Mathematical Foundations of Data Science

Course Project

Team 68

Ishan Pendse (ME19B191), Prathamesh Dabade (ME19B151), Shreyas Paranjape (ME19B035), Tadeparti Sidharth(ME19B052), Shreyas Pendse(ME19B047)

CH5019

Problem Statement

An instructor wants to grade answers to descriptive questions automatically. The instructor has a template best answer that she/he has developed and wants to use the same to grade descriptive answers of students. The comparison results between the template and the student provided answer could be categorical (right vs wrong) or continuous (75% similarity and so on). You are expected to develop an AI algorithm that can do this comparison automatically. Please develop an algorithm through any appropriate concept and demonstrate the results on ten test cases. You can use any training approach and test it on any 10 test cases that you feel are appropriate. Your algorithm should take two paragraphs, one correct answer and one student provided answer and return a result. Since this is a rather open-ended problem, the solutions will be largely evaluated based on the creativity associated with problem formulation, solution approach (including feature engineering), and the achieved results. Please remember that there is no single correct method. Please use existing AI/ML libraries as much as possible.

Approach 1

Using TF-IDF

The first approach we use involves a classic technique in text processing, namely the use of Term Frequency and Inverse Document Frequency. TF-IDF is a cruicial tool using which semantic information about a piece of text can be extracted and processed further to yield useful insights. In this approach, we have made use of the TF-IDF data and applied certain mathemtical operations that help us understand how similar a query is to a target response.

What is TF-IDF?

- TF-IDF consists of two parts; TF which is Term Frequency and IDF which is Inverse Document Frequency.
- **Term Frequency** : This refers to the frequency with which a word appears. This is typically computed sentence wise for each word present in the corpus/document. The formula:

$$TF_{ij} = \frac{\text{Number of Instances of a Word i in Sentence j}}{\text{Number of Words in Sentence j}}$$

- **Inverse Document Frequency**: IDF tries to identify how prevalent or common a given word is, in a given document, a word that appears in several sentences is likely to be a common theme throughout the document while a word that appears only one sentence might carry more semantic meaning given it's rare occurrence. The IDF is typically computed word wise for the entire document. The formula to compute IDF is as follows:

$$IDF_i = \log\left(\frac{\text{Number of Sentences in Document}}{\text{Number of Sentences in Document that has word i}}\right)$$

- **TF-IDF:** The value of TF-IDF is computed by multiplying both the TD times the IDF, this effectively results in a values of TF-IDF for each of the I unique words in the context of each of the J sentences. The formula can be given by.

$$TF - IDF_{ij} = TF_{ij} \times IDF_i$$

Putting TF-IDF to use:

Once we compute the TF-IDF for a given document, we are left with a matrix of dimensions $(J \times I)$, where J denotes the number of sentences in the document and I denotes the number of unique words in the document.

What we effectively have is data matrix, that holds certain semantic information in the form of TF-IDF values, we now apply matrix operations to compute a score that tells us how close a paragraph is to a target or model paragraph. This in addition to matrix operation, requires a similarity metric, in this approach we make use of the **cosine similarity metric** evaluate closeness in meaning. The matrix is a follows.

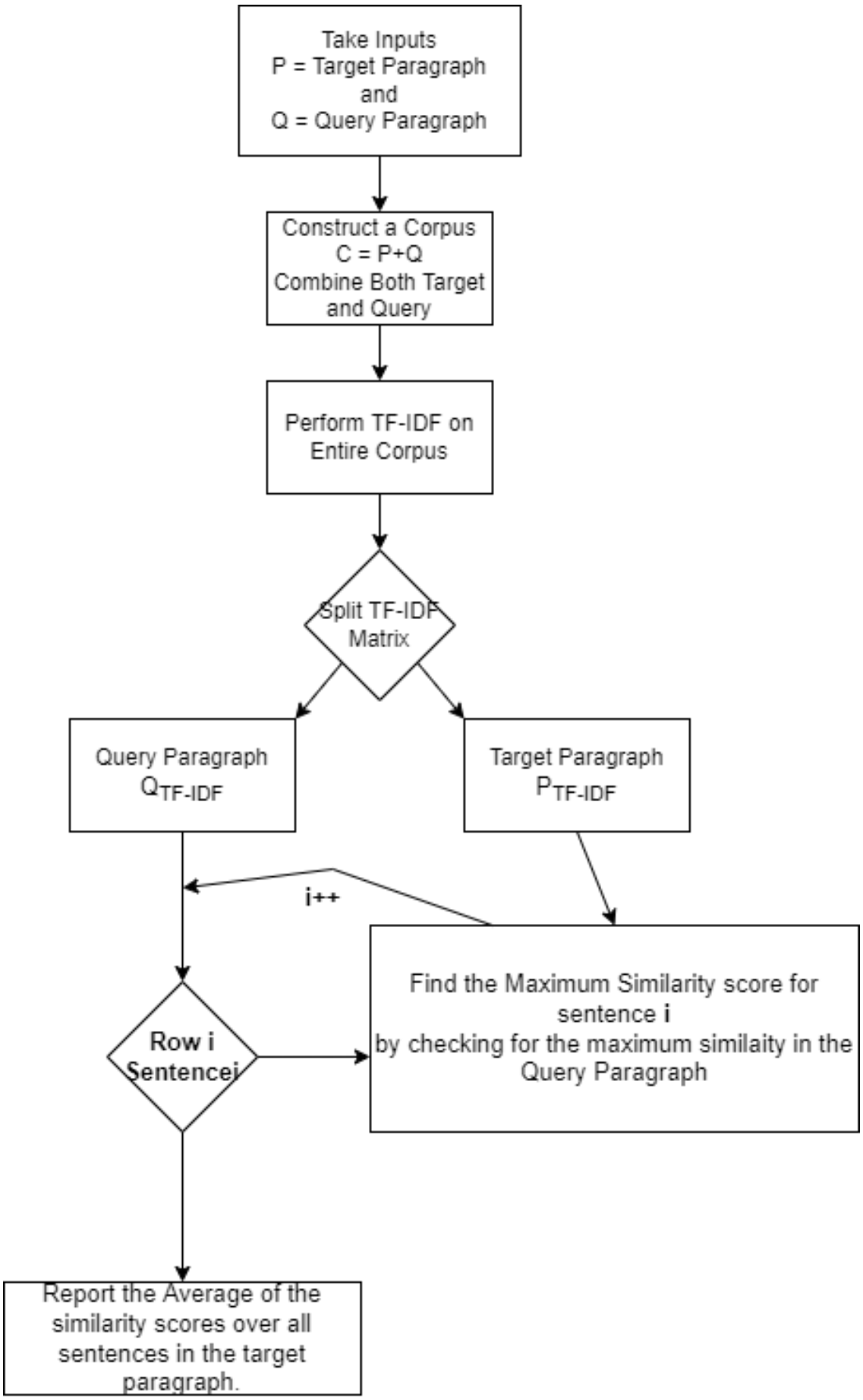
$$\begin{bmatrix} \text{Word1 - Sentence1} & \text{Word2 - Sentence1} & \text{Word3 - Sentence1} & .... \\ \text{Word1 - Sentence2} & \text{Word2 - Sentence2} & \text{Word3 - Sentence2} & .... \\ . & . & . & .... \\ . & . & . & .... \end{bmatrix}$$

To calculate the similarity between two word vector representations:(cosine similarity)

$$\text{Similarity}_{P,Q} = \frac{P \cdot Q}{|P||Q|}$$

Using the above ideas, the following method was used to compute the score:(Also shown in the following flowchart)

- Both the query and the target paragraphs were first combined, processed and their TF-IDF matrix was computed.(the query refers to the text that has to be evaluated, and the target refers to the model text)
- The prepossessing though not shown in the flow chart consists of removing the stop words,punctuation, lemmatization and stemming.
- The TF-IDF Matrix is then split into the target and query parts.
- The query TF-IDF Matrix is then split sentence by sentence (i.e row by row), and the maximum similarity of each of the sentences is calculates by checking it with each sentence of the Target paragraph.
- Finally the score for the query paragraph is computed by taking the average of all the individual maximum scores of the query sentences.


Take Inputs
P = Target Paragraph
and
Q = Query Paragraph
Construct a Corpus
C = P+Q
Combine Both Target
and Query
Perform TF-IDF on
Entire Corpus
Split TF-IDF
Matrix
Query Paragraph
Q_{TF-IDF}
Target Paragraph
P_{TF-IDF}
i++
Row i
Sentencei
Find the Maximum Similarity score for
sentence i
by checking for the maximum similaity in the
Query Paragraph
Report the Average of the
similarity scores over all
sentences in the target
paragraph.

Additionally, we also explore using Singular Value Decomposition, in order augment results, in this slightly different approach continue to use the same method to identify similarity however, we exploit the possible benefits that SVD might have.

SVD allows us to break matrix down into a sum of several rank 1 matrices. Typically the matrices that have the largest singular values are the most significant, and form the deterministic part of the data. In the same vein, SVD is applied TF-IDF matrices once they are split and the most significant component alone is used to reconstruct the split matrices. This is seen to yield superior results in some cases.

Sample Inputs and Results:

Random Text:

```
paragraph_target =  """I like python, it is a powerful programing language.
It has several uses.
"""
paragraph_response = """Python is a very likeable programing language.
  It has strong computing power.
"""
The Score : [[0.39783228]]
```

Text on History

```
paragraph_target =  """Victoria was Queen of the United Kingdom of Great Britain
and Ireland from 20 June 1837 until her death in 1901. Known as the Victorian
era, her reign of 63 years and seven months was longer than any previous British
monarch."""
paragraph_response = """Victoria was queen of the United Kingdom of Great Britain
and Ireland (1837{1901) and empress of India (1876{1901). Her reign was one of
the longest in British history, and the Victorian Age was named for her
"""
The Score : [[0.43577276]]
```

Text from Paleontology

```
paragraph_target =  """Dinosaurs are a diverse group of reptiles of the clade
Dinosauria. They first appeared during the Triassic period, between 243 and 233.23
million years ago, although the exact origin and timing of the evolution of dinosaurs
is the subject of active research.
"""

paragraph_response = """Dinosaurs are a group of reptiles that dominated the land
for over 140 million years (more than 160 million years in some parts of the world).
They evolved diverse shapes and sizes, from the fearsome giant Spinosaurus to
the chicken-sized Microraptor, and were able to survive in a variety of ecosystems.
"""
The Score : [[0.1504266]]
```

Text from Game Theory

```
paragraph_target =  """The prisoner's dilemma is one of the most well-known concepts
in modern game theory. The prisoner's dilemma presents a situation where two parties,
separated and unable to communicate, must each choose between co-operating with
the other or not.
"""

paragraph_response = """A prisoner’s dilemma is a decision-making and game theory
paradox illustrating that two rational individuals making decisions in their own
self-interest cannot result in an optimal solution. The paradox was developed
by mathematicians M. Flood and M. Dresher in 1950, and the modern interpretation
was conceptualized by Canadian mathematician A.W. Tucker.
"""
The Score : [[0.08548837]]
```

Text from Art History

```
paragraph_target =  """Renaissance art is marked by a gradual shift from the abstract
forms of the medieval period to the representational forms of the 15th century.
```

```
Subjects grew from mostly biblical scenes to include portraits, episodes from
Classical religion, and events from contemporary life."""
paragraph_response = """Both classical and Renaissance art focused on human beauty
and nature. People, even when in religious works, were depicted living life and
showing emotion. Perspective, as well as light and shadow techniques improved;
and paintings looked more three-dimensional and realistic.
"""
```

```
The Score: [[0.02317263]]
```

Conclusions:

- The TF-IDF approach is successful at prediciting scores for all kinds of text from science to history and gives fairly accurate score on lucid examples
- This approach however struggles when information is presented in a numerical manner, for example the example on history, essentially provides the information of years in two different ways in the target and the query, but the method fails to identify that they are similar
- The approach also fails when complicated phrases and synonyms are used, this is seen the text on Art History where, the two texts suggest the same meaning, but common words are absent this resulting in a very low score
- The approach also lack an understanding of the relational aspects of the language, this is seen in the random text, where even though the two sentences are suggesting different things, it gives it a relatively high score mere due the fact that a few common words are used although in different contexts
- While, there are several limitations of the TF-IDF approach, it still can serve as an effective way to compare and score two paragraph. A particular advantage of this method is its computational simplicity, it involves simple mathematical expressions and operations that allow for fast computation, and results in scalability. To address the complexities listed above, we explore more advanced techniques.

Approach 2

Using Doc2Vec

Word Embeddings

Word embedding is one of the most popular representation of document vocabulary. It is capable of capturing context of a word in a document, semantic and syntactic similarity, relation with other words and other semantic information. Word embeddings are also vectors hence making it easy for us to compare them using various mathematical techniques like the cosine similarity used in the TF-IDF approach.

How does Doc2Vec work?

Doc2vec builds upon another algorithm called word2vec, let us explore this first. word2vec is a well known concept, used to generate representation vectors out of words. when we build some model using words, simply labeling the words or one-hot encoding them is a possible way to go but this looses a lot of subtlety and contextual information of the language. word2vec attempts to capture much more of such relations. word2vec representation is created using 2 algorithms: Continuous Bag-of-Words model (CBOW) and the Skip-Gram(SG) model.

- Continuous bag of words creates a sliding window around current word, to predict it from "context"(surrounding words). Each word is represented as a feature vector. After training, these vectors become the word vectors. This allows us to encapsulate even numerical relations like $king - man + women = queen$.
- The second algorithm is actually the opposite of CBOW. Instead of predicting one word each time, we use one word to predict all surrounding words ("context"). Skip gram is much slower than CBOW, but considered more accurate with infrequent words.

Now, coming back to doc2vec, our goal is to create a vectorised representation of our paragraph(Also referred to as documents), regardless of its length. Unlike words, documents do not come in logical structures such as words, so the same concept as word2vec wont work . The concept that Mikilov and Le used in their 2014 paper was- they used the word2vec model, and added another vector(Paragraph vector).It is a small extension to the CBOW

model, but instead of using just words to predict the next word, we also added another feature vector, which is document-unique. So, when training the word vectors, the document vector is also trained, and at the end of training, it holds a numeric representation of the document. The model above is called Distributed Memory version of Paragraph Vector. It can be seen as a memory vector which holds the overall context of the paragraph(the topic of the paragraph). While the word vectors represent the concept of a word, the document vector intends to represent the document as a vector. Similar to word2vec, another algorithm, which is similar to skip-gram may be used. Distributed Bag of Words version of Paragraph Vector (PV-DBOW),this algorithm is actually faster (as opposed to word2vec) and consumes less memory, since there is no need to save the word vectors.

Summing up

Essentially, Doc2Vec uses a neural network approach to create vector representations of variable-length pieces of text, such as sentences, paragraphs, or documents. These vector representations have the advantage that they capture the semantics, i.e. the meaning, of the input texts. This means that texts which are similar in meaning or context will be closer to each other in vector space than texts which aren't necessarily related.

The implementation

So now that we have an idea of what doc2vec does, let's take a look at how we can train a doc2vec model on our data. we're going to be using the 20Newsgroups data set. This data set consists of about 18000 newsgroup posts on 20 different topics. We limit ourselves to 4 categories to save on computation time. As we want to be able to measure how similar the documents are to each other semantically, what we want to do is transform our text documents into a numerical, vectorised form, which can later be used to calculate a similarity score.

- First, We need to pre-process the data. we are using dictionaries to keep track of which documents belong to which categories. The function `tokenize()` will transform the data into a list of strings so we can compare for stopwords(common words that add no contextual meaning). with our training documents ready, we can proceed.
- We then create a doc2vec object, which will serve as our model, and initialize it with different hyperparameter values. `epochs` is how many times the training loop executes. the `vector_size` is how large the resulting document vector will be. the `min_count` and `window` are to drop less frequent words and to set how many words will be consid-

ered a word's context.

- We then build the vocabulary. Essentially a dictionary that contains the occurrence counts of all unique words in the training corpus. Finally, the model is trained.
- Now comes the inference part. first, we push the paragraphs through the same preprocessing steps we did earlier and obtain the input for our inference.
- We then use the `model.infer_vector()` on processed input data to obtain the document vectors of our documents.
- Now, we can calculate the cosine similarity similar to what we did above,

$$\text{Similarity}_{P,Q} = \frac{P \cdot Q}{|P||Q|}$$

Sample Inputs and Results:

Random Text:

```
paragraph_target =  """I like python, it is a powerful programing language.
It has several uses.
"""
paragraph_response = """Python is a very likeable programing language.
  It has strong computing power.
"""
The Score : [[0.60056061]]
```

Text on History

```
paragraph_target =  """Victoria was Queen of the United Kingdom of Great Britain
and Ireland from 20 June 1837 until her death in 1901. Known as the Victorian
era, her reign of 63 years and seven months was longer than any previous British
monarch."""
paragraph_response = """Victoria was queen of the United Kingdom of Great Britain
and Ireland (1837{1901) and empress of India (1876{1901). Her reign was one of
the longest in British history, and the Victorian Age was named for her
"""
The Score : [[0.87123221]]
```

Text from Paleontology

```
paragraph_target =  """Dinosaurs are a diverse group of reptiles of the clade
Dinosauria. They first appeared during the Triassic period, between 243 and 233.23
million years ago, although the exact origin and timing of the evolution of dinosaurs
is the subject of active research.
"""

paragraph_response = """Dinosaurs are a group of reptiles that dominated the land
for over 140 million years (more than 160 million years in some parts of the world).
They evolved diverse shapes and sizes, from the fearsome giant Spinosaurus to
the chicken-sized Microraptor, and were able to survive in a variety of ecosystems.
"""
The Score : [[0.69176495]]
```

Text from Game Theory

```
paragraph_target =  """The prisoner's dilemma is one of the most well-known concepts
in modern game theory. The prisoner's dilemma presents a situation where two parties,
separated and unable to communicate, must each choose between co-operating with
the other or not.
"""

paragraph_response = """A prisoner’s dilemma is a decision-making and game theory
paradox illustrating that two rational individuals making decisions in their own
self-interest cannot result in an optimal solution. The paradox was developed
by mathematicians M. Flood and M. Dresher in 1950, and the modern interpretation
was conceptualized by Canadian mathematician A.W. Tucker.
"""
The Score : [[0.63339579]]
```

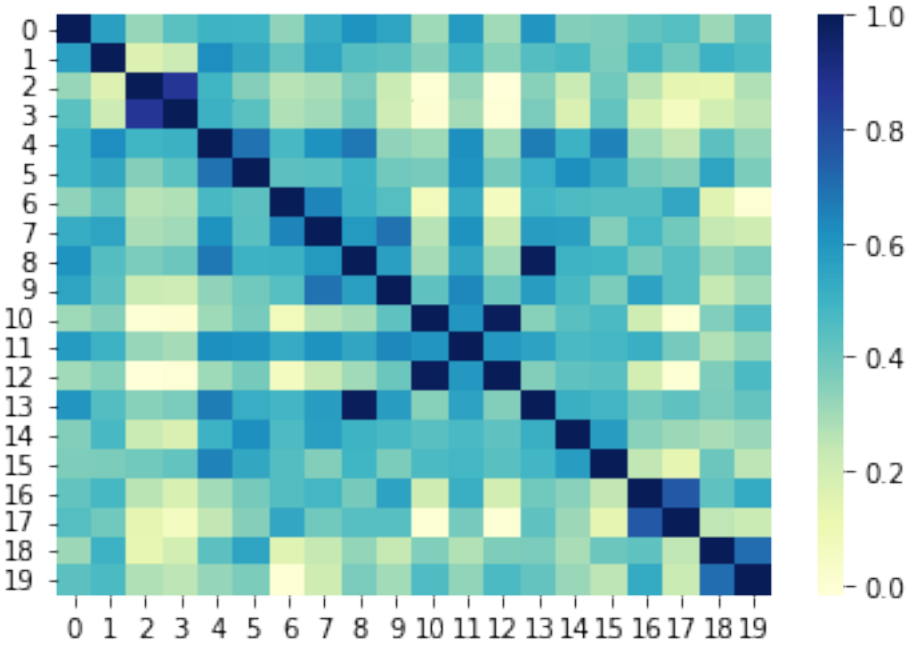


Figure 1: Pairwise similarity between the sentences

Text from Art History

```
paragraph_target =  """Renaissance art is marked by a gradual shift from the abstract
forms of the medieval period to the representational forms of the 15th century.
Subjects grew from mostly biblical scenes to include portraits, episodes from
Classical religion, and events from contemporary life."""
paragraph_response = """Both classical and Renaissance art focused on human beauty
and nature. People, even when in religious works, were depicted living life and
showing emotion. Perspective, as well as light and shadow techniques improved;
and paintings looked more three-dimensional and realistic.
"""

The Score: [[0.50242537]]
```

Conclusions:

- The algorithm works very well in predicting similarity scores for all kinds of text from science to history.

- Stark difference in the similarity score values can be observed when the algorithm is applied to two related and two unrelated documents.

Differences between TF-IDF and Doc2Vec Algorithms:

	TF-IDF Algorithm	Doc2Vec Algorithm
Corpus Size	Designed for modest size corpora	Designed for ultra-large corpora
Corpus Content	The corpus used is relatively focused; limited vocabulary usage; synonyms used are well-known	The corpus used is wide-ranging; huge possible range of synonyms and interpretations
Salient Features and Limitations	Irrelevant words can be eliminated; Synonyms; substantially 'equivalent terms' can be identified and mapped	Difficult to discern term relevance; Difficult to create high-accuracy equivalent terms

Please find all the necessary codes here:
https://colab.research.google.com/drive/1MVenoUohuj4voV04CmcoUDSVmOTaiIA5?usp=sharing
https://github.com/prathamdabade/CH5019-MFDS-Project

Individual Contributions

Ishan Pendse	Ideation, Representation of Results obtained via Heat Maps and other interactive Methods
Prathamesh Dabade	preprocessing and Doc2Vec Algorithm, An explanation of doc2vec (and word2vec) implementation
Shreyas Paranjape	Working on Doc2Vec and Inference Making, Test Code Generation, Similarity Score Calculation and Guidelines for usage of Algorithms
Tadeparti Siddharth	Implementation of TF-IDF without SVD, Report Writing for TF-IDF without SVD
Shreyas Pendse	Working on TF-IDF with SVD, Report writing for TF-IDF with SVD

References

[1] https://radimrehurek.com/gensim/models/doc2vec.html

[2] https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa

[3] https://towardsdatascience.com/multi-class-text-classification-with-doc2vec-logistic-regression-9da9947b43f4

[4] https://towardsdatascience.com/detecting-document-similarity-with-doc2vec-f8289a9a7db7

[5] https://medium.com/wisio/a-gentle-introduction-to-doc2vec-db3e8c0cce5e

[6] https://www.youtube.com/watch?v=iSkbq6Tjkj0