Czech Technical University in Prague
Faculty of Nuclear Sciences and Physical Engineering

Evolution of meaning of words in time

Evoluce významu slov v jazyce

MASTER'S THESIS

Author:	Bc. Dominika Zogatová
Supervisor:	Ing. Tomáš Mikolov, Ph.D.
Academic year:	2022/2023

České vysoké učení technické v Praze, Fakulta jaderná a fyzikálně inženýrská

Katedra: matematiky Akademický rok: 2021/2022

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student:	Bc. Dominika Zogatová
Studijní program:	Aplikované matematicko-stochastické metody
Název práce (česky):	Evoluce významu slov v jazyce
Název práce (anglicky):	Evolution of meaning of words in time

Pokyny pro vypracování:

1) Seznamte se s matematickými základy metod strojového učení pro zpracování přirozeného jazyka.
2) Podejte souhrn těchto metod.
3) Prostudujte zdroje textových dat, jejichž pomocí je možné vytvořit korpusy pro různá časová období (např. Common Crawl nebo Twitter).
4) Vytvořte modely jazyka (založené například na statistice, teorie grafů, ...) pomocí korpusů vytvořených v bodě 2.
5) Prozkoumejte změny vztahů mezi vybranými klíčovými slovy v čase, například pomocí vizualizace jazykové mapy.
6) Najděte příklady slov, jejichž význam se v posledních letech nejvíce posunul.

Doporučená literatura:

1) S. Bird, E. Klein, E. Loper, J. Steele, R. Romano, Natural Language Processing with Python. O'Reilly Media, Inc, USA. 2009.
2) A. Clark, C. Fox, S. Lappin, The Handbook of Computational Linguistics and Natural Language Processing. Blackwell Publishing Ltd. 2010.
3) T. Wolf, et al., Transformers: State-of-the-Art Natural Language Processing. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, 38–45.
4) X. Qiu, T. Sun, Y. Xu, et al., Pre-trained models for natural language processing: A survey. SCIENCE CHINA Technological Sciences 63, 2020, 1872–1897.

Jméno a pracoviště vedoucího diplomové práce:

Ing. Tomáš Mikolov, Ph.D.
Český institut informatiky, robotiky a kybernetiky, ČVUT v Praze (CIIRC), Jugoslávských partyzánů 1580/3, 160 00 Dejvice

Jméno a pracoviště konzultanta:

Datum zadání diplomové práce: 31.10.2021

Datum odevzdání diplomové práce: 2.5.2022

Doba platnosti zadání je dva roky od data zadání.

V Praze dne 01.11.2021

Milan Krbálek

garant oboru

[illegible]

vedoucí katedry

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta jaderná a fyzikálně inženýrská -1-

[illegible]

děkan

Author's declaration:

I declare that this Master's Thesis is entirely my own work and I have listed all the used sources in the bibliography.

Prague, January 4, 2023

[illegible]

....

Bc. Dominika Zogatová

Acknowledgment:

I would like to express my gratitude to Ing. Tomáš Mikolov, Ph.D. for his expert guidance.

Bc. Dominika Zogatová

Název práce:
Evoluce významu slov v jazyce

Autor: Bc. Dominika Zogatová

Program: Aplikované matematicko-stochastické metody

Druh práce: Diplomová práce

Vedoucí práce: Ing. Tomáš Mikolov, Ph.D.
Český institut informatiky, robotiky a kybernetiky, ČVUT v Praze

Abstrakt: Jazyk je primárním způsobem komunikace a každý den přijdeme do styku z mnoha jeho formami. Zaznamenávání a kvantifikování změn významu určitých slov je dobrým způsobem jak monitorovat vztah veřejnosti k těmito slovům. Tato práce přestavuje nový způsob jak analyzovat a kvantifikovat evoluci významu slov. Součástí této diplomové práce je rozsáhlý úvod do strojového učení a zpracování přirozeného jazyka. Dále tato práce prezentuje zpracování dat Common Crawl z různých časových období, vytvořené Word2Vec modely a způsob měření změny významu slova založený na technikách automatického překladu textu.
Klíčová slova: Common Crawl, evoluce významu slov, překlad jazyka, Word2Vec, zpracování přirozeného jazyka

Title:
Evolution of meaning of words in time

Author: Bc. Dominika Zogatová

Abstract: Language is a primary means of communication and each day everybody comes into contact with its many forms. Being able to notice and quantify changes in the meaning of words is a good way of capturing the changes in attitude towards those words. This thesis presents a new way of analysing and quantifying evolution of meaning of words. This thesis consists of a theoretical and mathematical introduction to machine learning and natural language processing. Additionally, this work presents a form of processing Common Crawl data from different time corpora, creates multiple Word2Vec models based on those corpora and introduces a method of analysing the evolution of meaning of words based on automatic translation techniques.
Key words: Common Crawl, evolution of meaning of words, language translation, natural language processing, Word2Vec

Contents

Introduction **11**

1 Introduction to natural language processing **13**

1.1 Machine learning 13

1.1.1 Neural networks 14

1.1.2 Learning 18

1.1.3 Performance metrics 22

1.2 Natural language processing 24

1.2.1 Tokenization 25

1.2.2 Word embedding 25

1.2.3 Recurrent neural networks 27

1.2.4 Convolutional neural networks 28

1.2.5 Transfer learning 28

2 Analysis **31**

2.1 Common Crawl 31

2.2 Data and processing 31

2.3 Data analysis 31

2.3.1 Time clusters 33

2.3.2 Models 35

2.3.3 Language translation 35

2.3.4 Word mappings 39

3 Results **41**

Conclusion **51**

Bibliography **52**

Appendix **55**

A List of Acronyms **55**

Introduction

Language is something all of us use in our everyday life. Each of us uses language for various reasons, as a form of exchanging information, for pleasure or as a way of achieving a goals. There are thousands of languages being spoken today all around the world and all languages keep on evolving every day.
During recent years there has been a noticeable change in attitude towards many people, places and other words. The evolution of the semantic meaning of the words changes and this change can be detected when examining the sentence in which the word appears. One easy example is seen during the election period. Each candidate's name exists in many sentences during the pre-election period. The sentiment of the sentences in which the name Donald Trump or Joe Biden appeared has evolved based on the events such as their or their opponent's speeches, public affairs or people expressing their support for or criticism of each candidate. Another example of a word that changes its sentiment as the attitude of people towards it changes is Bitcoin. In fact, sentiment analysis of textual data linked to Bitcoin was used to predict the price of this currency many times [8], [13]. The recent energy crisis was one of the biggest topics of the past year and words such as *oil* or *gas* appeared more in media, and the attitude towards energy has shifted. Being able to capture this change can help quantify the otherwise unquantifiable phenomenon.
The meaning of a word does not often change drastically. The change in attitude towards some words, people, brands or companies over time can be leveraged in economics, politics, finance and many others. Marketing is another example worth mentioning. The attitude towards a brand over time changes based on information available to the public and media exposure (good and bad) of that particular brand. By monitoring the attitude of people towards a brand the company can change and validate the marketing strategy.

The birth of the machine learning discipline is linked to the publication made in 1943 by logician W. Pitts and neuroscientist W. McCulloch who tried to mathematically describe the decision-making process in humans. They described a model known today as a *perceptron.* Famous computer scientist Alan Turing formalized a test in 1950, which tells if a machine can be considered intelligent called *Turing Test.* During the test, a judge carries two conversations, one with another human being and one with a machine. If the judge cannot distinguish between a machine and a human, the machine passed the test and is considered intelligent. In 1957, F. Rosenblat first implemented a perceptron introduced a few years earlier in 1943 by Pitts and McCulloch. Developed in 1997, Long Short-Term memory neural networks

revolutionized speech recognition later in the first decade of the 21st century. In 2010 *Kaggle* was founded, a website hosting machine learning competitions and offering numerous datasets making creating machine learning models more accessible to the public. Facebook reached another huge milestone in the field of artificial intelligence in 2014. A team of researchers working at Facebook have created *Deep face*, an algorithm which reached the accuracy of 97.35% in recognizing human faces. This level of accuracy is equal to the performance of humans. Nowadays, machine learning and neural networks can be used for face recognition, self-driving cars, fraud detection, dynamic pricing, personalized advertisements or decision-making and many others. The field of machine learning is one of the most quickly growing disciplines and there are machine learning tools that can be leveraged to effectively analyze data called neural networks.
Artificial neural networks Artificial neural network (ANN) are computing systems inspired by biological neural networks which can be found in human or animal brains. The use of ANN has changed many industries. Today, neural networks are used in almost every industry from computer science or biology to weather forecasting or medicine. Neural networks can be applied to many sorts of data including images, text, numeric data, videos and speech.
The portion of machine learning which uses neural networks and other tools to analyze textual data is called natural language processing.
One of the most difficult steps of building any machine learning algorithm is finding good-quality data that is large enough for the task at hand. Most real-life data is very sparse, meaning there are not enough data points to describe the whole feature space model is trying to learn. There are some datasets widely used in academia and research communities such as MNIST [2] or ImageNet [3]. These datasets are available to anyone. Using the same data and the same performance metric makes comparing different models. The biggest source of all the data on the internet available to the public is the Common Crawl dataset. Common Crawl was used in the practical part of this thesis.

The goal of this thesis is to give the reader an introduction to machine learning and natural language processing. Further this thesis aims to develop models which capture the evolution of meaning of words mathematically and create a way of measuring the way in which the meaning of a word changes over time.

This thesis is divided into the following chapters. Chapter 1 is an introduction to machine learning and natural language processing. This chapter also summarizes the mathematical foundations of both. The most popular neural networks for language processing are also described in chapter 1. Chapter 2 discusses the data used in this thesis and provides an analysis of the data and model summaries. Chapter 4 discusses the results and the last chapter concludes.

Chapter 1

Introduction to natural language processing

1.1 Machine learning

Machine learning (ML) can be classified into three categories supervised learning, unsupervised learning and semi-supervised learning. During supervised learning, an algorithm is trained on labelled data to predict a correct label and then used to predict a label of unlabeled data. On the other hand, sets of unlabeled data are used during unsupervised learning where the algorithm learns patterns in the data. An example of unsupervised learning can be clustering or Principal component analysis (PCA). Combination of supervised and unsupervised learning is semi-supervised learning. During semi-supervised learning, a small labelled data set is used along with a large unlabeled data set. An example of a semi-supervised algorithm is self-training. During self-training, a machine learning model learns on a small amount of labelled data and then makes predictions about large data sets. The large datasets together with predicted labels are then used again to train the model. Later, this model makes predictions on test data.
One more class of ML algorithms can be distinguished called reinforcement learning algorithms. In Reinforcement learning (RL) an agent tries to maximize an objective function by taking certain actions in the environment that they are in. The agent learns correct actions by getting rewarded.

Deep learning

Deep learning (DL) is a subset of machine learning. The difference between DL and ML is that DL is based on artificial neural networks (ANN) whereas ML includes also algorithms other than ANN. Generally, due to the high number of hyperparameters, DL algorithms require more data than ML algorithms [15]. ML needs to design features manually which rapidly increases the time and cost of such methods, DL is in that way much more scalable [14]. The word *deep* in deep learning refers to the depth of a neural network. A network is considered deep if it consists of more than three layers.

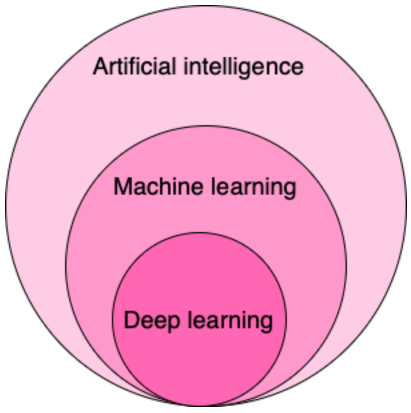

Figure 1.1: The relationship between artificial intelligence, machine learning and deep learning.

1.1.1 Neural networks

Sometimes called artificial neural networks to distinguish them from their biological equivalent found in the brains of animals, neural networks form the base unit of any deep learning models. In order to understand what neural networks are, first one should examine the very basic building block of any artificial neural network, a perceptron.

Perceptron is the simplest example of a neural network built of only one neuron. Perceptron is a binary classifier and also an example of a supervised learning algorithm. Given a number of numerical inputs, the perceptron learns *weights* of each input variable to correctly predict the desired output. Different weights of each input variable correspond to its importance. Figure 1.2 is a visualization of a perceptron.

The perception is constructed in the following way. First, select a threshold t when the perception gets activated and predicts a positive value. Let x_i denote the i-th input variable of the perceptron and w_i a corresponding weight of that variable. All input variables get multiplied by the corresponding weight and the multiplication results get summed. If the final sum is greater than the threshold t then the final prediction is equal to 1 otherwise the output is 0. This can be rewritten mathematically as follows:

$$g(\boldsymbol{x}) = \begin{cases} 1, & \text{if} \quad \boldsymbol{w} \cdot \boldsymbol{x} = \sum_i w_i x_i > t \\ 0, & \text{otherwise} \end{cases} \tag{1.1}$$

Multilayer perceptron (MLP) is a more complex artificial neural network based on the same principles as a perceptron explained above. MLP consist of at least

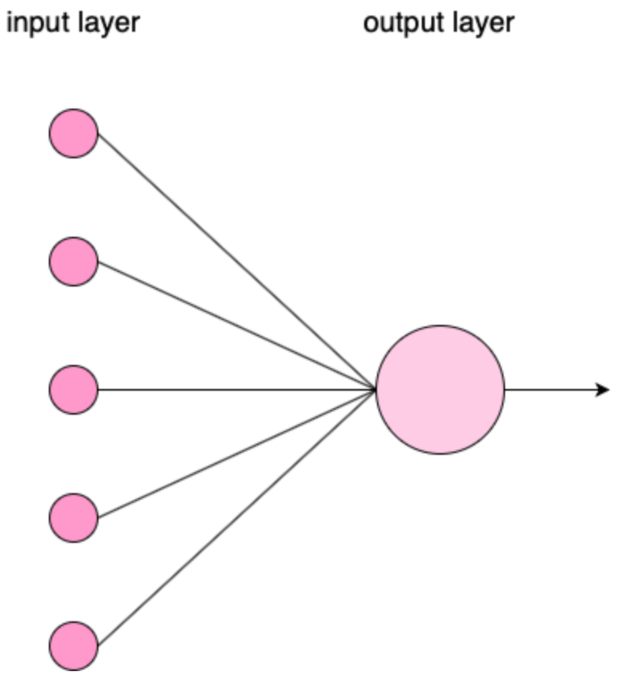

Figure 1.2: Graphic visualization of a perceptron.

three layers as shown in figure 1.3 an input layer, one hidden layer and an output layer. MLP is an example of a fully connected neural network which means that each neuron in one layer is connected to each neuron in the next layer.

Different layers in the MLP can have different activation functions. Let $\phi^{(l)}$ be the activation function of l-th layer and $h_i^{(l)}$ be the output of a i-th node in the l-th layer of the MLP. The mechanism of computing the final output can be formulated mathematically as follows:

$$h_i^{(1)} = \phi^{(1)}(\sum_j w_{ij}^{(1)} x_j + b_i^{(1)}) \tag{1.2}$$

$$h_i^{(2)} = \phi^{(2)}(\sum_j w_{ij}^{(2)} h_j^{(1)} + b_i^{(2)}) \tag{1.3}$$

$$y_i = \phi^{(3)}(\sum_j w_{ij}^{(3)} h_j^{(2)} + b_i^{(3)}), \tag{1.4}$$

where $h_i^{(l)}$ denotes the output of the l-th layer and y_i is the i-th element of the output vector.

Artificial neural networks (ANN) are usually composed of a large number of interconnected computational neurons which collectively learn to optimize the output. The mechanism is identical to the mechanism described above, the vector is loaded to the input layer and the hidden layers make decisions based on knowledge from the previous layer and pass their decisions to the next layer. Nowadays, there are several types of neural networks such as convolutional neural networks (CNN), recurrent neural networks (RNN), graph neural networks and others. Some of the mentioned neural networks will be discussed later in this thesis.

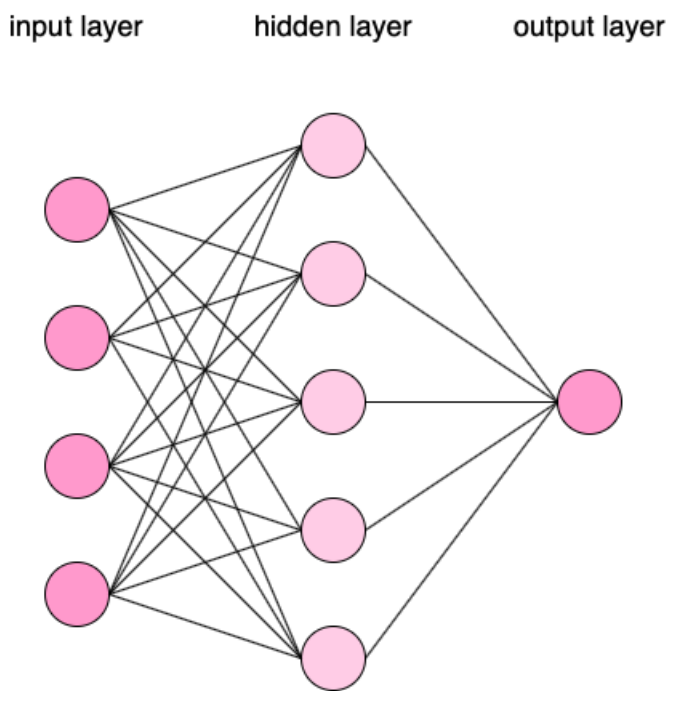

Figure 1.3: Graphic visualization of a multilayer perceptron with one hidden layer.

Activation function

The activation function has already been mentioned in the description of MLP before. The activation function is a tool used to determine whether a neuron should be activated by a given input or not. The activation function also determines the value of the output of that neuron. Equation 1.5 shows the application of the activation function f when calculating the output y. Symbol x_i denotes the i-th element of the input vector and w_i is the corresponding weight. b_i is a bias. There are two kinds of activation functions, linear and non-linear, but nowadays non-linear activation functions are usually used. There are many functions used for this purpose and the following summary includes some advantages and disadvantages of them.

$$y = f(\sum_i w_i \cdot x_i + b_i) \tag{1.5}$$

Linear activation function The linear activation function creates an output that is proportional to the input of the function. It is sometimes called also as *no-activation* or *identity* function (for $a = 1$). The mathematical notation of an identity function is the following:

$$f(x) = a * x \tag{1.6}$$

The range of this function is the whole set of real numbers, $f(x) \in \mathbb{R}$.

The biggest limitation of a linear activation function is the fact that its derivative is a constant. This means that the derivative does not have any relation to the size of an input and the gradient used to calculate the new set of weights is a constant as well.

Non-linear activation function

1. **Sigmoid**
   The formula of a sigmoid function is the following:
   $$f(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{1+e^x}. \tag{1.7}$$
   The output of a sigmoid function is always between 0 and 1.

2. **Hyperbolic Tangent**
   The hyperbolic tangent function produces an output in the range from -1 to 1. The formula of this function is:
   $$f(x) = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{1.8}$$

3. **Rectified Linear Unit (ReLU)**
   The ReLU function is equal to zero whenever the input is smaller than zero otherwise it is equal to the input.
   $$f(x) = \max(0, x) \tag{1.9}$$
   The problem with ReLU is that if many inputs into the ReLU function are negative the outputs are zeros. This is called the *Dying ReLU problem* as the network cannot perform *backpropagation*.

4. **Leaky ReLU**
   The Leaky ReLU was introduced to solve the dying ReLU problem. In this case, if input is smaller than zero, the function does not return zero but a small number very close to zero. The formula of the Leaky ReLU is the following:
   $$f(x) = \begin{cases} 0.01 \cdot x & \text{if } x < 0 \\ x & \text{else} \end{cases} \tag{1.10}$$

5. **Softmax**
   Sometimes also called the soft *argmax* function, the *softmax* function is a generalization of a *sigmoid* function that can be used for multiclass classification.
   $$f(x)_i = \frac{e_i^x}{\sum_j e^{x_j}} \tag{1.11}$$

Data

All machine learning algorithms rely heavily on data. The data and its quality often determine the quality of the final model. There are a few phases in which data is used. First, the model must be trained on some data to predict a label or learn a pattern. Then different models must be compared against each other and the best one is selected. In the final stage of the model development, the model's performance is evaluated. For those reasons the initial data set is usually divided into three subsets: *training data*, *validation data* and *testing data*.

1.1.2 Learning

The process in which a neural network updates its weights is called learning.

Loss function

The loss function is a tool used during the training of the neural network to evaluate how well a particular model fits the data. Common loss functions used in machine learning algorithms can be classified as regression losses (such as mean square error or mean absolute error) and classification losses (cross-entropy loss). Let y_i be a label and $\hat{y_i}$ prediction of a model.

Mean square error (MSE) is the average of the squared difference between the predicted value and the real value.

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y_i})^2}{n} \tag{1.12}$$

Mean absolute error (MAE) is the average absolute value of the difference between the predicted value and the real value.

$$MAE = \frac{\sum_{i=1}^{n}|y_i - \hat{y_i}|}{n} \tag{1.13}$$

Cross entropy loss function comes from information theory. Cross-entropy is a measure of the difference between two probability distributions and is based on the information $H(x)$ which can be calculated as :

$$H(x) = -\sum_{i} x_i log(x_i) \tag{1.14}$$

Cross entropy loss XE can then be calculated as:

$$XE = -\sum_{i=1}^{n}(y_i \log(\hat{y_i}) + (1 - y_i \log(1 - \hat{y_i})) \tag{1.15}$$

Optimizers

Weights of a neural network get updated during learning process as the loss function decreases and the predictions get more accurate. One can choose different optimizers in deep learning to make changes in your weights and learning rate.

Gradient descent (GD) is an optimization algorithm to find extremes of a function. GD is used in machine learning to minimize the loss function of the ML algorithm. Two requirements have to be satisfied for GD to work. The function that GD is optimizing has to be 1) convex and 2) differentiable. Let f be a loss function. Function f is convex on the set X if $\forall x_1, x_2 \in X$ and $\lambda \in < 0, 1 >$ the following equation holds true:

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2). \tag{1.16}$$

The function $f : X \to \mathbb{R}$ is differentiable in at $x_1 \in X$ if the following limit exists:

$$\lim_{h \to 0} \frac{f(x_1 + h) - f(x_1)}{h} \tag{1.17}$$

Function f differentiable at x_1 is continuous in x_1.

As the name of the GD algorithm suggests, gradient descent relies on the calculation of a gradient of a given function. Gradient of $f(\boldsymbol{x})$ is calculated as:

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}, \tag{1.18}$$

where $\boldsymbol{x} = (x_1, ..., x_n)$.

Gradient descent calculates the neural network weights using the current network weights and gradient scaled by a number, called the learning rate. The calculation is done in the following way:

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n - \alpha \cdot \nabla f(\boldsymbol{w}_n), \tag{1.19}$$

where α is a learning rate. A smaller learning rate increases the time needed to reach the optimum but a learning rate too large might not converge at all.

Stochastic gradient descent (SGD)
Computing a gradient for every point in a dataset may take too much time. This makes it more difficult to use GD on a large number of data. SGD is a stochastic approximation of a GD algorithm. SGD calculated the gradient only on a randomly selected subset of the training data which speeds up the whole training process. Let $x_1, ..., x_n \in X$ be a subset of training data. Weights are updated in the following way:

$$\boldsymbol{w}_{n+1}^S = \boldsymbol{w}_n^S - \alpha \cdot \nabla f(\hat{\boldsymbol{w}_n^S}) \tag{1.20}$$
$$= \boldsymbol{w}_n^S - \alpha \cdot \nabla \frac{1}{n} \sum_{i=1}^{n} f(x_i, \boldsymbol{w}_n^S)), \tag{1.21}$$

where $f(x_i, \boldsymbol{w}_n^S)$ is a loss function associated with weights $\boldsymbol{w}_n^S$ and observation $x_i \in X$.

Adaptive moment estimation (ADAM) is an another popular optimizer. ADAM algorithm has 4 parameters: α - the learning rate, β_1 - the exponential decay rate for the first moment, β_2 - the exponential decay rate for the second moment and ϵ - a small number to prevent division by zero. ADAM updates the weights of a neural network in the following way:

$$\boldsymbol{w}_{n+1}^A = \boldsymbol{w}_n^A - \frac{\alpha}{\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon} \cdot \hat{\boldsymbol{m}}_t, \tag{1.22}$$

where estimates $\hat{m_t}$ and $\hat{v_t}$ are computed from m_t and v_t which at the beginning of the algorithm are vectors of zeros. $\hat{m_t}$ and $\hat{v_t}$ are computed in the following way:

$$\boldsymbol{g}_t = \nabla f(\boldsymbol{w}_n^A) \tag{1.23}$$
$$\boldsymbol{m}_t = \beta_1 \cdot \boldsymbol{m}_{t-1} + (1 - \beta_1) \cdot \boldsymbol{g}_t \tag{1.24}$$
$$\boldsymbol{v}_t = \beta_2 \cdot \boldsymbol{v}_{t-1} + (1 - \beta_2) \cdot \boldsymbol{g}_t^2 \tag{1.25}$$
$$\hat{\boldsymbol{m}}_t = \frac{\boldsymbol{m}_t}{1 - \beta_1^t} \tag{1.26}$$
$$\hat{\boldsymbol{v}}_t = \frac{\boldsymbol{v}_t}{1 - \beta_2^t} \tag{1.27}$$

Over-fitting

The goal of machine learning models and neural networks is to perform well on training data as well as new data. Some models fail to generalize and do not perform well on new data due to *over-fitting*. This happens when a model learns the training data too well but does not perform accurately on the evaluation set. The art of training a model is creating a model complex enough to fit data well a simple enough to generalize on new data. This is called the bias-variance trade-off.

There are techniques that can be used during the training of a machine learning algorithm to avoid over-fitting. A very straightforward solution is to train on more data. Very often this is not possible and one of the following techniques must be implemented to decrease the chance of over-fitting:

K-fold cross-validation During K-fold cross-validation the initial data set is split into k subsets. One subset is used for validation and the remaining $k-1$ subsets are used for training. The training and testing are then repeated k-times using different data each time. The final error is then calculated as a mean of k errors.

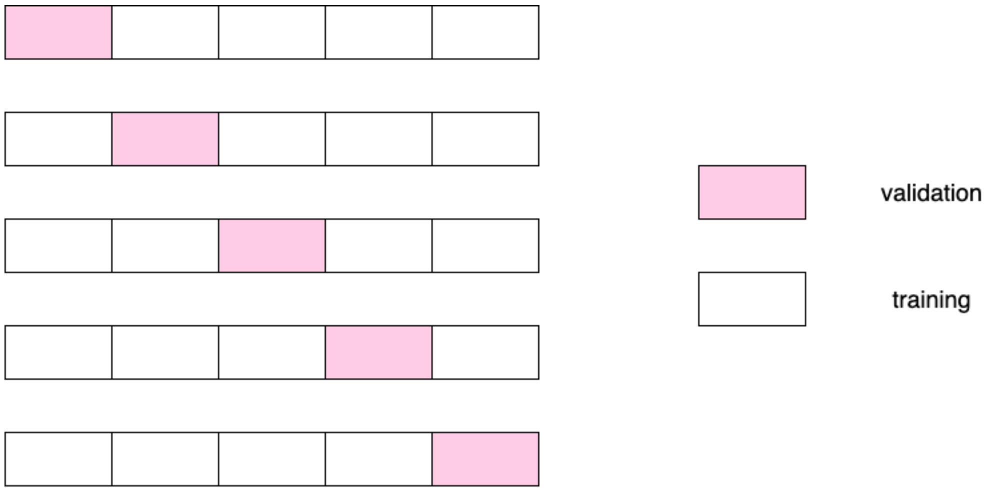

Figure 1.4: Graphic visualization of k-fold cross-validation (k=5).

Adding noise to input data Sometimes, in order to describe the input space well, one might need a lot of training data. Big datasets might not always be possible. When the space is not represented well, the model cannot learn how to generalize and does not perform well on new data. Adding noise to input data often leads to better generalization abilities of the model and better tolerance to mistakes [12].

Feature selection A more complex model does not guarantee better performance. Complex models tend to be more prone to over-fitting. Reducing the number of input data is called feature selection. Not only can feature selection improve the performance of the model but it also speeds up the training process.

Early stopping This technique stops the model before it over-fits the training data. The risk of stopping the learning process is that the maximum performance will not be achieved. The goal of early stopping is to stop training the model when the performance of the model on validation dataset starts to drop.

Adding dropout layers Another regularization technique which prevents over-fitting is adding dropout to your model. During training, random connections between nodes or alternatively whole nodes can be deactivated. This happens in each training epoch which prevents the model from over-fitting the data.
Let $w_1, ..., w_n$ be the weights of a network. Then mathematical notation of adding a dropout is equal to:

$$w_i = \begin{cases} 0, & \text{with probability } p \\ 1, & \text{with probability } 1-p \end{cases} \tag{1.28}$$

A good way of representing a dropout visually is shown on 1.5. 1.5 shows a simple fully connected neural network with one hidden layer. After dropout, some connections were removed from the second neural network on 1.5. Removing connections represents the weights that were set to zero with probability p.

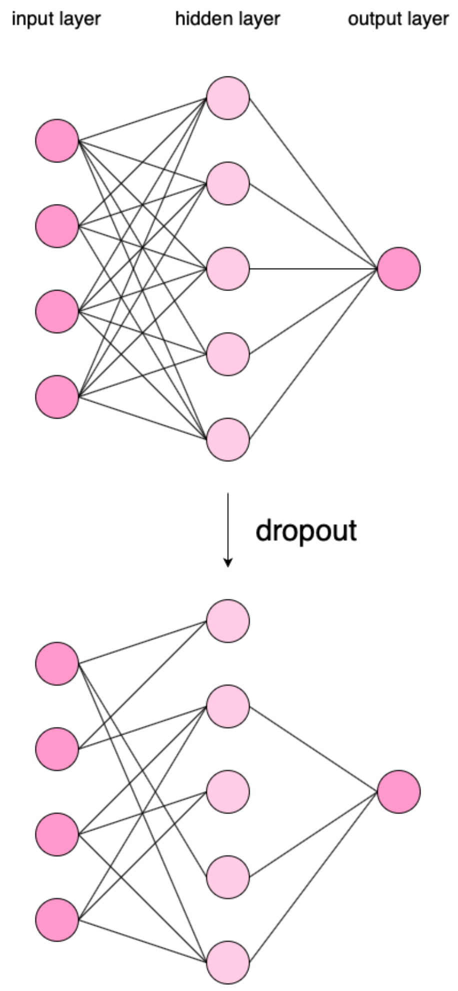

Figure 1.5: Dropout - deactivating random connections in the neural network.

1.1.3 Performance metrics

To choose the best model for each task one must measure its performance. There are a few performance metrics to evaluate a model. The following section summarizes the most common performance metrics.

To simplify the notation, let the classifier K be a classifier with two labels. The classification made by a classifier can be either a True positive (TP), False positive (FP), True negative (TN) or False negative (FN). Figure 1.6 explains the meaning of TP, FP, TN and FN terms.

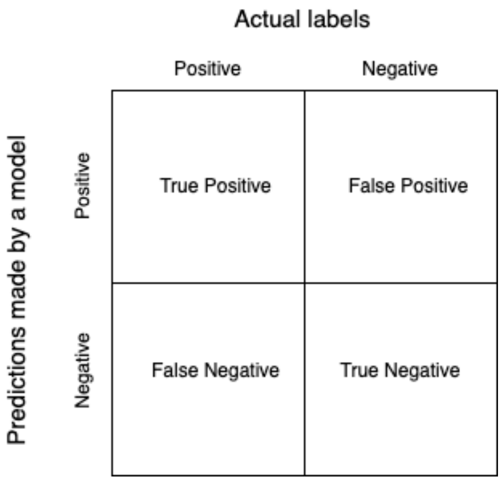

Figure 1.6: Table explaining the meaning of different classification terms.

Accuracy

For binary classification accuracy can be calculated as:

$$A := Accuracy = \frac{TP + TN}{TP + FP + TN + FN}. \tag{1.29}$$

Generally, accuracy is calculated as a ratio of correctly classified inputs to all classifications:

$$A = \frac{CP}{AP} = \frac{\sum_i CP_i^i}{\sum_i CP_i^i + \sum_{i,j} IP_j^i}. \tag{1.30}$$

CP denotes all correct predictions and CP_i^i are all correct predictions of class i. AP denotes all predictions made by the classifier. IP is the number of incorrect predictions and IP_j^i is the number of incorrect predictions of class j predicted as class i. The same notation and indexation are used in the rest of this section.

Accuracy does not take into account the dataset itself and is useful only when there is an equal distribution of classes on the classification. If the data set is imbalanced (meaning, it contains more examples of a certain label) the final accuracy does not describe the true performance of the model well. Let us consider a binary classification problem with 100 data points in the testing dataset with only positive labels. A classifier which predicts a positive class for all the input data will have a high accuracy. Although accuracy is high, the truth is that the model does not generalize well and does not make decisions in an intelligent manner.

For this reason there are other performance metrics to consider when training a model.

Precision

Precision is calculated as follows:

$$P := Precision = \frac{TP}{TP + FP}. \tag{1.31}$$

In the case of a model with $FP = 0$, which means that the model does not classify any negative objects as positive, the precision of that model is 1. In a multi-class classification precision of a class i is calculated as:

$$P_i = \frac{CP_i^i}{CP_i^i + \sum_j IP_j^i} \tag{1.32}$$

Recall

Recall is calculated as follows:

$$R := Recall = \frac{TP}{TP + FN} \tag{1.33}$$

In the case of a model with $FN = 0$, which means that model does not classify any negative objects as positive, the recall of that model is 1. In a multi-class classification recall of a class i is calculated as:

$$R_i = \frac{CP_i^i}{CP_i^i + \sum_j IP_i^j} \tag{1.34}$$

F1 score

Both precision and recall have their drawbacks. Therefore a combination of the two previous metrics called the F1 score is used. The F1 score is calculated as follows:

$$F1 = 2 * \frac{P \cdot R}{P + R}. \tag{1.35}$$

The goal of the F1 score is to create a better metric which would work well on imbalanced data sets. Again, a higher F1 score of a model indicates better performance. Medium F1 score is obtained if one of the metrics used is low.
For a multi-class classifier, the F1 score for class i is equal to:

$$F1_i = 2 \cdot \frac{P_i \cdot R_i}{P_i + R_i}. \tag{1.36}$$

1.2 Natural language processing

Language modelling is described by [1] as a process of learning the distribution over a set of tokens taken from a vocabulary. Let the sequence of tokens be noted as

$(x_1, ..., x_n)$. The goal is to learn the probability $P(x_1, ..., x_n)$. The join distribution is usually computed as:

$$P(x) = \prod_t P(x_t | x_{<t}). \tag{1.37}$$

$x_{<t}$ are $\forall x_i \in (x_1, ..., x_n)$, where $i < t$.

1.2.1 Tokenization

Tokenization is an important aspect of working with language data. In order for a model to understand the human language it needs to be pre-processed first. This is when tokenization comes into the picture. Tokenization is a way of dividing the text model is working with into smaller units such as sentences, words or even characters. These units are called tokens. Tokens are then formed into a set of tokens called a vocabulary.
Word tokenization is commonly used during data processing. For example, tokenization can be performed on letter, word or sentence level. The biggest drawback of tokenization on the word level is that the model can encounter words that are not in the primary vocabulary. These words are called the Out of Vocabulary (OOV) words. There are methods to escape the OOV words, but usually, the information the word carries gets lost in the process. Tokenization on the character level overcomes the OOV problem. Additionally, the vocabulary on the character level is significantly smaller due to the limited number of characters in a language. The drawback of this approach is that the length of input is considerably larger than in word tokenization.
State-of-the-art (SOTA) models in NLP rely on a tokenization method that solved all of the issues mentioned above. Tokenization on a sub-word level splits words into multiple parts (sub-words) and in that way tackles both the OOV problem of the word tokenization as well as the input length issue of the character tokenization.

1.2.2 Word embedding

Word embedding requires no labelled data and is considered to be one of the most successful applications of unsupervised machine learning. Word embedding can map large dimensional data into lower dimensions while maintaining semantic relationships between words. These techniques represent words as vectors in a pre-defined vector space. Words with similar meanings have similar representations in a given vector space and in this way, a word meaning is captured.

Word2Vec

One of the most iconic and major works in this field is Word2Vec [9]. Word2Vec algorithm results in a vector space where words with similar semantic meanings are close to each other. One of the findings of word embedding is that one can perform mathematical operations with the vectors from its vector space. An example of a mathematical operation with vectors representing words in vector space is in the 1.8. In 1.8 words represent vectors in the vector space. When the value of a vector representing the word *men* is subtracted from the word *king* and a value of the word *woman* is added, the resulting word vector represents (or is very close to) the word *queen*.

Team of scientist lead by Tomas Mikolov at Google proposed two model architectures for learning distributed representations of words that try to minimize computational complexity. These two architectures are presented on figure 1.7. *CBOW* uses context words to predict the value of a given word. On the other hand, *Skip-gram* uses a certain word to predict its context. Word2Vec is a shallow, two-layer neural networks. It is trained to reconstruct contexts of words as explained on figure 1.7. A by product of learning the context of words are the hidden weights of the model. These weights are then used as word embeddings.

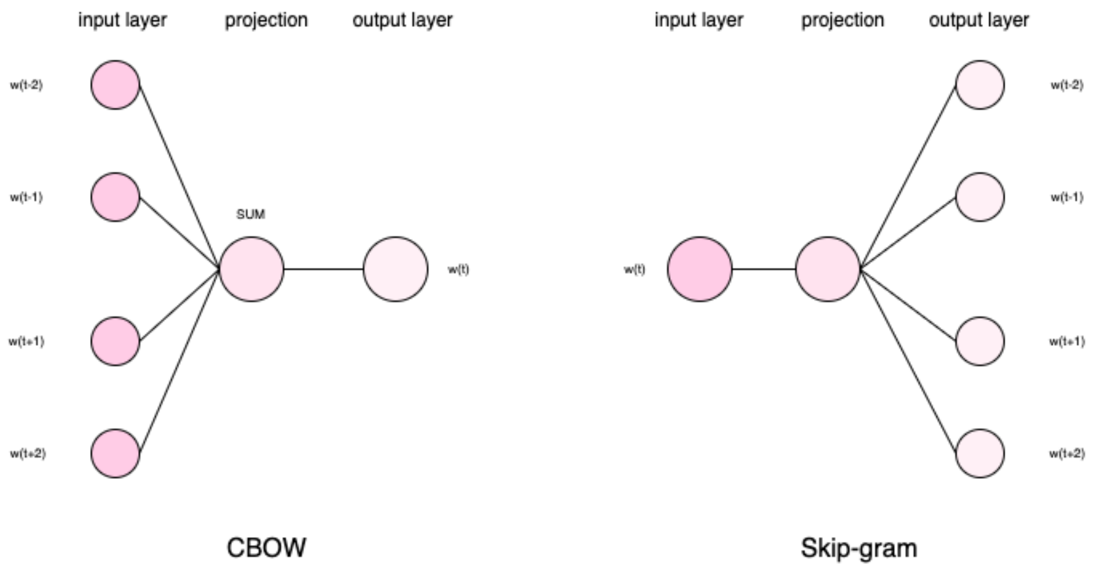

Figure 1.7: CBOW architecture predicts a word based on the surrounding words, Skip-gram uses a given word to predict the context words.

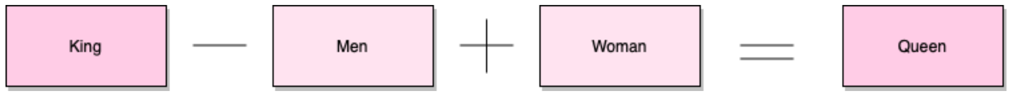

Figure 1.8: Example of a relationship between a vector in the vector space.

Following subsections present a few model architectures used in natural language processing such as recurrent neural networks (section 1.2.3), convolutional neural networks (section 1.2.4) and a short comment on transfer learning (section 1.2.5).

1.2.3 Recurrent neural networks

Recurrent neural network (RNN) was created for processing of sequential data. RNN's architecture is derived from feedforward neural networks with addition of a concept of memory. Ordinary neural networks were create to process independent data but sequential data such as text or speech rely heavily on previous data. Figure 1.9 visually shows how the concept of memory is represented in RNNs. At each step RNN's output is used as an input in the next step.

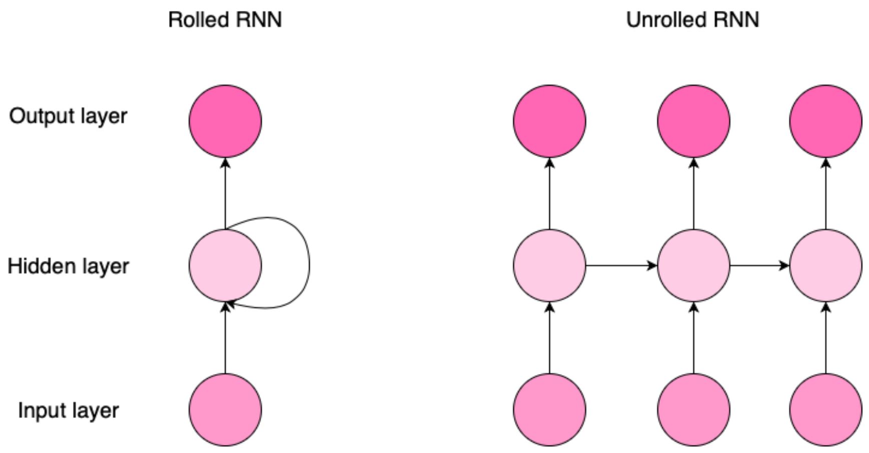

Figure 1.9: Recurrent neural network

Like many neural networks, RNNs are optimized using back-propagation [16]. During back-propagation through the recurrent layers, errors decay very fast, this is called the *vanishing gradient problem* [6]. The two ideas were introduced to fight the vanishing gradient problem. Replacing the sigmoid activation function with ReLU is one of them [5]. The other way how to combat the vanishing gradient is to use a special kind of recurrent neural network called a Long Short-Therm memory network [4].

Long short-term memory

Long short-lerm memory (LSTM) neural networks are used to process sequential data such as text, time series or video data. The biggest disadvantage of the LSTM is a number of parameters that need to be trained thus slowing the whole process. For that reason, the Gated recurrent unit was developed as a faster version of the LSTM neural network [1].

Gated recurrent unit

Even though GRU's architecture is more simple it can outperform the LSTM on some tasks [1].

1.2.4 Convolutional neural networks

RNNs suffer from being too slow due to the fact that they process data sequentially [1]. A simple ANN, when given a large number of inputs may easily over-fit on the training data. Convolutional neural networks were invented to combat problems or slow training and over-fitting. Traditionally convolutional neural networks (CNN) were applied to image processing. The main component of CNN are convolutional and pooling layers. The overall architecture of CNNs is the following:

1. Input layer - numerical data
2. Convolutional layers - extract features
3. Pooling layer - used to reduce the dimension
4. Fully-connected layers - learn to solve the task

The convolutional layers are usually stacked followed by pooling layers. Another common architecture is repeating two convolutional layers followed by a pooling layer several times [11]. Convolutional layers are used for extracting features from the feature space and in NLP are used to extract features from word embeddings [1]. Stacking multiple convolutional layers extracts more complex features. These layers' parameters focus on the use of learnable kernels [11].
The pooling layers help to reduce the size of the feature space as pooling is equivalent to dimension reduction. An example of pooling is shown on 1.10, this pooling is done using the *max* function and therefore is called max-pooling. For example, a max-pooling performed with a kernel of 2x2 size applied with stride 2 will reduce the activation map to 25% of its original size. Stride is the number of pixels the kernel shifts each time. In the case of a 2D filter the size of the output layer can be calculated using the following formula:

$$\frac{N-F}{S+1}, \tag{1.38}$$

where N is the size of the image, F is the size of the filter and S is the stride. In figure 1.10 the parameters are the following: $N = 16,\ F = 4,\ S = 2$ therefore the size of the output is $\frac{16-4}{2+1} = \frac{12}{3} = 4$.

In image processing the input into a CNN are pixels, in NLP the convolutional neural network is fed with the matrices made from vectors representing each word.

1.2.5 Transfer learning

Nowadays, the machine learning and deep learning community have to tackle two problems: data availability and computing resources. Collecting large datasets is time-consuming and very expensive. For some domains collecting datasets large enough to train complex neural networks is not possible. Furthermore, the time and financial costs as well as the environmental costs of training big complex models

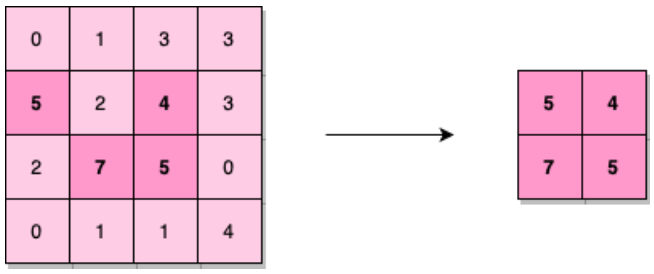

Figure 1.10: An example of pooling - max-pooling.

are massive. Both of these problems can be reduced using transfer learning. The goal of transfer learning is to keep the knowledge gained on one task and apply this knowledge to a different task. In practice, transfer learning is widely used in Natural language processing (NLP) and computer vision (CV). In both NLP and CV, models trained on very large data sets learn specific patterns and then this knowledge is applied to a new task. Transfer learning relaxes the condition that the training and testing data must be independent and identically distributed [14].

Following the same notation as [1] transfer learning is defined as follows. Let D be the domain and X be the feature space. Domain D can be defined by a tuple $(X, P(X))$, where $P(X)$ is a marginal probability of the feature space X. Let y be the labels and $P(y|x)$ a conditional distribution that the model is trying to learn. The task model is trying to solve can be described by a tuple $(y, P(y|x))$. During transfer learning two different domain-task tuples can be distinguished, source (D_s, T_s) and target (D_t, T_t) domain tuples. Using a source domain and task during the learning process for the target task is called transfer learning.

Chapter 2

Analysis

2.1 Common Crawl

Common Crawl (CC) is an open-source dataset accessible online to anyone. Data can be downloaded using Amazon S3 or HTTPS requests. The CC dataset contains petabytes of high-quality data collected on the internet from 2008 until today. Raw data is stored using the Web ARChive (WARC) format. Plain text data extracted from WARC are stored in WET files. WAT files store metadata. Only WET files were downloaded and analysed for the purposes of this thesis.

2.2 Data and processing

The CC datasets contain petabytes of data. Only a subset of data was processed in the analysis part of this thesis. Each subset of data was around 10% of the original dataset. The analysis was done using resources available by Google. The size of the dataset was too large to process using free resources and additional resources were purchased. The final storage available for the analysis was 200GB and 32GB of RAM. The main language analyzed was English. Additionally, data in Spanish were used for model and method validation.

2.3 Data analysis

A random sample of the Common Crawl data from each month has been downloaded using HTTPS requests. Only textual data was analyzed, therefore data such as code or images were not considered. First, the data was cleaned by removing characters that do not belong to the English language. By doing so, languages such as Russian, Chinese, Korean, Arabic and many others were filtered out. In the next step sentences that were too long (longer than 90 words) or too short (shorter than 10 words) were filtered out as well, as they do not carry much useful information that can be processed in the next steps. A lot of sentences were duplicated in the dataset due to the nature of the data on the internet, so an important step in data preparation was the deduplication of sentences.

Two languages were filtered out of the prepared data, English and Spanish. The English language was used in the main part of the data analysis to capture and measure the evolution of the meaning of words in time. The Spanish language was used in to validate the method presented in this thesis and also show that the implementation of this approach was correct. Both languages were filtered out using a sets of ten thousand most common words extracted from English and Spanish data found on the internet. Although using this approach might omit some English or Spanish sentences, this approach was chosen as it is quicker than using libraries available in Python.

Graphic visualization of the steps of English language data analysis is shown in figure 2.1 below.

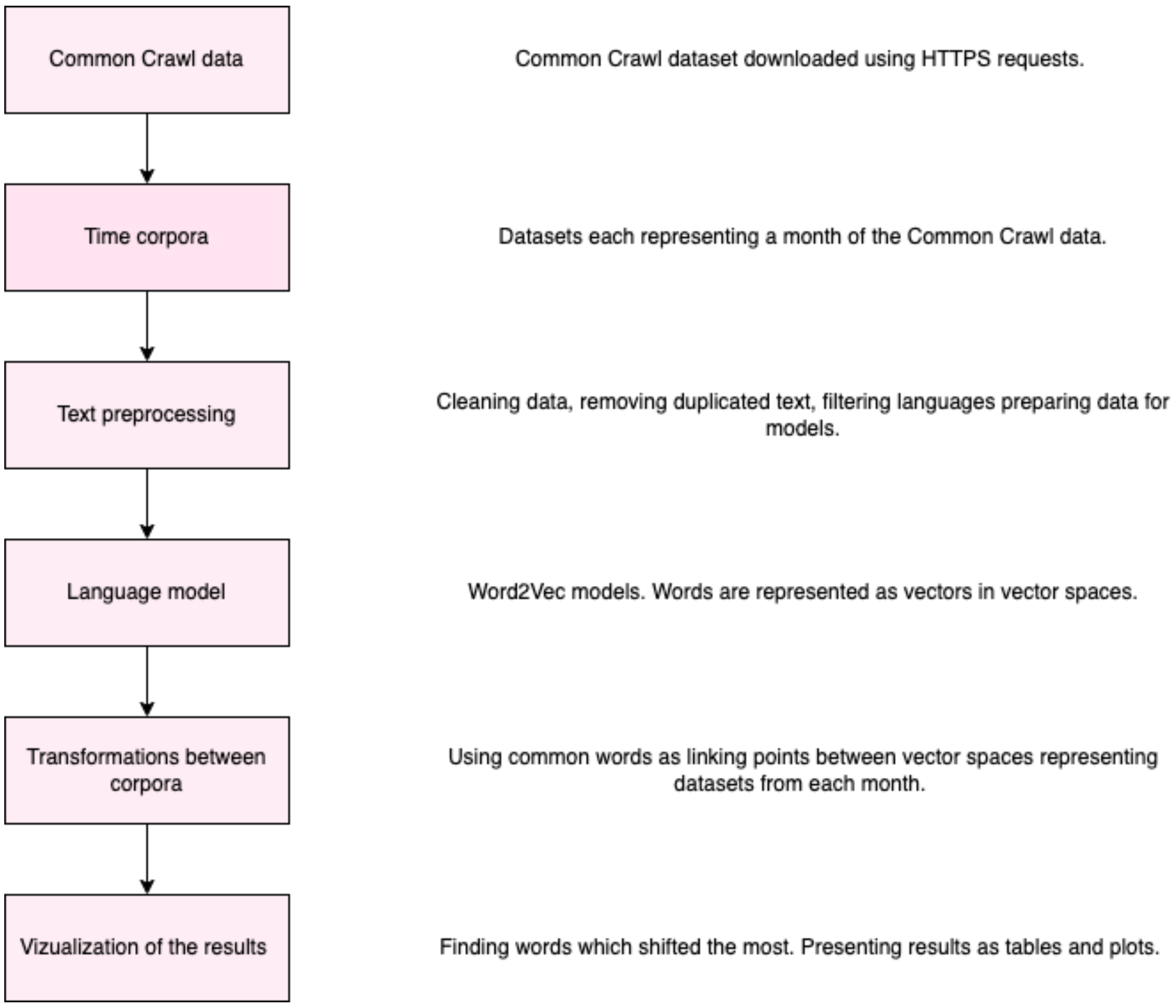
Common Crawl data
Common Crawl dataset downloaded using HTTPS requests.
Time corpora
Datasets each representing a month of the Common Crawl data.
Text preprocessing
Cleaning data, removing duplicated text, filtering languages preparing data for models.
Language model
Word2Vec models. Words are represented as vectors in vector spaces.
Transformations between corpora
Using common words as linking points between vector spaces representing datasets from each month.
Vizualization of the results
Finding words which shifted the most. Presenting results as tables and plots.

Figure 2.1: A simple visualization of data processing and analysis (English language).

2.3.1 Time clusters

Common Crawl downloads data from the internet regularly. Most often datasets are created monthly or bimonthly. The number of pages changes over time and does not necessarily increase. Many web pages get deleted each month and new websites are added. Each dataset contains new web pages crawled for the first time as well as older pages crawled in the previous months or years. In the analysis, each crawl was taken as a separate time cluster. This is the best representation of data available on the web in any given month. Although web pages and data may be present in multiple crawls and therefore multiple time clusters, the portion of web pages added each month is substantially large and will influence the monthly dataset significantly. In this way, it is possible to capture the evolution of the meaning of words over time.

The evolution of WET file sizes is shown in figure 2.2, figure 2.3 shows the number of new URLs scraped each crawl, and figure 2.4 shows the total number of pages in each dataset.

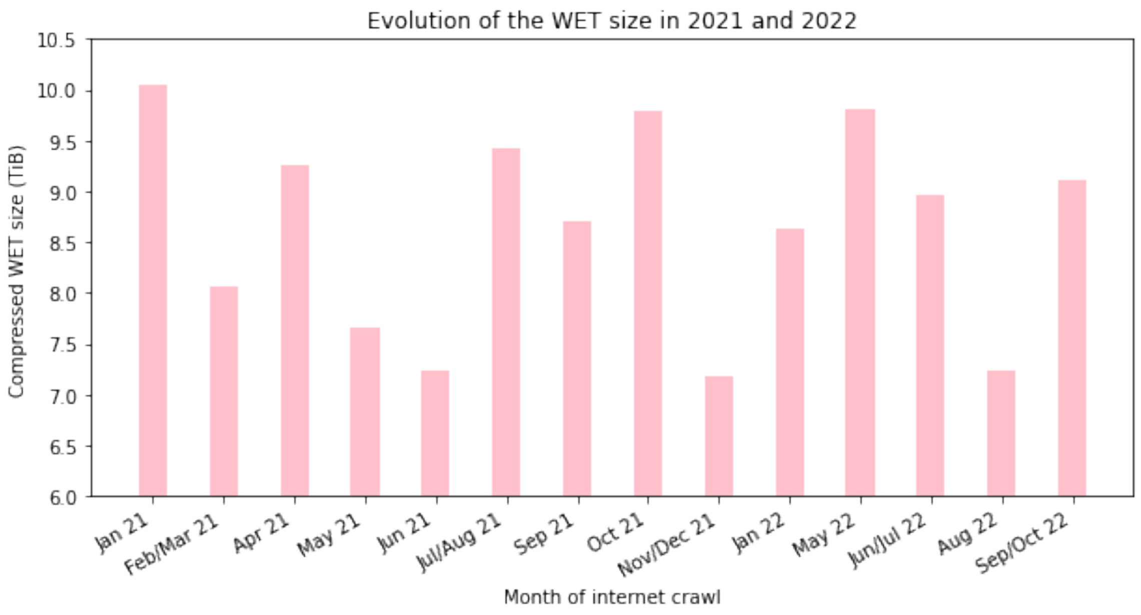

Figure 2.2: Size of compressed data in 2021 and 2022.

Table 2.1 presents information about each dataset and presents a notation that is used in further analysis. For example, the symbol EN_1 denotes the first dataset of the English language crawled in January 2021. For simplicity symbol, EN_i denotes the English dataset of the i-th crawl as well as the model built on this dataset. The symbol ES denotes the dataset containing only the Spanish language. In order to achieve this amount of Spanish data, more than 10% of the January 2021 crawl was processed.

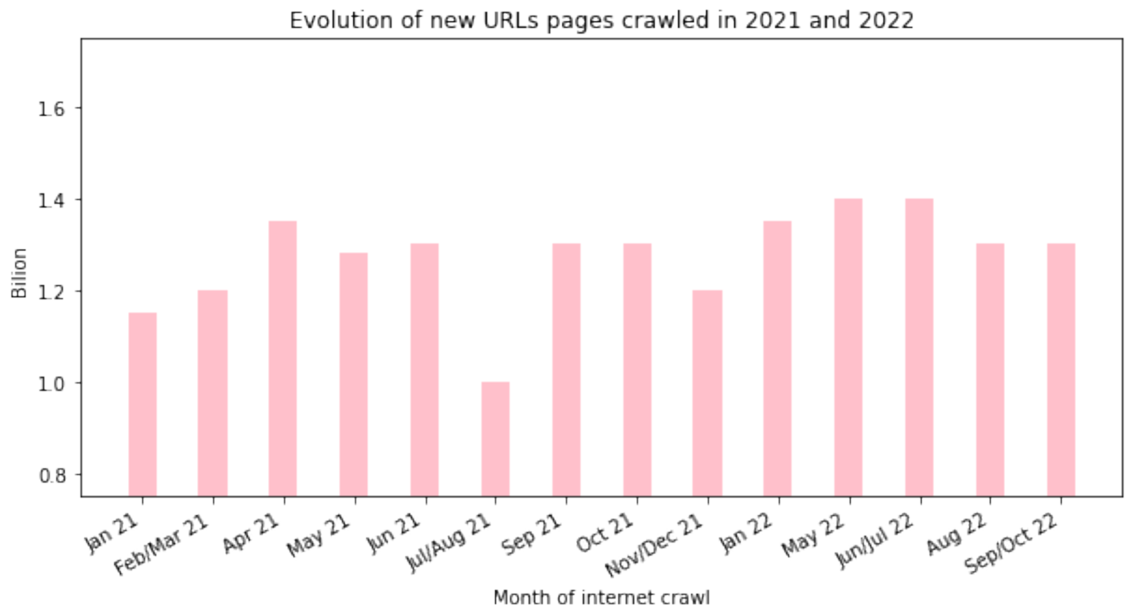

Figure 2.3: New ULRs scraped in each crawl in 2021 and 2022.

Table 2.1: 14 English datasets and one Spanish dataset used to train models.

model	**date**	**sentences**	**words**
EN_1	January 2021	14 079 776	29 065 500
EN_2	February/March 2021	13 459 793	28 796 000
EN_3	April 2021	14 732 509	30 150 500
EN_4	May 2021	12 569 176	27 803 500
EN_5	June 2021	11 440 936	26 556 000
EN_6	July/August 2021	12 098 726	26 590 500
EN_7	September 2021	11 635 755	26 296 500
EN_8	October 2021	13 898 464	28 869 500
EN_9	November/December 2021	11 213 365	26 041 000
EN_{10}	January 2022	12 323 643	27 160 500
EN_{11}	May 2022	11 699 690	26 117 500
EN_{12}	June/July 2022	10 855 253	25 116 500
EN_{13}	August 2022	9 046 566	23 082 500
EN_{14}	September/October 2022	11 968 787	16 165 350
ES	January 2021	9 760 899	27 833 000

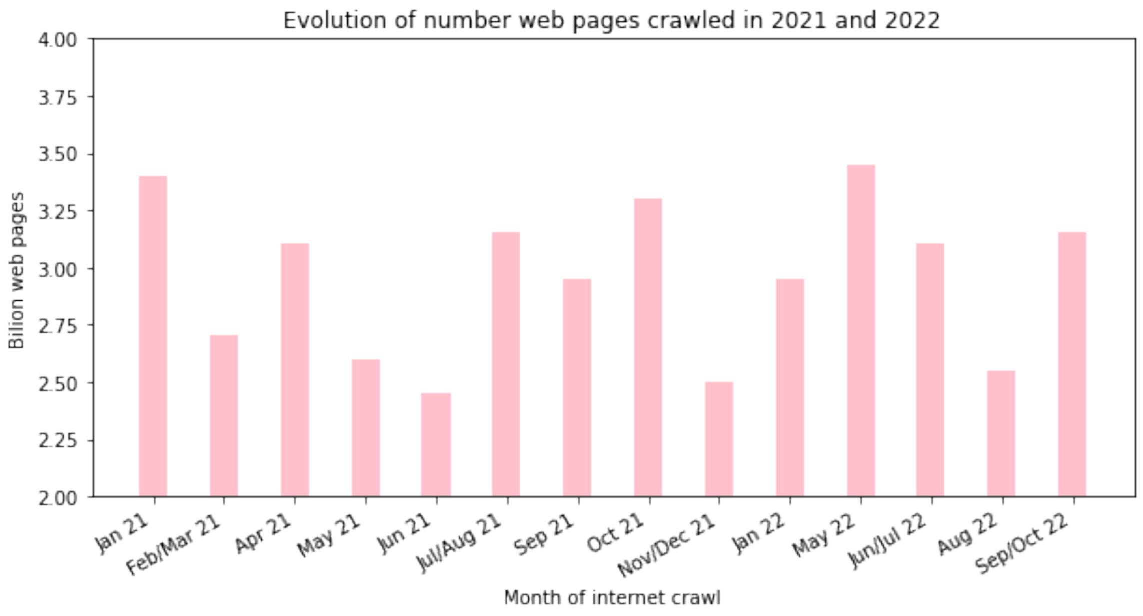

Figure 2.4: Number of pages in crawls in 2021 and 2022.

2.3.2 Models

Word2Vec model was used for the analysis of the data. Models and their different settings were validated on word analogies and their accuracy is presented in the tables below. These models were trained on dataset EN_1. Word2Vec models were implemented using *Gensim* (Python) library and have many parameters but only some were fine-tuned as they have shown to have a noticeable effect on the accuracy score on word analogies. Other parameters were set to their default value. Following parameters were fine-tuned: vector dimension (*dim*), minimal word occurrence (*min_count*) and a number of training *epochs*. For all models the *window* was set to 10 as increasing this parameter made little difference in the final accuracy and increased the training time. Tables 2.2, 2.3 and 2.4 present the accuracy of 50, 100 and 200-dimensional models with different parameters. For all 50, 100 and 200-dimensional models higher *min_count* and more training epochs achieved better accuracy. Also, the higher the vector dimension, the better the model performs on word analogies. A summary of the best model from each vector dimension size is shown in the table 2.5. For further analysis and word mappings, models with the highest performance on word analogies were chosen. Same dataset was used to validate these models as it was in [9].

2.3.3 Language translation

Mikolov et al. [10] found that word representations learned in different languages can be transformed, so the word vectors overlap or are very close to each other. This method can be used to develop large dictionaries between two languages using little

Table 2.2: Accuracy of models with 50-dimensional vectors.

dim	**window**	**min_count**	**epochs**	**accuracy**
50	10	5	5	35%
50	10	5	10	36%
50	10	5	20	37%
50	10	10	5	35%
50	10	10	10	36%
50	10	10	20	37%
50	10	100	5	40%
50	10	100	10	40%
50	10	100	20	41%

Table 2.3: Accuracy of models with 100-dimensional vectors.

dim	**window**	**min_count**	**epochs**	**accuracy**
100	10	5	5	48%
100	10	5	10	49%
100	10	5	20	50%
100	10	10	5	47%
100	10	10	10	50%
100	10	10	20	50%
100	10	100	5	53%
100	10	100	10	54%
100	10	100	20	55%

assumptions about the actual languages.

This method consists of two steps:

1. build two monolingual models on large datasets
2. learn linear projection between the two vector spaces using a smaller dictionary

Let X be a set of vectors representing data available from one language and Y represent a second language. Let $x_i, y_{i}{}_{i=1}^{n}$ be a dictionary of n translations where $x_i \in X$ and $y_i \in Y$. The linear mapping between the two languages (represented by

Table 2.4: Accuracy of models with 200-dimensional vectors.

dim	**window**	**min_count**	**epochs**	**accuracy**
200	10	5	5	58%
200	10	5	10	60%
200	10	5	20	60%
200	10	10	5	58%
200	10	10	10	60%
200	10	10	20	61%
200	10	100	5	62%
200	10	100	10	64%
200	10	100	20	65%

Table 2.5: Models with the best accuracy score among 50, 100 and 200-dimensional vectors.

dim	**Window**	**min_count**	**epochs**	**accuracy**
50	10	100	20	41%
100	10	100	20	55%
200	10	100	20	65%

vector spaces) is learned by finding a transformation matrix W, such as Xx_i is y_i. In practice, one tries to find the best approximation. W can be learned by optimizing the following:

$$\min_W \sum_{i=1}^{n} ||Wx_i - y_i||^2. \tag{2.1}$$

In the implementation of this method, optimization was done using gradient descent.

In the experiments W was initialized from Gaussian distribution with mean 0 and variable 0.1, $\forall i, j, W_{ij} \sim N(0, 0.1)$. Each element of the matrix W was updated using the gradient descent method. Let Wx be the prediction of the vector y. The difference between predicted and true value z can be calculated for one word as:

$$z = Wx - y. \tag{2.2}$$

The value of W_{ij} element of the matrix W at step $n+1$ is updated using the gradient descent described in equation (1.19):

$$W_{ij}^{n+1} = W_{ij}^{n} - \alpha \cdot z_i \cdot x_j, \tag{2.3}$$

where W_{ij}^n is the W_{ij} element at step n, z_i is the i-th element of vector z, x_j is the j-th element of vector x and α is the learning rate.

A dataset of 3 thousand most common Spanish words were extracted and translated using Google Translate. First 2000 words were used to learn the matrix W, the remaining were used to validate how well this implementation preforms. Results are discussed further in this section.

It is worth noting that these resulting models can suffer from the fact that some word vectors tend to be the nearest neighbours of the abnormally high number of other words [7]. This problem is called the *hubness problem.* In this case translations can be inaccurate.

English to Spanish translation results are summarised in table 2.6. Higher dimensional vector spaces perform better than lower ones. Symbol A@1 denotes correct translations within the closest word to the expected position and A@1 correct translations within the five closest words of the expected position.

Table 2.6: Accuracy of English to Spanish translation of models with different vector dimensions.

model	**A@1**	**A@5**
$EN_{50} \rightarrow ES_{50}$	23%	40%
$EN_{100} \rightarrow ES_{100}$	30%	51%
$EN_{200} \rightarrow ES_{200}$	35%	58%

Spanish to English translation results are summarised in table 2.7. Higher dimensional vector spaces perform better than lower ones. Noticeably better accuracy was achieved in Spanish to English translation.

Table 2.7: Accuracy of Spanish to English translation of models with different vector dimensions.

model	**A@1**	**A@5**
$ES_{50} \rightarrow EN_{50}$	30%	50%
$ES_{100} \rightarrow EN_{100}$	39%	56%
$ES_{200} \rightarrow EN_{200}$	46%	66%

Experiments were run only for two vector spaces of the same dimensions. Additionally, experiments of translating between different dimensions could be done as they were done in [10]. Better accuracy in language translation would not benefit the goal of this thesis and therefore they were not done.

2.3.4 Word mappings

The same idea of language translation when applied to multiple datasets of the same language can be used to measure the distance between the expected and actual position of a word. Models can assign vectors to words and those vectors corresponding to the same word can be compared.

The following example between two vector spaces, EN_1 and EN_2 can be applied to others as well. Let $x_i \in EN_2$ and $y_i \in EN_1$. The transformation matrix can be found by optimizing the following:

$$\min_{W_{1,2}} \sum_{i=1}^{n} ||W_{1,2}x_i - y_i||^2. \qquad (2.4)$$

This optimization was done using gradient descent. In this case, 10 thousand most common words in our datasets were extracted and 3 thousand were used to learn the best matrix $W_{k,l}$. The remaining words were used to validate the approach.

The hypothesis is that some words, which appear in all monthly datasets $EN_i, \forall i \in 1, ..., 14$ can be represented by slightly different vectors in each model corresponding to each dataset. The difference between vectors of the same words in time can be one of the ways to represent the change in the meaning of these words.

To validate the method of mapping vectors to one vector space using transformation matrices accuracy was calculated in the same way as in section 2.3.3. Table 2.8 summarizes the results of the mapping of vector spaces. Symbol A@1 denotes again the accuracy within the first closest word of the expected position of each word and A@5 in accuracy within the closest five words. All results perform very well scoring more than 98% on A@1 each time and more than 99% on A@5.

Using those 13 learnt mappings between $EN_i, i \in 2, ..., 14$ and EN_1, all words can be mapped into one vector space and their vectors can be compared.

Table 2.8: Accuracy of monolingual mapping between vector spaces created on different datasets.

model	**A@1**	**A@5**
$EN_1 \rightarrow EN_2$	99.186 %	99.857 %
$EN_1 \rightarrow EN_3$	99.143 %	99.143 %
$EN_1 \rightarrow EN_4$	99.086 %	99.771 %
$EN_1 \rightarrow EN_5$	98.957 %	99.7 %
$EN_1 \rightarrow EN_6$	98.942 %	99.728 %
$EN_1 \rightarrow EN_7$	98.557 %	99.614 %
$EN_1 \rightarrow EN_8$	98.971%	99.742 %
$EN_1 \rightarrow EN_9$	98.686%	99.571 %
$EN_1 \rightarrow EN_{10}$	98.771%	99.571 %
$EN_1 \rightarrow EN_{11}$	98.757%	99.614 %
$EN_1 \rightarrow EN_{12}$	98.629 %	99.5 %
$EN_1 \rightarrow EN_{13}$	98.343 %	99.457 %
$EN_1 \rightarrow EN_{14}$	98.614%	99.7 %

Chapter 3

Results

In chapter 2 the mappings between two Word2Vec models were trained on the 3000 most common words in the whole portion of the Common Crawl dataset. This chapter presents results obtained analysing the following 7000 most common words. Learnt mapping between each two vector spaces was used to predict the expected vector of a certain word and its actual position in the vector space. To measure the distance between the two (expected and actual) positions of a vector, two metrics were used.

Euclidean distance is a way of measuring the change between the expected position of a vector in one vector space and its actual position. Euclidean distance of two n-dimensional vectors u and v can be calculated in the following way:

$$d(u,v) = \sqrt{\sum_n (u_i - v_i)^2}. \tag{3.1}$$

Let w be a word that is being analyzed. The vector representation of the word w in model M_1 is $v_1 \in V_1$ and this word is represented by $v_2 \in V_2$ in model M_2. The transformation matrix learned on a small dictionary is $W_{1,2}$. Then the expected position of word w in the vector space V_1 using the value of vector v_2 is:

$$\tilde{v_1} = W_{1,2} v_2. \tag{3.2}$$

The euclidean distance between vectors $\tilde{v_1}$ and v_1 is calculated using equation 3.1 as:

$$d(v_1, \tilde{v_1}) = \sqrt{\sum_n (v_{1i} - \tilde{v_1}_i)^2}. \tag{3.3}$$

Using a euclidean distance to measure the shift between two vectors of the same word would be a good metric for normalized vectors, but vectors of the Word2Vec models are not normalized.

Cosine distance is a metric that can be used to measure the distance between two vectors without needing to normalize them first. The cosine distance between two vectors u and v, is defined as follows:

$$d_{cd}(u, v) = 1 - \frac{u \cdot v}{\|u\|_2 \|v\|_2}. \tag{3.4}$$

Both euclidean distance and cosine distance were used to measure the change in the positions of different words. The distance between vectors in each vector space was determined and the sum of distances were calculated for each word. Words with the biggest and the smallest change calculated using euclidean distance are in table 3.1. Table 3.4 shows words with the biggest and the smallest change over all datasets calculated using cosine distance. These two tables are very similar when it comes to the words with the biggest change. On the other hand, euclidean and cosine distances have determined different sets of words with the smallest change.

Nine of the words with the biggest change presented in tables 3.1 and 3.4 were selected and their frequencies in each model are summarized in table 3.7 and 3.8. These words are keywords to some world events that happened in the years 2021 and 2022. These words include words linked to the coronavirus pandemic such as *delta* or *mask* and words linked to political events such as *hong*, *kong*, or *scotland*. Possibly, the word *harry* could be linked to the events of Prince Harry and the British royal family. Although the change of these words was quantified in the analysis, there is no simple way of determining what the change means or if each word is really linked to the specific event.
Further analysis of the nine words presented in tables 3.7 and 3.8 was done. Each word from vector spaces of models $EN_2, ..., EN_{14}$ was mapped using the corresponding transformation matrix to the vector space of the EN_1 model. The fourteen obtained vectors were reduced to two dimensions using the principal component analysis (PCA). The resulting two-dimensional vectors were plotted in the scatter plots presented in this chapter. It is important to note that each plot has different axis sizes for better readability.

Table 3.1: The change of English words measured by Euclidian distance.

Table 3.2: Biggest change

	word	$\sum$ **d**
1.	hong	403
2.	kong	393
3.	francisco	376
4.	diego	356
5.	santa	303
6.	saudi	302
7.	scotland	275
8.	escorts	252
9.	mold	246
10.	erectile	243
11.	wheat	238
12.	escort	237
13.	bearings	237
14.	keto	226
15.	mask	224
16.	harry	224
17.	replica	222
18.	dysfunction	217
19.	acids	215
20.	ebook	213

Table 3.3: Smallest change

	word	$\sum$ **d**
1.	apparently	63
2.	realized	71
3.	consistently	71
4.	essentially	72
5.	somewhat	73
6.	yesterday	73
7.	impressive	74
8.	demonstrate	74
9.	producing	75
10.	hence	76
11.	demonstrated	76
12.	somehow	76
13.	involves	76
14.	surely	76
15.	sees	76
16.	hoping	77
17.	initially	77
18.	exceptional	77
19.	whereas	77
20.	gorgeous	78

Table 3.4: The change of English words measured by cosine distance.

Table 3.5: Biggest change

	word	$\sum d_{cd}$
1.	kong	7.76
2.	hong	7.56
3.	diego	7.37
4.	francisco	6.79
5.	saudi	5.11
6.	santa	4.86
7.	replica	4.01
8.	harry	3.75
9.	delta	3.50
10.	louis	3.38
11.	wheat	3.32
12.	erectile	3.31
13.	cape	3.30
14.	dysfunction	3.3
15.	jean	3.23
16.	phoenix	3.21
17.	saint	3.13
18.	taylor	3.13
19.	maria	3.1
20.	francis	3.1

Table 3.6: Smallest change

	word	$\sum d_c d$
1.	focuses	0.25
2.	behalf	0.26
3.	involves	0.27
4.	grew	0.27
5.	interact	0.27
6.	rely	0.27
7.	depend	0.28
8.	weren (weren't)	0.29
9.	comply	0.29
10.	aims	0.3
11.	involve	0.3
12.	lets	0.3
13.	occurs	0.31
14.	remained	0.31
15.	picked	0.31
16.	demonstrate	0.31
17.	invited	0.31
18.	shouldn (shouldn't)	0.31
19.	ensuring	0.31
20.	impressive	0.31

Table 3.7: Frequency of interesting words in each dataset.

	kong	**hong**	**diego**	**francisco**
EN_1	8716	8376	7785	8799
EN_2	8166	7946	6366	8150
EN_3	9132	8455	8158	8804
EN_4	8475	7937	7288	8149
EN_5	7254	6991	5631	7346
EN_6	6355	6537	5229	7177
EN_7	6352	6234	5233	7022
EN_8	7881	7710	5568	8304
EN_9	6572	6568	4414	6671
EN_{10}	7185	7062	5144	7229
EN_{11}	5827	5500	4588	6595
EN_{12}	6669	6515	4570	6558
EN_{13}	5093	5089	3892	7200
EN_{14}	6019	5828	4731	7327

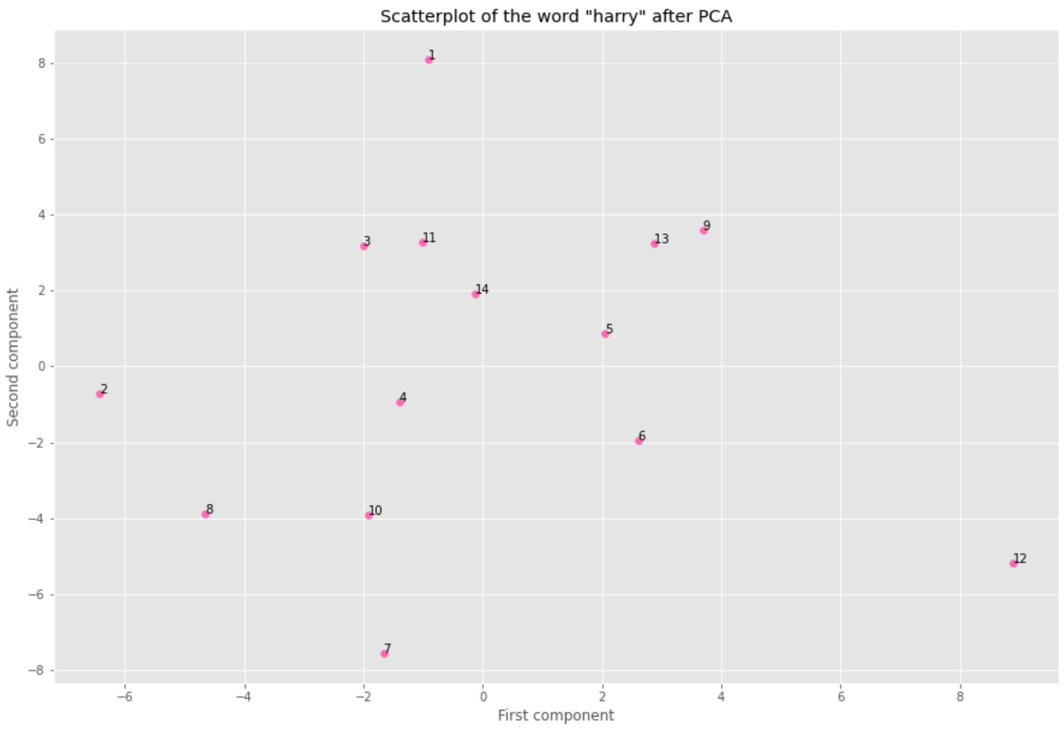

Figure 3.1: The two-dimensional visualization of the change in the meaning of the word *harry*.

Table 3.8: Frequency of interesting words in each dataset.

	saudi	**harry**	**delta**	**scotland**	**mask**
EN_1	5599	7858	4918	11860	34085
EN_2	5613	7106	6274	7631	27769
EN_3	6154	8460	6108	12677	39133
EN_4	4541	6966	5229	6778	27708
EN_5	5130	6151	4534	7125	19816
EN_6	4220	6268	5258	6872	16154
EN_7	4682	5855	5727	6734	12339
EN_8	5930	7576	5887	7129	17370
EN_9	5225	5762	5536	7487	13610
EN_{10}	4980	7616	5642	12098	12865
EN_{11}	4148	6295	5126	6438	11158
EN_{12}	4423	5068	4766	5624	10691
EN_{13}	3407	4329	4233	4703	7674
EN_{14}	4509	5687	5348	5225	9606

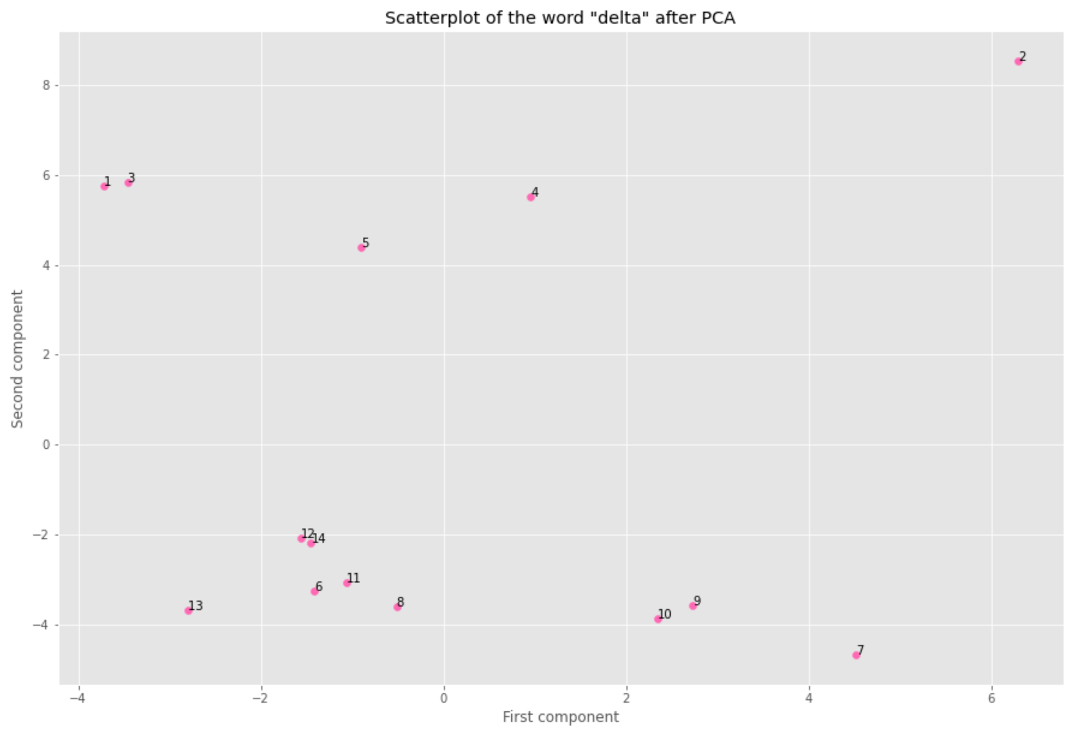

Figure 3.2: The two-dimensional visualization of the change in the meaning of the word *delta*.

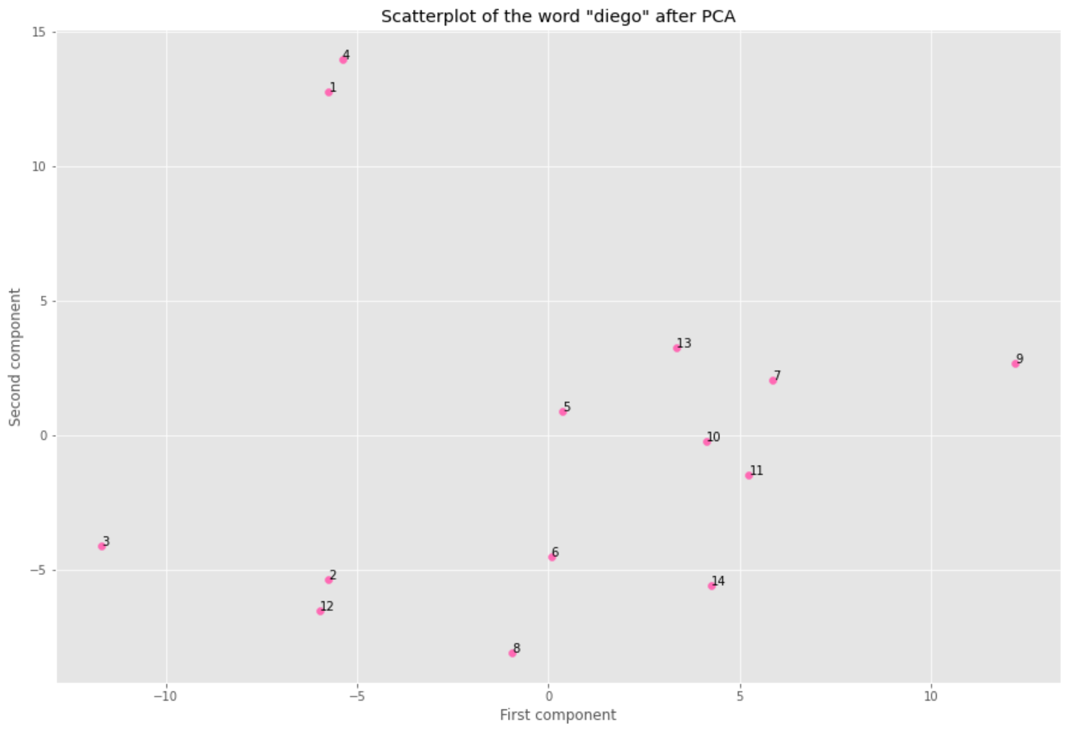

Figure 3.3: The two-dimensional visualization of the change in the meaning of the word *diego*.

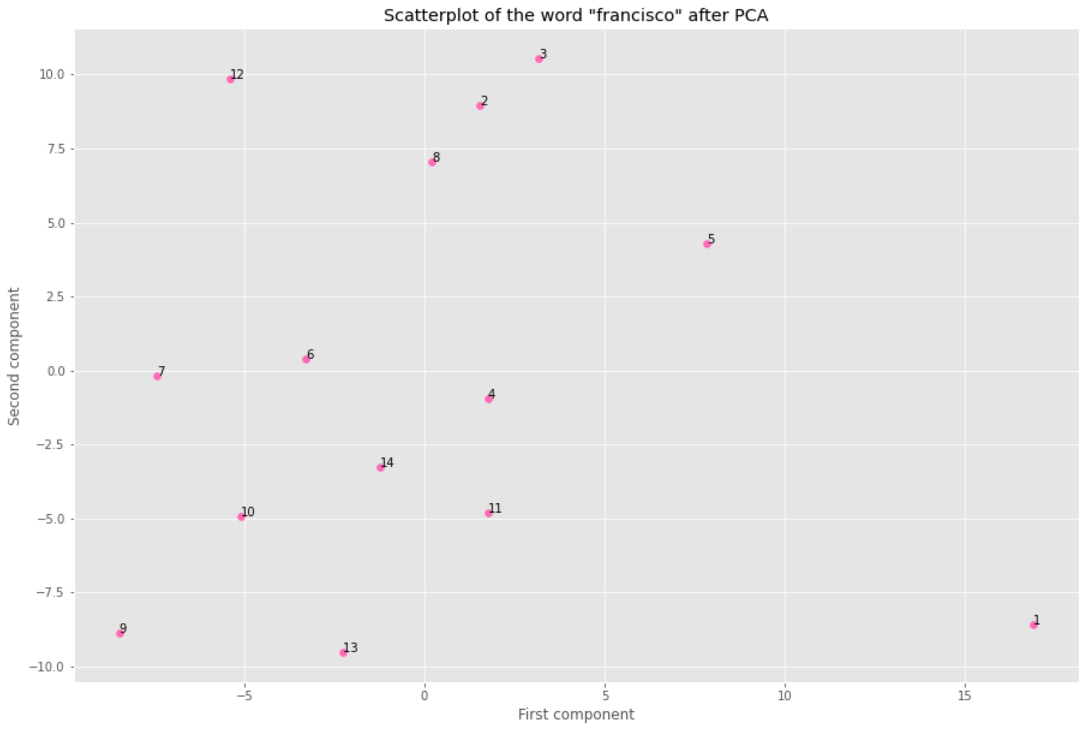

Figure 3.4: The two-dimensional visualization of the change in the meaning of the word *francisco*.

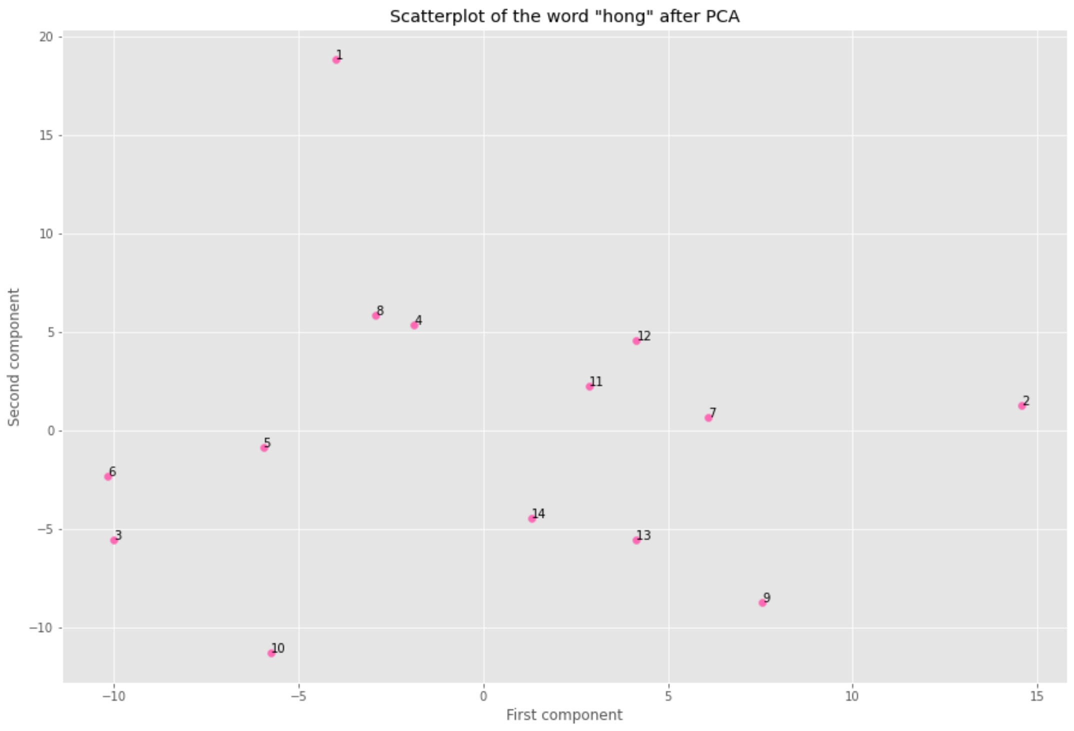

Figure 3.5: The two-dimensional visualization of the change in the meaning of the word *hong*.

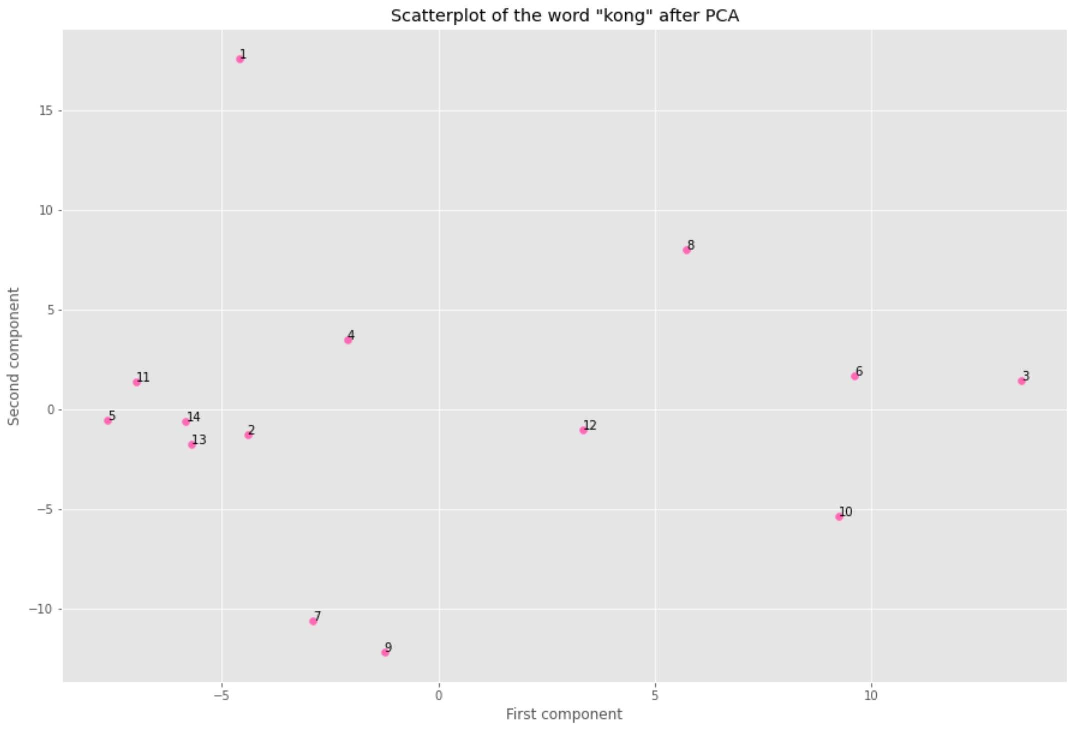

Figure 3.6: The two-dimensional visualization of the change in the meaning of the word *kong*.

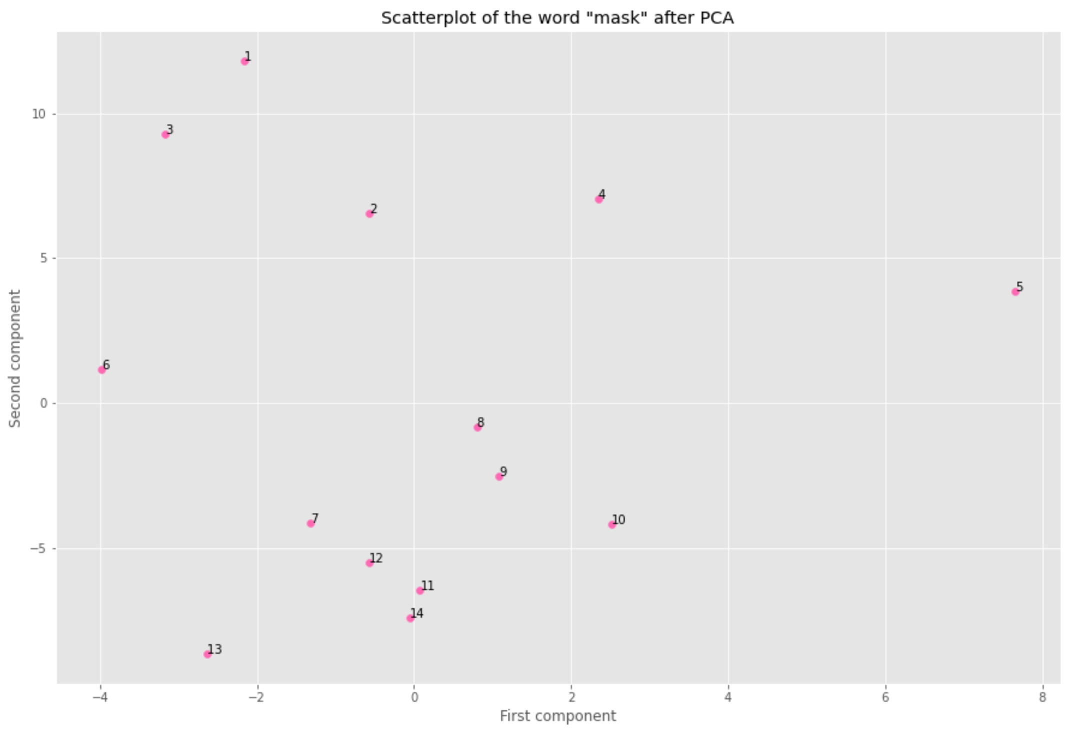

Figure 3.7: The two-dimensional visualization of the change in the meaning of the word *mask*.

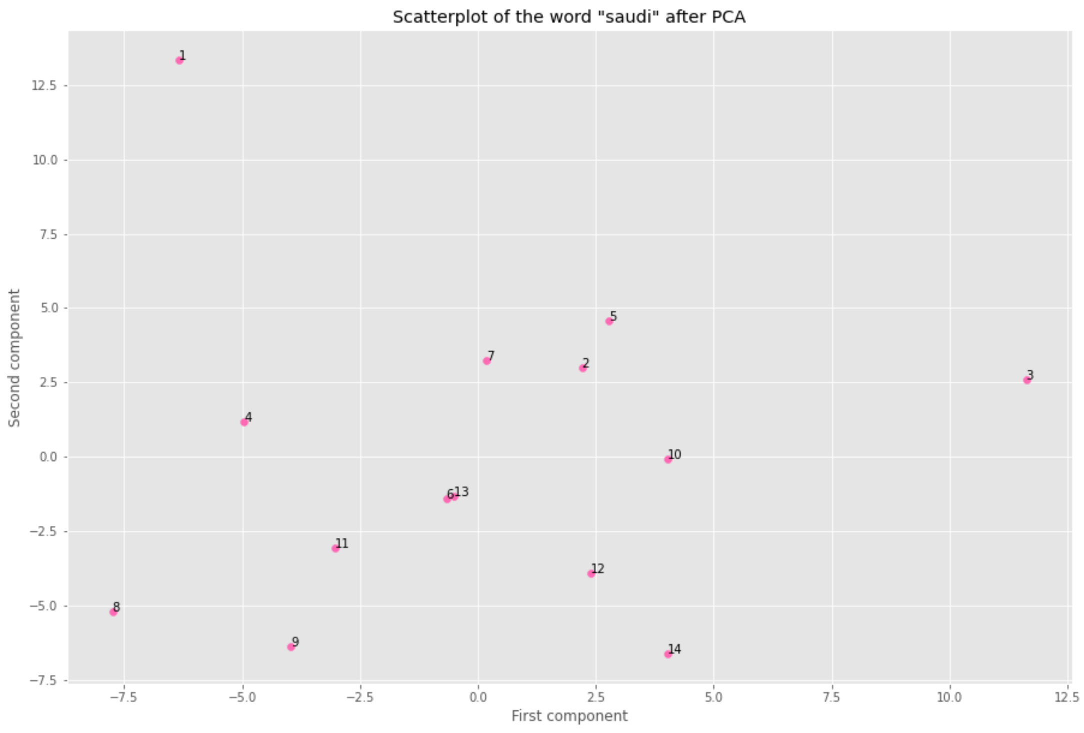

Figure 3.8: The two-dimensional visualization of the change in the meaning of the word *saudi*.

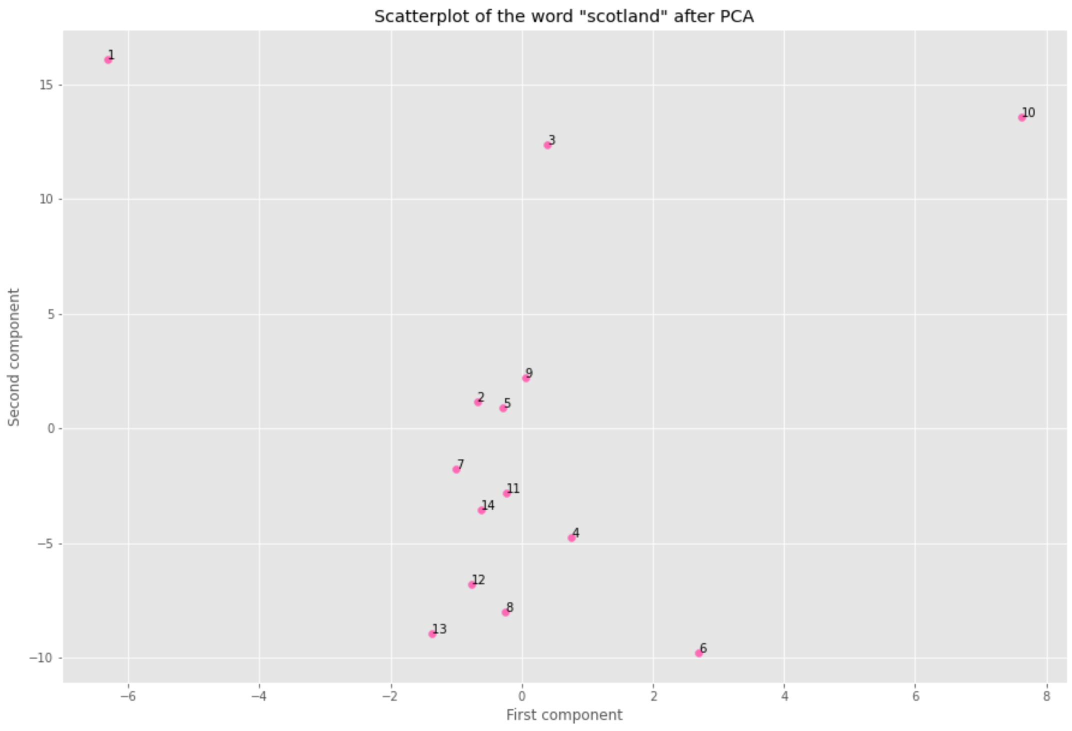

Figure 3.9: The two-dimensional visualization of the change in the meaning of the word *scotland*.

Conclusion

This thesis had two goals. The first goal was to introduce the reader to machine learning and natural language processing. The second goal was to analyze data from the internet using techniques described in the theoretical part of the thesis and create a mathematical method to quantify the change in a particular language. Word2Vec models created in 2.3.2 proved their quality by accurately understanding relationships between words (word analogies). Techniques from [10] were implemented using Python programming language. The results in Spanish to English and English to Spanish translation have proven to be comparable with the ones produced in [10]. This approach of automatic language translation was used on multiple monolingual models. These models were trained on datasets created using data scraped from the internet in different months. Using this technique of monolingual *translation* the change in semantic word meaning was captured and measured. Further, a few interesting words were selected and analyzed more in-depth. The final results do correspondent with events that happened in the years 2021 and 2022 such as the pandemic, and many other political and cultural events all over the world.
The analysis of the data had its limitations due to resource availability. Although the amount of data processed in the data analysis is a good representative sample, not all Common Crawl data was analyzed. More data would produce more robust results and create bigger models with possibly better performance on word analogies and language translation. Additionally, these models might capture the change in the word meaning more accurately. Results achieved with limited resources could also be further analyzed and combined with knowledge about the world in 2021 and 2022 might yield more insight. Capturing the change of the word meaning presented in this thesis involves in-depth machine learning and mathematical knowledge, but in order to completely understand the evolution of the word, one must be aware of the world's political, cultural and economic events.
Despite resource limitations, this work has achieved all its goals and can be used in future research. Future work could reproduce the data analysis on the whole Common Crawl dataset from the time interval used in this thesis or create a bigger time interval and analyze data over many years. Further analysis of the most changed words including more visualizations could help to understand the evolution of each word.

Bibliography

[1] Alyafeai Z., AlShaibani M. S., Ahmad I. *A survey on transfer learning in natural language processing.* arXiv:2007.04239. 2020.

[2] Deng, L. *The mnist database of handwritten digit images for machine learning research.* IEEE Signal Processing Magazine, 29(6), 141–142, 2012.

[3] Deng J., Dong W., Socher R., Li L.-J., Li K., Fei-Fei L. *Imagenet: A large-scale hierarchical image database.* In 2009 IEEE conference on computer vision and pattern recognition (pp. 248–255), 2009.

[4] Gers F. A., Schmidhuber J., Cummins F. *Learning to forget: continual prediction with LSTM.* Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470), pp. 850-855 vol.2, 1999.

[5] Glorot X., Bordes A., Bengio Y. *Deep Sparse Rectifier Neural Networks.* International Conference on Artificial Intelligence and Statistics. 2011.

[6] Hochreiter S. *The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions.* International Journal of Uncertainty, Fuzziness and Knowledge-Based SystemsVol. 06, No. 02, pp. 107-116, 1998.

[7] Joulin A., Bojanowski P., Mikolov T., Jegou H., Grave E. *Loss in Translation: Learning Bilingual Word Mapping with a Retrieval Criterion.* Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 2979–2984, 2018.

[8] McNally S., Roche J., Caton S., *Predicting the Price of Bitcoin Using Machine Learning.* 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), pp. 339-343, 2018.

[9] Mikolov T., et al. *Distributed Representations of Words and Phrases and their Compositionality.* NIPS'13: Proceedings of the 26th International Conference on Neural Information Processing Systems, Vol. 2, pp. 3111–3119, 2013.

[10] Mikolov T., Le Q. V., Sutskever I. *Exploiting Similarities among Languages for Machine Translation.* arXiv:1309.4168, 2013.

[11] O'Shea K., Nash R. *An Introduction to Convolutional Neural Networks.* arXiv:1511.08458, 2015.

[12] Reed R., Marks R. J. *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*. The MIT Press. 1999.

[13] Shah D., Zhang K., *Bayesian regression and Bitcoin*. 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton), pp. 409-414, 2014.

[14] Tan C., Sun F., Kong T., Zhang W., Yang C., Liu C. *A survey on deep transfer learning*. ICANN. 2018.

[15] Tehrani S. F., Calvello M., Liu Z., Zhang L., Lacasse S. *Machine learning and landslide studies: recent advances and applications*. Nat Hazards 114, 1197–1245, 2022.

[16] Werbos, *Backpropagation: past and future*. IEEE 1988 International Conference on Neural Networks, pp. 343-353 vol.1, 1988.

Appendix A

List of Acronyms

ANN	Artificial neural network	12
ML	Machine learning	13
DL	Deep learning	13
PCA	Principal component analysis	13
RL	Reinforcement learning	13
MSE	Mean square error	18
MAE	Mean absolute error	18
GD	Gradient descent	19
SGD	Stochastic gradient descent	19
ADAM	Adaptive moment estimation	20
TP	True positive	22
FP	False positive	22

TN	True negative	22
FN	False negative	22
RNN	Recurrent neural network	27
LSTM	Long short-lerm memory	27
CNN	convolutional neural networks	28
CV	computer vision	29
NLP	Natural language processing	29
CC	Common Crawl	31