National Research University Higher School of Economics

Faculty of Computer Science
Bachelor's Programme in Data Science and Business Analytics

GROUP TERM PAPER

(Software Project)

The Model for Identifying Statements That Mean Quantitative Estimates of the Value of Global Markets from Scientific and Technological Texts in English and Their Visualization

Prepared by students: of Groups 191 and 193 in Year 3 (year of study),
Dzhkha Anika, Group 193, Year 3,
Fishman Maxim, Group 191, Year 3

Term Paper Supervisor:
Olga Veniaminovna Maksimenkova,
Candidate of Technical Sciences: specialty Information Systems and Processes,
Deputy Center Director: Center for Strategic Analytics and Big Data

Moscow
2022

Table of Contents

Abstract

Directly in the texts of many reports with market analytics, professional articles on specialized resources often contain statements about quantitative estimates of the value of global markets. For example, "According to IDC analysts' forecasts, the global market volume of artificial intelligence technologies, including software, hardware and services, will grow to $554.3 billion by 2024". However, different sources give different estimates of the same parameters and sometimes quote the same values on the contrary. However, it is not uncommon for such descriptions to contain errors in measurement units or to indicate the wrong order of values due to typographical errors, etc. Identifying and systematizing all available quantitative valuations of the various global markets is one of the challenges facing modern data analysts.

We will study the existing experience of solving this problem, propose and justify the optimal approach and criteria for identifying quantitative market valuations, develop a practical implementation of such a tool, which provides on request lists of detected valuations of the market of interest, as well as visualization of distribution of such valuations (as icons with signatures of valuation sources) according to two parameters: year specified in the valuation and size of market volume (scatter diagram, which we call the valuation corridor).

Introduction

Subject area

All work is related to NLP, preprocessing is based on spacy and nltk libraries, vectorization uses TfidfTransformer, CountVectorizer, TfidfVectorizer libraries. The Hugging face and Gensim kernels are also loaded for clustering texts and subsequent model building for n-grams. All other libraries are auxiliary and are not very important, although the project is Rule-based, we still use smart systems of

vectorization and clustering, because modern research with good metrics can not do without it.

Relevance of topic

The work of analysts is very much in demand these days, as are the financial exchanges with which analysts work in many countries. With the help of our tool and articles that appear in the news, analysts can identify particular factors affecting the market, see statistics on the market from the news and have an idea of how this or that market will behave in the future, which gives an advantage over those who manually analyze all the articles and news. This project reduces the analysis time for analysts and provides an automatic summary of the market, which has a positive impact on their work.

Objectives

The results of such a tool will allow analysts to get an overview of the size of a particular market, to see significant outliers of claims where errors have been made, and even to assess the retrospective feasibility of past forecasts.

Tasks

Tasks and its allocation among the team members will be described further in the plan.

Plan (Algorithm)

1. Receive the data that was automatically created and proposed by our mentor.

2. “Golden” data sample: manually mark up ("contains/shall not contain") the test sample and save it. (Anika)

3. Preprocessing: clean up the data before feeding it into the model-vectorizing. (Anika)

4. Word embedding: get vectors-embedding from the vectorizing model. (Maxim)

5. Clustering: perform clustering of these vectors with parameter analysis. (Anika)

6. Searching for best results: choose the clustering method with the best result and decide which cluster is our target. (Anika)

7. Searching for N-grams: save the most valuable n-grams from the best cluster into a list. (Maxim)

8. Create rule based model with such functionality: (Maxim)

   a. take the “Golden” sample

   b. check each text from the sample for n-grams from the list

   c. if any, put the text into the category 1-"contains statements of world market values".

   d. count accuracy metric by true-sampling

Data observation and preprocessing

Initial dataset

Let us start from a short data description that we are going to use during our project. It is a set of more than 28 thousand individual sentences: some of them are just meaningless, some of them fit our target and some of them have different meanings. It is quite a massive volume of data that was collected previously by a machine from different financial and other articles and therefore is needed to be processed in a proper way.

Here is an example from our dataset:

"Chairman Eric Xu said Huawei's sales revenue for 2019 was likely to reach 850 billion yuan (US$121 billion) — up roughly 18 percent from the previous year."[1]
"The AI and cognitive market in the UAE is set to grow between 25% and 30% year over year in the next two to three years."[2]
"Just looking at autonomous vehicles, for example, a latency of 1 millisecond means that data can be shared between cars traveling at 70 miles per hour in 31 millimeters."[3]

Table 1.

Preprocessing

Text preprocessing is a technique for cleaning text data and preparing it for use in a model. Text data comprises noise in the form of emotions, punctuation, and text in a different case, among other things. At first glance, there is a lot of noise in our data, such as excessive spaces, several hyphen marks, different cases, and so on. Let's begin by outlining the strategies that we used to clean the raw data:

1) Lower case: If the text is in the same case, it is easy for a machine to interpret the words because the lowercase and uppercase are treated differently by the machine.
2) Removing punctuations and numbers: we can directly use the string module

[1] (*Huawei Says 'Survival' Top Priority as Sales Fall Short*, 2019)
[2] (Hall & Violino, 2019)
[3] (Cherrayil, 2019)

with a regular expression to replace any punctuation in text with an empty string. But we should replace dollar and euro signs with words since they have specific meaning.

3) Removing Extra Spaces
4) Removing non-informative words: abbreviations, particles and prepositions (stopwords), words of length 2 and 3 (not informative)
5) Replacing big "bad" words, correcting mistakes: mistakes could appear as a result of deleting punctuations or ends of sentences.

Here is how sample of our data looks like after previous steps:

	sentence	Clean_sentence
0	Chairman Eric Xu said Huawei's sales revenue for 2019 was likely to reach 850 billion yuan (US$121 billion) — up roughly 18 percent from the previous year.	chairman eric said huawei sales revenue likely reach billion yuan dollar billion roughly percent previous year
1	The AI and cognitive market in the UAE is set to grow between 25% and 30% year over year in the next two to three years.	cognitive market grow percent percent year year next three years
2	Just looking at autonomous vehicles, for example, a latency of 1 millisecond means that data can be shared between cars travelling at 70 miles per hour in 31 millimeters.	looking autonomous vehicles example latency millisecond means data shared cars travelling miles hour millimeters
3	Chairman Eric Xu said Huawei's sales revenue for 2019 was likely to reach 850 billion yuan (US$121 billion) -- up roughly 18 percent from the previous year.	chairman eric said huawei sales revenue likely reach billion yuan dollar billion roughly percent previous year
4	And in fiscal 2020, it expects this number to grow 7.8% on a year-over-year basis.	fiscal expects number grow percent year year basis

Table 2.

For the next step in data preprocessing we need to choose between Lemmatizing and Stemming.

Stemming is the process of reducing a word to its basic stem, such as run, running, runs, and runed, all of which are derived from the same word as run. What stemming does is remove the prefix or suffix from words such as ing, s, es, and so on. The NLTK library is used to stem the words. The stemming technique is not utilized in production since it is inefficient and frequently stems undesired words. As a result,

another technology known as lemmatization entered the market to fix the problem. Lemmatization is a method of systematically reducing words to their lemma by comparing them with a linguistic dictionary. Therefore, we decided to proceed to lemmatization rather than to stemming, comparing two different lemmatization methods from two different libraries: NLTK and SpaCy.

Let's look at the diagram describing time performance of this two methods in three main test parameters (ran on the text of the Wikipedia article on NLP, which contains about 10 kB of text)[4]:

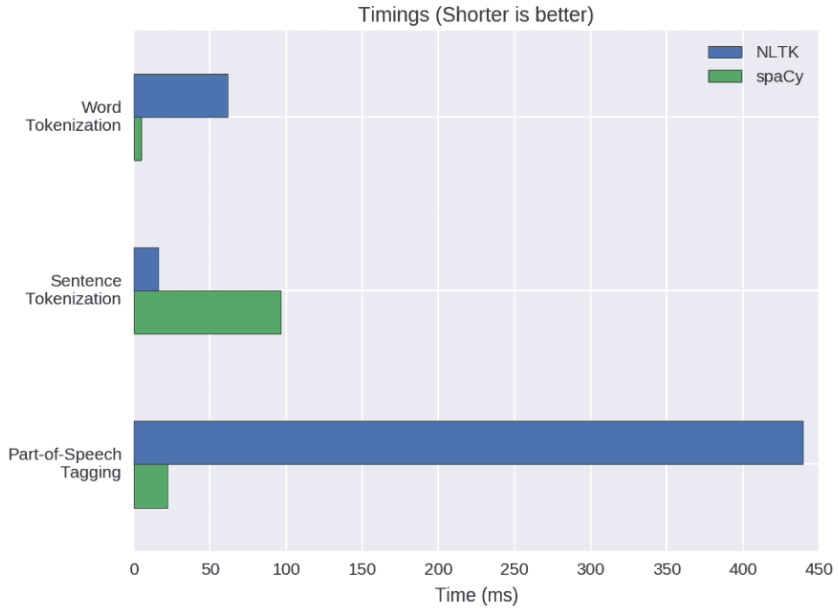


Figure 1.

In terms of word tokenization and part-of-speech tagging, spaCy clearly outperforms NLTK. Its low performance in sentence tokenization is due to disparities in approach: NLTK merely tries to divide the text into sentences. SpaCy, on the other hand, constructs a syntactic tree for each phrase, a more robust strategy that offers far more information about the text. After testing both methods on our firstly-preprocessed data we got the next result:

[4] (*Natural Language Processing in Python: NLTK Vs. SpaCy*, 2016)

	sentence	Clean_sentence	Clean_sentence_lemm_nltk	Clean_sentence_lemm_spacy	d_
30	T-Mobile reported revenue of $33.1 billion in the first nine months of 2019 compared to $31.9 billion during the same period in 2018, reflecting 3.9% YoY growth.	mobile reported revenue dollar billion first nine months compared dollar billion period reflecting percent growth	mobile reported revenue dollar billion first nine month compared dollar billion period reflecting percent growth	mobile report revenue dollar billion first nine month compare dollar billion period reflect percent growth	
31	According to Gartner's preliminary results, worldwide PC shipments fell 8.3% year-over-year in the calendar fourth quarter of 2015.	according gartner preliminary results worldwide shipments fell percent year year calendar fourth quarter	according gartner preliminary result worldwide shipment fell percent year year calendar fourth quarter	accord gartner preliminary result worldwide shipment fall percent year year calendar fourth quarter	
32	The latest data suggests gamers account for 18.6% of mobile users, which is expected to rise to 22.5% by 2024.	latest data suggests gamers account percent mobile users expected rise percent	latest data suggests gamers account percent mobile user expected rise percent	late datum suggest gamer account percent mobile user expect rise percent	
33	And back in fiscal 2016, Gaming was its second-fastest growing segment with	back fiscal gaming second fastest growing segment percent year year	back fiscal gaming second fastest growing segment percent year year	back fiscal gaming second fast grow	

Table 3.

On these examples we can see that with SpaCy we got better results, that is why we will further use the SpaCy variant of data. The last step of our preprocessing was cleaning bad splitted words after lemmatization which was not really necessary.

Word embedding - Vectorizing

Word embedding is a technique used in natural language processing (NLP) to characterize the representation of words for text analysis, mostly in the form of a real-valued vector that encodes the meaning of the word so that words that are close in the vector space are considered to be close in meaning.

Techniques

1) The easiest way to represent sentences as vectors is to use our dictionary length vector and place only one unit at the position corresponding to the word number in the dictionary, a method known as **one-hot encoding** (OHE).
   However, this method lacks the semantic closeness properties that are crucial for our clustering. In this method the terms are independent from one another.

2) A **bag of words** (BoW) is a vector representation that defines the appearance of words in a document. We only keep records of the number of words, ignoring grammatical nuances and word order. Because all information about

the sequence or structure of words in the text is deleted, it is referred to as a "bag" of words.

Even though the Bag-of-Words method is relatively efficient and simple to construct, there are several drawbacks to this technique, such as the fact that it does not respect the semantics of words and the vocabulary range constraints.

3) Another strategy is to rescale the frequency of words based on how frequently they appear in all texts, such that common terms like "the" that are likewise frequent across all publications are punished. This technique is known as the **term frequency-inverse document frequency**, or TF-IDF approach of scoring, and it is designed to indicate how significant a word is in a specific document.

   The Term-Frequency can be calculated as $TF(i,j) = n(i,j)/\Sigma\, n(i,j)$, where n(i,j) is the number of times the nth word occurred in a document, Σn(i,j) is the total number of words in a document. IDF score is calculated as follows: $IDF = 1 + log(N/dN)$,

   where N is the total number of documents in the dataset, dN is the total number of documents in which the nth word occurs. By multiplying these two features we got the TF-IDF score of a word in a document. The higher the score, the more valuable that term is in that specific paper.

   TF-IDF allows us to correlate each word in a document with a number that signifies the importance of a particular word in that document. Documents containing comparable, relevant phrases will have similar vectors, exactly what we are trying to get. However, the TF-IDF approach has a very limited vocabulary, rules could only be defined on words that had been observed in numerous training samples.

4) **Word2vec** is a widely known method for learning word embeddings with a two-layer neural network (Skip Gram and Common Bag Of Words (CBOW)). It takes a text corpus as input and returns a set of vectors as output. The main

difference between these two algorithms is that CBOW predicts a target word based on context, whereas skip-gram predicts a target context based on a word. The main disadvantage is that Word2Vec models create embeddings that are context-independent: that is, there is only one vector representation for each word. Different meanings of the one term are integrated into a single vector.

5) **Bert vectorization or Hugging face** vectorization uses the semantic kernel 'all-MiniLM-L6-v2', Bert itself as a model uses this kernel but is built not only on it, but on a bunch of similar semantic kernels. It is an illustration of a sentence-transformers model: It translates sentences and paragraphs into a dense vector space and may be applied for such tasks as clustering and semantic search. These embeddings may then be compared, for example, with cosine-similarity to detect phrases with similar meanings. The closer words are to sense, the less distance between their vectors will be.
   The advantage of this kernel is that it can very well identify the meaning of whole sentences and separately taken from the context words and vectorization at this stage is supposed to be better at this method. BERT embeddings are context-dependent because they create embeddings that allow us to have many vector representations for the same word based on the context in which the word is used. It also works much faster than all other methods of vectorization.
   The disadvantage of such a kernel is that we can't look at and change the settings of semantic load of our individual words, because some words have a great sense in the context of world markets estimation, and the given kernel most likely does not understand that and treats them as a regular word.

Implementation details of vectorizing methods:

We applied 2-5 methods on our data and saved each result in the corresponding variables which will be used further in clustering: ‘bow’(BoW embeddings), ‘X_train_tfidf’(TF-IDF embeddings), ‘sent_vectors’(W2V embeddings),

'corpus_embeddings' (Bert embeddings).In this block we are working with the data that was preprocessed lemmatized previously by SpaCy ('df.Clean_sentence_lemm_spacy').

As for the BoW algorithm: firstly, we need to find all unique words in the data; secondly, we compose words and their frequency of occurrence in data into pairs (here we can see that the most frequent are 'million', 'percent', 'year', 'dollar', 'billion'), then, we visualize the most common (Figure 1) and rare words. Let's observe results for the further work:

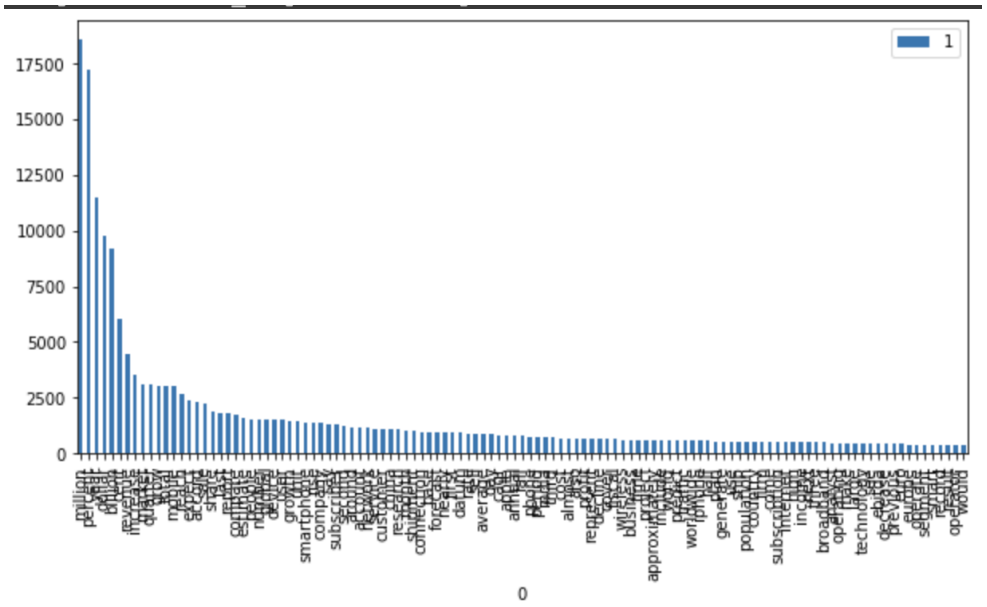


Figure 2.

On the figure above we can see that the words million and percent we can meet almost everywhere, and there are about the top 6 most popular words, and other frequent words are much rarer, but with approximately the same frequency. There are 4929 words in the data that can be found less than 2 times which is quite a big number, and therefore the histogram for this one is very inaccurate.

While generating a bag of words, we need to set parameters for the 'CountVectorizer()' method from the 'sklearn.feature_extraction' library. We set 'max_features=1500' that is how many terms we take to consider, 'min_df=5' if

the word occurs less number of times we ignore it and 'max_df=0.6' is an upper threshold, i.e. if a term appears in a larger percentage of the documents then the model ignores it. Then we just fit the model with clean sentences and get our embeddings. To observe words again, we can just use the 'get_feature_names()' method.

Considering the next vectorizing model, TF-IDF has a similar algorithm: we use 'CountVectorizer()' firstly, train the model, but then we use 'TfidfTransformer()' method from the same library without tuning any parameters. Representations are stored in the 'X_train_tfidf' matrix variable, which can be encoded back using the 'TfidfVectorizer()' method.

In the next vectorization, Word2Vec, we are using the model 'Word2Vec()' from the 'gensim' library which is usually known as one of the fastest for training by vector embeddings. The parameters of the model are our dataset of «clean lemmatized sentences», 'size' ('vector_size') - dimensionality of the word vectors, 'window' which is the maximum distance between the current and predicted word within a sentence, 'workers' – how many worker threads to train the model, 'min_count' – ignores all words with total frequency lower than this. The gensim semantic model transforms our sentences based on w2v based on the internal rules of the gensim kernel; it does it in multiple threads (workers=4) and in a large space (the distance between vectors is max 5) .

Finally, the Bert model implementation we use is quite simple and also very takes the minimal time for running: we just need to install 'sentence_transformers' library and use the model 'SentenceTransformer' with the only parameter which is a semantic core ('all-MiniLM-L6-v2'). Then we just feed our sentences to the model and get the result. We are using a contrastive goal to fine-tune the model. Technically, we evaluate the cosine similarity between

each potential sentence pair in the batch. Next, we apply the cross comparing them with real pairings. In fact, all semantic kernels on HaggingFace have similar parameters, but the one chosen works faster.

Unsupervised learning - Clustering

The training data in certain pattern recognition tasks consists of a set of input vectors x with no matching target values. In such unsupervised learning problems, the objective may be to find groupings of comparable examples within the data and this is called clustering. In our paper we will use 3 different clustering methods to each of embedding representations obtained earlier to receive the best result.

The optimal number of clusters into which the data may be grouped is a basic step for any unsupervised technique. One of the most popular approaches for determining the optimal value of k is the Elbow Method[1]. After visualizing this method, we must choose the value of k at the "elbow", or the point when the distortion/inertia begins to decrease linearly. Distortion is determined as the average of the squared distances from the respective cluster centers. The Euclidean distance measure is generally implemented here. Inertia is defined as the total of the squared distances between samples and their nearest cluster center. In our case the optimal number of clusters can be not one number but several

Clustering methods that we use:

1) **K-Means:**

The most basic, as well as the most inaccurate. It separates the vector space element set into a fixed number of clusters k. The concept behind K-means clustering is that a successful clustering is one with as little within-cluster variation as possible. The main downside of K-means clustering is that we have to pre-specify the number of clusters. But even with this number it can be inconclusive.

To illustrate some results we plot the dependencies of the sum of squares of distances between an element and the center of its cluster for a different number of clusters.The distances also depend on the method of embedding.

Let's look at some of the visualizations of KMeans clustering on embeddings generated by BERT model:

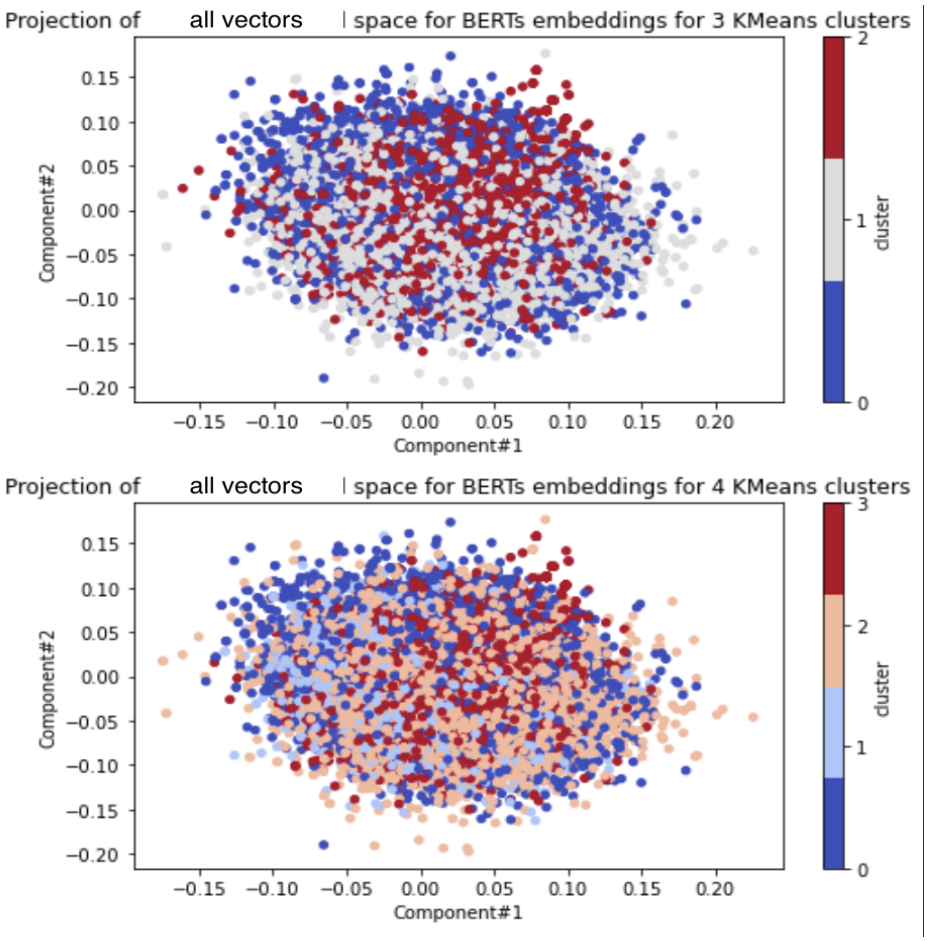


Figure 3.

Observing plots above we can assume that clusters are not isolated at all or that KMeans clustering just does not fit our embeddings. However, it should be noted that vectors are placed in an unusual way and are clustered from the center to the outer shell, thus, let's examine others and see how visualizations are sometimes deceptive and why it was worth looking at the assigned clusters manually.

The next visualization of the same clustering method we can look at is for W2V vector representation:

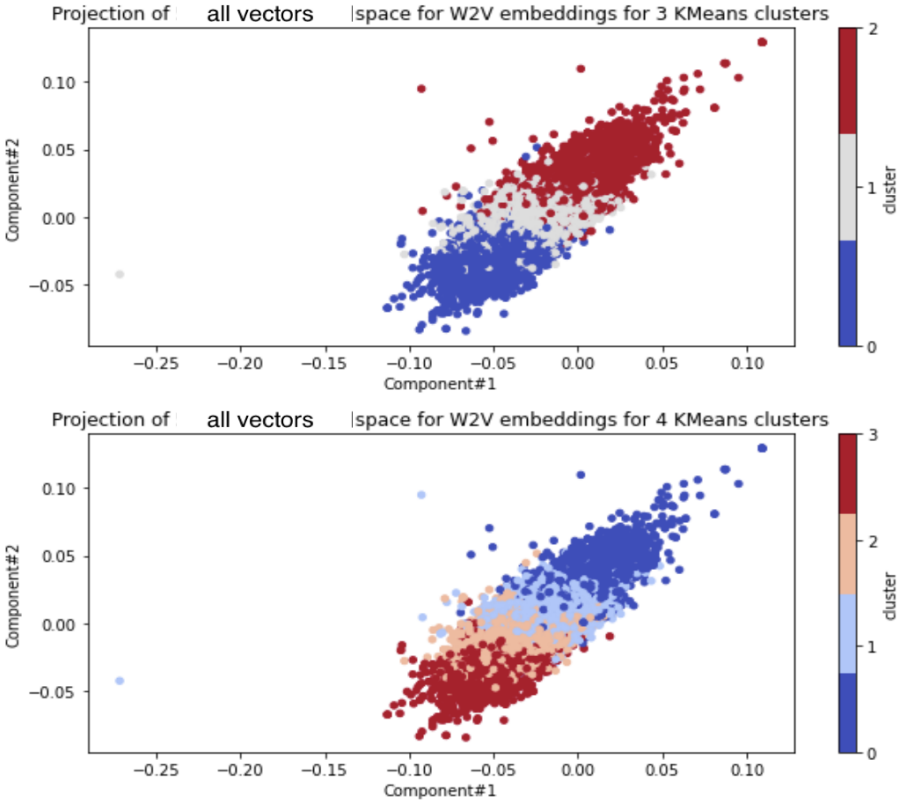


Figure 4.

Here we can draw the conclusion that clustering worked well, even though they are not isolated, it seems that there are clear clusters and the embedding model was also productive, therefore the false prediction could be that in the result there should be at least one good cluster that we need.

Visualizations for BoW and TF-IDF mean nothing useful to us, since we cluster vectors like [1,0,0,0,0,0,0,0,1,0,0,0,0,0...] in a matrix form, consequently, we will just consider conclusions for them, rather than plots.

2) DBSCAN:

Density-based spatial clustering of applications with noise. It's a non-parametric density-based clustering technique that takes a set of points in a space and clusters them together that are closely packed together (points with many nearby neighbors), while labeling points that are alone in low-density regions as outliers (whose nearest neighbors are too far away). Unlike k-means, it does not require a beforehand specification of the number of clusters. One more advantage is that the algorithm only

requires two parameters and is generally unaffected by the ordering of the points in the database. The concern here is that determining a meaningful distance criterion epsilon might be challenging if the data and scale are not well understood.

The first two illustrations after DBSCAN clustering we are going to consider only the first 10% of clustered vector space of W2V embeddings for visualization with 2 different epsilon hyperparameters :

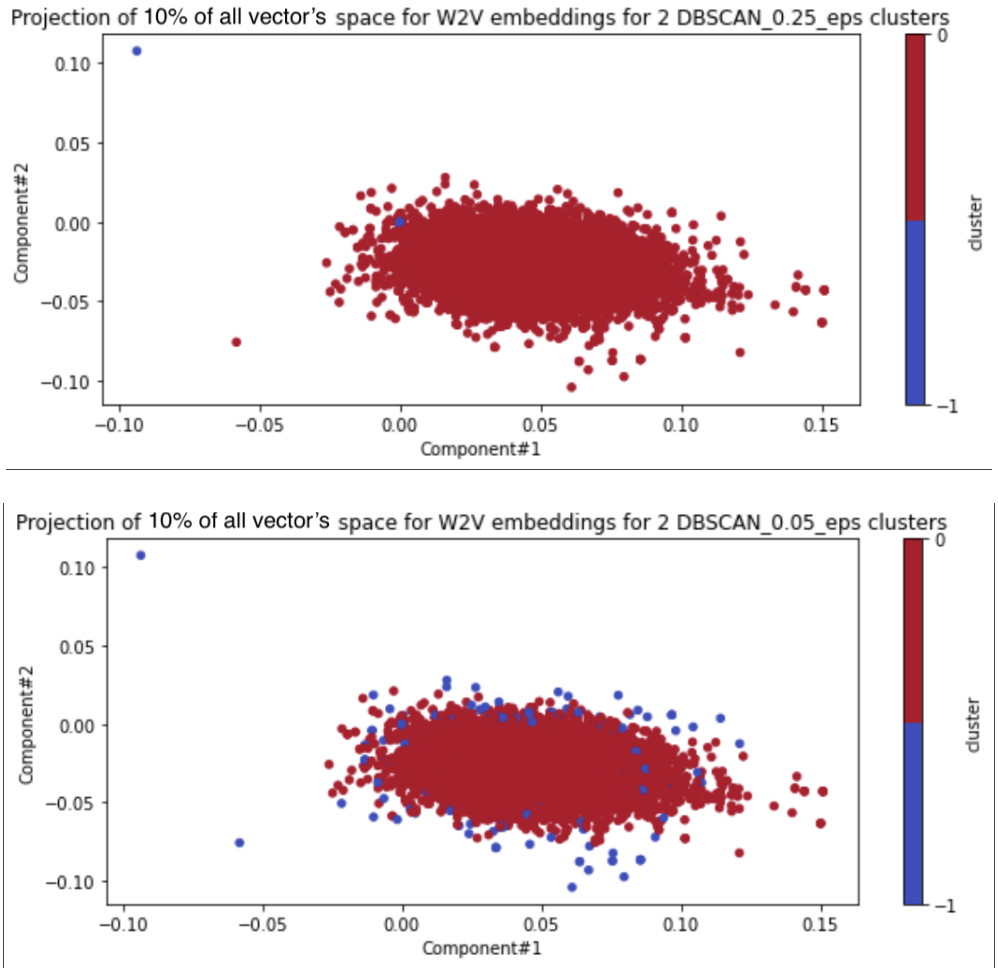


Figure 5.

The only notable point is that in the second case we see more elements of -1 cluster, they are not highly isolated from the other cluster, but the blue cluster practically surrounds the red cluster on the borders.

The following two graphs reflect the same as two above, just for the Bert

embeddings with the two possible epsilon parameters that we considered.

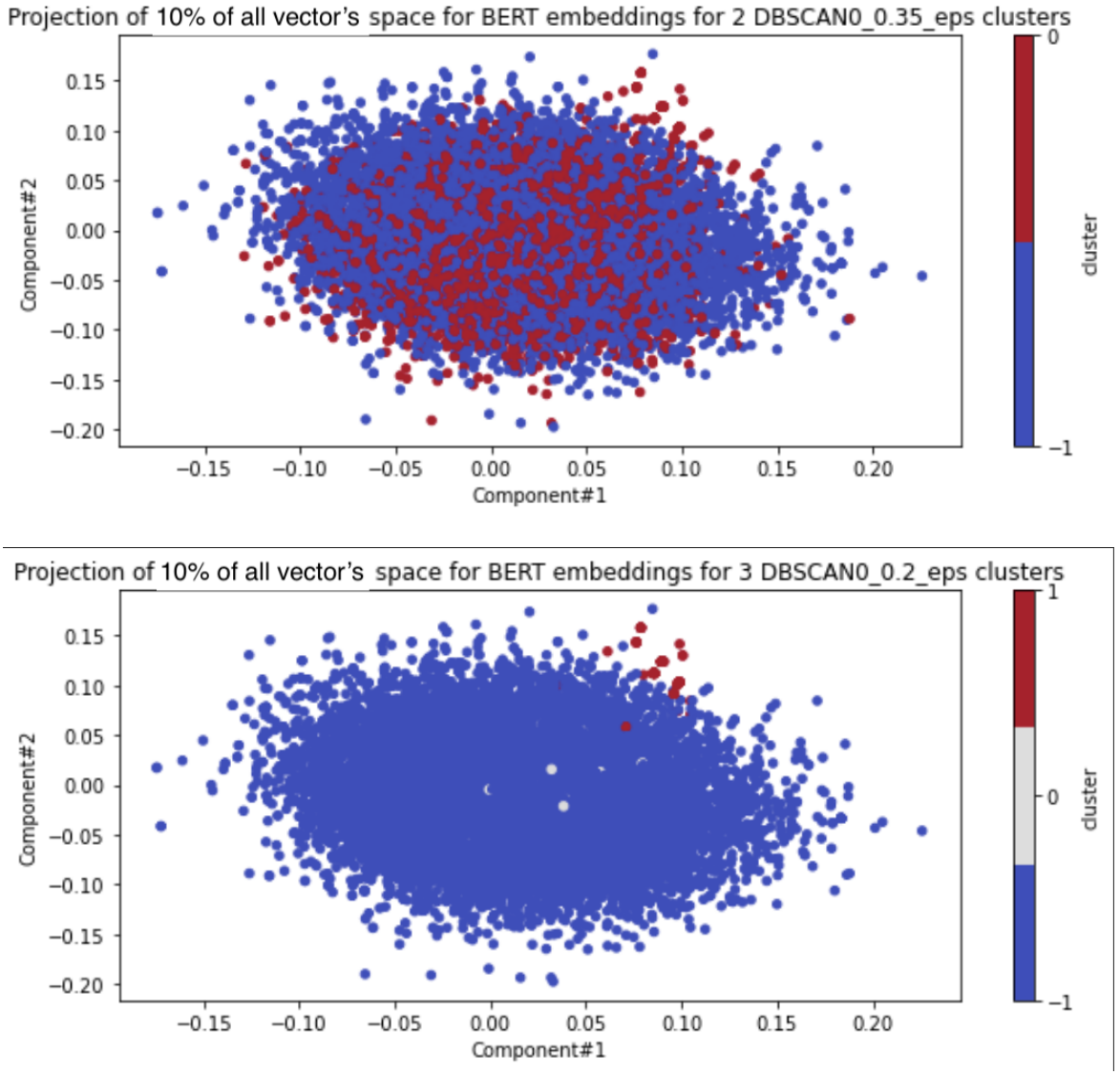


Figure 6.

It can be seen that the first graph is excessively mixed and does not provide us with clear results. As for the plot with three clusters, almost insignificant visualization, perhaps this is because of the limitation on the vector space that we have set, it turns out here too difficult to establish any assumptions about how good clusters we got. It is also possible that this clustering method did not produce good results in this case and all clusters are not suitable for us at all.

3) Hierarchical agglomerative clustering (HAC)

is an alternative strategy that does not require us to settle to a certain value of K. Hierarchical clustering outperforms K-means clustering in that it produces an

appealing tree-based representation of the observations known as a dendrogram. In an agglomerative approach each observation begins in its own cluster, and pairs of clusters are combined as one advances up the hierarchy. The classic algorithm for HAC has a time complexity of $O(n^3)$, considering it still slow even for moderate datasets. In practice it takes a lot more time to process than the two methods before. However, this method has the unusual benefit of being able to apply any valid measure of distance. In reality, the observations themselves are unnecessary: all that is required is a distance matrix.

Few more visualizations we are about to consider of HAC clustering for W2V and Bert are as follows:

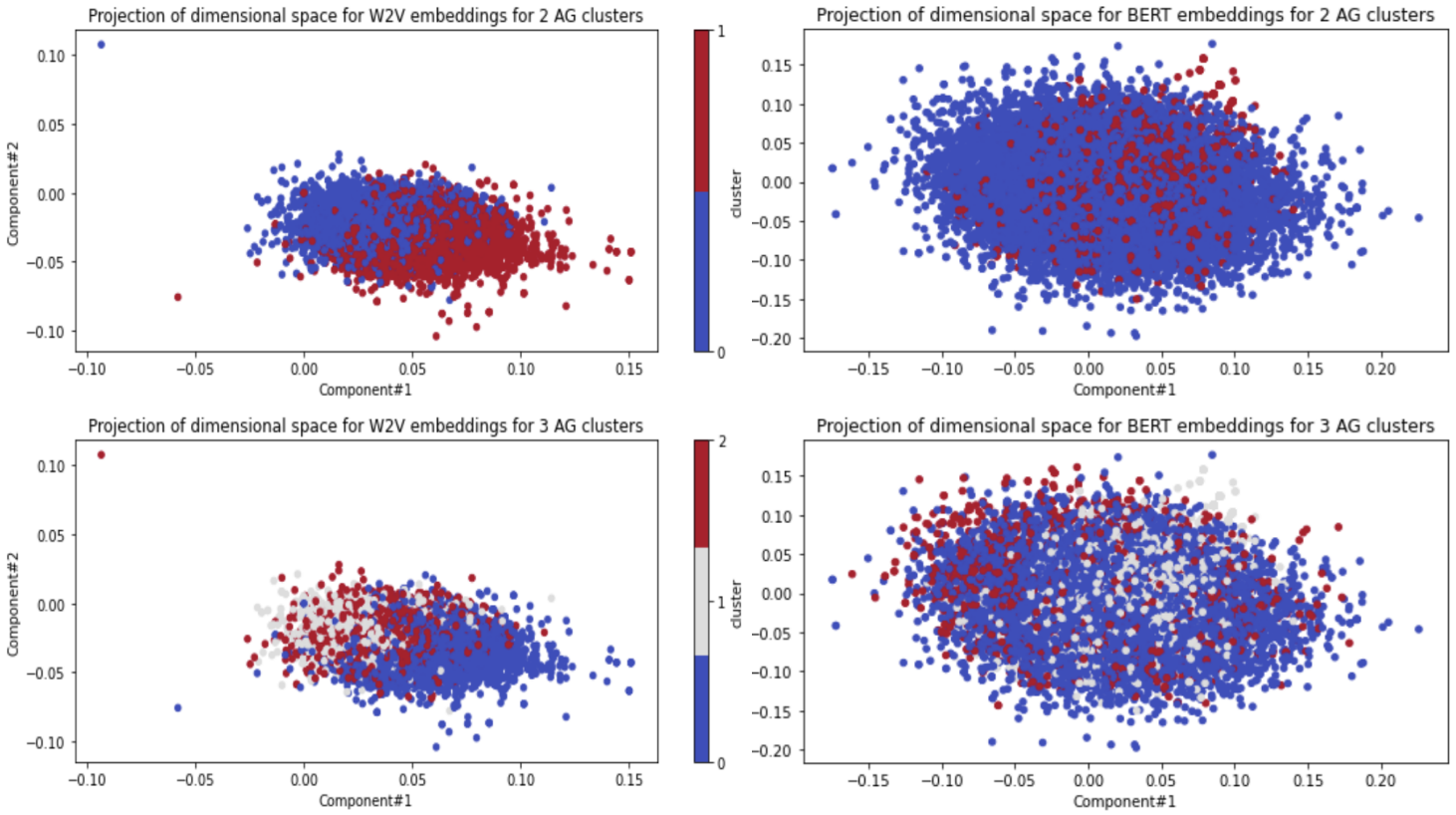


Figure 7 and Figure 8.

The graphs displayed are different from the observed before, here we can see that the density of cluster "2", for example, on the right bottom graph is closer to the Component 2, whereas in KMeans the points of each cluster are more scattered and hardly can be described with the same main pattern. Plots on the right-hand side are

also not such separated as in the K-Means, so it could be the reflection of some mix in the clusters, but we still cannot not draw this conclusion before the hand check.

Let us now look at our actual cluster results and see which clustering method and which embedding composed gave us the best result and which did not.

Exploring clustering results

Here we have to manually examine ready-made clusters searching for the target one in the output file that consists of: original sentences, two lemmatized versions (not important in this block), and columns with the cluster of each embedding method with each clustering method (with different parameters, e.g. number of clusters or epsilon tuning). Let us remind you how the sentences containing quantitative estimates of the value of global markets should look: "According to IDC analysts' forecasts, the global market volume of artificial intelligence technologies, including software, hardware and services, will grow to $554.3 billion by 2024" .

Important points in determining the right cluster:

1) suitable statements must be in one or at most two clusters

2) if the cluster seems to be suitable, you should check the other clusters and check if there is another criterion by which they were classified

After looking through all the clusters there are illustrative examples of a poor clustering, good but not suitable, and very good appropriate one. The bad example can be illustrated by 3-KMeans clustering with BoW embeddings, 6-KMeans with W2V: all clusters are just mixed with all types of texts (20-30% or more not target) even though the cluster “1” has the largest number of target texts.

At this point, we also can conclude that the best number of clusters will be 2-3, because 2 clusters are unlikely to show a clear separation between the target and unnecessary texts, and more than 3 will yield trash clusters, which contain both targeted and non-target texts and will cause a problem with the search n-grams in the

future.

Our next observation is the result of DBSCAN(epsilon=0.2) with Bert embeddings: here we have three distinct clusters describing the state of the market, but the problem is that all three clusters contain both target and non-target phrases, which is extremely problematic for us and provides no useful clustering.

	sentence	BERT_embed_DBSCAN_0.2_eps
3386	were NIS 47 million (US$ 12 million) in Q4 2016, a decrease of 16% from NIS 56 million in Q4 2015.	1
3392	Annual in 2016 totaled NIS 2,752 million (US$ 716 million), a decrease of 8% from NIS 2,992 million in 2015.	1
3393	In 2016, were NIS 3,544 million (US$ 922 million), a decrease of 14% from NIS 4,111 million in 2015.	1
3395	in 2016 were NIS 105 million (US$ 27 million), a decrease of 27% compared with NIS 143 million in 2015.	1
3396	in Q4 2016 totaled NIS 498 million (US$ 130 million), a decrease of 9% from NIS 550 million in Q4 2015.	1
3397	totaled NIS 45 million (US$ 12 million) in 2016, compared to NIS 47 million in 2015, a decrease of 4%, mainly reflecting a decrease in income from the unwinding of trade receivables.	1

Figure 9.

Cluster "-1" reflects more about market conditions, "0" is about growth, "1" is more about decline. That is why we can say that the result is good, but not meaningful for our goal.

Moving forward to our best result, we can state that the best possible outcome was obtained using 3-KMeans clustering with Bert vectorization. In the cluster "1" with 8648 rows we got the target and the other 2 are trash clusters. This is the most accurate result with good specifications and parameters.

	sentence	Bert_embed_K_means_3_clusters
0	Chairman Eric Xu said Huawei's sales revenue for 2019 was likely to reach 850 billion yuan (US$121 billion) — up roughly 18 percent from the previous year.	1
1	The AI and cognitive market in the UAE is set to grow between 25% and 30% year over year in the next two to three years.	1
3	Chairman Eric Xu said Huawei's sales revenue for 2019 was likely to reach 850 billion yuan (US$121 billion) -- up roughly 18 percent from the previous year.	1
4	And in fiscal 2020, it expects this number to grow 7.8% on a year-over-year basis.	1
5	So Nvidia's automotive revenue has doubled from $320 million in fiscal 2016 to $640 million in fiscal 2019.	1
6	Game consoles were added in fiscal 2017.Nvidia's Tegra revenue rose almost threefold from $559 million in fiscal 2016 to $1.54 billion in fiscal 2019.	1
7	Back in fiscal 2016, Nvidia earned 5% of its revenue—or $264 million—from licensing.	1
8	Moreover, it's expected to outperform the market, even in 2019.The WSTS (World Semiconductor Trade Statistics) expects global semiconductor revenue to fall 12.8% in 2019.	1

Figure 10.

Searching for N-grams

N-gram refers to an unbroken sequence of n elements from a given sample of text. Those elements might be morphemes, syllables, letters, words.

In our case, the n-grams will represent the words that somehow influenced the sentence to be definite to a certain class.

We had to come up with an algorithm to find these n-grams so that they would cover the sentences of the class we wanted (Evaluative sentences) quite clearly and not occur in sentences of another class (Sentences of another nature)

The algorithm for finding such n-grams:

1) Let's find unique n-grams that can cover most of the clustering of sentences under the desired class
   1.1 Print unique n-grams for the class we want and for
   all other classes and see their statistics.
   look at 2-5 diagrams
   1.2 Delete repeating n-grams.

2) Let's see n-grams that are not unique, i.e. they occur in several clusters, but more often in the target cluster
   2.1 Let's get visual statistics for 2-5-grams which ones occur more often repeated
   2.2 Output the statistics of maximum occurrence of grams in the target and non-target texts
      2.2.1 Choose an optimal coefficient and put these n-grams in defining for target class

3) Create and write into the file unique and "preunique" n-grams

N-grams observation

Unique n-grams

We will consider as unique n-grams those grams that occur only in a particular cluster. We created two variables ‘aim’ and ‘other’ to represent grams from the target cluster and non-target cluster, so the grams from the target cluster will belong to class 1 and the others to any class but not 1.

Thus a block was written with descriptive statistical functions that output:

1)Maximum frequency of n-gram mentions

2)Minimum frequency of mentioning the n-gram

3)Average mentioning frequency of the n-gram

4)Modal frequency of n-gram mention

5)Standard deviation of n-gram mention

```
found 1915 unique n-grams in the target texts
maximum frequency of mention of the n-gram in the target texts 131
minimum frequency of mention of the n-gram in the target texts 3
average frequency of n-gram mentions in the target texts 4.8877284595300265
modal frequency of n-gram mention in the target texts 3
standard deviation of the frequency of n-gram mention in the target texts 4.826864640245



found 9483 unique n-grams in non-target texts
maximum frequency of n-gram mention in non-target texts 151
minimum frequency of n-gram mention in non-target texts 3
average frequency of n-gram mentions in non-target texts 5.400189813350206
modal frequency of n-gram mention in non-target texts 3
standard deviation of the frequency of n-gram mention in non-target texts 5.952448568319376
```

Figure 11.

There is a summary of bigrams above, i.e. we take 2 words in a sentence and see that the number of unique ngrams in the target cluster, much lower than unique in other clusters, this information gives us little, but information about how often there are n-grams suggests that it is worth looking at other grams to play on the word mention barrier and set it different values in the future. So let's display the information on all n-grams and compare it:

Unique aim n-grams	bigrams	tergrams	fourgrams	fivegrams
Maximum frequency of n-gram mentions	131	121	121	109
Minimum frequency of mentioning the n-gram	3	3	3	3
Average mentioning frequency of the n-gram	4.8877284595300265	4.907598197037991	4.749545178896301	4.384480337078652
Modal frequency of n-gram mention	3	3	3	3
Standard deviation of n-gram mention	4.826864640245	5.7939640607447815	5.467726700300944	4.077031875684038

Table 4.

As you can see, there is little to see from the single table on target n-grams, we can only understand that the minimum number of mentions and the modal frequency of mentions will not play any role in further research.

Other aim n-grams	bigrams	tergrams	fourgrams	fivegrams
Maximum frequency of n-gram mentions	151	220	38	17
Minimum frequency of mentioning the n-gram	3	3	3	3

Average mentioning frequency of the n-gram	5.400189813350206	4.2719546742209635	3.741441321433329	3.5964680640369697
Modal frequency of n-gram mention	3	3	3	3
Standard deviation of n-gram mention	5.952448568319376	3.6287866988458006	1.5346075486413961	1.122646033830429

Table 4.

From the comparisons of tables 4 and 5 you can notice a couple of interesting facts:

1) The maximum frequency of n-grams in target and non-target grams differs by times, which gives us to understand that maybe 4-5 grams will be very useful for us, because the maximum frequency in target class they have 121 and 109 when in non-target they have 38 and 17, which will play into our hands when accounting for them in the Rule-based method.
2) Distributions do not play a special role for our task, because all the necessary observations are visible from other statistical indicators.

Then all the lists of n-grams are displayed just for clarity, they are displayed immediately with an indicator of how often they occur in the text through a function that counts the maximum occurrence and minimum, i.e. works through the variable-counter count:

Thus it was noticed that there are repeated n-grams in the whole plurality (2-5), such as dollar dollar, million million million etc. Such grams appeared due to lemmatization and removal of bad words, most likely it appeared due to removal of punctuation marks and small words like company abbreviations or the "year to year" proverbs were combined. Delete such n-grams if they have at least 2 repeated words.

For this purpose we wrote a function find_dublicate_word which ran each n-gram through itself and returned a value either false or try and then deleted or left the n-gram. Subsequently, we did not remove many n-grams, but we were able to increase the accuracy of the model by several percent thanks to the following n-grams.

The stage with unique n-grams comes to an end, let's check how many unique n-grams from the target cluster cover sentences for clustering. What it means:

If we encounter an n-gram that we have collected from the target cluster, the sentence that will have this n-gram can immediately be attributed to the cluster containing an estimated description of the market condition, because such an n-gram is unique to the target cluster.

For this check:

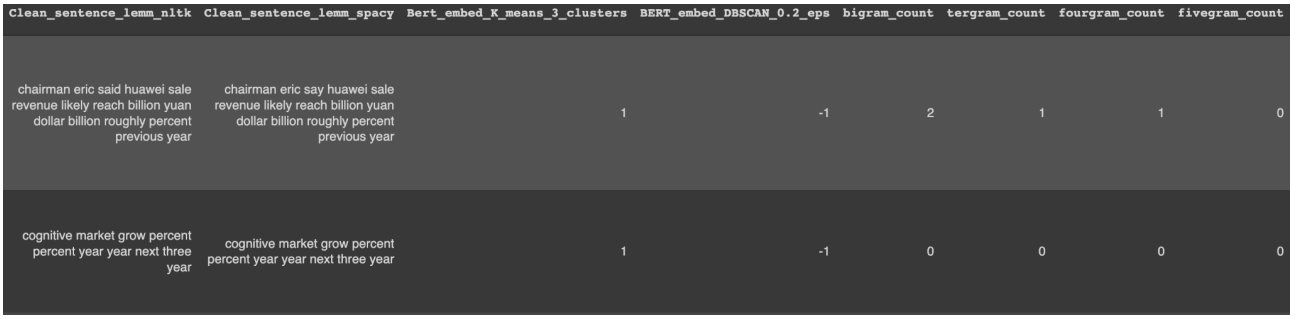
1) Let's derive the number of sentences from the target cluster: 8648;
2) Output the number of sentences from the non-target cluster:20287;
3) Let's add columns with grams to the sentences, which will show how many unique grams each sentence contains;
4) Check by rows which have cluster 1(target) but do not have any unique grams, thus we will get the share of presumably incorrectly defined sentences by the
5) Future model and we will see that such sentences are only 2220, which is only 25% of the whole target cluster.

Thus, we can claim that our model will work with 75% accuracy if we cluster only by unique n-grams, but we can improve the accuracy of the model if we take n-grams that occur in both the target and non-target cluster, but only consider those that occur more often in the target cluster.

Not unique n-grams

From the previous observations, we see that the number of n-grams in non-target clusters prevails, and that adding non-unique n-grams may not only improve the accuracy of the model for us, but may also screw it up because of the number of repetitions of them in the texts

After displaying the table, there were some places that are not visible with unique n-grams, for example:

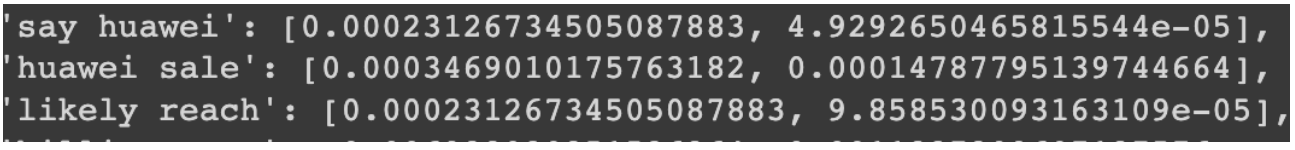


Clean_sentence_lemm_nltk	Clean_sentence_lemm_spacy	Bert_embed_K_means_3_clusters	BERT_embed_DBSCAN_0.2_eps	bigram_count	tergram_count	fourgram_count	fivegram_count
chairman eric said huawei sale revenue likely reach billion yuan dollar billion roughly percent previous year	chairman eric say huawei sale revenue likely reach billion yuan dollar billion roughly percent previous year	1	-1	2	1	1	0
cognitive market grow percent percent year year next three year	cognitive market grow percent percent year year next three year	1	-1	0	0	0	0

Figure 12.

Let's take the second sentence from the table, you can see that it has cluster 1, but 0 unique n-grams of all kinds, so we see proof that not all sentences can be covered by unique n-grams and such gaps should be solved.

Function pre_aim_ngrams, which collects a ngram dictionary where key is ngram, value - list with specific frequency of occurrence in target texts and non-target texts specific in meaning *number of mentions in a group of texts*/*total number of texts in the group* there are 2 values in the list: 1 - in the target group of texts, 2 - in the non-target one.



```
'say huawei': [0.00023126734505087883, 4.9292650465815544e-05],
'huawei sale': [0.0003469010175763182, 0.00014787795139744664],
'likely reach': [0.00023126734505087883, 9.858530093163109e-05],
```

Figure 13.

Thus we see a non-unique n-gram, in this case a bigram where the first value is the specific frequency in target sentences and the second value is the specific frequency in non-target sentences. It is clear here only that some are more common in both target and non-target sentences.

Next, by the same principle as in the unique ngrams, let's remove the repeating grams and there are much more of them by about 100-200 more repeating grams for each gram from 2 to 5.

Unnamed: 0	sentence	Clean_sentence	Clean_sentence_lemm_nltk	Clean_sentence_lemm_spacy	Bert_embed_K_means_3_clusters	BERT_embed_DBSCAN_0.2_eps	uniq_aim_ngrams	pre_bigram_count	pre
0	Chairman Eric Xu said Huawei's sales revenue for 2019 was likely to reach 850 billion yuan (US$121 billion) — up roughly 18 percent from the previous year.	chairman eric said huawei sales revenue likely reach billion yuan dollar billion roughly percent previous year	chairman eric said huawei sale revenue likely reach billion yuan dollar billion roughly percent previous year	chairman eric say huawei sale revenue likely reach billion yuan dollar billion roughly percent previous year	1	-1	4	10	
1	The AI and cognitive market in the UAE is set to grow between 25% and 30% year over year in the next two to three years.	cognitive market grow percent percent year year next three years	cognitive market grow percent percent year year next three year	cognitive market grow percent percent year year next three year	1	-1	0	8	

Figure 14.

Here we add the same method as in the unique n-grams columns with pre-grams (Non-unique which occur in both target and non-target texts) And our problem is solved with the fact that the sentence has no n-grams.

Next, there is a research work on what minimum occurrence of n-grams to choose so that this coefficient does not spoil the accuracy of our model, consider, for example, bigrams:

```
maximum frequency of mentioning n-gram in the texts 19
minimum frequency of mentioning n-gram in the texts 0
average frequency of mentioning an n-gram in texts 6.604301572617946
modal n-gram mention rate in texts 6
standard deviation of the frequency of n-gram mentioning in the texts 2.946601547955196
```

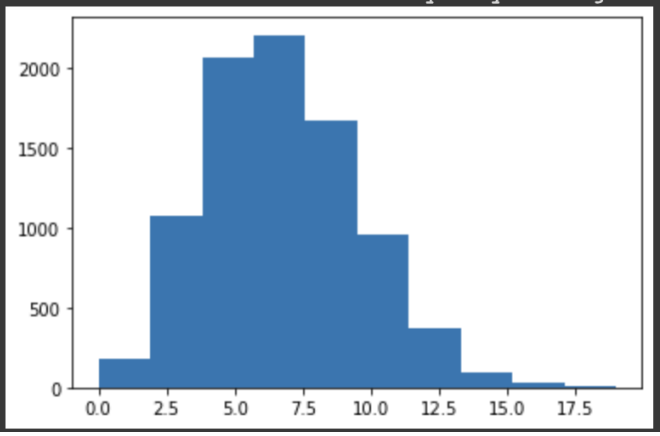


Figure 15.

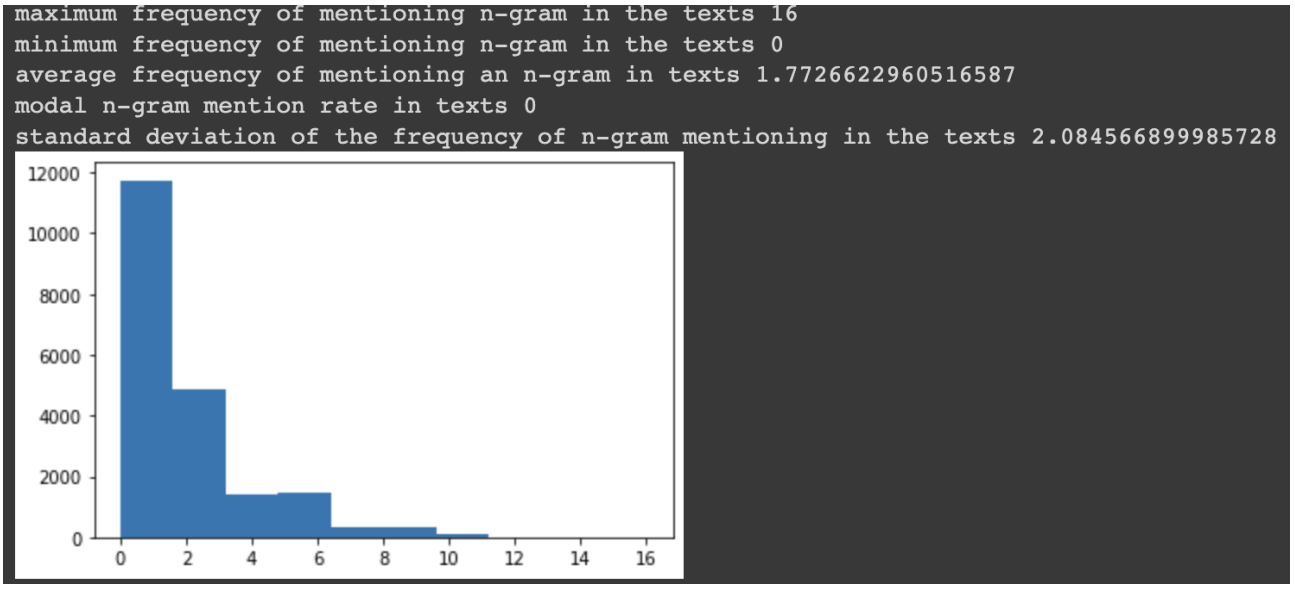


Figure 16.

The upper graph shows the ratio of the number of n-grams to their occurrence in sentences for the target cluster, the lower graph for non-target clusters and here we see that if we set the model to 17, then we will have only those that should be in the target clusters, ie they will be unique, but we must choose a parameter that we have the majority of target proposals and as little as possible for non-target, so we stopped at 8.

The values we obtained with this parameter:

- how many target texts contain more than 8 pre-target ngrams: 2216
- how many non-target texts contain more than 8 pre-target ngrams: 219
- if we make the threshold for these bigrams at 8, then there is a 90% probability that the markup will be correct and thus we will add 223 n-grams.

For tergrams the accuracy will be 79% for four-grams 45% and for five-grams 25%, these conclusions are based on the enumeration of parameters for this barrier n-grams, so our main grams will be bigrams, despite the fact that in the beginning you might think that this will be five-grams or four-grams because of their uniqueness. We add these n-grams to a file called preunique n-grams and use it as a file with unique n-grams for the target cluster in the rule-based model.

Rule-based model

A rule-based model is composed of a combination of rules that may be performed by general-purpose simulation and data analysis.

In our case, the rule-based model will work on the rules associated with n-grams.

The first thing we do is preprocess the sentences again, since each part in our code is independent and in order to be able to test different parts of the code separately (This makes the process very easy, since we don't need to restart the whole project) we get the same thing as after the preprocessing part from the beginning of the code)

In the model there are two independent cycles applied:

1) Take n-grams that we have marked as unique for the target class and if we find one, we assign class 1(target) to the sentence, if not, we assign 0;
2) Take pre-n-grams and if we cross a barrier (in this case it is more than 8 in the sentence for bigrams) then we assign a class 1(target) to such a sentence, if not, then 0.

So we get a model that works only on conditions, in this case for, and does not use smart machine learning models to define classes of prepositions.

After we have loaded the functions, we open the n-gram files and our model simply walks through them and labels the sentences True-False and then replaces them with 0 and 1 with a separate function.

In the final part, we have to compare the markup between the golden markup that we did in the file with a separate class column and what the model has marked up, the metric we chose is accuracy and it's 0.806 which means almost 81%, so this is a really good result for a rule-based model.

Conclusions

The result of our work can be observed on the Github source: https://github.com/FishmanMax/-Model-for-Identifying-Statements-That-Mean-Quantitative-Estimates

Task was wider than expected, so the main purpose was not just to create and tune a model, but to use and compare different Machine Learning and NLP tools and methods to achieve the best result, so that our tool could be reused for different analytic purposes.

Wide comparative analysis of methods was made and the review of Bert embedding model has significantly improved model consistency and should be noted for further developments of our tool.

The accuracy of our model does not depend in any way on the model itself. In this project, it is not possible to simply prescribe all sorts of conditions in the model itself without the auxiliary subparagraphs that were carried out earlier so that there is good accuracy.

As we have seen, the accuracy of the model is significantly affected by several factors:

- Good vectorization, preferably using pre-trained semantic kernels (Part of Burt's kernels and gensim), otherwise we will not be able to make advantageous clusters in the future.
- A good choice of cluster, without it you can not normally select unique n-grams from clusters, which means that with a bad choice of clusters model accuracy will be significantly lower than presented in the project.
- Accurate selection of the barrier n-grams for non-unique ones, because otherwise the accuracy may not increase, but drop and confuse the model when partitioning sentences.

Thus, in this project, research work with each file plays a big role, not all the work can be done by the code because it is necessary to determine the meaning of the sentences themselves due to the undeveloped artificial intelligence and lack of knowledge in the field of common sense in it.

To improve this algorithm we could consider other kinds of n-gram selection, such as for example tf-idf and look at the significance of each n-gram to determine the sentence which contains this gram to the desired class, maybe the metric would be better than ours.

For additional development also consider models based not only on branching with semantic kernels, but also which are able to cluster based on the meaning of sentences (Such a model is Bert, ElMO).

Bibliography

(n.d.). SentenceTransformers Documentation — Sentence-Transformers documentation. Retrieved June 10, 2022, from https://www.sbert.net/

[1906.01502] How multilingual is Multilingual BERT? (2019, June 4). arXiv. Retrieved June 10, 2022, from https://arxiv.org/abs/1906.01502

Cherrayil, N. K. (2019, December 31). *Top five trends that will shape the technology sector in UAE in 2020*. TechRadar. Retrieved June 7, 2022, from https://www.techradar.com/news/top-five-trends-that-will-shape-the-technology-sector-in-uae-in-2020

Columbus, L. (2021, June 21). *2021 Roundup Of AI And Machine Learning Market Forecasts Show Strong Growth*. Software Strategies Blog. Retrieved June 10, 2022, from https://softwarestrategiesblog.com/2021/06/21/2021-roundup-of-ai-and-machine-learning-market-forecasts-show-strong-growth2021-roundup-of-ai-and-machine-learning-market-forecasts-show-strong-growth/

Hall, R., & Violino, B. (2019, December 31). *5G technology demands could cause a data revolution*. Information Management. Retrieved June 7, 2022, from https://www.information-management.com/opinion/5g-technology-demands-could-cause-a-data-revolution

Hastie, T., Witten, D., James, G., & Tibshirani, R. (2021). *An Introduction to Statistical Learning: With Applications in R*. Springer US.

Huawei says 'survival' top priority as sales fall short. (2019, December 31). The Financial Express. Retrieved June 7, 2022, from

https://www.financialexpress.com/industry/huawei-says-survival-top-priority-as-sales-fall-short/1808532/

Jurafsky, D., Kehler, A., LINDEN, K. V. A., Ward, N., Martin, J. H., & Jurafsky, J. D. S. (2000). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall. https://web.stanford.edu/~jurafsky/slp3/

models.word2vec – Word2vec embeddings — gensim. (2022, May 6). Radim Řehůřek. Retrieved June 10, 2022, from https://radimrehurek.com/gensim/models/word2vec.html

Natural Language Processing in Python: NLTK vs. spaCy. (2016, April 27). The Data Incubator. Retrieved June 10, 2022, from https://www.thedataincubator.com/blog/2016/04/27/nltk-vs-spacy-natural-language-processing-in-python/

Reimers, N., & Gurevych, I. (2019, August 27). *[1908.10084] Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. arXiv. Retrieved June 10, 2022, from https://arxiv.org/abs/1908.10084