PALACKÝ UNIVERSITY OLOMOUC

Faculty of Science

Department of Biochemistry

Performance evaluation of Machine Learning approaches for identifying parts of scientific affiliations.

DIPLOMA THESIS

Author:	**Bc. Jan Macháň**
Study Program:	N0512A130013 Biochemistry
Study Branch:	Bioinformatics
Form of Study:	Full-time
Supervisor:	**doc. RNDr. Karel Berka, Ph.D.**
Year:	2023

Prohlašuji, že jsem diplomovou práci vypracoval/a samostatně s vyznačením všech použitých pramenů a spoluautorství. Souhlasím se zveřejněním diplomové práce podle zákona č. 111/1998 Sb., o vysokých školách, ve znění pozdějších předpisů. Byl/a jsem seznámen/a s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, ve znění pozdějších předpisů.

V Olomouci dne ....

Podpis studenta

Acknowledgments

I would especially like to thank my diploma thesis supervisor, doc. RNDr. Karel Berka, Ph.D. for his warm approach, the time he devoted to me, and for the opportunity to work on the project. My thanks also go to researchers from Department of Computer Science and Laboratory for Inherited Metabolic Disorders for their guidance and insightful advice during consultations over statistical parts of this work.

Bibliografická identifikace

Jméno a příjmení autora	Bc. Jan Macháň
Název práce	Hodnocení výkonnosti metod strojového učení pro identifikaci částí vědeckých afiliací.
Typ práce	Diplomová
Pracoviště	Katedra biochemie
Vedoucí práce	doc. RNDr. Karel Berka, Ph.D.
Rok obhajoby práce	2023

Abstrakt

Tato diplomová práce zkoumá vhodnost několika volně dostupných natrénovaných modelů ke tvorbě *word embeddings* pro úlohu geolokalizace částí vědeckých afiliací. Analýza využívá statistické metody, jako jsou PCA a ANOVA, k určení nejvhodnějšího *embeddings* modelu. Tyto modely se používají v kombinaci s modely strojového učení, jako jsou neuronové sítě a další klasifikátory. Jako nejvýkonnější se ukazuje kombinace modelů neuronové sítě + uncsd-BERT *embeddings*. Jeho přesnost a průběh procesu učení jsou dále prozkoumány. Práce je případovou studií ilustrující možný postup výběru nejlepšího modelu pro konkrétní klasifikační úlohu a poskytuje pohled na současný stav výkonnosti vybraných *embeddings* modelů v této úloze. Primárním cílem je vyvinout model strojového učení pro geolokalizaci afiliací, bez nutnosti spoléhat na komerční nástroje.

Klíčová slova	afiliace, geolokalizace, *embeddings*, *pre-trained word embeddings* modely, modely strojového učení, klasifikace, statistické vyhodnocení, výběr vhodného modelu, analýza dat
Počet stran	46
Počet příloh	0
Jazyk	Anglický

Bibliographical identification

Author's first name and surname	Bc. Jan Macháň
Title	Performance evaluation of Machine Learning approaches for identifying parts of scientific affiliations.
Type of thesis	Diploma thesis
Department	Department of Biochemistry
Supervisor	doc. RNDr. Karel Berka, Ph.D.
The year of presentation	2023

Abstract

This diploma thesis investigates the suitability of multiple freely available pre-trained word embeddings (PWE) models for the task of geo-localizing parts of affiliations. The analysis employs statistical methods, such as PCA and ANOVA, to identify the most suitable PWE model. PWE models are used in combination with ML classifiers such as Neural Networks, Random Forests, Support Vector Classifier, and K-Nearest Neighbors. The Neural Networks together with uncsd-BERT embeddings model emerges as the best performing combination. Its classification performance and learning process are further evaluated. The thesis serves as a case study illustrating the selection of the best model for a specific classification task, and it provides insights into the state-of-the-art performance of selected embeddings models on this task. The primary goal is to develop a ML model for geo-localizing affiliations without commercial tools.

Keywords	affiliations, geo-localization, embeddings, pre-trained word embeddings, machine learning models, classification, statistical evaluation, model selection, data analysis
Number of pages	46
Number of appendices	0
Language	English

CONTENTS

OBJECTIVES OF THE WORK

Goals in the theoretical part the goals are as follows:

- describing the advantages, disadvantages, and difficulties of solving problems by means of machine learning approaches,
- summarizing existing models and approaches for text vectorization.

Main goals of the practical part of this works are:

- the preparation of training and test sets,
- the deployment of various pre-trained embeddings models,
- analysis of performance of said models in combination with machine learning models,
- evaluation of the results and indicating the best performing model combination,
- discussion of the implementation of selected model into the pipeline for geo-localization.

1 INTRODUCTION

Artificial Intelligence (AI) and ***Machine Learning*** (ML) have become widely recognized and frequently discussed concepts in the public recently. Notably, their versatility and ability to tackle complex problems that defy simplistic "If A, then B" descriptions render them inherently advantageous.

Machine learning (ML) relies on procedures where the program learns to solve a given problem. This learning occurs mostly through adjusting the system's state to accept input and generate an output as a response. If the answers are incorrect, the state is modified, and the process is repeated on a training dataset until a system setting is achieved that yields reliable results for most data. With sufficient training data and validation process, the trained program can be expected to provide reasonably accurate outcomes.

For instance, the AlphaFold2 AI, which utilizes deep learning (DL) to predict protein folding with accuracy comparable to physical methods, like X-ray crystallography, has gained significant attention (Jumper et al., 2021; Skolnick et al., 2021). Similarly, image generation models based on verbal descriptions, often indistinguishable from the work of skilled graphic artists, have gained renown (Borji, 2022).

However, a main drawback of ***neural networks*** (NN) is that they lack transparency in their problem-solving approach. While this may not be problematic in some cases, it can be critical in others. The ethical implications and rate of progress also raise concerns among scientists.

A significant breakthrough has been made in the field of ***natural language processing*** (NLP), with the emergence of models that can learn to understand the meanings of the words through so called embeddings. Among these, ChatGPT stands out as one of the most prominent, owing to its use of the latest third-generation transformer neural network, known as GPT3 (Brown et al., 2020). This chatbot is easily accessible online and has already proven useful in performing practical tasks. Its capabilities extend beyond mere text generation, with recent reports indicating that it can also help programmers by writing and correcting code, thereby increasing their efficiency and productivity (Narasimhan et al., 2021). With GPT4 becoming available for users via ChatGPT Plus and its predecessor gaining more widespread use, its impact on various fields cannot be ignored.

In the experimental part of this thesis, I aim to prepare a training dataset and train various ML models, including ***Neural Networks*** (NN), ***Random Forest*** (RF), ***Support Vector Machine*** (SVM), and ***K-Nearest Neighbors*** (KNN), to identify the names of institutions, states, or cities. An important aspect of this process is selecting the appropriate vectorization approach for affiliations, which will be used across all models. By doing so, I hope to determine the optimal combination ***pre-trained word embedding*** (PWE) model and ***machine learning*** (ML) model to achieve the best results. This research builds on my bachelor's thesis, where I was able to identify the countries in information related to molecules from the ChEMBL and PubMed databases (Macháň, 2021). However, brute force identification would be highly impractical for detailed geo-localization down to the level of cities and institutions due to the size of the datasets spanning millions of affiliations. Hence, I intend to utilize machine learning methods to tackle this challenge.

My goal is to present my master's thesis as a case study that showcases several machine learning models in combination with embeddings models and evaluates their performance in solving the geo-localization of textual data. After assessing their effectiveness, I will select the most suitable model pairing for processing large amounts of data. These models will be trained to recognize the parts of the text that correspond to the names of countries, cities, or institutions.

Although this topic may not be strictly considered as a bioinformatics research area, it is worth noting that the geo-localization of affiliations was initially motivated by the desire to create a web application highlighting region-specific chemical research on biological targets. Moreover, the study of embeddings is becoming increasingly popular and applicable in various domains beyond natural language processing (NLP). For instance, embeddings have shown their usefulness in chemical structure representation (Coley et al., 2017; Morris et al., 2020), and are likely to find application in other fields as well. Thus, gaining knowledge about embeddings can be highly beneficial in a broader scientific context.

2 THEORETICAL PART

2.1 Solving Problems by Means of Machine Learning

In this chapter, I aim to explore some of the challenges one may encounter when solving tasks using machine learning, as well as categorize the typical approaches employed by numerous ML models. These models utilize large datasets to discover hidden patterns, enabling them to make informed decisions over time through the learning process. The model's learning is typically expressed through its ability to gradually change its state to make statistically better predictions.

2.1.1 Supervised vs Unsupervised Learning

Typically, machine learning approaches are categorized into two main types: ***supervised*** and ***unsupervised learning***. Supervised learning involves a training-set with N objects represented by vector x of their features and crucially, y representing the class or value that the model should learn to predict. In contrast, unsupervised learning has no y, and the task is descriptive. The objective of unsupervised learning is often to recognize interesting patterns in the data, making it a form of knowledge discovery (Murphy, 2012). Importantly, both approaches can learn advantageous data representations as a side effect of solving specific tasks (Goodfellow et al., 2016).

Some consider a third type of machine learning, ***reinforcement learning***, which learns on the basis of occasional positive or negative feedback (Murphy 2012). However, as this approach is not used in this thesis, it is unnecessary to delve into more details.

The primary type of machine learning employed in this work is supervised learning. Thus, obtaining a training-set that consists of both vector x and desired y is essential. In addition, to evaluate performance, it is necessary to create a similarly defined test-set.

Lastly, the creators of freely available embeddings models performed unsupervised learning, yielded advantageous data representation, as previously mentioned. These pretrained models were utilized as generators of features for vectors x since their representation, also known as embedding, behaves in a beneficial manner, as described in subsequent chapter.

2.1.2 Regression vs Classification

Supervised learning problems are typically divided into regression and classification tasks based on the predicted *y*. If *y* is categorical, the task is referred to as ***classification***, and the predicted category is often considered the label or class. On the other hand, if *y* is real-valued, the problem is known as ***regression***. (Murphy 2012)

Despite the apparent differences, both regression and classification require the ML model to approximate a function - in regression, the regression function, and in classification, the function representing probabilities of memberships of different classes. (Bishop 1995)

This is why ML models can perform both regression and classification tasks, and why the preparation of data or performance estimation differs only slightly.

The goal of the experimental part is to label different parts of affiliations, which categorizes it as a classification problem.

2.1.3 Typical Learning Issues

When using supervised learning, common problems that one may encounter are underfitting and overfitting.

Underfitting refers to the inability to capture the relationship between inputs (vectors *x*) and outputs (*y*), resulting in the model approximating a complex function by a simpler one (Kruse et al., 2016). In other words, underfitting occurs when the model is too simple to capture the patterns in the data. This may be caused by using an inadequate model or inappropriate settings of hyperparameters, such as setting the number of neurons in a neural network too low. Underfitting is not a major issue since it is easy to detect and typically fixable with relative ease.

Conversely, ***overfitting*** is often more challenging to detect and fix. It occurs when the model approximates a simple function with a way too complex one. This is a common occurrence, as data frequently includes noise, making it easy to focus on wrong features. Another reason for overfitting is relatively small size of the training set, which can describe the relationship between inputs and outputs in a restricted manner, resulting in a possibly distorted reflection of reality. Therefore, the model may inadvertently learn specific characteristics of the training set, including errors, noise, and deviations, instead of revealing the true underlying relationship, which may be much simpler. (Kruse et al., 2016)

Other aspects that require attention include class balancing within the training-set, dimensionality reduction of input vectors, and various other considerations that may impact the learning process. These considerations are highly dependent on the nature of the data being used.

2.2 Problem of Vectorization

Before cleaning, normalizing, encoding categorical attributes, or filtering for relevant attributes, it is necessary to first vectorize any data that is not already represented by numerical vectors. This can be a challenge in the case of text or image data, which require special treatment, and which is the point of interest in this thesis. The issue of vectorizing text data for ***Natural Language Processing*** (NLP) tasks will be discussed in this chapter.

2.2.1 Why BOW and TF-IDF do not work here.

The simplest vectorization techniques include ***Bag of Words*** (BoW) and ***Term Frequency – Inverse Document Frequency*** (TF-IDF). Although not directly relevant to this thesis, they form the basis of text vectorization approaches and are therefore worth mentioning. However, these techniques have a significant limitation in that they do not consider the context of words well enough, resulting in a narrow and limited representation of the text or document.

The BoW method counts the number of occurrences of each instance of the word overlooking the order, grammar, or context. This enables the classification of texts and coming from the number of occurrences then calculating weights for each word. (Qader, 2019) However, the apparent difficulties with using this method are the size of the created vectors, which increases with every new word and the sparsity of such vectors. Furthermore, since BoW approach by definition ignores the context of the words, it may not be suitable for certain applications, such as this one.

Secondly, TF-IDF is an approach of text vectorization that calculates the weights of terms based on their frequency in a document. The calculation involves the intuitive term frequency (TF) and the inverse document frequency (IDF), which is the logarithm of the ratio of documents in the corpus to the number of documents where term occurred. (Sammut and Webb, 2011) This method allows for filtering of words specific to certain documents and avoids the use of frequently occurring words in all documents. Consequently, describing these documents by vectors with increased weights for those specific and relevant words. Again, this approach considers context of the words in a limited document-wise way.

2.2.2 Word Embeddings and Transformers

As previously discussed, models can often learn useful data representations while solving other tasks. These representations can then be utilized for more complex tasks.

The curse of dimensionality is a fundamental problem in language modeling. If the goal is to solve a problem on large training data, the resulting corpus will include large number of words which will be represented by discrete vectors, which will inevitably complicate the learning process. This can consume a significant portion of the model's learning capacity just to understand the data, since every answer can be either correct or incorrect. However, representing words with continuous vectors where distance represents level of dissimilarity can result in a smoother learning process and better performance. These models can learn from large textual datasets to comprehend the typical context of a word, which is then expressed in its continuous vector representation. (Bengio, 2003)

N-gram models create conditional probabilities for the next word given some history or context (Bengio, 2003). Specifically, they calculate the frequency of a particular word following a given history of words in the corpus. Unsupervised training of a neural network on large text data can produce projection of words into a continuous space, such that the distance of the word vectors reflects on similarity/dissimilarity of the words. (Jurafsky and Martin, 2023) These real-valued vectors that encode the meanings of words with such quality are known as ***word embeddings***.

The effectiveness of word embeddings is not only influenced by the training data but also by the task being tackled by the ML model. Additionally, since the embedding vector length derives from the number of neurons in the hidden layer the sufficiency is also directly influenced by architecture of the model, besides other architectural factors that influence it as well.

Various other models have been developed based on the idea of projecting words into continuous vector space. Tomáš Mikolov's earlier works (Mikolov, 2008, 2009) stemmed from the need for better language models for highly inflective languages such as Czech and laid the foundation for his later creation of the ***Word2Vec*** model.

Word2Vec is a predictive model that can be realized by two different models: ***Continuous Bag of Words*** (CBOW) and ***Continuous Skip-Gram***. (Mars, 2022; Mikolov et al., 2013a)

The neural networks of CBOW and Skip-Gram models have opposing objectives: while CBOW predicts the current word given its context, Skip-Gram predicts the context given a target word. Thus, their architectures mirror each other. Without going into too much detail, these and other related models use architecture of an input layer, a hidden layer that describes embeddings, and an output layer. Each model has advantages and disadvantages during the learning process, and while their derived embeddings are similar, they may exhibit slightly different levels of accuracy across various tasks. Nevertheless, the resulting embeddings for different words display even very subtle syntactic and semantic relationships. (Mikolov et al., 2013a) Their practicality is further evidenced by their ability to organize concepts and learn relationship among words, despite using unsupervised data. For example, if one finds a vector representation of x (vec(x)) that is the closest to the result of the equation vec("Berlin") - vec("Germany") + vec("France"), the resulting embedding will be the closest to that of "Paris". (Mikolov et al., 2013b)

In addition to Word2Vec, other popular word embedding models include GloVe: ***Global Vectors for Word Representation*** (Pennington et al., 2014) and ***FastText*** (Mikolov et al., 2017), the latter can be perceived as successor of Word2Vec (Mars, 2022).

These ideas were further explored, and the field of Natural Language Processing (NLP) saw a breakthrough when the ***Transformer model*** surpassed all previous models in terms of accuracy and training speed, setting a new state-of-the-art. (Vaswani et al., 2017) The simplified architecture of Transformers, which relies solely on the attention mechanism, seems to significantly quicken the learning process. This mechanism emulates the attention function by assigning weights to values based on a given key and query (Vaswani et al., 2017), and can also be found in other modern systems like the aforementioned AlphaFold2 (Jumper et al., 2021).

Pretrained Language Models (PLMs) that are based on the Transformer architecture, such as BERT and GPT, have proven to be highly successful and have become the mainstream approach for solving various NLP tasks (Mars, 2022). Their usability is derived from their training data, which proves valuable in instructing models to comprehend distinct text types, such as scientific papers. This is presented in the works of Tshitoyan et al. (2019) or Meijer & Truong & Karimi (2021), for instance.

2.3 Used Pretrained Word Embeddings Models

In the next chapter, I will briefly describe the different models that were selected for the experimental part of this work. The objective was to compare multiple models from various sources. The models were chosen based on their availability and my ability to configure and run them successfully within python environment.

It is also worth acknowledging that the models being used in this context often generate embeddings for larger text units beyond individual words. Despite this, I will continue to use the abbreviation PWE (pre-trained word embeddings) for convenience, even though it may not be entirely accurate. Ultimately, the final embeddings are often derived from the word embeddings.

2.3.1 Bidirectional Encoder Representations from Transformers (BERT)

Three ***BERT models*** were selected from TensorFlow Hub repository (https://tfhub.dev) to represent this family of models. They are based on Transformer architecture and were trained on Wikipedia and BooksCorpus dataset. First two use settings of L=12 transformer blocks, a hidden size of H=768 that corresponds with the length of embeddings and A=12 attention heads. Only difference being if they use cased or uncased input text. Last of the BERT models was chosen to be an uncased and bigger version with L=24, H=1024 and A=16.

The fundamental idea behind BERT's creation is to surpass models like GPT or ELMo by overcoming their unidirectional constraint during the learning process, which is caused by their left-to-right approach. BERT accomplishes this through the utilization of a Masked Language Model (MLM) objective, which randomly masks some parts of the text, and the objective is to guess the original masked word from its context. BERT's notable performance can be attributed to the utilization of this bidirectional pre-training for language representations, which is also the source of its name. (Devlin et al., 2018)

It should be noted that BERT has gained significant attention in the research community, leading to the development of various iterations that have been trained with slightly different architectures or on data specific to certain fields of interest. These include RoBERTa, ALBERT, DeBERT, and SciBERT, among others (Mars, 2022; Beltagy et al., 2019).

2.3.2 Doc2Vec

I also decided to try a model that was created with longer texts in mind and test his performance on much shorter parts of the affiliations, in a hope that this more complex model will have bigger capacity to store more information communicated in affiliation.

Doc2Vec is a model build upon the basis of Word2Vec model described earlier. The main addition is a paragraph vector that in combination with word vectors contributes on preserving the information. Paragraph vector is in a way another word vector, that remembers what is missing from the current context or what is the topic of current paragraph. (Le and Mikolov, 2014) The model was used via Gensim library (https://github.com/RaRe-Technologies/gensim) with default inference hyper-parameters and in two settings based on the two original training datasets: English Wikipedia DBOW and Associated Press News DBOW (Distributed Bag-of-Words).

2.3.3 Universal Sentence Encoder

The ***Universal Sentence Encoder*** (USE) model is one of two models described in the paper and it is based on Transformer architecture. The training of the model was performed on dataset consisting of multiple web sources such as Wikipedia, web news, etc. and was later improved by multiple supervised corpora. USE is available on the TensorFlow Hub repository website, as was BERT. The universal-sentence-encoder-large in version 5 was used. This model can encode input of variable length into a 512-dimensional embedding vector. (Cer et al., 2018)

2.3.4 InferSent

InferSent model stems from ***natural language inference*** (NLI) task. NLI task tries to determine whether a hypothesis sentence is true, false, or neutral given a premise sentence. The main distinction of InferSent model being that NLI task requires annotated data and thus is a representative of supervised learning. (Conneau et al., 2017) This layer of training is added on already unsupervised models of GloVe and FastText. InferSent was downloaded from its GitHub repository (https://github.com/facebookresearch/InferSent). Two versions of the model are presented with one being trained with GloVe (Pennington et al., 2014) and one with FastText (Mikolov et al., 2017).

InferSent was used with the same parameters as was suggested by the authors in their GitHub repository examples (Conneau, n.d.).

2.3.5 MiniLM

The authors were concerned about the sizes of models like BERT and proposed a ***deep self-attention distillation*** approach. Knowledge distillation (KD) approach is to use a pre-trained large model as a teacher to help student model to tackle the same task with fewer parameters (Hinton et al., 2015). With MiniLM, the student model focuses on the last layer of the Transformer model, which serves as its teacher. This strategy allows for the compression of a strong but sizable model into a MiniLM model that is much quicker to deploy, while still achieving comparable results. (Wang et al., 2020, 2021)

I decided to use two variants of ***MiniLM model***, All-MiniLM-L6-v2 and All-MiniLM-L12-v2. The prefix All- meaning that they were trained on all available training data authors had and suffix -v2 stating their version. These models are available at sbert.net website or on their GitHub (https://github.com/UKPLab/sentence-transformers).

2.4 Used Supervised Learning Models

This section serves as a connection between the theoretical and experimental parts of this thesis, as it briefly summarizes the machine learning (ML) models and their hyperparameters used in the experimental part.

All the models used originate in the scikit-learn library (https://scikit-learn.org/stable), which is an open-source software for tackling both supervised and unsupervised learning. It is built in Python and utilizes other scientific libraries such as NumPy and SciPy. Scikit-learn provides a wide range of popular ML models. Additionally, some of them, like Support Vector Machine (SVM), are written in Cython to achieve C-like performance. (Jolly, 2018)

I installed this library via Anaconda environment in version 1.2.1 of scikit-learn.

2.4.1 Multilayer Perceptron and Neural Networks

Multilayer Perceptron Classifier (MLPClassifier) model can learn a non-linear function to approximate given task. MLPClassifier uses numerous perceptrons in multiple layers to model network-like structure. For purposes of reproducibility, it is important to use specified random seeds, since different weights on initiation can cause different local minima found. In order to perform multi-class classification softmax function is applied on output layer, which normalizes the output values that their sum is one. I used MLPClassifier with default hyper-parameters except maximum of iterations, which I set to 300. (scikit-learn, n.d.)

2.4.2 Random Forests

Random Forest Classifier (RFClassifier) is an estimator that fits multiple decision trees on sub-samples of the training dataset. This process as well as the selection of features in decision trees nodes is dependent on random seed. The settings used for RFClassifier was number of estimators equal to 1000 and maximal depth of decision tree equal to 20. (scikit-learn, n.d.)

2.4.3 Support Vector Machines

Support Vector Classifier (SVC) finds the best decision boundary in the vector space to separate desired classes. SVC model is in scikit-learn implemented based on libsvm and is advised not to use this implementation for datasets with large numbers of samples. However, this is not a concern for this work as the dataset is not large. The only parameter that was changed from its default value was gamma that is set to ‘auto’. (scikit-learn, n.d.)

2.4.4 K Nearest Neighbors

For the last model, ***K-Nearest Neighbors*** (KNN) was used, which classifies new inputs by the majority class of the nearest neighbors. This means that it is not necessary to change this model in any way for multiclass classification. This approach also needs no training, since it only stores the training set instances and then classifies all new inputs based on those samples. Finally, various k values, including 1, 3, 5, 7, and 9, were tested, and k=3 was chosen due to its superior performance. (scikit-learn, n.d.)

3 EXPERIMENTAL PART

The goal of the experimental part of this work is to evaluate performance of several freely available PWE models on given task, which is a classification of parts of paper affiliations. Identify models which are statistically significant candidates for further analyses and additionally test these candidates in combination with ML models to evaluate which pair outperforms others and thus forms the best combination of models to be used in pipeline for process of geo-localization.

Both training-set and test-set were manually created from affiliations retrieved from PubMed (https://pubmed.ncbi.nlm.nih.gov/) for articles mentioning chemical structures and further refined. Embeddings for the annotated parts were created using several free PWE models. Suitability of embeddings was inspected through PCA and then statistically evaluated using ANOVA from K-fold Cross-Validation data over multiple supervised ML models. Candidate embedding models (cPWE) were chosen on which other ANOVA was performed to determine best performing ML model (cML). Candidate pairs (cP) of PWE and ML model were chosen and the confirmation of the results via test-set was performed to determine best performing pair (Figure 1). The performance of chosen combination was further investigated and described using confusion matrix and related measures typically used for evaluating machine learning models.

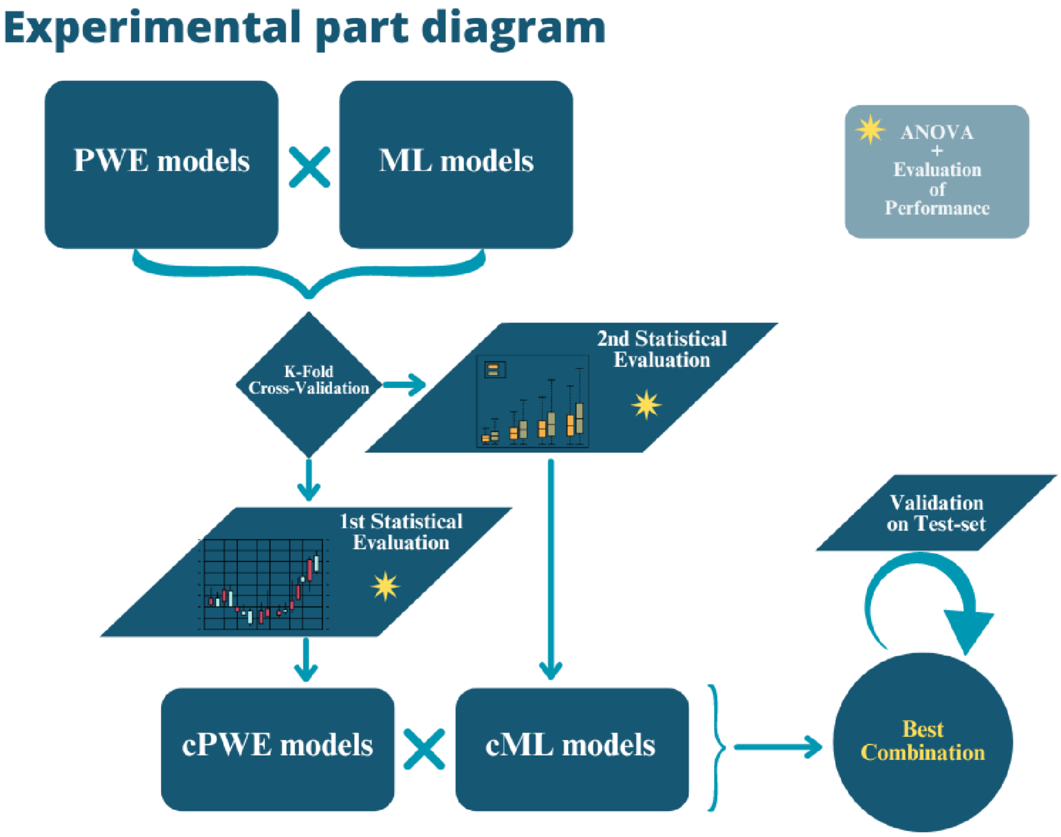


Figure 1: Diagram of the intended flow of work for experimental part of this thesis.

All workflows (written in `python`, specifically `jupyter notebook`), all datasets, graphs, `Statistica` workbooks and excel sheets with results and analyses are shared on GitHub (https://github.com/Najlaron/Diploma-Thesis) to ensure reproducibility and full transparency of this work.

3.1 Training-set and test-set

To get curated training and test-set, I created script that runs through affiliations and lets me label its parts manually. The affiliations were randomly shuffled to prevent any biases towards specific years, countries, or institutions. Contentious parts were removed or altered to refer to only one specific label to not compromise further analysis. Non-specific and vague terms such as *'Department of Biochemistry'*, *'College of Pharmacy'*, etc. were labeled '*None*' to increase specificity of identified institutions, since highlighting said parts would not lead to their localization. Lastly, during this whole process each label was assigned in even amounts, usually once per affiliation, if possible, to guarantee roughly balanced composition. Along those lines, as a means to preserve this balance, some redundant and often repeated parts were removed from training and test-set.

Created training-set comprises of total 2997 parts of 602 affiliations. Out of these, 746 were labeled as *'None'*, 704 as *'Institution'*, 642 was the number of both *'City'* and *'Country'* labels and the remaining 263 were classified as *'State'*. *'State'* label was used mainly for states of US and then provinces or other parts of specific countries. This label is naturally not that prevalent in dataset and thus it is imbalanced regarding the other labels. Since it is not necessary for localization it is often omitted from affiliations, more importantly this also means that the precision of its identification should not have such impact on the precision of the whole model for geo-localization. Other classes seem to be balanced enough to prevent biases during training.

Test-set counts for 316 parts from 60 affiliations and hence it is roughly tenth of the training-set size. Numbers of instances of labels are as follows: *'None'*: 97, *'Institution'*: 77, *'Country'*: 65, *'City'*: 59, and then *'State'*: 18. This distribution follows the trend of training set, this means being mostly balanced for all labels except *'State'* which is less frequent and *'None'* slightly outperforming. This similarity should help with guaranteeing reliable performance measures.

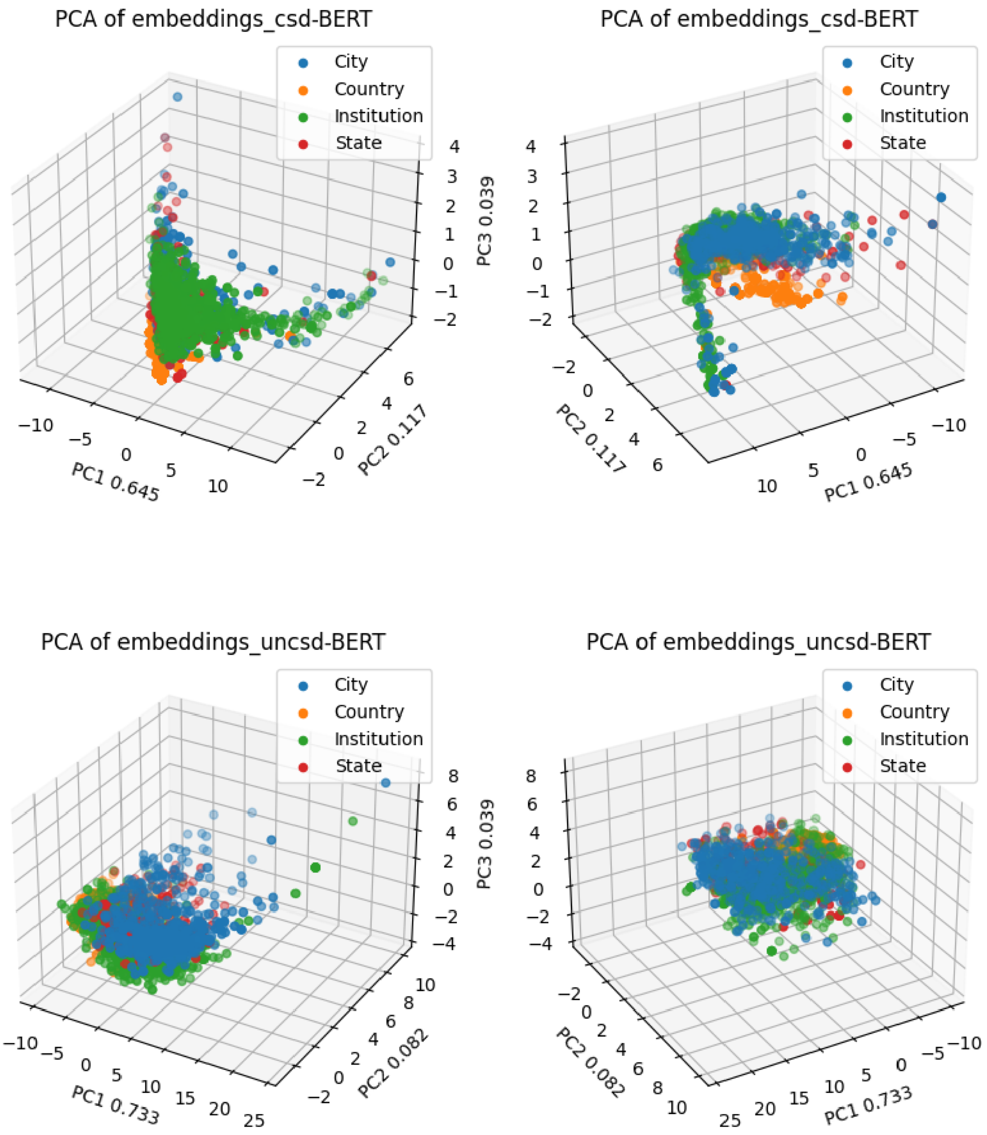
4 RESULTS

4.1 Suitability of Pretrained Embeddings Models

In this part I present how well different PWE models cluster vectors based on their assigned label. The task is to successfully classify embedding vectors, which solely depends on their location in *n*-dimensional space, or in other words their *n* values, where *n* is determined by the PWE model architecture. Considering our classes consist of geographic terms and therefore occur in similar context, it is not necessarily true, that they will be clearly separated in this space. Also, one cannot expect increasing distinction of clusters for larger *n*, since the embedding vectors depend mainly on training data of said PWE model. To get an insight into the clustering of such multidimensional data Principal Component Analysis (PCA) was performed and data was displayed in form of 3D and 2D graphs. PCA was performed both with and without vectors classified as *'None'* to highlight important differences. All graphs can be found also in GitHub repository (https://github.com/Najlaron/Diploma-Thesis). I decided to present mainly results of PCA realized on data without *'None'* vectors, for the sake of clarity of graphs.

4.1.1 BERT

Three embedding models based on BERT were included in experimental part, namely cased-BERT-726, uncased-BERT-726 and uncased-BERT-1024. Values 726 and 1024 represent number *n* of the dimensions of embedding space. It can be seen on the following graphs, that all three models cluster data quite similarly, but not very noticeably (Figures 2-4). There are visible signs of clusters, which are potentially more distinctly separated in higher dimensions, but it must be notted, that first 3 components account for 0.80, 0.85, and 0.74 of the total variance of said models in order. Additionally, suggesting that such difference in remaining dimensions may not be substantial enough. Exact percentages can be seen in axis descriptions in forms of explained variance ratios. It would certainly be desirable to see these models exhibit clearer data clustering already in these lower dimensions.



Figures 2-3: Results of PCA shown as 3-Dimensional graphs (from 2 angles) depicting clustering of embeddings vectors from csd-BERT-726 and uncsd-BERT-726 models colored by their designated class (excluding *None*). The axes belong to the first 3 components from PCA and describe total variance explained by said components.

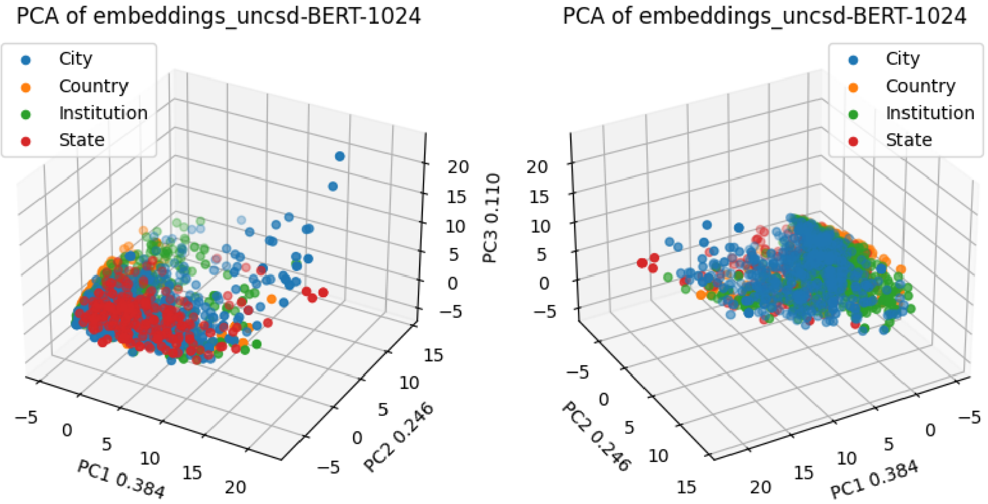
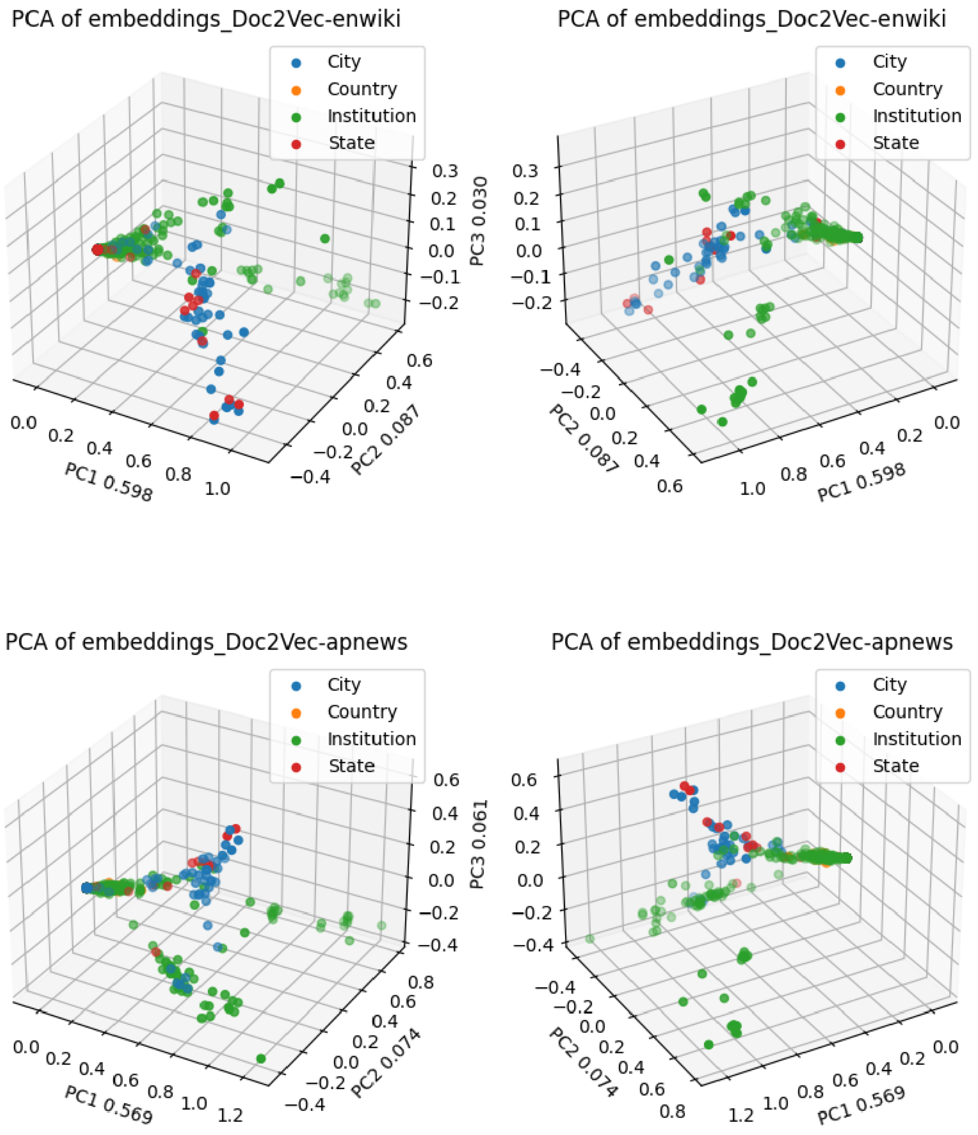


Figure 4: Result of PCA shown as 3-Dimensional graph (from 2 angles) depicting clustering of embeddings vectors from uncsd-BERT-1024 model colored by their designated class (excluding *None*). The axes belong to the first 3 components from PCA and describe total variance explained by said components.

4.1.2 Doc2Vec

Despite describing approximately 70 percent of total variance for both Doc2Vec-enwiki and Doc2Vec-apnews, the following 3D graphs do not exhibit any clustering, except for a single cluster that encompasses all the data points (Figures 5-6). It is now obvious, that the attempt to use model for embedding whole documents for only small parts of affiliations did not yield favorable results.

While the unsuitability of the model for this task may be a possible explanation, it is also plausible that the model is simply impractical for this particular approach to the problem and may perform better under a different perspective. Furthermore, it cannot be discounted that my potential misuse of the model or suboptimal settings may be the root cause of the lackluster outcomes, since the model is expected to yield results comparable or better than Word2Vec embeddings (Lau and Baldwin, 2016). Consequently, the ensuing analyses are likely to suffer from very poor results.



Figures 5-6: Results of PCA shown as 3-Dimensional graphs (from 2 angles) depicting clustering of embeddings vectors from both Doc2Vec models colored by their designated class (excluding *None*). The axes belong to the first 3 components from PCA and describe total variance explained by said components. Most points are indistinguishable as they seem to be projected into very similar areas, this clusters all data to one cluster instead of showing some class-wise clustering.

4.1.3 Universal Sentence Encoder (USE)

Right from the start, the capacity of USE model to effectively cluster vectors with matching labels appears to be very potent. While describing only around 20 % of total variance, this model clearly separates *'Institution'* class. Albeit closely, USE groups the rest of the classes into recognizable tight spatial patterns (Figure 7). Undoubtedly there is a great potential for this model to pronounce edges of these subspaces and it is worth to acknowledge its apparent suitability for this task. On the contrary as is shown in 2D plot, if we do PCA including *'None'*-labeled data, *'Institution'* class is not so isolated, and it is blending with *'None'* (Figure 8). This will certainly not help with the classification task, but at least *'None'* class is not entirely joined into the bundle of remaining classes to hinder their chances of separation.

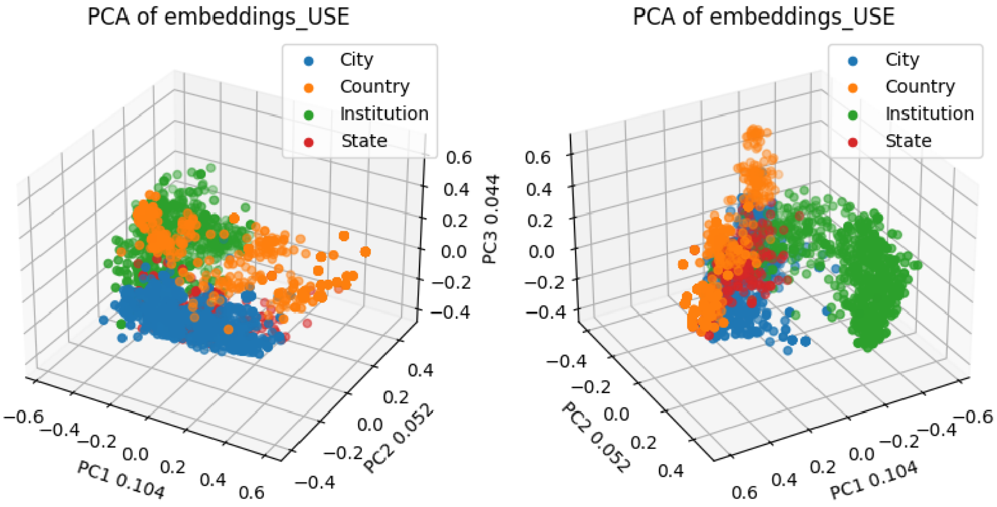


Figure 7: Result of PCA for USE model shown as 3-Dimensional graph (from 2 angles) depicting clustering of its embeddings vectors colored by their designated class (excluding *None*). The axes belong to the first 3 components from PCA and describe total variance explained by said components.

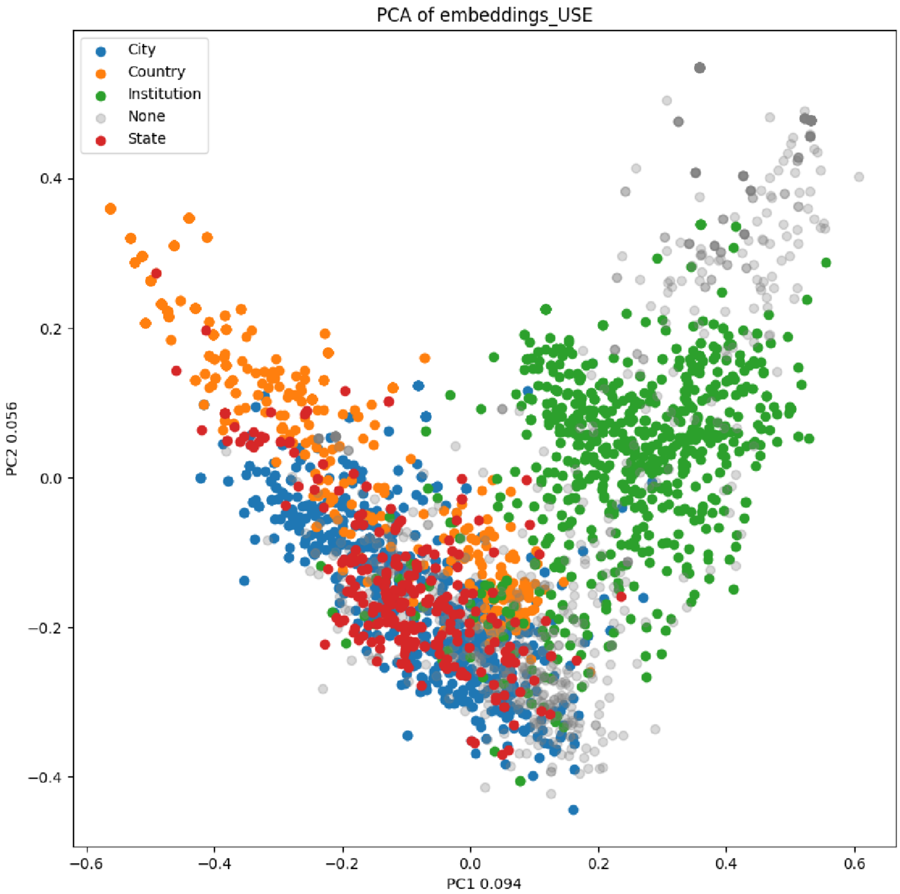
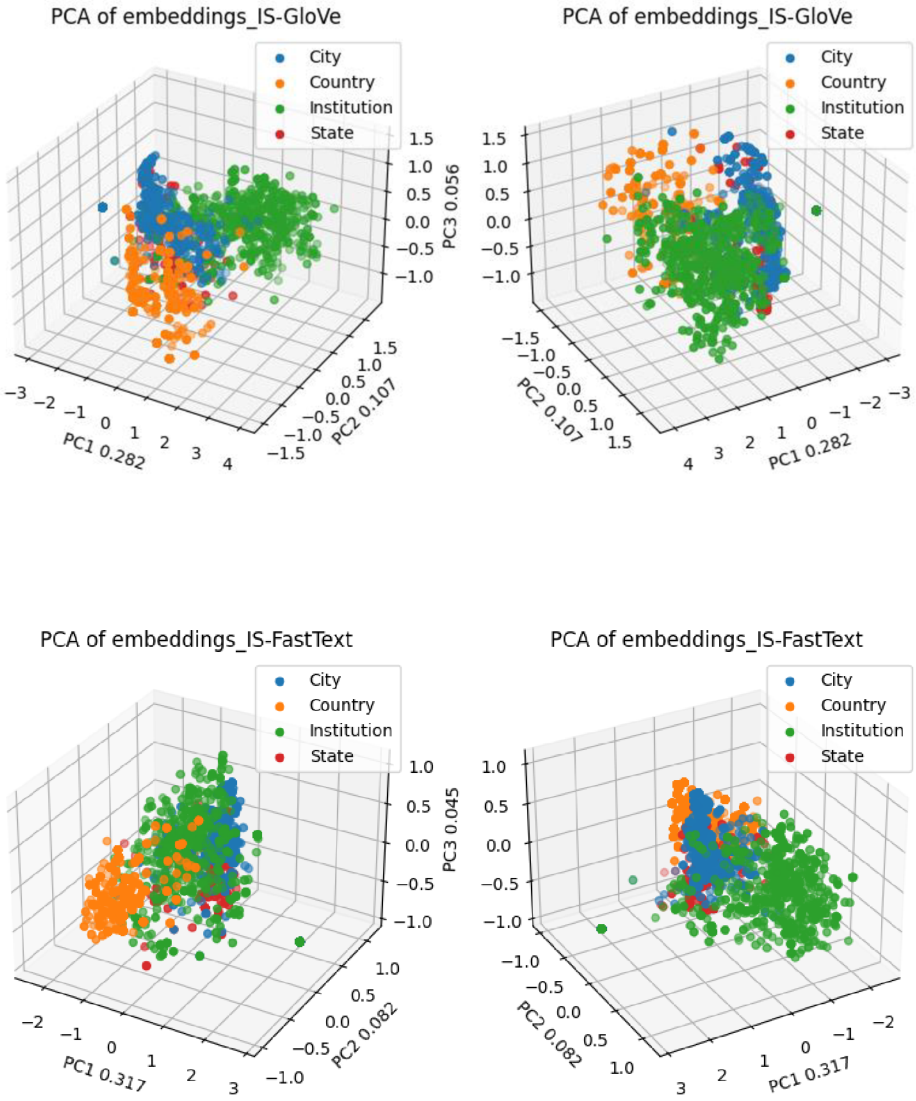


Figure 8: Result of PCA for USE model shown as 2-Dimensional graph depicting clustering of its embeddings vectors colored by their designated class including *None* in transparent grey. The axes belong to the first 2 components from PCA and describe total variance explained by said components.

4.1.4 InferSent (IS)

As in the previous case, IS models separate majority of *'Institution'* labels and additionally *'Country'* (Figures 9-10). Whereas not exactly evident in 3D plots, it is clear from 2D versions that *'City'* and *'State'* classes are hard to differentiate (Figures 11-12). However, in comparison with USE model, IS models are more extensively described by first 3 calculated components, reaching 45 % of total variance described by both GloVe and FastText version. It will be seen in further analysis if this makes a difference or proves insignificant in the end.


PCA of embeddings_IS-GloVe

PCA of embeddings_IS-GloVe

PCA of embeddings_IS-FastText

PCA of embeddings_IS-FastText

Figures 9-10: Results of PCA for both IS models shown as 3-Dimensional graphs (2 views) depicting clustering of their embeddings vectors colored by their designated class (excluding *None*). The axes belong to the first 3 components from PCA and describe total variance explained by said components.

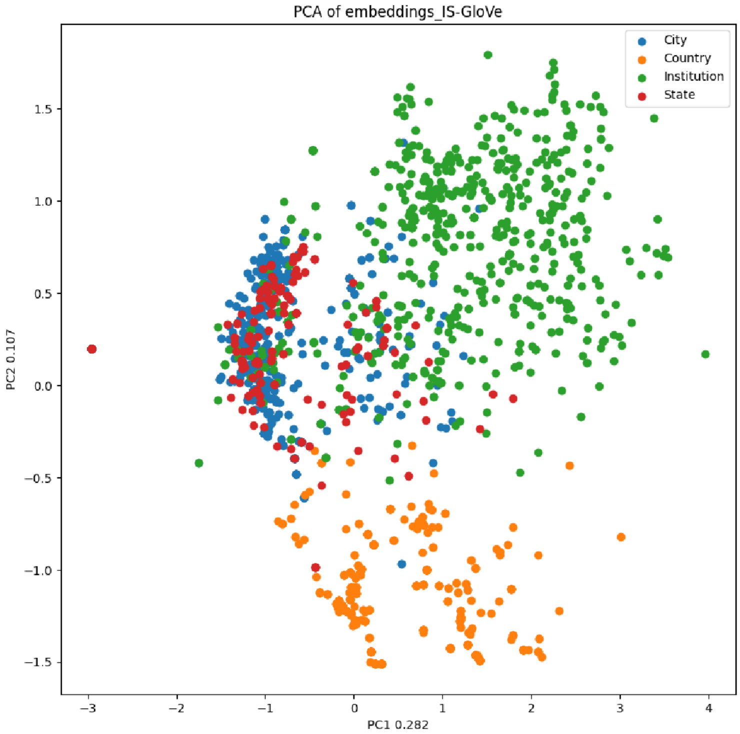


Figure 11: Results of PCA for IS-GloVe model presented as 2-Dimensional graph depicting clustering of its embeddings vectors colored by their designated (excluding *None*). The axes belong to the first 2 components from PCA and describe total variance explained by said components.

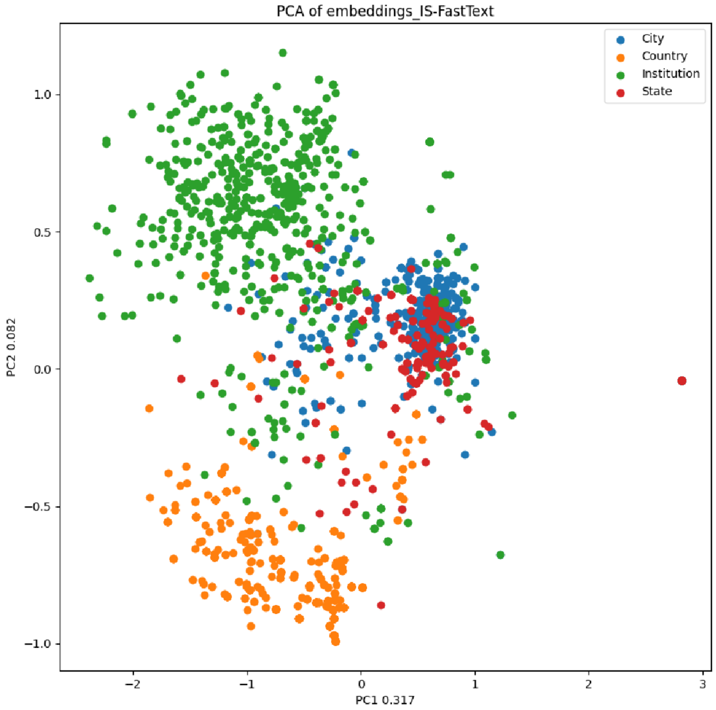
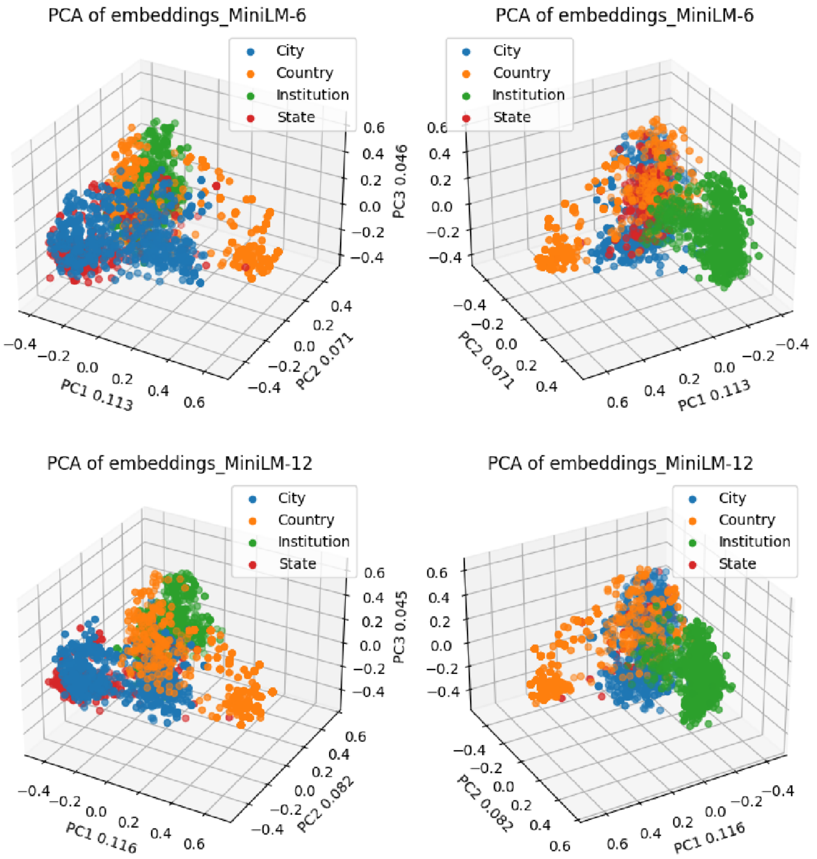


Figure 12: Results of PCA for IS-FastText model presented as 2-Dimensional graph depicting clustering of its embeddings vectors colored by their designated (excluding *None*). The axes belong to the first 2 components from PCA and describe total variance explained by said components.

4.1.5 MiniLM

Both MiniLM models exhibit distinct clusters in 3D PCA graphs (Figures 13-14) and crucially already for the first two components (Figrues 15-16). Additionally this clustering does not distinguish *'City'* and *'State'* classes very well, while *'Institution'* and *'Country'* clusters are well pronounced (Figures 13-16), similarly to clustering of last models. It can be seen on plot of PCA done with points labeled *'None'*, that these clusters remain to be distinguishable and that *'None'* class is predominantly on the negative side of first component axis, which helps with separating vectors labeled as *'Country'* (Figures 15-16). The same can be said for both MiniLM-6 and MiniLM-12. What seems to be an advantage of these models over the previously described clustering is the fact, that first two components acount for under 20 % of total variance which leaves enough room for other dimensions to distinctly separate desired classes. Therefore these models seem to be suitable for given task and one can expect them to perform well in following analyses.



Figures 13-14: Results of PCA for both MiniLM-6 model, and MiniLM-12 model shown as 3-Dimensional graphs (2 views) depicting clustering of their embeddings vectors colored by their designated class (excluding *None*). The axes belong to the first 3 components from PCA and describe total variance explained by said components.

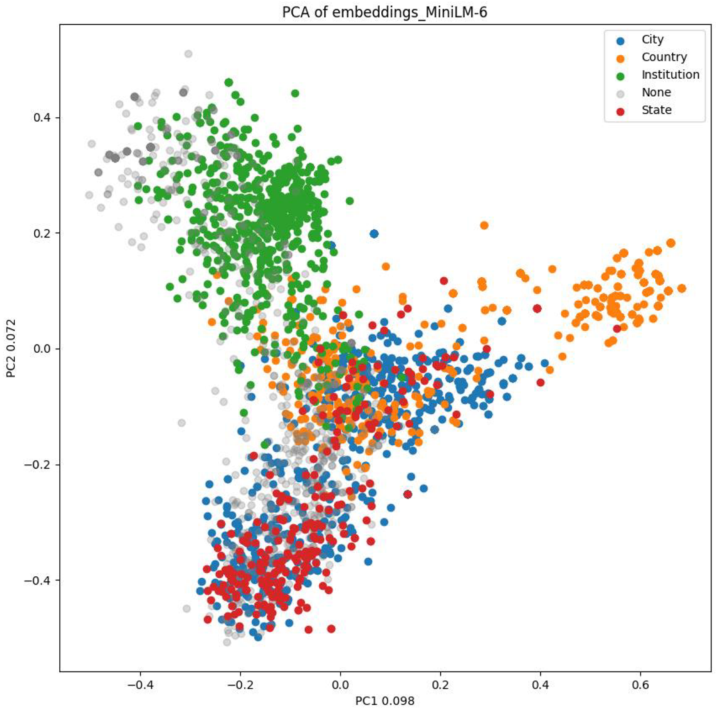


Figure 15: Results of PCA for MiniLM-6 model presented as 2-Dimensional graph depicting clustering of its embeddings vectors colored by their designated including *None* in transparent grey. The axes belong to the first 2 components from PCA and describe total variance explained by said components.

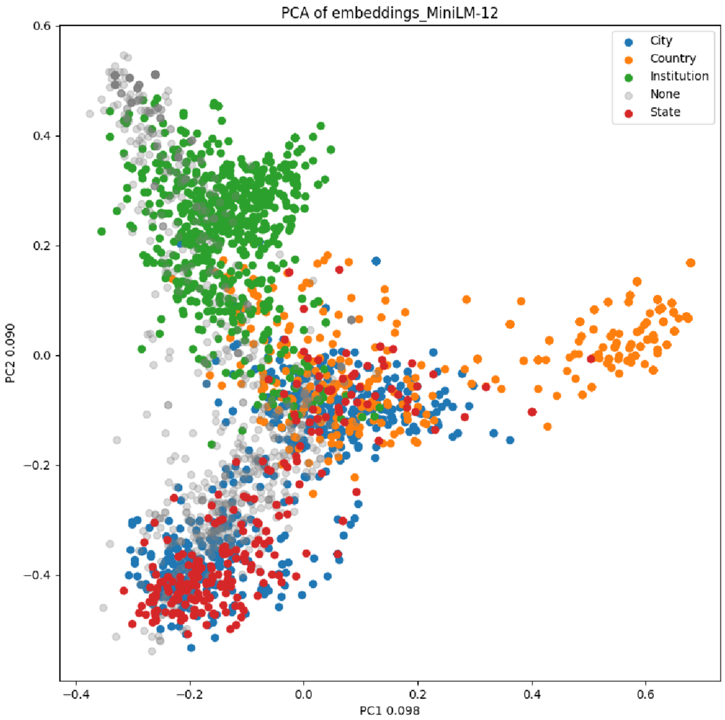


Figure 16: Results of PCA for MiniLM-12 model presented as 2-Dimensional graph depicting clustering of its embeddings vectors colored by their designated including *None* in transparent grey. The axes belong to the first 2 components from PCA and describe total variance explained by said components.

4.2 Performance of Pretrained Embeddings Models

With an idea about the potential suitability of different models, I performed Analysis of variance (ANOVA) to inspect the real performance of embedding models on the data.

Concerning statistical significance, it was needed to create enough data to analyze. This was achieved by K-fold Cross-Validation with k = 5 performed on training-set 5 times with different random states. This means that for every ML model used, 25 accuracy measurements were performed. Four ML classifiers were used, namely Neural Networks (NN) in form of Multilayer perceptron, Random Forest classifier (RF), Support Vector classifier (SV) and lastly K Nearest Neighbors (KNN). Therefore 100 scores were measured for every embedding model.

Nonparametric Kruskal-Wallis one-way ANOVA was performed due to the distribution of analyzed data not being normal, which is arguably a by-product of using 4 different ML models.

All models used were drawn from `scikit-learn` library for `python` and the scripts are shared via GitHub repository. Eventually ANOVA itself was performed on generated data in software `Statistica` in version 14.0.0. In following table p-values rounded to 4 digits after the decimal are presented (Table 1). Highlighted in red are p-values lower than 0.05, which shows the result being statistically significant, meaning that the performance of those two models is significantly different. On the other hand, p-values with changed background color ranging from yellow to blue emphasize that the similarity in performance cannot be denied with enough statistical evidence, suggesting their alikeness.

It is necessary to combine this information with the view of boxplots for the same accuracy data. As can be clearly seen in the first boxplot, both Doc2Vec models perform way worse than other models (Figure 17). This was expected from problems identified for them in previous section. This is obviously reflected also in the p-value matrix, where it is evident, that this couple of models is significantly different then remaining eight. Let's remove these two and continue to investigate the remaining models.

Table 1: Results of Analysis of Variance (ANOVA) for all used pre-trained word embeddings models (PWE). Input data for ANOVA were accuracies achieved for different machine learning (ML) models trying to perform classification task with embeddings vectors from said PWE models. P-values depicted in table represent how significant the difference in performance between PWE models is. Statistically significant difference (p-value ≤ 0.05) is highlighted with red color while for higher p-values, where one cannot deny the similarity of their performance from yellow to blue.

p-values	csd-BERT	uncsd-BERT	uncsd-BERT -1024	Doc2Vec -enwiki	Doc2Vec -apnews	IS-GloVe	IS-FastText	USE	MiniLM -6	MiniLM -12
csd-BERT	1.000	1.000	0.000	0.000	0.000	1.000	0.833	1.000	0.107	0.001
uncsd-BERT	1.000	1.000	0.000	0.000	0.000	1.000	0.181	1.000	0.533	0.007
uncsd-BERT-1024	0.000	0.000	1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000
Doc2Vec-enwiki	0.000	0.000	0.000	1.000	1.000	0.000	0.000	0.000	0.000	0.000
Doc2Vec-apnews	0.000	0.000	0.000	1.000	1.000	0.000	0.000	0.000	0.000	0.000
IS-GloVe	1.000	1.000	0.000	0.000	0.000	1.000	0.357	1.000	0.277	0.003
IS-FastText	0.833	0.181	1.000	0.000	0.000	0.357	1.000	0.000	0.000	0.000
USE	1.000	1.000	0.000	0.000	0.000	1.000	0.000	1.000	1.000	1.000
MiniLM-6	0.107	0.533	0.000	0.000	0.000	0.277	0.000	1.000	1.000	1.000
MiniLM-12	0.001	0.007	0.000	0.000	0.000	0.003	0.000	1.000	1.000	1.000

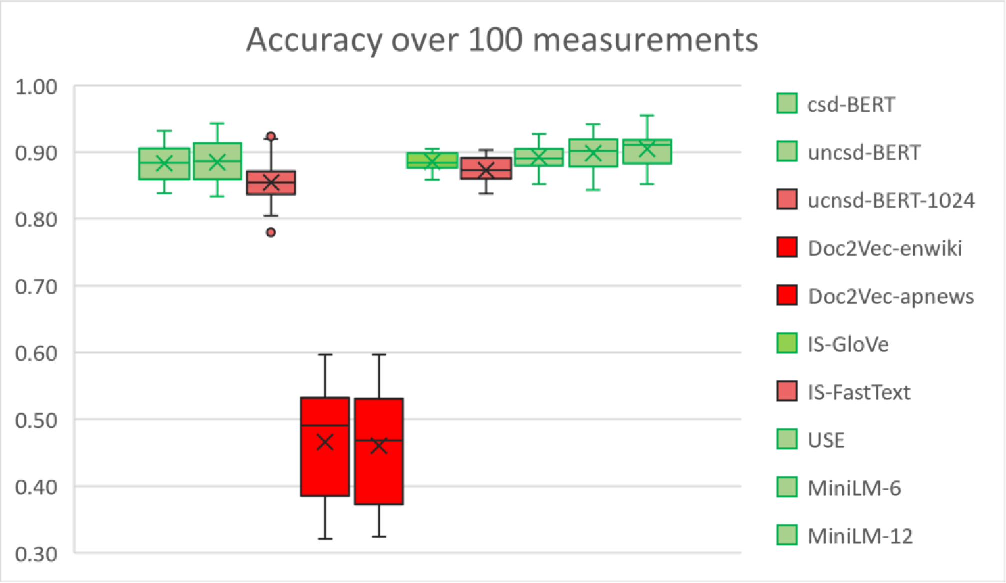


Figure 17: Boxplot of accuracies achieved by all 10 pre-trained word embeddings (PWE) models over multiple machine learning (ML) models. Models highlighted in green were deemed better for the task and eventually labelled as candidates (cPWE) while (depicted in dark red) both Doc2Vec models were omitted for their lackluster performance for this task.

If we look in more detail into the performances of remaining eight models in next boxplot, two models seem to achieve lower average accuracy (Figure 18). Model uncsd-BERT-1024 behaves like almost no other model and this is supported by p-values, where it is statistically different then every other model, expect IS-FastText.

For IS-FastText the results are that it is significantly distinctive from only USE and MiniLM models. While for BERT models and IS-GloVe no existing difference can be concluded, or else null hypothesis cannot be rejected.

Nevertheless, this does not disprove its possibility to be grouped with uncsd-BERT-1024. Based on their shared worse average accuracies and ANOVA data (Table 1) it can still be insisted, that both uncsd-BERT-1024 and IS-FastText should be omitted from the pool of candidate models.

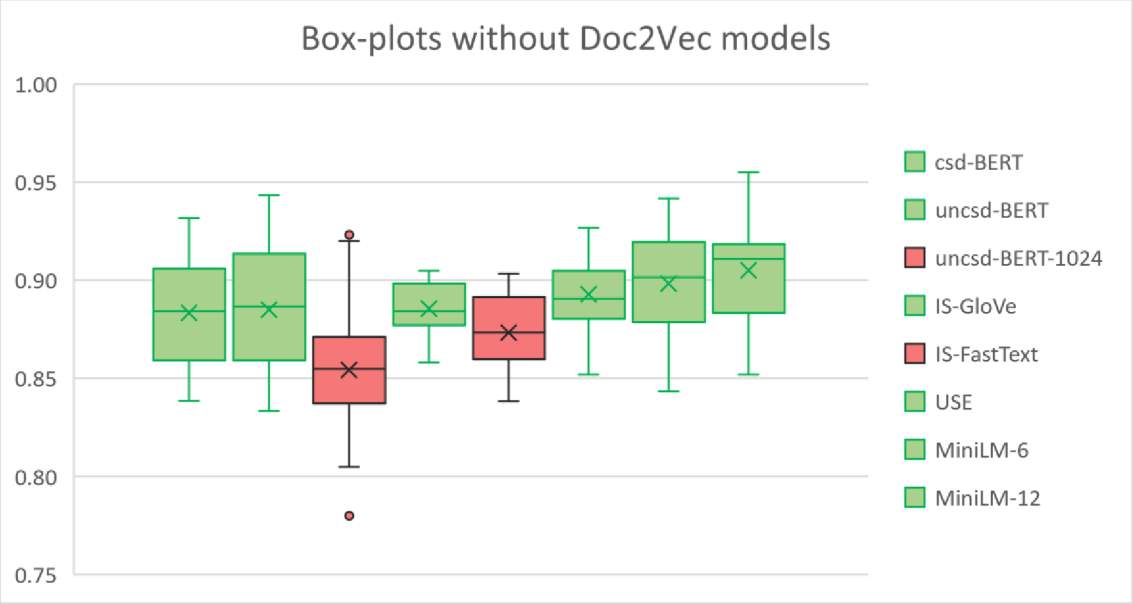


Figure 18: Boxplot of accuracies achieved by 8 better performing pre-trained word embeddings (PWE) models over multiple machine learning (ML) models. Models highlighted in green were deemed better for the task and labelled as candidates (cPWE). Additionally red-colored models were left out.

Table 2: Description of the performance of all pre-trained word embeddings (PWE) models via mean and median accuracies, standard deviation and finally minimal and maximal accuracy achieved.

model	mean	median	std	min	max
csd-BERT	0.883	0.884	0.027	0.838	0.932
uncsd-BERT	0.885	0.886	0.031	0.833	0.943
uncsd-BERT-1024	0.854	0.855	0.034	0.78	0.923
Doc2Vec-enwiki	0.466	0.49	0.085	0.321	0.597
Doc2Vec-apnews	0.46	0.469	0.082	0.324	0.597
IS-GloVe	0.886	0.884	0.013	0.858	0.905
IS-FastText	0.873	0.873	0.019	0.838	0.903
USE	0.893	0.891	0.017	0.852	0.927
MiniLM-6	0.898	0.902	0.026	0.843	0.942
MiniLM-12	0.905	0.911	0.025	0.852	0.955

In summary, I opted to keep 6 models highlighted in green as candidates (cPWE) owing to their high average accuracy. Their accuracy means and medians all exceed 0.88 (Table 2). Standard deviation ranging from 0.013 for IS-GloVe to 0.031 for uncsd-BERT. These models provide convenient accuracy while not being distinguishable with enough statistical evidence from one another. Candidates will help to determine which ML models achieve best results regarding their pairing with said candidate PWE models (cPWE).

4.3 Performance of ML Models on Candidates

After the evaluation of the data from the perspective of ML models, it became apparent that the distributions deviated significantly from the normal distribution. This means that, it was necessary to utilize a nonparametric version of ANOVA once again. However, given that only four ML models were involved, the use of ANOVA may have been unnecessary in this situation, as the results were not particularly interesting or surprising. In fact, a simple inspection of boxplots would have sufficed to identify that the NN and SV models outperformed the RF and KNN models (Figure 19).

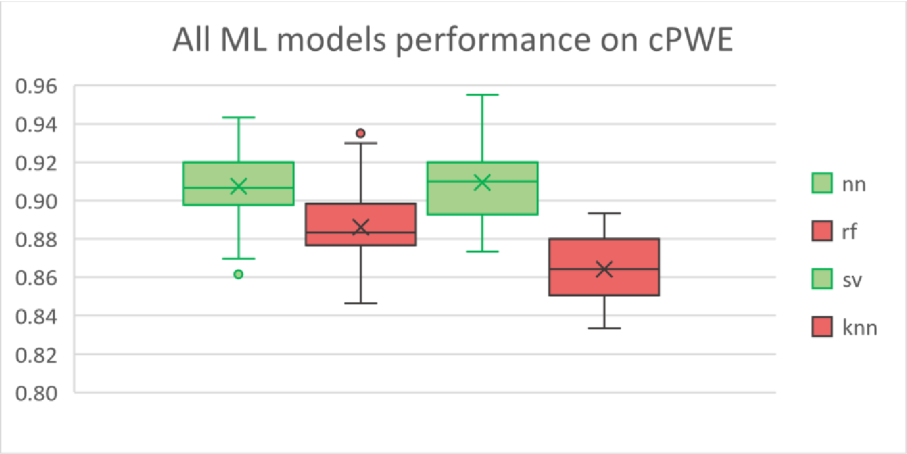


Figure 19: Boxplot of accuracies achieved by all machine learning (ML) models over chosen candidate pre-trained word embeddings (cPWE) models. ML models highlighted in green were deemed better for the task and labelled as candidates (cML). Additionally red-colored ML models were left out in further analyses.

Let's begin by discussing poorly performing ML models. Although KNN performed slightly worse than RF, its advantage lies in the fact that it does not require any training time, unlike RF which takes an extensive amount of time, increasing with the number of trees built. Results for KNN with k = 3 are presented, which yielded the best results for this setting. Additionally, results for k = 1 and k = 5 were also quite good, but performance declined for higher k values. The RF model was configured with hyperparameters that specified the number of estimators to be 1000 and maximum depth of trees to be 20. This was the best performing setting from a few tested, but it cannot be denied, that both models could potentially perform better with an extensive search for hyperparameters. Different models with various settings can be utilized, and ANOVA can be performed on multiple settings of the same model to make this analysis more robust. This study intentionally limits its scope to this narrow set of hyperparameters, given that its intended goal is to find only one adequate model.

It is important to note, that SV and KNN models do not rely on random seeding while training, this means, that the number of unique measured accuracies is 5 times smaller, than for NN and RF, which are dependent on random seeds. The random seeds in this case influence only the process of Cross-Validation. Additionally, it should be emphasized that the accuracy was calculated 5 times as for NN and RF to preserve the balance of the data in creation of previous statistics, especially the earlier table presenting values such as mean or median. For following analyses, the data could be reduced into its fifth since we are comparing each ML model separately.

Finally, both the NN and SV models achieved an accuracy rate of over 90% in terms of both median and mean (Figure 20). Upon closer examination, it is evident that the difference between them is statistically insignificant and the slight overperformance of SV is likely due to the lack of measurements for this model. This indicates that NN and SV are the models that should be continued with. The combination of the two candidate ML models (cML) with cPWE models creates candidate pairs (cP), which must be investigated to determine the best combination to solve the given task.

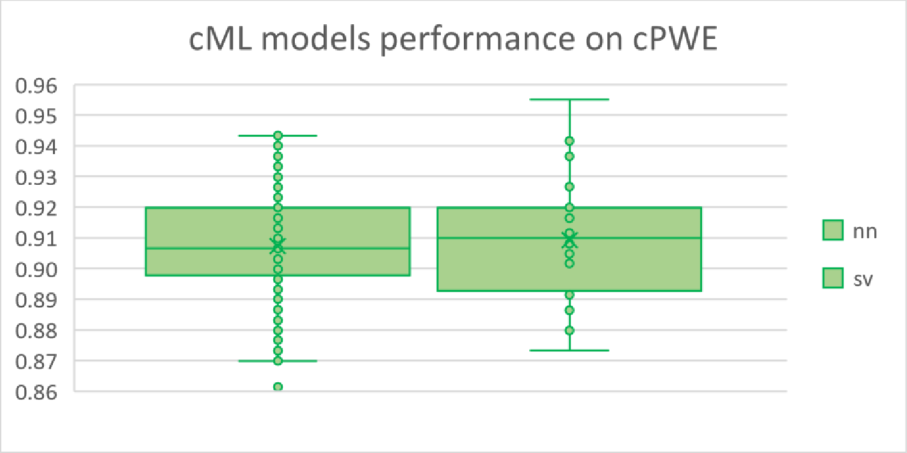


Figure 20: Detailed view of accuracies achieved by selected candidate machine learning (cML) models over candidate pre-trained word embeddings (cPWE) models in form of a boxplot.

Since the goal is to choose model combination to use in the final pipeline for classifying parts of the affiliations and finally geo-localizing the affiliation, it is necessary to pick single pair from the available cP. As one can see in the following table and boxplot (Table 3; Figure 21), there is a compelling argument for choosing almost any pair. For example, combining SV with any MiniLM model could be the first idea, as the accuracy of both PWE models is notably high, with one combination, SV+MiniLM-12, even scoring as high as 0.955 once. However, it is important to note that the data for SV is limited, and the maximum score achieved is even considered to be an outlier in the default boxplot graph produced in `Statistica`. This suggests that the promised accuracy may, in fact, be much lower than what the median of only 5 values promises.

To assure sufficient accuracy I decided to go with NN as the ML model of choice. Specifically in combination with uncsd-BERT, where, as can be seen in table and boxplot, the median accuracy is 0.920 and mean 0.919 with standard deviation being only 0.012. For one iteration the accuracy went down to 0.890 which seems to be an outlier. Although only slightly, this appears to be the best performing cP, furthermore, supported by 25 measurements.

Table 3: Description of the performance of selected candidate machine learning models (cML) in relation to the candidate pre-trained word embeddings (cPWE) models via means and medians of accuracies and standard deviations. The combination of cPWE and cML model was labelled as candidate pair (cP). Outlined in bold font is the best performing cP (NN+uncsd-BERT) which was selected to be used in a pipeline.

	NN				SV			
model	n	mean	median	std.dev.	n	mean	median	std.dev.
csd-BERT	25	0.907	0.902	0.017	5	0.897	0.893	0.018
uncsd-BERT	**25**	**0.919**	**0.920**	**0.012**	5	0.902	0.903	0.015
IS-GloVe	25	0.890	0.893	0.011	5	0.894	0.892	0.010
USE	25	0.895	0.898	0.010	5	0.911	0.910	0.009
MiniLM-6	25	0.914	0.917	0.013	5	0.924	0.920	0.014
MiniLM-12	25	0.918	0.915	0.013	5	0.927	0.918	0.016

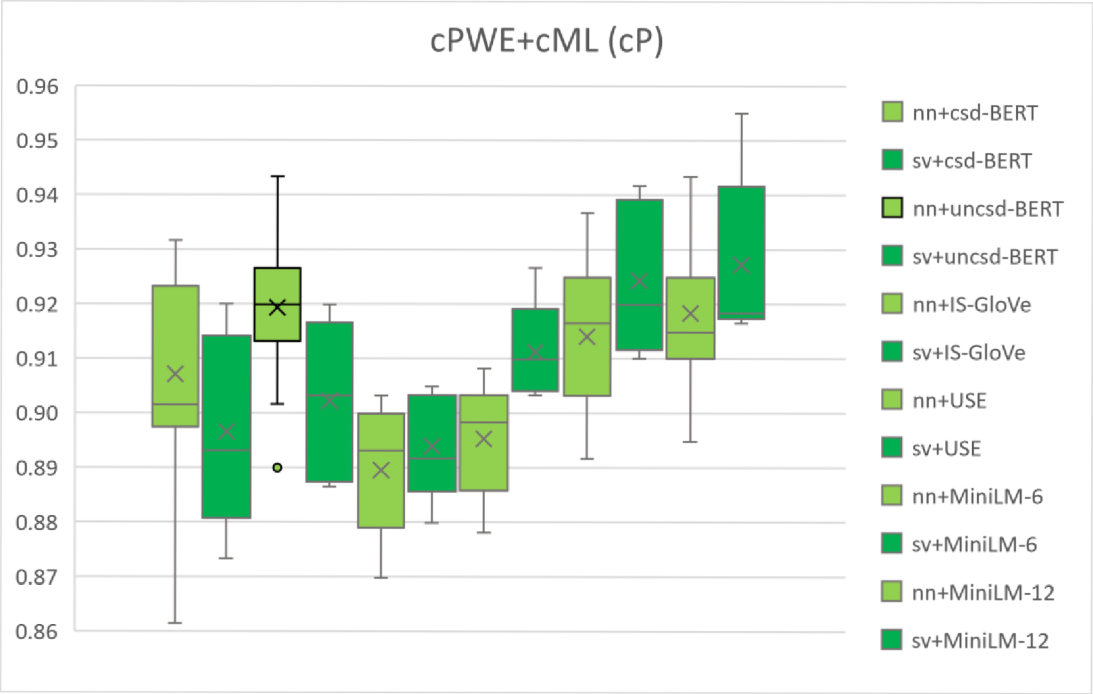


Figure 21: Detailed view of accuracies achieved by candidate pairs (cP), which is a combination of candidate pre-trained word embeddings (cPWE) model and a candidate machine learning (cML) model. Outlined cP was chosen as a best performing model combination.

4.4 Validation via Test-set

To confirm the desired accuracies achieved with the chosen cP, a final validation was performed on an external test-set. Although this dataset may have some similarities with the training set by the nature of the data, no specific part was used in the previous K-fold Cross-Validation. The test-set consists of approximately one-tenth of the training set, with 316 parts from 60 affiliations. The table below shows the accuracies obtained by all candidate pairs (cP) (Table 4). As observed earlier SV is not affected by random seedings, resulting in fewer measurements. However, all results align with the trends observed during Cross-Validation, with an average score of around 90%. Moreover, the chosen combination of SV+uncsd-BERT is one of the top performers, even on the test-set, which further proves the decision.

Table 4: Inspection of the accuracies of selected candidate pairs (cP) on the test-set data. Outlined in bold font is the best performing cP (NN+uncsd-BERT) which was selected to be used in a pipeline.

cP accuracy on test-set	**avg_nn (5)**	sv (1)
csd-BERT	0.907	0.930
uncsd-BERT	**0.929**	0.902
IS-GloVe	0.890	0.896
USE	0.909	0.927
MiniLM-6	0.930	0.918
MiniLM-12	0.911	0.918
avg	**0.913**	**0.915**
std	**0.015**	**0.014**

5 DISCUSSION

After choosing the combination of NN+uncsd-BERT as the model to use in the annotation pipeline, training on the whole training-set was performed and evaluated on test-set. In order to describe the learning process in more detail Cross-Validation was performed as part of validation_curve function. Heatmaps of confusion matrix were created as well as report table.

5.1 Pipeline inclusion.

This model will serve as a primary classification tool for identifying important parts of affiliations. It is worth noting that the model's implementation into the pipeline will most likely improve its performance, as the performance was evaluated ignoring the position of the identified part in the affiliation. This information was ignored to simplify the architecture of the classification model. Furthermore, we only need to predict one of each label, excluding 'None', which could simplify the problem and increase real accuracy. This means that the overall accuracy of the final geo-localization might be even higher than the accuracy achieved by NN+uncsd-BERT on its own. This is because the real task is in a way simpler than the version model learnt to solve. With the added information of the order of the parts and with the certainty of most classes appearing once in all correct affiliations, the final pipeline or a model build upon the basis of NN+uncsd-BERT can perform very well for the affiliation geo-localization task and thus its inclusion in the pipeline may fine-tune the model to correctly geo-localize all typical affiliations.

Ultimately, the goal is to prepare the important parts of the affiliation as an input to tools like Here: Geocoder API (https://developer.here.com/). This API utilizes data to produce geocoordinates and depends on annotated data to achieve optimal performance. However, it is noticeably more cost-effective than Google's API, which performs better on more disorganized data.

To reduce the number of queries for the geocoding API, clustering in the space of identified classes may be necessary to receive groups of possible affiliations from the same institution. This approach also enables projecting annotated affiliations into fewer dimensions.

5.2 Performance Measures (Confusion Matrix, F1-score, …).

The overall accuracy of the NN+uncsd-BERT on the test is 93% in other words, it classified 93% of the instances in the dataset correctly. The confusion matrix and classification report table summarize the model's performance (Figure 22; Table 5).

Looking at the precision and recall scores for each label in the following table, it is apparent that the model combination performs particularly well for *'Country'* and *'Institution'*, specifically over 95 %. These two labels, together with *'City'* are the most important for geo-localization. The *'City'* and *'None'* labels also have relatively high precision and recall scores, both above 90 %. Consequently, the identification of the 'State' class, where the model struggles and scores worse, is not that problematic for the entire classification process.

In conclusion, the macro-averaged and weighted-average F1-scores both exceed 91 %, indicating that the model is performing consistently across all labels.

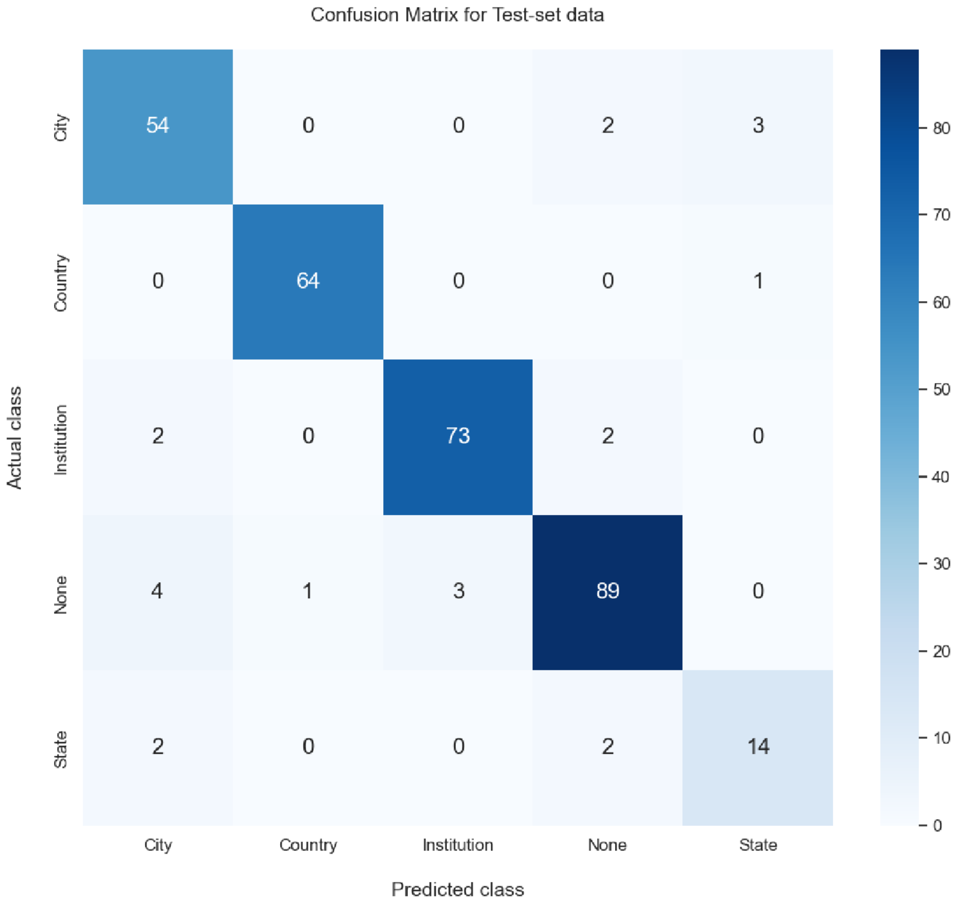


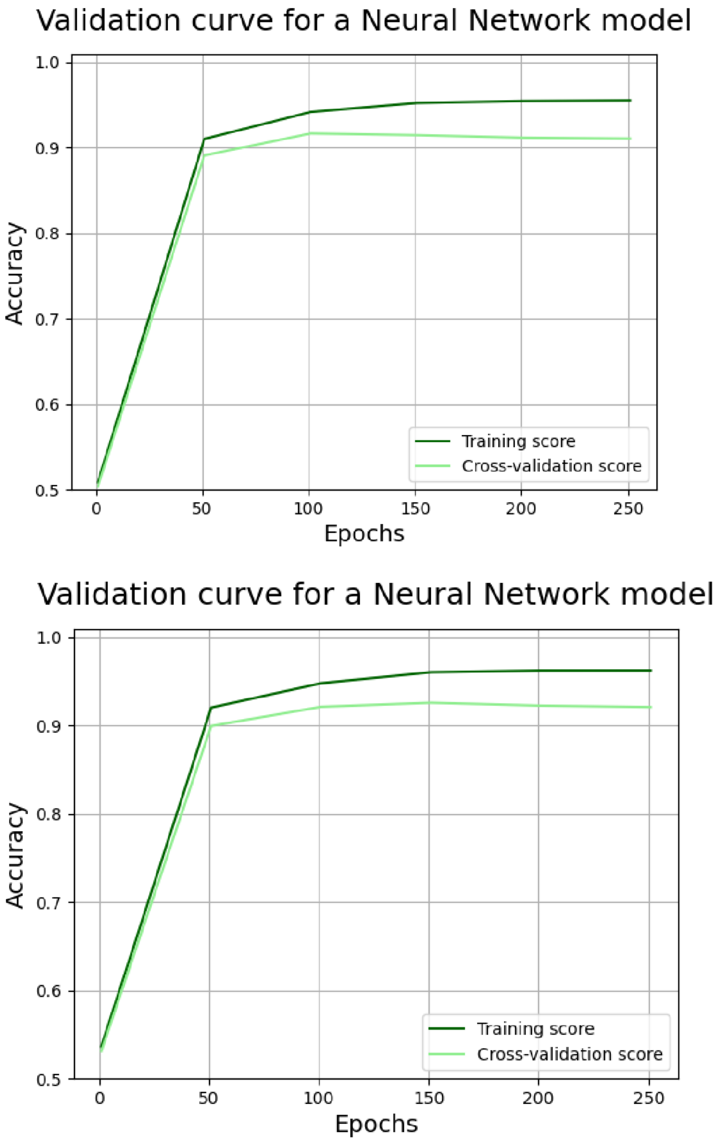
Figure 22: Confusion matrix depicting the performance of NN+uncsd-BERT combination on classification task. The heatmap shows the actual classes of the vectors in the dataset and which classes were predicted for these vectors by the model.

Table 5: Performance report table for the NN+uncsd-BERT combination is presented. Typical measures such as precision, recall, or F1-score are shown for each class. Support describes the number of instances in the test-set for certain class or overall. Accuracy, macro-average, and weighted-average F1-scores are included as well.

label	precision	recall	f1-score	support
City	0.87	0.92	0.89	59
Country	0.98	0.98	0.98	65
Institution	0.96	0.95	0.95	77
None	0.94	0.92	0.93	97
State	0.78	0.78	0.78	18
accuracy			0.93	316
macro avg	0.91	0.91	0.91	316
weighted avg	0.93	0.93	0.93	316

5.3 Learning and Speed of Convergence.

Validation curves were generated for an increasing number of epochs (iterations) and different random seeds. None of the curves seemed to deviate from the overall trend. Neural network scores around 90 % accuracy after first 50 epochs and then slowly converges reaches the maximum. However, after 100 to 150 epochs its accuracy on omitted part of the set decreases slightly, indicating possible overfitting. Decreasing the number of iterations or neurons to avoid overfitting should be considered.



Figures 23-24: Validation curves of the learning process of NN+uncsd-BERT for two initial random states. Unsurprisingly, both show steep progress for first 50 epochs and then mild increase in accuracy in the following 50. After 100-150 epochs slight decrease in Cross-validation score can be observed suggesting possible overfitting.

5.4 Reflecting on predicted potential of respective PWE models.

Based on the performance of ML models using various PWE models, it can be assumed that the accuracy of classification reflects the predictions made about the usability and natural clustering of embedding vectors. It was clear that Doc2Vec models were not suitable for this task, as evidenced by their poor performance. Some models, such as uncsd-BERT-1024 and IS-FastText, performed slightly worse than expected, but further analysis would be required to determine the reasons for this. The remaining models met the expectations made about their suitability. It is safe to say, that the initial view through PCA or any other multidimensional analysis offers highly valuable, yet not complete information about the potential of embedding models, or any vectors for that matter.

5.5 Comparison with naïve use of GPT3

What seems like an interesting comparison to include is to relate the performance to the one of GPT3. I used its API to perform the same classification task and measured the accuracy. Importantly, the code used to perform this task was also written by GPT3 itself and only necessary change was to add personal OpenAI API key. The performance was as follows:

Accuracy: 0.69; Precision: 0.78; Recall: 0.69; F1-score: 0.69.

Even though it can be argued that the prompt and parameters such as temperature, might be adjusted to improve performance, each of my attempts ended in similar or worse performance on test_data than the first script suggested by ChatGPT3 itself. I don't deny that with more experience, understanding of mistakes the model makes and by using better prompts its performance will increase, but such insight would require more time and the point was to calculate performance of naive use of GPT3.

This result is somewhat impressive considering there was no added training performed and was in fact unsupervised. Where GPT3 might have had problems are instances, where mixed data like for example: *"country name + email"* in the same input occurred and since this model had no information about correct labels, it might have predicted *'None'*, as it deemed email to be more important. It is still encouraging to note that results of statistically chosen, and extensively trained model is not outperformed by naïve version of GPT3 script.

6 CONCLUSION

Multiple freely available models for generating embeddings using pre-trained word embeddings (PWE) were used in this study to analyze the suitability of the embeddings for the task of classification. Statistical methods, such as PCA or ANOVA, were utilized to identify the most suitable models. These models were combined with different machine learning (ML) models, including Neural Networks, Random Forests, Support Vector Classifier, and K Nearest Neighbors model. The NN+uncsd-BERT combination was found to be the best performing model, and its learning process and classification performance were further evaluated.

This work is a case study that demonstrates how to choose a suitable model for a specific goal and provides a modest insight into the state-of-the-art performance of such models. The primary aim of this study was to develop a machine learning model for geo-localizing affiliations without relying on any commercial tools.

In conclusion, this work presents a successful approach for identifying and geo-localizing affiliations using only freely available tools. The developed model is a valuable addition to the data-analysis pipeline for the prepared web-application. Although there may be larger models or commercial tools that can achieve higher accuracy, the statistical evaluation and careful selection of appropriate models for specific tasks emphasized in this work are of a significant importance.

7 REFERENCES

7.1 Literature

- Beltagy, I., Lo, K., Cohan, A. (2019) SciBERT: A Pretrained Language Model for Scientific Text. *In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 3615–3620, Hong Kong, China. Association for Computational Linguistics. doi: 10.18653/v1/D19-1371.

- Bengio, Y., Ducharme, R., Vincent, P. (2003) A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137-1155. doi: 10.5555/944919.944966.

- Bishop, C. M. (1995) *Neural Networks for Pattern Recognition*, Oxford: Claredon Press., Available at: Book-Bishop-Neural Networks for Pattern Recognition.pdf (sabanciuniv.edu)

- Borji, A. (2022) Generated Faces in the Wild: Quantitative Comparison of Stable Diffusion, Midjourney and DALL-E, *ArXiv*. doi: 10.48550/arXiv.2210.00586.

- Brown, T.B. et al. (2020) Language Models are Few-Shot Learners, *NIPS'20: Proceedings of the 34th International Conference on Neural Information Processing* Systems, 33(159), 1877-1901. Available at: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

- Cer, D., Yang, Y., Kong, S., Hua, N., Limtiaco, N., et al. (2018) Universal Sentence Encoder, *ArXiv*, 1803.11175. doi: 10.48550/arXiv.1803.11175.

- Coley, C. W., Barzilay, R., Green, W. H., Jaakkola, T. S., Jensen, K. F. (2017) Convolutional Embedding of Attributed Molecular Graphs for Physical Property Prediction, *Journal of Chemical Information and Modeling*, 57 (8), 1757-1772. doi: 10.1021/acs.jcim.6b00601.

- Conneau, A., Kiela, D., Schwenk, H., Barrault, L., Bordes, A. (2017) Supervised Learning of Universal Sentence Representations from Natural Language Inference Data; *ArXiv*. doi: 10.48550/arXiv.1705.02364.

- Devlin, J., Chang, M-W., Lee, K., Toutanova, K. (2018) BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, *ArXiv*. doi: 10.48550/arXiv.1810.04805.

- Goodfellow, I., Bengio, Y., Courville A. (2016) *Deep Learning*. Cambridge, MA: MIT Press. Available at: https://www.deeplearningbook.org/.

- Hinton, G., Vinyals, O., Dean, J. (2015) Distilling the Knowledge in a Neural Network. *ArXiv*. doi: 10.48550/arXiv.1503.02531.

- Jolly, K. (2018) *Machine Learning with Scikit-Learn Quick Start Guide: Classification, Regression, and Clustering Techniques in Python*. Birmingham, UK: Packt Publishing., Available at: https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,shib&db=e000xww&AN=1936459&authtype=shib&site=eds-live&scope=site&authtype=shib&custid=s7108593. (Accessed: 25.04.2023).

- Jumper, J., Evans, R., Pritzel, A. et al. (2021) Highly accurate protein structure prediction with AlphaFold., *Nature*, 596, 583–589. doi: 10.1038/s41586-021-03819-2.

- Jurafsky, D., Martin, J. H. (2023) *Speech and Language Processing (3rd ed. draft)*, Available at: https://web.stanford.edu/~jurafsky/slp3/. (Accessed: 25.04.2023).

- Kruse, R., Borgelt, C., Braune, C., Mostaghim, S., Steinbrecher, M. (2016) *Computational Intelligence A Methodological Introduction (Second Edition)*, London: Springer, doi: https://doi.org/10.1007/978-1-4471-7296-3.
- Lau, J. H., Baldwin, T. (2016) An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation. *In Proceedings of the 1st Workshop on Representation Learning for NLP*, Available at: https://github.com/jhlau/doc2vec.
- Le, Q. V., Mikolov, T. (2014) Distributed Representations of Sentences and Documents, *Proceedings of the 31st International Conference on Machine Learning*, PMLR 32(2):1188-1196. Available at: https://proceedings.mlr.press/v32/le14.html.
- Machàň, J. (2021) Mapování struktur současných chemických látek., *Bachelor's thesis (Bc.).* Olomouc, Palacký University Olomouc. Faculty of Science. Available at: https://theses.cz/id/doscu6/?lang=cs.
- Mars, M. (2022) From Word Embeddings to Pre-Trained Language Models: A State-of-the-Art Walkthrough. *Appl. Sci*., 12 (17), 8805. doi: 10.3390/ app12178805.
- Meijer, H. J., Truong, J., Karimi, R. (2021) Document Embedding for Scientific Articles: Efficacy of Word Embeddings vs TFIDF. *ArXiv* Volume: 2107.05151. doi: 10.48550/arXiv.2107.05151.
- Mikolov, T. (2008) Language models for automatic speech recognition of Czech lectures, *Master's thesis*, Brno University of Technology. Available at: https://www.fit.vutbr.cz/research/groups/speech/publi/2008/mikolov_eeict2008.pdf
- Mikolov, T., Kopecký, J., Burget, L., Glembek, O., Černocký, J. (2009) Neural network-based language models for highly inflective languages, In: *Proc. ICASSP 2009*, Available at: http://www.fit.vutbr.cz/research/groups/speech/publi/2009/mikolov_ic2009_nnlm_4.pdf.
- Mikolov, T., Corrado, G., Chen, K. & Dean, J. (2013a) Efficient estimation of word representations in vector space. *ArXiv*. doi: 10.48550/arXiv.1301.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J. (2013b) Distributed Representations of Words and Phrases and their Compositionality, *ArXiv*. doi: 10.48550/arXiv.1310.4546.
- Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., Joulin, A. (2017) Advances in Pre-Training Distributed Word Representations. *ArXiv*. doi: 10.48550/arXiv.1712.09405.
- Morris, P., St. Clair, R., Hahn, W. E., Barenholtz, E. (2020) Predicting Binding from Screening Assays with Transformer Network Embeddings; *Journal of Chemical Information and Modeling*, 60 (9), 4191-4199; doi: 10.1021/acs.jcim.9b01212.
- Murphy, K. P. (2012) *Machine Learning: A Probabilistic Perspective*, Cambridge, MA: MIT Press. Available at: noiselab.ucsd.edu/ECE228/Murphy_Machine_Learning.pdf.
- Narasimhan, A. & Rao, K. & M.B, V. (2021). CGEMs: A Metric Model for Automatic Code Generation using GPT-3., *ArXiv*. doi: 10.48550/arXiv.2108.10168.
- Pennington, J., Socher, R., Manning, C. D. (2014) GloVe: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. doi: 10.3115/v1/D14-1162.
- Sammut, C., Webb, G. I. (2011) *Encyclopedia of Machine Learning*, Boston, MA.: Springer, doi: 10.1007/978-0-387-30164-8_832.
- Skolnick J., Gao M., Zhou H., Singh S. (2021) AlphaFold 2: Why It Works and Its Implications for Understanding the Relationships of Protein Sequence, Structure, and Function., *Journal of Chemical Information and Modeling*, 61(10), 4827-4831. doi: 10.1021/acs.jcim.1c01114.

- Tshitoyan, V., Dagdelen, J., Weston, L. et al. (2019) Unsupervised word embeddings capture latent knowledge from materials science literature. *Nature* 571, 95–98. doi: 10.1038/s41586-019-1335-8.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., Polosukhin, I. (2017) Attention is all you need, *ArXiv*. doi: 10.48550/arXiv.1706.03762.
- Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., Zhou, M. (2020) MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. *ArXiv*. doi: 10.48550/arXiv.2002.10957.
- Wang, W., Bao, H., Huang, S., Dong, L., Wei, F. (2021) MiniLMv2: Multi-Head Self-Attention Relation Distillation for Compressing Pretrained Transformers. *ArXiv*. doi: 10.48550/arXiv.2012.15828.
- Wisam, A. Q., Ameen, M. M., Bilal, I. A. (2019) An Overview of Bag of Words;Importance, Implementation, Applications, and Challenges. *International Engineering Conference (IEC)*, Erbil, Iraq, 2019, pp. 200-204, doi: 10.1109/IEC47844.2019.8950616.

7.2 Software, repositories, hubs, and documentations

The GitHub repository with my code, tables, and figures available at:

- https://github.com/Najlaron/Diploma-Thesis

Else:

- https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/4
- https://tfhub.dev/tensorflow/bert_en_cased_L-12_H-768_A-12/4
- https://tfhub.dev/tensorflow/bert_en_uncased_L-24_H-1024_A-16/4
- https://github.com/RaRe-Technologies/gensim
- https://tfhub.dev/google/collections/universal-sentence-encoder/1
- https://github.com/facebookresearch/InferSent
- https://www.sbert.net/docs/pretrained_models.html
- https://github.com/UKPLab/sentence-transformers/blob/master/docs/pretrained_models.md
- scikit-learn: machine learning in Python — scikit-learn 1.2.2 documentation
- https://scikit-learn.org/stable/modules/neural_networks_supervised.html#multi-layer-perceptron
- https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html#sklearn.neural_network.MLPClassifier
- https://scikit-learn.org/stable/modules/neural_networks_supervised.html#multi-layer-perceptron
- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier
- https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC
- https://scikit-learn.org/stable/modules/neighbors.html#nearest-neighbors-classification
- https://pubmed.ncbi.nlm.nih.gov
- TIBCO Software Inc. (2020). Data Science Workbench, version 14. http://tibco.com.

8 LIST OF ABBREVIATIONS

ML = Machine Learning

AI = Artificial Intelligence

NN = Neural Networks

MLP = Multilayer Perceptron

RF = Random Forests

SV(M/C) = Support Vector (Machine/Classifier)

KNN = K-Nearest Neighbors

DL = Deep Learning

PWE = pre-trained word embedding

(C)BoW = (Continuous) Bag of Words

TF-IDF = Term Frequency – Inverse Document Frequency

NLP = Natural Language Processing

BERT = Bidirectional Encoder Representations from Transformers

USE = Universal Sentence Encoder

PCA = Principal Component Analysis

ANOVA = Analysis of Variance

cPWE model = candidate pre-trained word embedding model

cML model = candidate machine learning model

cP = candidate pair (of models)