, slightly to the positive side in this case.



Positive Tweets	Negative Tweets
23312	21941

Fig. 22 Polarity of data

In Fig. 23 and 24 we have presented the 15 most common words used in the collected tweets which reflects the overall topics - after removing stopwords. The main topics seem to be US related -“trump”, “biden”, “us” - and health-related - “covid”, “19”, “pandemic”, “case”. We can notice that the most common word is “rt” - retweet - thus, the dataset might contain duplicated texts even though they don’t have the same id.



```
           Word  Frequency
0            rt       4599
1   coronavirus       3972
2         trump       3488
3         covid       2855
4            19       2317
5      pandemic       2199
6        people       2040
7     president       1995
8            us       1817
9         state       1597
10        first       1595
11         time       1558
12         live       1409
13        biden       1397
14         case       1271
```

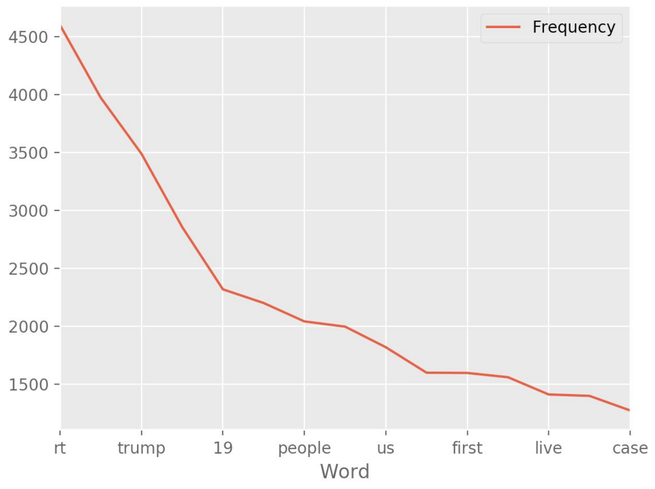


Fig. 23 Word Frequency

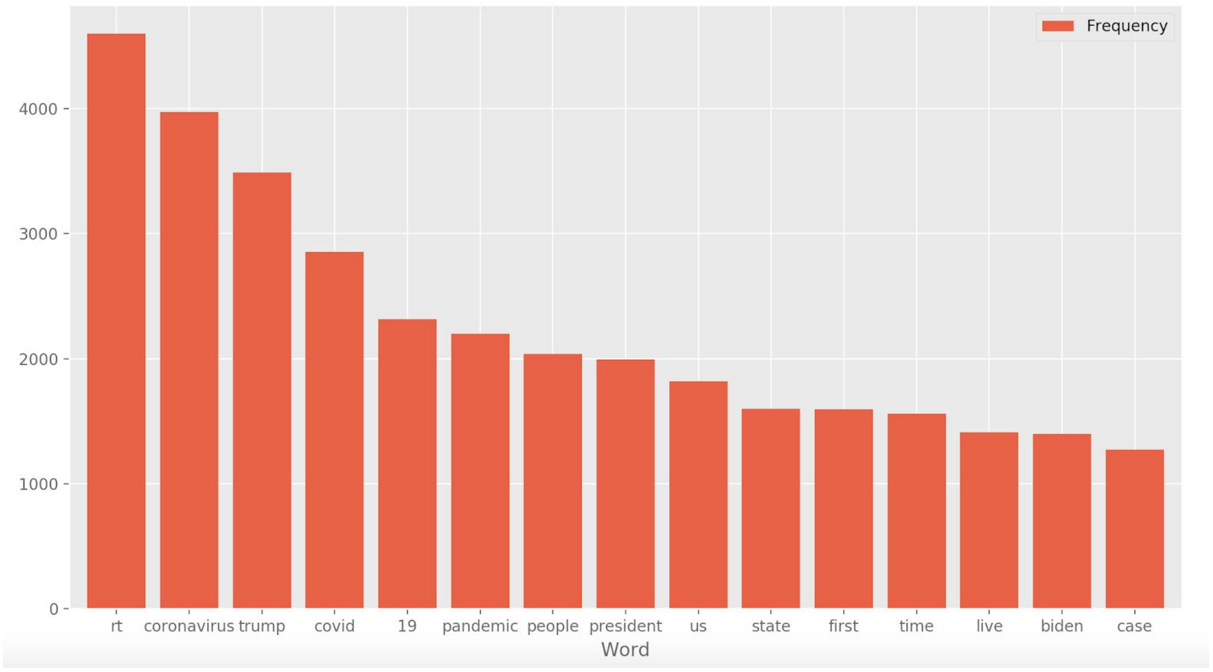


Fig. 24 Word Frequency Bars

6. Results, Analysis and Critical Evaluation

In this section we will first define our testing corpus, the methodology underlying it and a justification for this specific choice, then we delve into the actual hardware and device settings, present the results for each method in a structured manner and then draw conclusions based on them. However, it is important to state right from the start that the algorithms are dependent on additional training data sets - that were defined in the implementation chapter - and that the nature of our dataset is inclined to the topics of politics, economy and healthcare.

For the purpose of clarity, we have decided to use the following methodology with respect to testing, mostly because of a lack of a well defined and extended enough dataset of tweets to reveal supportivity - namely, an annotated dataset with examples of tweets that support each other. However, we believe that our solution could be used to reduce the number of manual inspections necessary for the compiling of such a dataset. Such improvements stem from the ability to generate artificial yet meaningful datasets with respect to supportivity of a tweet by clustering and sorting with respect to similarity. As a brief explanation, this operation focuses the need for human verification only on a finite subset of tweets from a cluster. Such a dataset could be theoretically used for training any sort of machine learning model and scenarios as described in section **2.3.** because it would contain a representation of the concept of supportivity. Having such a representation, it would be much easier to determine texts that support each other.

In these circumstances, we were forced to choose a different approach, that can somehow be compared with the retrieving power of a search engine:

1. For illustrative purposes, we chose to test five different queries
2. The queries are being used as input sentences for each described algorithm.
3. The algorithms return sets of tweets that are “supportive” to the input query, in decreasing order. Because these sets require manual inspection, only a fixed subset of 15 results is being used for evaluation.
4. 3 categories of supportivity are defined: supportive, unclear and not related
5. The subset of tweets returned is analysed with respect to the 3 categories defined above, and a relevance score is defined as a weighted average of the 3, using the following weights [2, 1, 0]. Given the size of the subsets evaluated using this categorisation, the relevance score is a best-case orientated metric.
6. To compute the relevance we follow the equation:

   relevance = (2*nr_related_sentences + 1*nr_unclear)/nr_results
7. The total relevance is computed as the arithmetic mean of the evaluated sentences’ relevance

It is important to take into account that the underlying hardware used in the testing environment has a direct impact on the running time of the algorithms. Our testing environment consists of an Apple MacBook Pro Retina 2015 with 8 GB RAM 1867 MHz DDR3, integrated graphics card - Intel Iris Graphics 6100 1536 MB - and 2,7 GHz Dual-Core Intel Core I5. The software was developed with Python 3.7.

In the following tables, we present a runtime, storage and accuracy comparison between well-known algorithms used for determining the similarity of texts.

Name	Training time	Prediction Time	Optimization time	Model size
Jaccard	None	9.94 sec	None	None
Cosine with CBOW embedding	None	19.08 sec	None	None
Cosine with SG embedding	None	20.25 sec	None	None
Cosine with GloVe	None	8.67	None	None
LSI with Cosine	6.33 sec	19.38 sec	174.93 sec	4 MB
LDA with Shannon	17.01 sec	34.67 sec	None	7 MB
Siamese LSTM with Manhattan	9 hours 12 min	0.64 sec	None	103.9 MB
Word Mover's distance	None	171.28 sec	None	None
Word Mover's Sentiment	None	207.3 sec	None	6.4 MB
k-means CBOW with Word Mover	k-means with optimization technique: 256.7 sec	32.61 sec	k-means prediction of query: 0.8 sec	183 KB
k-means SG with Word Mover	k-means with optimization technique:	31.65 sec	k-means prediction of query: 0.3 sec	183 KB

	229.7 sec			
Combined similarity	None	inf	None	None
Combined similarity with k-means CBOW	k-means with optimization technique: 256.7 sec	inf		183 KB
k-means CBOW with Word Mover and Sentiment analysis	k-means with optimization technique: 256.7 sec SVM: 10 sec	40.6 sec	None	6.5 MB
k-means SG with Word Mover and Sentiment analysis	k-means with optimization technique: 229.7 sec SVM: 10 sec	55.62	None	6.5 MB

Fig. 25

It is important to note that besides the training time, the preprocessing and embedding parts took an extensive period of 861 seconds. However, now the dataset consists of all necessary features to work with all the models presented in **Fig. 25**.

We defined the next five sentences to be evaluated by the models described in **Fig. 25**:

1. Stock market reached a historical low after the covid pandemic hit economies.
2. Joe Biden is fighting Donald Trump’s conspiracy theories.
3. The black lives matter show the extent of racial inequality.
4. The covid vaccine search continues.
5. China’s territorial claims in the south china sea spark new conflicts.

	Q1	Q2	Q3	Q4	Q5	Total relevance
Jaccard	0.5	0.83	0.9	0.7	0.63	0.712 out of 2
Cosine with CBOW embedding	0.2	0.1	0.15	0.2	0.4	0.21 out of 2
Cosine with	0.1	0.3	0.2	0.25	0.15	0.2 out of 2

SG embedding						
Cosine with GloVe	0.73	0.86	0.8	0.4	1.26	0.811 out of 2
LSI with Cosine	1.06	0.93	0.73	1.26	0.3	0.85 out of 2
LDA with Shannon	0.53	0.26	0.12	0.93	0.11	0.39 out of 2
Siamese LSTM with Manhattan	1.06	0.6	0.5	0.73	0.4	0.65 out of 2
Word Mover's distance	1.6	1.26	1.46	1.33	1.73	1.47 out of 2
Word Mover and Sentiment analysis	1.53	1.33	1.86	1.66	1.86	1.648 out of 2
k-means CBOW with Word Mover	0.8	0.93	1.6	0.8	1.2	1.07 out of 2
k-means SG with Word Mover	1	1.13	0.46	0.6	0.53	0.77 out of 2
k-means with Word Mover and Sentiment analysis	1.33	0.4	1.26	0.4	1.13	0.904 out of 2

Fig. 26

Analysing Fig. 26, we can draw some interesting observations. First of all, it can be clearly seen that the Word Mover's Distance performs much better than all the other distances: without any enhancements, it reaches an R-value of 1.47, whereas the second-best distance seems to be the Cosine Similarity, which has a value of 0.81 if used with Glove Embeddings. However, when used with CBOW or SG embeddings, the Cosine Similarity scores much lower, having scores of 0.21 or 0.2, probably due to the fact that the information in these embeddings alone is not enough for the cosine to clearly distinguish topics. This phenomenon got replicated in the usage of the Kmeans clustering: to be able to clusterise the tweets based on their topics, they required the embeddings containing semantic information, which resulted in lower R scores when used with the CBOW. Even though the combined similarities showed promising results on small subsets randomly sampled from the database, its complexity is of a polynomial with a degree higher than 5, so its computation didn't complete in due time on the full database. However, we believe that further research on this method can lead to improving its running time and therefore leading to increasing its usability on large datasets. Interestingly enough, the Jaccard Similarity outperformed other well-advertised methods, probably because of the length, complexity and lemmatization of the texts. The sentences retrieved by it contain the basic words related to the searched topic, but it lacks in determining supportivity. At the other end of the spectrum, the Siamese LSTM network and the LDA with Shanon Distance retrieve some very interesting and complex examples that can meaningfully support the input text, but a big part of the retrieval still contains noise, so their relevance decreases sharply, having scores of 0.65 and 0.39. However, we do believe that having a bigger and more Twitter related database for training the Siamese LSTM with respect to sentiments regarding a given topic can greatly increase its score - its main flaw was that the sentiments were not diverse enough. The SVM sentiment classifier proved to be a valuable addition to the regular WordMover's Distance in the sense that it increased its overall R score to a value of 1.648 and making it our best supportivity determination method, yet it also proved slightly detrimental when used in conjunction with Kmeans. This was an unfortunate event, caused by the fact that the clustering misassigned tweets to different clusters and these clusters were furtherly pruned by the use of the SVM. In addition to the high R score, the WordMover's Distance comes as a pre-trained Google model and therefore requires no actual training time, but can be quite slow when making predictions - the average prediction speed was 171.28 seconds - so it might not be suitable for use in real-time situations. Depending on the situations, the WordMover's Distance can be used after k-means clustering for an average prediction time of 32.61 seconds, or a simpler alternative of Cosine Similarity and Glove embeddings for 8.67 seconds, but the R score decreases to 1.07 and 0.81 respectively. A tradeoff relevance-prediction time can be seen in Fig. 27.

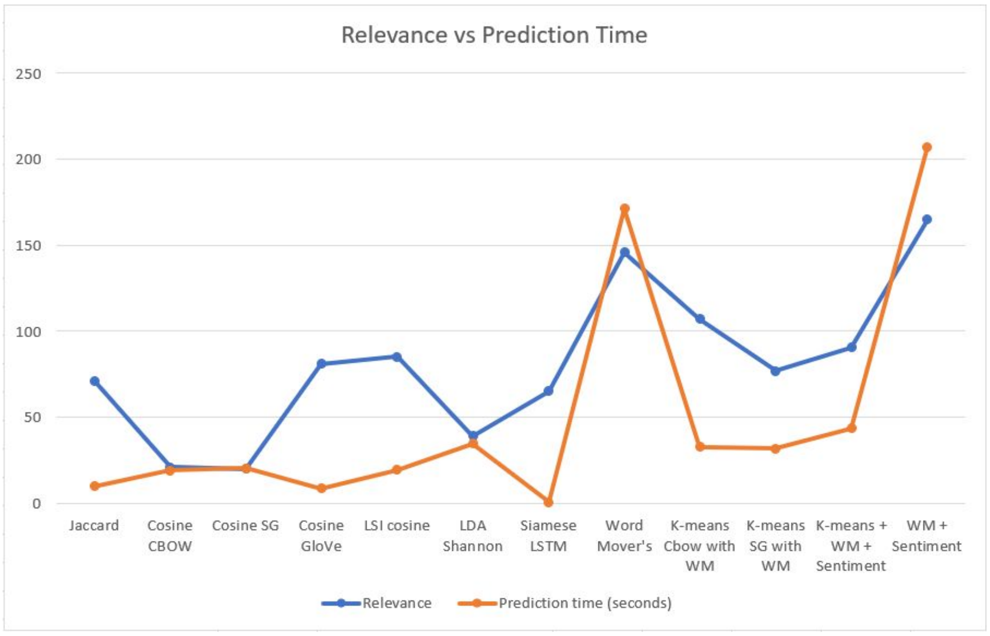


Fig. 27 Tradeoff Relevance - Prediction Time

Taking these into account, we can clearly state that sentiment analysis is a promising method for improving the supportivity retrieval metric. Furthermore, clustering algorithms seem a valuable candidate for reducing the overall prediction time but suffer from issues related to correctly identifying topics that mainly stem from the embedding underneath them. Last but not least, the Siamese LSTM did not produce the expected results, but cannot be fully considered inappropriate for this task, mainly because of the nature of the dataset on which it was trained on. Finally, we propose a configuration of k-means clustering on Glove embeddings, with sentiment analysis and WordMover's Distance for future testing.

Finally, we have tested our proposed solution which lowered the prediction time significantly to 21.09 seconds with the note that the prediction time is highly correlated with the cluster size. However, the reduction is not significant enough to pass other faster algorithms that we discussed, and a tradeoff between accuracy and time must be done. Our proposal scored 1.26, 1.13, 1.8, 1.1 and 1.73 for query 1, 2, 3, 4 and 5 respectively, averaging at 1.404, which does not pass simple Word Mover with Sentiment Analysis but makes a difference in time as shown in Fig. 28.

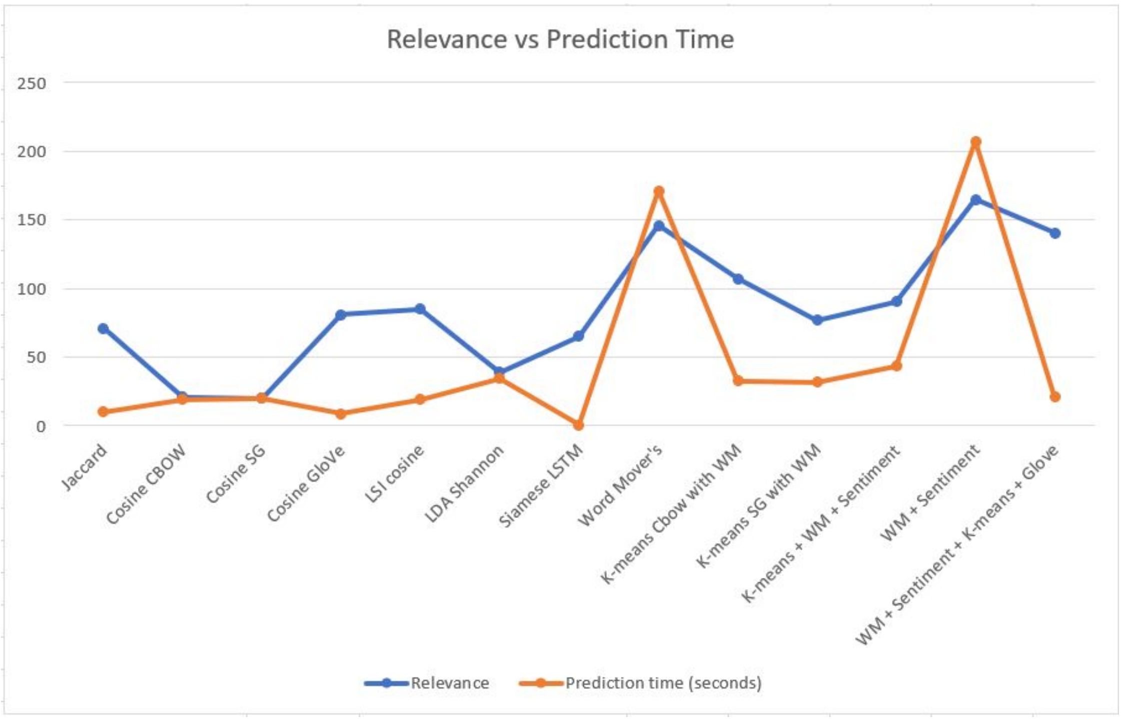


Fig. 28 Tradeoff comparison including our proposal

All exact results can be found in the folder "results" from TextSimilarityProject.

7. Conclusion & Future Work

This paper has examined several techniques that were well known in the literature for their use in determining text similarity and tried to determine their appropriateness for text supportiveness. In support of this goal, we have constructed a political and economical tweet database using Twitter API, analysed the received packets, pruned the necessary metadata, and implemented scripts for preprocessing it. We implemented the above-described algorithms using a mixture of personal code and well-known libraries and performed some optimisations to increase the accuracy and decrease the running time of our solution. By analysing the produced results, we determined some of the desirable metrics that should be further studied with respect to this topic - namely the WordMover's Distance -, some of the possible candidates that require additional optimisations - such as the SVM sentiment classifier or k-means clustering - and some of the main impediments that might arise while trying to improve the solution - the fact that the word embeddings either do not contain enough information regarding the topic of the texts they embed, or the distances calculated using them are not entirely able to use the available information fully. As a personal opinion, we believe it is the first issue since some distances manage to produce some better results even without embeddings, but that also might be an indication that bigger databases of examples would be required

.

However, the work done in this paper is by no means complete, rather it offers a starting point for future projects and improvements built upon what has already been done. With respect to functionalities, we could focus on the results and processing to be stored in a cloud environment, facilitating parallel processing via distributed computing - offer faster results for both training and prediction time. In such a setting, the access could be granted to multiple parties via a web API or interface. Another quite interesting functionality would be to create a more visual interpretation of the results as an application or website that will retrieve a list of supportive tweets for the user's input. Moreover, semantically, our solution could improve by offering meaning to links or other accounts references, emoji, memes, hashtags or popular phrases by using an expression dataset, as they can have underlying information about the user's sentiments towards the topic. Additionally, even more, semantic information could be inferred by building topic distributions that contain the sentiment polarity of the user towards them, for multiple tweets by the same user. This notion could be extended even further if semantic graphs of topics and sentiments can be constructed at the level of multiple linked users. In a sense, this would mean using information about the user's best friend's opinion of a certain topic to improve the chance of finding the user's. From a sociology point of view, this seems possible, at least on social networks that favour the clustering of users into very specialised groups supporting various causes or ideologies. Following this idea, the scenarios presented in section **2.3.** could also be subjects for other researches. However, the implications for user privacy should be taken into consideration.

8. References

[1] Jure Leskovec, Anand Rajaraman, Jeffrey David Ullman, "Mining of Massive Datasets", Chapter 3, Stanford University, California, Available online at: http://infolab.stanford.edu/~ullman/mmds/ch3.pdf

[2] Darwis Robinson Manalu, Edward Rajagukguk, Rimbun Siringoringo, Desmon Kristanto Siahaan and Poltak Sihombing, "The Development of Document Similarity Detector by Jaccard Formulation", IEEE 2019 International Conference of Computer Science and Information Technology (ICoSNIKOM), Available online at: https://ieeexplore.ieee.org/document/9111494

[3] Pawan Kumar Verma, Shalini Agarwal, Mohd Aamir Khan, "Opinion mining considering roman words using Jaccard similarity algorithm based on clustering", IEEE 2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Available online at: https://ieeexplore.ieee.org/document/8204041

[4] IEDP Editorial, "The Big Data Revolution - Time to embrace data driven analytics", Available online at: https://www.iedp.com/articles/the-big-data-revolution/

[5] Saikumar, "3 Trends Enabling The Big Data Revolution", Technative, 2018, Available online at: https://www.technative.io/3-trends-enabling-the-big-data-revolution/

[6] Foster Provost and Tom Fawcett, "Data Science for Business", 2013. Available online at: https://www.researchgate.net/publication/256438799_Data_Science_for_Business

[7] Social Media One, "The development of social networks worldwide: History and future", Available online at: https://socialmediaagency.one/development-social-networks-worldwide-history-future

[8] Esteban Ortiz-Ospina, "The rise of social media", Our World in Data, 2019, Available online at: https://ourworldindata.org/rise-of-social-media

[9] KungFu AI, Data.AI, "Coronavirus Twitter Analysis", Available online at: https://data.world/kungfuaiteam/coronavirus-twitter-analysis?fbclid=IwAR0IidtrOfWvnMb1S9WyLHP2Xgg75fMzF00MS7g-NPN-tgz7yO-232d0DDs

[10] Amardeep Kumar, Keggle: "Aossie: Fake News Detection Datasets", Available online at: https://www.kaggle.com/ad6398/aossie-fake-news-detection-datasets?fbclid=IwAR0PRinx0K3gX-xIzMlBJWR5e3ak6CQ1DAfikuBeFIPQusvacuJcJzIc22M

[11] Ray Jackendoff, "Semantics and Cognition", MIT Press 1983, Cambridge, MA.

[12] Aminul Islam and Diana Inkpen, "Semantic Text Similarity Using Corpus-Based Word Similarity and String Similarity", University of Ottowa, ACM Transactions on Knowledge Discovery from Data, Vol. 2, 2008, Available online at:

https://static.aminer.org/pdf/PDF/000/369/439/classification_of_rss_formatted_documents_using_full_text_similarity_measures.pdf

[13] Juan M. Huerta, "Vector based Approaches to Semantic Similarity Measures", IBM T. J. Watson Research Center, Available online at: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.123.1818&rep=rep1&type=pdf

[14] Yuhua Li, David McLean, Zuhair A. Bandar, James D. O'Shea and Keeley Crockett, "Sentence Similarity Based on Semantic Nets and Corpus Statistics", IEEE Transactions on knowledge and data engineering, Vol. 18, 2006, Available online at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1644735

[15] T. Landauer, P. Foltz and D. Laham, "Introduction to latent semantic analysis", 1998, Discourse Processes, Vol. 25, Available online at: https://www.tandfonline.com/doi/abs/10.1080/01638539809545028

[16] João Carlos Alves dos Santos and Eloi Luiz Favero, "Practical use of a latent semantic analysis (LSA) model for automatic evaluation of written answers", Journal of the Brazilian Computer Society, 2015, Available online at: https://journal-bcs.springeropen.com/articles/10.1186/s13173-015-0039-7

[17] Lushan Han, Abhay kashyap, Tim Finin, James Mayfield and Jonathan Weese, "UMBC_EBIQUITY-CORE: Semantic Textual Similarity Systems", Association for Computational Linguistics, 2013, Available online at: https://www.aclweb.org/anthology/S13-1005.pdf?fbclid=IwAR1ixxNVNj64iuLf5aXXWZKPbHruqogS0VWROthxkblQvaRUSMrs3qR_6AE

[18] Pei-Yuan Zhou, Keith C.C. Chan, "A Model-Based Multivariate Time Series Clustering Algorithm", PAKDD, 2014, Available online at: https://www.researchgate.net/publication/273063437_A_Model-Based_Multivariate_Time_Series_Clustering_Algorithm

[19] Zhu Huan, Zhang Pengzhou and Gao Zeyang, "K-means Text Dynamic Clustering Algorithm Based on KL Divergence", IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS), 2018, Available online at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8466385

[21] Faisal Rahutomo, Teruaki Kitasuka and Masayoshi Aritsugi, "Semantic Cosine Similarity", The 7th International Student Conference on Advanced Science and Technology ICAST, 2012, Available online at: https://www.researchgate.net/publication/262525676_Semantic_Cosine_Similarity

[22] Singular Value Decomposition (SVD) tutorial, Available online at: https://web.mit.edu/be.400/www/SVD/Singular_Value_Decomposition.htm

[23] Syamsul Bahri, Surya Sumpeno and Supeno Mardi Susiki Nugroho, "An Information Retrieval Approach to Finding Similar Questions in Question-Answering of Indonesian Government e-Procurement Services using TF*IDF and LSI Model", 10th International Conference on Information Technology and Electrical Engineering (ICITEE), 2018, IEEE, Available online at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8534856

[24] Fawaz S. Al-Anzi and Dia AbuZeina, "Enhanced Search for Arabic Language Using Latent Semantic Indexing (LSI)", International Conference on Intelligent and Innovative Computing Applications (ICONIC), IEEE, 2018, Available online at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8601096
[25] Nik Nur Amirah, Tuan Mohamad Rahim, Zulaile Mabni, Haslizatul Mohamed Hanum and Nurazzah Abdul Rahman, "A Malay Hadith translated document retrieval using parallel Latent Semantic Indexing (LSI)", Third International Conference on Information Retrieval and Knowledge Management (CAMP), IEEE, 2016, Available online at:
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7806346
[26] Diego Buenano-Fernandez, Mario Gonzalez, Sergio Lujan-Mora, "Text mining of open-ended questions in self-assessment of university teachers: an LDA topic modeling approach", IEEE Access, 2020, Available online at:
https://www.researchgate.net/publication/339368709_Text_mining_of_open-ended_questions_in_self-assessment_of_university_teachers_an_LDA_topic_modeling_approach
[27] Minglai Shao and Liangxi Qin, "Text Similarity Computing Based on LDA Topic Model and Word Co-occurrence", 2nd International Conference on Software Engineering, Knowledge and Information Engineering (SEKEIE), 2014, Available online at: https://www.atlantis-press.com/proceedings/sekeie-14/13639
[28] Nihit Saxena, "Word Mover's Distance for Text Similarity", Towards Data Science, 2019, Available online at:
https://towardsdatascience.com/word-movers-distance-for-text-similarity-7492aeca71b0
[29] Scott Cohen, "Finding Color and Shape Patterns in Images", Stanford University, 1999, Chapter 4, Available online at:
http://infolab.stanford.edu/pub/cstr/reports/cs/tr/99/1620/CS-TR-99-1620.ch4.pdf
[30] Mohammed Alshahrani, Spyridon Samothrakis and Maria Fasli, "Word mover's distance for affect detection", International Conference on the Frontiers and Advances in Data Science (FADS), IEEE, 2017, Available online at:
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8253186
[31] Xiaojun Chen, Li Bai, Dakui Wang and Jinqiao Shi, "Improve Word Mover's Distance with Part-of-Speech Tagging", Chinese Control And Decision Conference (CCDC), IEEE 2018, Available online at:
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8546241
[32] Shakir Mohamed, "Variational Inference for Machine Learning", Syllabus from David Duvenaud's course of "Differentiable Inference and Generative Models", Toronto University of Computer Science, 2016, Available online at:
https://www.shakirm.com/papers/VITutorial.pdf
[33] Karthik Rao, "Using Variational AutoEncoders to Analyse Medical Images", Qure.ai Blog, 2016, Available online at:
https://blog.qure.ai/notes/using-variational-autoencoders

[34] Zheng Gong, Xiangdong Su, Yujiao Fu and Heng Xu, “Deep Variation Autoencoder with Topic Information for Text Similarity”, 3rd International Conference on Computational Intelligence and Applications, IEEE, 2018, Available online at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8711495
[35] Stanford Contextual Word Similarity (SCWS) Dataset, Available online at: http://www.bigdatalab.ac.cn/benchmark/bm/dd?data=Stanford%20Contextual%20Word%20Similarity
[36] Wallace Dalmet, Abhishek Das, Vivek Dhuri, Moinuddin Khaja and Sunil Hm. Karamchandani, “Siamese Manhattan LSTM Implementation for Predicting Text Similarity and Grading of Student Test Papers”, Proceedings of International Conference on Wireless Communication, 2019, Available online at: https://link.springer.com/chapter/10.1007/978-981-15-1002-1_60
[37] Nouha Othman, Rim Faiz and Kamel Smaili, “Manhattan Siamese LSTM for Question Retrieval in Community Question Answering”, OTM Confederated International Conferences “On the Move to Meaningful Internet Systems”, 2019, Available online at: https://hal.archives-ouvertes.fr/hal-02271338/file/Manhattan_Siamese_LSTM_for_Question_Retrieval_in_Community_Question_Answering__1_.pdf
[38] Aditya Thyagarajan, “Siamese Recurrent Architectures for Learning Sentence Similarity”, Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, 2016, Available online at: https://www.researchgate.net/publication/307558687_Siamese_Recurrent_Architectures_for_Learning_Sentence_Similarity#fullTextFileContent
[39] C. Leacock and M. Chodorow, “Combining local context and WordNet sense similarity for word sense identification”, MIT Press, 1998, Available online at: https://www.researchgate.net/publication/200045856_Combining_Local_Context_and_WordNet_Similarity_for_Word_Sense_Identification
[40] M. Lesk, “Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone”, Proceedings of the SIGDOC Conference, 1986, Available online at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.178.2744&rep=rep1&type=pdf
[41] Z. Wu and M. Palmer, “Verb semantics and lexical selection”, Proceedings of the Annual Meeting of Association for Computational Linguistics”, 1994, Available online at: https://www.aclweb.org/anthology/P94-1019.pdf
[42] P. Resnik, “Using information content to evaluate semantic similarity”, Proceeding of the 14th International Joint Conference on Artificial Intelligence, 1995, Available online at: https://www.ijcai.org/Proceedings/95-1/Papers/059.pdf
[43] D. Lin, “An information-theoretic definition of similarity”, Proceedings of the International Conf. on Machine Learning, 1998, Available online at: https://www.cse.iitb.ac.in/~cs626-449/Papers/WordSimilarity/3.pdf

[44] J. Jiang and D. Conrath, "Semantic similarity based on corpus statistics and lexical taxonomy", Proceedings of the International Conference on Research in Computational Linguistics, 1997, Available online at:
https://www.aclweb.org/anthology/O97-1002.pdf

[45] Rada Mihalcea, Courtney Corley and Carlo Strapparava, "Corpus-based and Knowledge-based Measures of Text Semantic Similarity", Proceedings of the 21st national conference on Artificial Intelligence, Vol. 1, 2006, Available online at:
http://web.eecs.umich.edu/~mihalcea/papers/mihalcea.aaai06.pdf

[46] Quora Question Pairs, Kaggle, Available online at:
https://www.kaggle.com/c/quora-question-pairs

[47] WordNet, NLTK Corpora, Available online at: http://www.nltk.org/nltk_data/

[48] Savvas Zannettou, Tristan Caulfield, Jeremy Blackburn, Emiliano De Cristofaro, Michael Sirivianos, Gianluca Stringhini and Guillermo Suarez-Tangil, "On the Origins of Memes by Means of Fringe Web Communities", AMC Internet Measurement Conference, 2018, Available online at:
https://arxiv.org/pdf/1805.12512.pdf?fbclid=IwAR3h0AwpV-ydLk6bC3VLCGlIVMSgy-E7VOoTh9ax9PO3MMUAj2UXkAFOKIE

[49] "Part of Speech (PoS) Tagging", Tutorialspoint, Available online at:
https://www.tutorialspoint.com/natural_language_processing/natural_language_processing_part_of_speech_tagging.htm

[50] Julian Gilyadov, "Word2Vec Explained", Hacker's Blog, 2017, Available online at: https://israelg99.github.io/2017-03-23-Word2Vec-Explained/

[51] Jeffrey Pennington, Richard Socher and Christopher Manning, "GloVe: Global Vectors for Word Representation", 2014, Available online at:
https://nlp.stanford.edu/projects/glove/

[52] Mohd Sanad Zaki Rivi, "Demystifying BERT: A Comprehensive Guide to the Groundbreaking NLP Framework", Analytics Vidhya, 2019, Available online at:
https://www.analyticsvidhya.com/blog/2019/09/demystifying-bert-groundbreaking-nlp-framework/