
# Large High-quality Corpus building from Web and Text Classification for Azerbaijan Language

Amir Adamov
Computer Science
School of IT and Engineering
ADA University
aadamov5055@ada.edu.az

Gunash Hajiyeva
Computer Science
School of IT and Engineering
ADA University
ghajiyeva2021@ada.edu.az

Ismayil Karimli
Information Technology
School of IT and Engineering
ADA University
ikarimli4586@ada.edu.az

Narmina Aghayeva
Information Technology
School of IT and Engineering
ADA University
naghayeva4607@ada.edu.az

# Contents

## I. Abstract

The project consisted of two stages — Data collection and Development of Text Classification algorithms based on the collected data. The collected data was solely in the Azerbaijani language and was gathered from 4 websites. The total number of books in the corpus is around 3090. The books were categorized among 22 categories. All the collected books were converted to utf-8 plain-text format using pdftotext tool from Poppler. The converted data was cleaned manually and in an automated way. In Natural Language Processing (NLP) text classification is determining the category of a given text based on its content. While text classification tools exist for some languages such as English, Spanish and Chinese, a robust version of such a tool is not available for the Azerbaijani language.

The thesis first discusses the collected data in-depth, then aims to compare the performance of four Machine Learning algorithms — Naive Bayes, Random Forest, Support Vector Machine and Recurrent Neural Network — in classifying the collected texts across 22 categories.

In total, approximately 78 hours was spent on training the models. In the end Recurrent Neural Network with LSTM algorithm was found to be the best performing among the four with the accuracy score of 67%.



***Keywords:*** *Text Corpus, Data collection, Machine Learning, NLP, Text classification, Naive Bayes, Random Forest, Support Vector Machine, Recurrent Neural Network, Azerbaijani*


## II. Introduction

### A. Definition

This thesis demonstrates our Final Senior Design Project report, and this final report defines the development of our project. During this interesting project we got a chance to examine and to have a deep research on our rich Azerbaijani language. First of all, we want to state that similar projects have been done by our graduated SITE students and this year we, as a group, decided to maintain it, the main aim was to make a great deal of improvements on the project and fix some bugs. The absolute necessary detail of the project is that we have created our databases which contain clean data (words and sentences) that is extracted from books in Azerbaijani. The first semester we started to collect the books in Azerbaijani, found similar patterns among them and lastly applied many regular expressions in order to achieve clean data that is needed for the second part of the project. The SDP project is about text classification of Azerbaijani books across 22 categories. In order to get the classification of any text, users can type the text and the classification of the text will be demonstrated on the screen of our project's website. The essential detail of our project is that a new large database of many Azerbaijani books from various categories has been collected by our group. In general, it contains 22 different categories (art, psychology, literature, politics and etc). Furthermore, approximately 3090 books were collected, and the project has got about 6256759 (6 million+) sentences, therefore the project contains about 77994547 (77 million+) Azerbaijani words from these books.

Text classification is the process of classifying the given text/document based on its content to a certain category. Text classification has become a significant technique for figuring out and organizing electronic texts in the native language, since it is known that electronic texts have grown rapidly. Depending on the document content, text classification attributes an invisible document to one or more pre-defined groups. In general, text classification is a construction problem where these models can classify new files into predefined classes (Lui, 2006; Manning, Raghavan & Schütze, 2008). Text classification is not a process that includes only model training, besides this, it includes other steps, such as data processing, transformation, and dimensionality reduction. It is famous in the usage of many areas, such as digital libraries, spam filtering, online news and etc. Text Classification is in high demand because of the large amount of text on the internet which can not be classified manually by humans in terms of cost and time constraints. The task of text classification can be done in three ways – Rule-based approach, Machine Learning approach and Hybrid approach. Machine Learning classification approach attempts to determine the category of a given text based on the past observations.

Machine Learning based text classification consists of two phases — training and predicting. In the training phase the features are extracted from the given text and fed to the Machine Learning model along with a tag which in turn creates certain patterns between certain words in the pre-classified training data and categories. In the prediction stage the features are extracted from the given text and used as an input to the classifying algorithm which outputs a tag based on the previous observations in the training stage.

## B. Purpose

Due to the absence of a quality text classification tool in Azerbaijani language the main purpose of this project was to develop a robust, open-source text classification tool for Azerbaijani language. Furthermore, prior to our work, the presence of a large, clean, plain-text data based on Azerbaijani books was almost nonexistent. Therefore, second goal of the project was to create a large plain-text corpus based on the books in Azerbaijani language across a number of categories. Another aim of the project was to create a clean and robust dictionary of words across a wide range of categories such as cities, countries, first and last names etc. The dictionary will also contain words across all parts-of-speech in Azerbaijani language. Additionally, the list of stop words existing in the Azerbaijani language was among the goals of the project as well. The final goal of the project was to make all the data used in the project as well as the code itself open-source. We believe that the availability of these resources will ease the works of the future projects that will need a large text data in the Azerbaijani language.

## C. Project Objectives

In the modern world, there are a small group of tools and algorithms that were implemented by individuals to improve and enhance the linguistics in order to make editors' and authors' lives easy. Possible options for any single word or check the words can be seen by the user via one of these algorithms. Such kinds of applications have not been improved for the Azerbaijani language. Therefore, the objectives of the project are the following:

1. Creation of a text-classification tool for the Azerbaijani language
2. Creation of a large and properly preprocessed text corpus based on Azerbaijani books
3. Creation of a quality list of words across numerous categories and parts-of-speech based on Azerbaijani language
4. Creation of a lemmatizer for Azerbaijani language
5. Creation of a high-quality UI for the project and making it accessible to everyone
6. Making all the above-mentioned points open-source

It should be outlined that all of the project objectives were achieved successfully.

## D. Significance

There are several use cases for a text classification tool. Availability of a well built text classification tool will allow the companies, individuals, linguistics to classify a large, raw, unorganized big data in Azerbaijani language into categories and extract some meaning out of it in an automated fashion. Moreover, the tool can be used in combination with tools such as sentiment analyzer to obtain even further meaning out of a raw data. For example, by using a text classification tool alongside a sentiment analysis tool, businesses can categorize as well as extract the intent of the reviews from their customers. Moreover, text classification is also used in improving SEO by classifying the contents of a web-page to an appropriate category. Vertical search engines are another area where text classification is heavily used.

Although the project uses Azerbaijani books as a source for training, it can be used to categorize data from other sources such as Wikipedia or news websites as well. Furthermore, the tools that have been developed alongside the project (lemmatizer, text corpus, dictionary) can be used in isolation from the main part of the project as well.

## E. Novelty

The project brings several novelties to the table. Firstly, being a open-source text-classifier in Azerbaijani language based on Azerbaijani books makes the project unique. Additionally, the project also provides a vast collection of plain-text data based on Azerbaijani books with over 6 million sentences which have been pre-processed to a certain degree and have been categorized among 22 various categories. Besides, the tool is capable classifying text using one of four different Machine Learning algorithms. Furthermore, a lemmatization tool in Azerbaijani language is a novelty in itself as well. Although there have been strives in developing a lemmatization/stemming tools for the Azerbaijani language in the past, there were several flaws in them. While the rule-based lemmatization tool developed for this project is not perfect by any means it still improves upon the flaws of the already existing tools and creates more accurate lemmas from a given text.

## F. Problem Statement

One of the leading Machine Learning implementation fields - Text Classification, has a vast of application areas that we all observe uncon-

sciously during a day. If we would like to get data sources, materials, papers to implement text classification for English, they would be easily reachable for us. Besides, text classification for English has already been applied for many fields, platforms and improved to the advanced levels since it is the international language. However, at some point, we need to solve digital problems using our language, and as technology improves in our country, now and in the near future, we will need to have our own language data sources and advanced implementation of technology on them. Even though most of the global platforms can be used in our language, we need to digitize language operations like text classification in Azerbaijani ourselves because none of the foreign scientists are obliged to train the data in our language and improve it. As a citizen of Azerbaijan and senior IT students, the main problems we are aiming to tackle are the absence of an effective text-classification tool for the Azerbaijani language and the lack of categorized, plain-text data from Azerbaijani books. Generally, there were two main goals of our project.

- First was the absence of a high quality, large, plain-text data corpus from Azerbaijani books. Furthermore, for us it was vital to categorize the collected data so that each data set could be used in isolation if needed.
- Second was to create a text-classification tool for Azerbaijani language.

We believe that in tackling these issues we will be making a great contribution to the digital presence of the Azerbaijani language.

## III. Literature review

### A. Text Categorization

When we talk about text categorization and its features, firstly, the word comes to the mind. It generally has two meanings; the one is used as a unit in order to describe a document or to index a document. The other is primarily concerned with how to assign a suitable weight to a given feature, an instance, "bag of words" can be taken into consideration. The feature is a single word, if we use its erstwhile meaning; however, the last-mentioned meaning is used as TF-IDF weighting. This Literature Review section will demonstrate the previous researches and scholars' papers about the features and techniques for the text categorization in terms of those two meanings. Many scholars have looked at syntactic phrases, in addition to single words, for the first meaning. Language grammars are used to extract a syntactic expression. Syntactic phrases did not increase the efficiency of standard "bag-of-word" indexing in general, according to experiments. Researchers have often paid close attention to statistical phrases. A statistical term is made up of a collection of terms that appear in a statistically interesting order in a text and is typically referred to as an n-gram. With the aid of a feature selection mechanism, improved results were recorded, when statistical phrases were utilized in order to enhance the text demonstration of a single word. The short statistical term was also found to be more useful than the long one, according to the researchers. In addition, for the first meaning, a word cluster was a different appealing element. The distribution of a word across various categories was used to define it.

Sauban and Phahringer introduced new approach for the representation that directly exploited word sequence knowledge. (M. Sauban and B. Pfahringer, 2003). They started by calculating an exclusionary score for each word. The text was then indicated as a curve representing the shift in the cumulative scores, with each word entered in order. Document Profiling was the name given to this curve. To convert a profile into a constant number of features, two different techniques were applied. One of them was to take a constant-gap sample from it; however, another one was to extract high-level description data. On the other hand, with a low cost, approximate results to the "bag-of words" approach were obtained. The weight, specifically, intra-document and inter-document terms are used for the second meaning. The information within the document is taken into account by intra-document-based weight, whereas the information from the corpus is taken into account by the inter-document-based weight. Looking at the TF-IDF, here TF can be thought as an intra-document part, and the IDF part as an inter-document part. Two different strategies particularly were used to determine the importance of a sentence. One of them was to determine the degree of resemblance amongst title and the given sentence, whereas the second strategy weighted the stature of all words that were used in a certain sentence as the last importance.

In order to substitute TF, a weighted term frequency was selected with each appearance being weighted by the value of the sentence in which it appeared. Researchers attempted to enhance the IDF from the unsupervised view and the supervised view when it came to the inter-document-based weight. In order to quantify the skewness of the distribution of a word's frequency, Leopold and Kingermann suggested the Redun-

dancy to calculate its value (E. Leopold and J. Kingermann, 2002). Lan, Sung, Low and Tan have also written the term "relevance weight" in the similar studies (M. Lan, S.Y. Sung, H.B. Low, and C.L. Tan, 2005). Rather than using the overall number of files in IDF, the expression of the relevance weight utilized collection of those files having a word to split the collection without this word. Numerous researchers, on the other hand, assumed that the IDF extracted from the text was unsuitable for the text categorization. Many supervised weights were suggested to concentrate on the text categorization. Shankar and Karypis calculated the distinguishing strength of each word using a method identical to the Gini Index ( S. Shankar and G. Karypis, 2000). The IDF were altered by Debole and Sebastiani by using several functions including Chisquare, Information Gain and Gain Ratio that are commonly for the selection (F. Debole and F. Sebastiani, 2003). Additionally, with Gain Ration function, the highest results were yielded. In addition to the functions, Soucy and Mineau have applied interval-based weighting method, as well as, that approach had the benefit of automatically selecting features. In addition to the functions, Soucy and Mineau have applied interval-based weighting method, as well as, that approach had the benefit of automatically selecting features (P. Soucy and G.W. Mineau, 2003). On comparisons, they showed a major enhancement over the traditional TF-IDF process.

### B. Naive Bayes for Text Classification with Unbalanced Classes

Due to its computational efficiency and reasonably good predictive ability, multinomial naive Bayes (MNB) is a standard tool for document classification. It has recently been discovered that appropriate data transformations will increase predictive accuracy even further. The writers propose another transition in this paper that is intended to address a possible issue with the application of MNB to unbalanced datasets. By changing attribute priors, it is suggested an effective adjustment. This correction can be applied as a second data normalization step, and this shows that it improves the region under the ROC curve significantly. It is also demonstrated in the article that the updated variant of MNB is very similar to the primary centroid-based classifier and empirically evaluate the two ways.

The type of naive Bayes known as multinomial naive Bayes (MNB) is widely used for text categorization problems. The article identifies a possible MNB deficit in the form of distorted class sizes in this article. When there are fewer data, as in the smaller classes in a text categorization problem, the traditional practice of initializing word frequencies for all classes to the same value—typically, a value of one—biases forecasts in favor of the larger class: initial word counts have a more significant effect on the expected likelihood when there is fewer data. The authors look at using different initial word counts for different-size groups and suggest a heuristic for selecting the right one for each. This update, which normalizes the word count vector associated with each class as a pre-processing phase, will significantly increase predictive accuracy as calculated by the region under the ROC curve. The updated variant of MNB is often compared to the centroid classifier, which is closely related.

This article established a possible MNB flaw in the form of unbalanced datasets and demonstrated that per-class work vector normalization is a viable solution. Empirical findings suggest that normalization can increase output dramatically. The article also demonstrated that when the class vectors are normalized to unit length, MNB with class vector normalization is very closely related to the regular centroid classifier for text classification, and empirically validated the relationship.

### C. Influence of Word Normalization on Text Classification

The comparison of different Lemmatization and stemming algorithms, commonly used in natural language processing, is the subject of this article (NLP). These two methods are often confused, but there is a significant distinction between them. Lemmatization is more widely used because it generates the simple word type required in many applications (i.e., cross-language processing and machine translation). On the other hand, Lemmatization is a complex process, particularly in highly inflected natural languages with several words for the same normalized word form. The article presents a new lemmatization algorithm that makes use of the Eurowordnet multilingual semantic thesaurus (EWN). Here authors explain the algorithm in-depth on two separate corpora and compare it to other commonly used word normalization algorithms. In contrast to other approaches, in the article it is shown that positive findings obtained by EWN-based lemmatization method. The authors also talk about how the term normalization affects classification tasks in general. In contrast to other word normalization approaches, the approach performs well overall and achieves equal

accuracy and recall. However, the research shows that word normalization has no substantial impact on the text classification mission.

Thesaurus EuroWordNet (EWN) is an essential part of the strategy. Words are transformed into their simplest form using EWN and Ispell (lemma). Besides, each term is linked to the EWN thesaurus node synset, representing a group of synonyms. The EuroWordNet thesaurus can be used in a variety of NLP applications. It is a multilingual word and relational database for most European languages. It includes synsets, or groups of synonyms, as well as relationships between them. Each synset is given a unique index. It links the languages together using an inter-lingual index, such that the same synset in one language has the same index in another. As seen in several papers, the EWN-based approach allows for additional processing strategies such as query extension, cross-language information extraction, and word sense disambiguation. It is essential to assign an EWN index to each word of a text to use EWN.

Lemmatization reduces vocabulary to its simplest forms. The category of dictionary lemmatization algorithms includes EWN-based Lemmatization. The most complex aspect of the EWN-based approach is constructing a dictionary. Using the EWN thesaurus and the Ispell dictionary, the article suggested a system for creating lemmatization dictionaries. The Ispell utility was used to derive word types for the lemmatization dictionary. From the stem contained in the Ispell dictionary, it is possible to produce all current word types.

Studies show that word normalization has a negative rather than a positive impact on text categorization for both morphologically complex and morphologically simple languages. Lemmatization and stemming increased the macro-F1 value in some situations, but the gains were not statistically substantial. On the other hand, stop-words elimination increased classification accuracy in most situations, although the findings were not statistically crucial in this situation. On the other hand, stop-words withdrawal limits the size of secret papers and speeds up the classification process.

Stop-words elimination and omission of term normalization seem to be the optimal preprocessing method for text classification. When term normalization was used, the reduction in description precision was usually apparent and sometimes statistically meaningful.

Porter's stemmer is the most suitable algorithm when word normalization for English is needed. It has little impact on processing precision and reduces the corpus dimension by 72

For morphologically rich languages, on the other hand, a dictionary approach is preferable. In the Czech language, an EWN-based algorithm without transformation to EWN indexes performs well. It is anticipated that as the consistency of EWN improves, so will the efficiency of the EWN-based system. EWN layout can be used in other NLP tasks, as described in section 2.1, which is a significant advantage of the methodology.

### D. Text Classification Techniques

This article aims to examine various text classification strategies used and their strengths and disadvantages to raise awareness of various information extraction options in the field of data mining. Artificial Intelligence is reshaping text classification methods to improve information acquisition. Even though AI is more widely used in science, its position in text mining is still uncertain.

Text classification techniques in AI were defined as research problems, and techniques were classified according to the algorithms involved. The learning technique was used to divide these algorithms. Finally, to visualize the relationship between learning procedures and algorithms, the results were plotted as a tree structure. Contribute to an advanced area like AI; this paper identifies the benefits, shortcomings, and emerging research developments in text classification. For data scientists, this information is essential. They may create personalized data models based on the results of this report. It also improves the industry's understanding of text mining techniques' operating effectiveness. It further contributes to project cost reduction and facilitates good decision-making. Conclusions It has been discovered that studying and comprehending the essence of data is more critical than mining. With the growing amount of data and the need for consistency, the text classification process must be automated. Another promising area of research is the development of complex text data models using deep learning systems. It can complete dynamic Natural Language Processing (NLP) tasks with semantic constraints.

Practitioners' Recommendations Any of the guidelines for clinicians include frame analysis, fraud identification, plot science (where evidence tells a story), clinical apps to identify disorders, and dialogue analysis. Researchers' Recommendation Researchers may look at developing simplified algorithms in terms of coding and execution and improved methods for informa-

tion distillation, multilingual text refining, domain knowledge integration, subjectivity identification, and contrastive perspective summarization.

Impact on Society text classification is the foundation of data analytics and the driver that enables information discovery. It encourages cutting-edge decision solving, such as anticipating an occurrence before it happens, classifying a trade as 'Fraudulent,' and so on. The findings of this study could be used to create apps that help people make better decisions. These well-informed decisions would aid in the optimization of capital and the maximization of human benefits. Research in the Future By choosing better parameters representing successful information exploration, better methods for parameter optimization can be found in the future. When it comes to text classification, the task of streaming data processing is still under-appreciated.

### E. Improving Text Classification with Word Embedding

It is impossible to render function reductions based on attribute definitions, which is a problem in text classification. The classification accuracy can also be harmed by inadequate feature reduction. Word2Vec, a word embedding tool, has recently gained prominence as a result of its high precision rate in analyzing semantic similarity between words at a low computational cost. However, there is only a small amount of literature on the use of Word2Vec for feature reduction. In this research, Word2Vec is used to build a method for reducing feature size while increasing classification accuracy. Using graph search methods, it is possible to reduce the number of features by loosely clustering related features. In order to pair and cluster the features, in the paper similarity thresholds above 0.5 is used. Finally, the influence of the approach using the Multinomial Nave Bayes classifier is evaluated, Support Vector Machine, K Nearest Neighbor, and Random Forest classifier. To test the process results, in the paper authors used four datasets with sizes ranging from 100,000 features to 400,000 documents. In terms of various datasets and classifiers, the result shows that about 4-10% feature reduction was accomplished with up to 1-4% increase in classification accuracy. Meanwhile, by integrating the approach with other classic feature reduction techniques, including chi-square and shared knowledge, feature reduction and classification accuracy are improved. According to the paper, using the system of using Word2Vec, it is possible to maximize feature reduction while still increasing classification accuracy. On four different datasets, results of the approach using four different types of classifiers and two different types of classic feature reduction techniques is tested. In terms of various datasets and classifiers, the result reveals that about 4-10% feature reduction is accomplished with up to 1-4% increase. In the analysis that has been conducted, various classifiers behave differently: SVM achieves the highest classification accuracy in all four datasets. NB is successful in the 20-newsgroups dataset, and KNN reveals the most significant increase in R52 of the Reuters-21578 dataset and WebKB dataset compared to the baseline. In most scenarios, the approach shows no significant improvement in RF because the uncertainty of its estimation outcome covers the improvement of our method, but when a subset of R52 of Reuters-21578 is used, a significant improvement is shown. Meanwhile, it is demonstrated that after CS and MI, the system indicated in the paper will effectively increase feature reduction and classification accuracy.

### F. Text Classification Using Long Short-Term Memory

Text classification tasks have a simple problem of outputting data with high dimension causing the final model to perform sub-par. This happens because of two reasons. First is that the training time increases at an exponential scale. Secondly, as the number of features increase, the risk of overfitting increases alongside it. In their study Winda Kurnia Sari et al. test the implementation of RNN with LSTM for classifying text in English. When used standalone to study long-distance connection in the data sequentially, vanishing gradient problem may surface. LSTM substitutes the hidden vectors of the RNN with memory blocks coupled with gates. These gates are known as forget, input and output gates. By correctly gating the weights LSTM is able to retain a long-term memory of the observed data. The researchers use confusion matrix for multi-label classification. There are two optimization options for deep learning models — Adam and RMSProp and the paper opts for the Adam optimizer. The researchers used the metadata from 46985 Web Of Science (WOS) publications. The authors conduct the experiment with and without dropout. Dropout refers to the technique of preventing overfitting. To train the model the dimension was 50, dropout 0.2, spatial dropout 0.5, recurrent dropout 0.2 and epoch count 10. The researchers perform the experiment with different number of input feature and word sequence

counts. In the end it was concluded that maximum word sequence of 250 with 10000 words grouped to be used as input feature and dropout active provided the highest accuracy score with 82.13%.

### G. Multilingual text categorization

As Yasotha and Charles (2015) stated, text categorization has been extensively researched, and it provides a proper study survey, which addresses various machine learning-based text categorization techniques. In addition, Jianfang and Hong-bin (2010) and Erlin, Rio and Rahmiati (2013) found that Support Vector Machines are one of the extreme algorithm. They start SVMs with a large amount of textual analysis and train quickly.

During the 1990s, as Chung-Hong Lee and Hsin-Chang Yang (2002) pointed out that most widely used allocation metric was a Reuters collection known as Reuters-21578, which included 12,902 files that needed to be categorized into around 100 different categories and it was also mentioned by Yasotha and Charlesthe (2015). In addition to this, Lei and Qiao (2012) explored that this standard is also used to assess the success of individual algorithms, as well as the problems that come with relocating against larger calibration certificate allocations.

There are several machine belief (ML) algorithms that have been successfully adapted up to this stage (Lengyel, 2013). Neural Networks (NN), Naive Bayes, Support Vector Machines (SVMs), and K-nearest Neighbors (k-NN) are among the topics covered. Each of these methods has its own set of benefits and drawbacks in terms of allocation and scalability. The algorithm that is used will be determined by the program and the volume of data that will be used. Because of the large number of users and data in web apps, performance is particularly critical.

## IV. Design Concept

### A. Alternative Solutions/Approaches/Technologies

#### A.1 Data set

For Data set we found numerous websites with Azerbaijani PDF and EPUB files. The final choices of the website list were made on the following criteria:

- Whether the website provided a download option
- How easy was it is to automate the data download process from the website
- Quality of the books in the website

#### A.2 Dictionary

For choosing the dictionary, we started with https://obastan.com/ which contains a vast collection of words across numerous categories. Since the website was easy to scrape using Bash and the number of words in the website was satisfactory, we decided to opt for this option only and not pursue any alternatives.

#### A.3 Algorithms

As for the algorithms, we were given the list of algorithms beforehand by the project mentor, therefore we did not consider any alternative choices. Since the classification algorithms were already readily available thanks to the scikit learn library, we did not consider any other implementations. However, the implementation of Naive Bayes classifier is available in NLTK as well as the scikit learn library. The one in the NLTK library is more or less the same as the Bernoulli Naive Bayes classifier of the scikit learn library. We considered both options, however, in the end decided to settle for the one in NLTK to have variety in our project.

### B. Detailed description of Solutions/Approaches/Technologies of choice

We tried to automate or at least semi-automate processes wherever possible to save time and energy. For this reason, we gathered a list of several utilities which are described in the next sub-section.

#### B.1 Description of the tools used during data gathering and pre-processing

- Selenium - For gathering the data Selenium framework was used. Usually, Selenium is used for automating the browser tests, however, it we used it for automating the download procedure.
- Terminal utilities (data collection) - Multiple terminal utilities such as curl, jq were used during the data collection process.
- Terminal utilities (data cleaning) - For cleaning the data sed utility and awk language were used. awk domain-specific language used for text processing and data extraction.
- re library - Python's re library allows for the usage of regular expressions on a given string. All the text files were clean using this library.

## B.2 Description of the classification algorithms used

**B.2.1** ***Naive Bayes*** For this project we are using Naive Bayes classifier of the NLTK library. As the name implies Naive Bayes classification algorithm is based on the Bayes' theorem. To find the probability for a label, this algorithm first utilizes the Naive Bayes rule to express $P(label|features)$ in terms of $P(label)$ and $P(features|label)$:

$$P(label|features) = \frac{P(label) * P(features|label)}{P(features)}$$

The algorithm then makes the "naive" assumption that all features are independent and have equal weight. The advantages of the algorithm are that it works well with large data sets and can deal with multi class classification, which is the case in our project, with ease.

**B.2.2** ***Support Vector Machine (SVM)*** Support Vector Machine (SVM) is a supervised learning algorithm that can be used in classification, regression and outliers detection problems. For this project we used the SVM classifier from Python's scikit-learn library (sklearn.svm.SVC). SVC classes is able to conduct binary as well as multi-class classification on a given data set. The foundational idea behind the SVM is to create a hyperplane that divides the given data set into classes. The goal is to find points in the hyperplane such that those points lie closest to the classes. These points are what is known as the support vectors. The next step involves finding the distance between the support vectors and the dividing plane. This distance is referred to as margin. The goal of the SVM slgorithm is to maximize this margin as much as possible. When the maximum margin is found, the hyperplane becomes the optimal hyperplane.

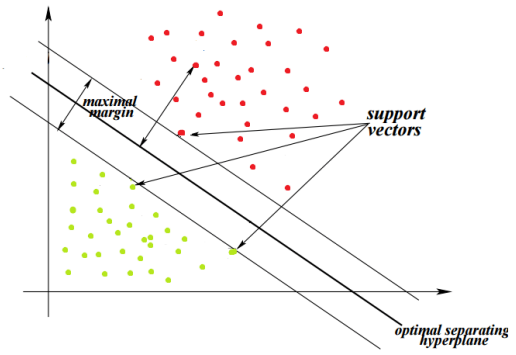


Figure 1: SVM diagram

**B.2.3** ***Random Forest*** For implementing the Random Forest classifier we used skicit-learn library (sklearn.ensemble.RandomForestClassifier) of Python. Random Forest is a supervised learning algorithm. To understand the idea behind the Random Forest it is essential to understand Decision Trees first as Random Forest itself is basically comprised of Decision Trees, hence the word "Forest" in the name. The Decision Tree takes some feature as an input and produces an output based on that feature. The aim of a decision tree is to create a model such that, that model predicts the output of a target variable by learning simple decision rules from the given data features. Furthermore, the decision trees can perform multi-class classification. The complexity of the algorithm is at an logarithmic scale making it an attractive choice.

What Random Forest does is it builds a collection of such Decision Trees from the randomly selected subset of the training data. The model then averages the outcomes of the decision trees to come to a conclusion. Below is a demonstration of how a Random Forest algorithm comes to a decision.

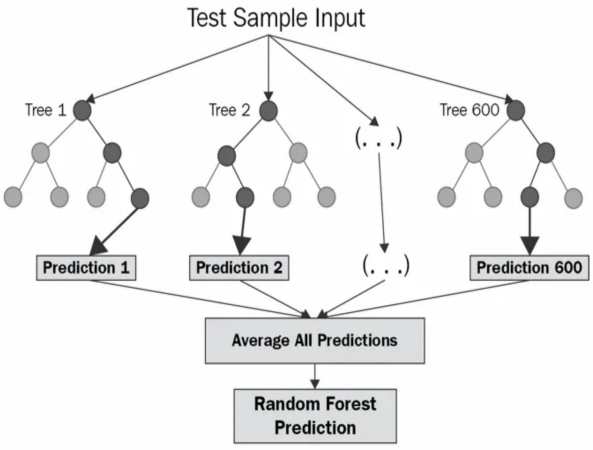


Figure 2: Working principle of Random Forest algorithm

**B.2.4** ***Recurrent Neural Network (LSTM)*** A recurrent neural network (RNN) is the first algorithm that remembers given input with the help of internal memory, which makes RNN applicable for machine learning problems that involve modeling sequential data. Despite the fact that recurrent neural networks were created in the 1980s, his true potential was revealed only recently and the invention of long short-term memory (LSTM), has brought RNNs to the foreground. Because of RNN's

internal memory, they can remember essential information about the input they received, which allows them to be very accurate in predicting what's coming next. That is why these deep learning algorithms are commonly used for ordinal or temporal problems, such as language translation, natural language processing (NLP), speech recognition, and image captioning. They are actively used with applications such as Google Translate, Apple's Siri and voice search. To understand how recurrent neural networks work it is necessary to compare RNN with feed-forward neural networks. In a feed-forward neural network, data only moves from the input layer, through the hidden layers, to the output layer. Through the network, data moves only in one direction and never touches the node twice. That is why feed-forward neural networks have no memory received input, respectively they are bad at predicting what will come next. In an RNN the information cycles through a loop. When it makes a decision, it considers the current input and also what it has learned from the inputs it received previously. The difference is demonstrated in the below illustration.

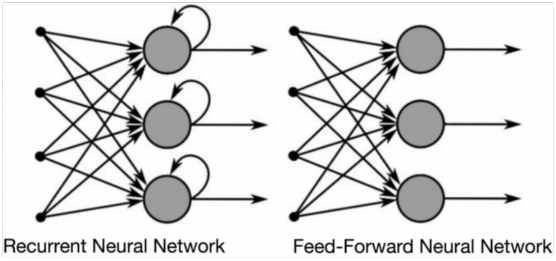


Figure 3: RNN vs Feed-Forward NN

RNN work by taking an input applying some function on it and then output a result. Then, that result is used as an input for the next layer. In a nutshell, the whole process resembles the following structure:

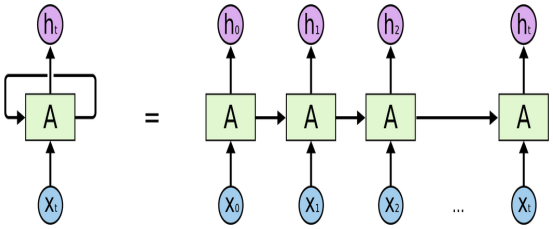


Figure 4: RNN structure

However, a usual RNN has a short-term memory due to a problem known as the vanishing gradient. After each layer, the gradients of the loss function becomes smaller and smaller. Consequently, there comes a point when the gradient is so insignificant that it is absolutely negligible, causing the model to basically not to learn hence, making it less useful for long-term learning tasks. Albeit, this problem can be solved by reducing the number of layers in the neural network, we opted for a different solution. In combination with a long short-term memory (LSTM). LSTM improves upon the shortcomings of the regular RNN by changing the structure of modules of the neural network which enables the model to perform at a high level even with the tasks that require long-term learning. A single LSTM module has 4 neural network layers, unlike regular RNN which has only 1 which is demonstrated below.

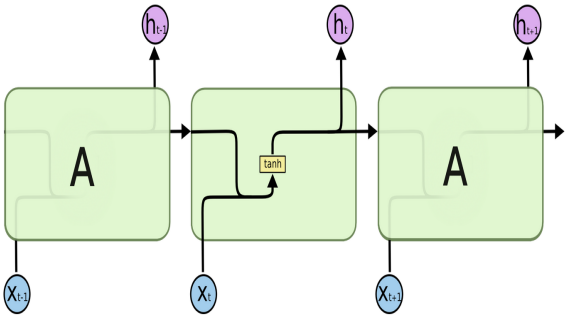


Figure 5: regular RNN module

As it can be seen from the above diagram, the structure of a regular RNN module is pretty simple. The same diagram for LSTM is placed below. As it can be observed, this one is much more complicated as it has 4 layers. These layers are discussed for the rest of this section.

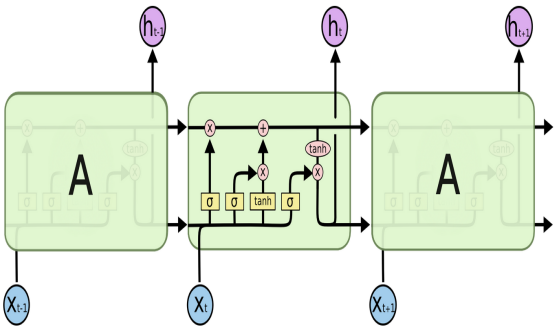


Figure 6: LSTM module

The first is the forget gate layer which is a sigmoid layer. The illustration of this gate is placed below.

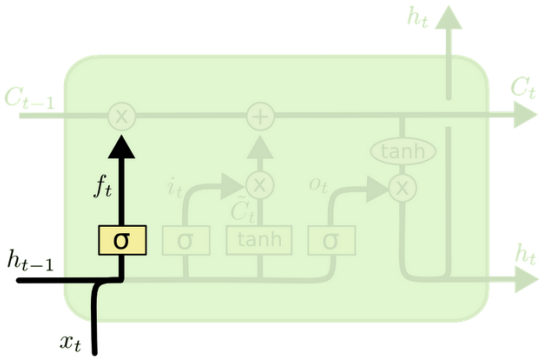


Figure 7: Forget gate layer

It analyzes the input and outputs a number between 0 and 1. 0 means that forget this completely, while 1 is the opposite — absolutely remember this. This layer uses the following function to output a value:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Figure 8: Forget gate function

The next step involves deciding what information will be stored in the cell state. This is done with the help of a sigmoid function known as the input gate layer and a tanh layer. Input gate layer decides which values will be updated, while the tanh function creates candidate values that could be added to the state. This process is demonstrated in the diagram below.

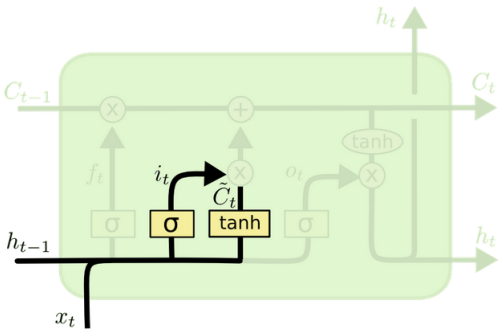


Figure 9: Input gate and the tanh function

After the cell is updated, an output needs to be generated. The cell state passes through a tanh, then is multiplied by the result of a sigmoid function which decides which parts of the cell state should be included in the output. The process is demonstrated below.

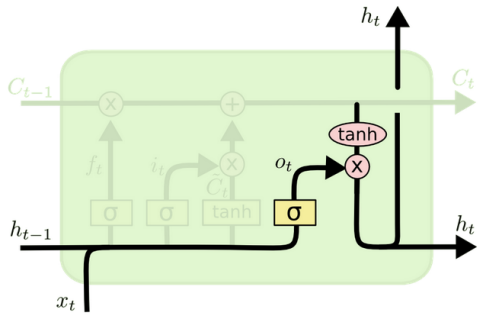


Figure 10: LSTM module producing output

The output is produced using the following formulae placed below. These additions over the regular RNN allow LSTM to retain data in the long-term.

$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Figure 11: LSTM output formulae

### B.3 Technologies used in Website

Technologies used for website: Figma - before creating the website, we needed to have a design prototype of it, and we used Figma for creating an interactive prototype of our website. Bootstrap (v.4.6) - bootstrap is a popular HTML and CSS framework, and we used it for better layout options and website responsiveness. jQuery - jquery is a widely used javascript library that we used to add up some interactivity to our website. Flask - flask is known as a micro web framework of python, and for the back end, we used this technology to integrate the project into our website.

## C. Research Methodology and Techniques

Majority of the research articles were obtained from IEEE Xplore website (https://ieeexplore.ieee.org/Xplore/home.jsp). Altough mainly text classification and text categorization papers were explored, we also consulted papers regarding sentiment analysis as well since the topics are closely related. The keywords used during the research include "Naive Bayes", "Neural Networks", "Support Vector Machine", "SVM", "Text Classification", "Text Categorization", "Sentiment Analysis"

D. Social impact

We do expect our project to have a social impact to a certain extent. First of all, because we will have an easy to use website, the tool will be accessible to anyone across the globe. Secondly, people who will need an access to a large data set in Azerbaijani language will be able to access ours with an ease. Finally, the other tools, such as the lemmatizer, which have been the byproducts of the text classification project can be used in isolation or integrated into other projects as well. Furthermore, this paper itself can be of a guidance to the people who decide to work on a similar project in the future as well.

## V. Implementation

A. Essential Components of the Project

There are several essential parts to the project. They are the followings:

1. Data set: Sentences, words, stopwords used in the project to train the model and create lemmatization tool
2. Lemmatizer: To make the algorithms function better a rule-based lemmatization tool was developed from the ground up
3. Text classification algorithms
   (a) Naive Bayes implementation: Text classification algorithm based on Naive Bayes algorithm
   (b) Support Vector Machine (SVM) implementation: Algorithm that uses classification algorithms for two-group classification problems
   (c) Random Forest implementation: A collection of decision trees that output average decision value
   (d) Recurrent Neural Net implementation: A type of neural networks which allows prior outputs to be used as inputs while having hidden states
4. Website: To make the created technology accessible to everyone

A.1 Data set

The data set used for the classifier was the text from the books in the Azerbaijani language. The entire data set consist of 3060 books, 6256757 sentences, around 77994551 words of which 1883105 are unique. There are 22 categories in the data set:

1. Architecture
2. Agriculture
3. Art
4. Culture
5. Communication and Informatics
6. Ecology
7. Education
8. Finance
9. Healthcare
10. History
11. Journalism
12. Law
13. Literature
14. Military
15. Philosophy
16. Politics
17. Programming
18. Psychology
19. Sociology
20. Sport
21. Religion
22. Tourism

**A.1.1** ***Data collection*** The process of data collection was mostly automated using Selenium library, Bash scripting and several Terminal utilities. The PDF and EPUB files used were collected from 5 various sources — https://ebooks.az, https://bookleaks.org, https://az.bookmate.com, http://www.e-derslik.edu.az and miscellaneous sources. The vast majority of the books were collected mainly from the Presidential Library website. The words for the dictionary file were gathered from https://www.obastan.com. Although data was collected throughout the entirety of the project lifetime, the majority of it was gathered in the first 2 days. Furthermore, some data was collected manually. The list of the inflectional as well as the derivational suffixes were collected and categorized in a manual fashion. Moreover, the website https://bookleaks.org did not provide any download options and contacting the site owner did not lead to anything, therefore, the data for about 100 books had to be collected manually from the website. For collecting the data from https://az.bookmate.com a custom devised Bash script and a Python code were used.

**A.1.2** ***Statistics*** Some of the statistics types were instructed to be extracted by the mentor, while some were extracted out of the will of the project team. The statistics of the final data set are described in the below attachments.

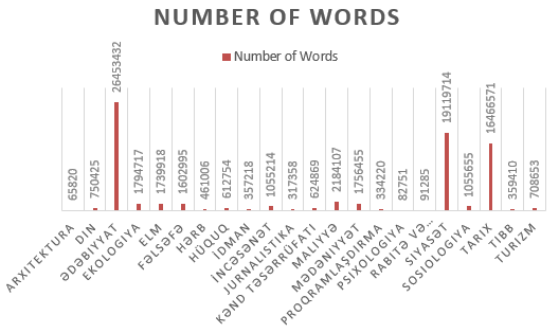


Figure 12: The number of words from each category

As it can be seen from the above picture, the majority of the words come from the categories of literature, politics and history.

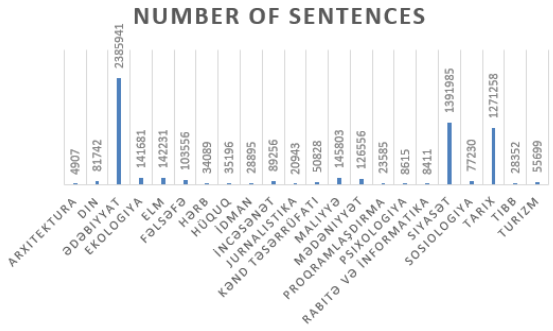


Figure 13: The number of sentences from each category

Consequently, these three categories contain the vast majority of the sentences as well which can be observed from the above picture.

Furthermore, we also wanted to know how the unique word distribution is among the different categories. Below are the results.

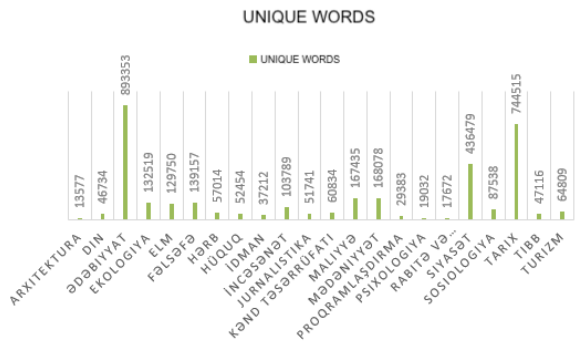


Figure 14: Unique word count for each category

It is interesting that, although the category politics has more words and sentences in contrast to the category history, the latter surpasses the former when it comes to the amount of unique words.

Finally, we also extracted statistics for the most frequently appearing adjective or noun from each category and the findings are placed below.

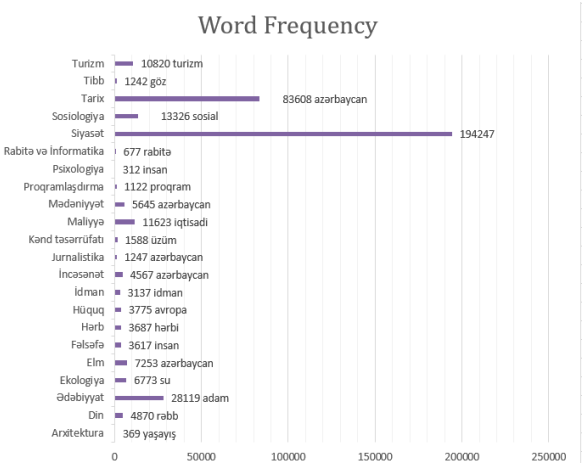


Figure 15: The number of unique words for each category

Our findings demonstrate that the word "Azərbaycan" was the most frequently appearing word in multiple categories.

**A.1.3** ***Data pre-processing*** The initial corpora contained a lot of irregularities. To mitigate the amount of such situations several techniques were used to make the data as immaculate as possible.

1. Categorization: Although around 2300 books were categorized during the data collection stage, around 790 books were uncategorized. Therefore, those books had to be categorized manually.
2. Data cleaning: Data cleaning process was automated for the most part. PDF files were converted to plain text file using pdftotext tool by Poppler, a PDF rendering library. Conversion of the EPUB files was conducted via epub2txt2 tool by Kevin Boone on GitHub (https://github.com/kevinboone/epub2txt2). A marginal amount of data cleaning was done manually. This was mostly the case in the earlier phases of the data cleaning stage and was done to spot the common patterns among the irregularities in the data. Overall, the data cleaning process can be divided into 4 stages:

3. Removal: Faulty and corrupted conversions were detected and removed from the text corpus. To clean the data Python, Perl, sed and tr tools were used. Numerous regex patterns were constructed to remove the faulty data.
4. Lowercasing: To make the data model consistent and prevent the same words with different cases being processed as different words, the data was converted to lowercase. Non-alphanumeric character removal: The characters that were not alphanumeric were removed from the data set using regular expressions
5. Lemmatization: To prevent the same words with different inflectional suffixes being processed as different, each word was lemmatized.
6. Stopword removal: There are some words in Azerbaijani that are common for every sentence regardless of category. A list containing around 430 such words was compiled to prevent them being fed into the classifier and damage the final result.

## A.2 Lemmatizer

Lemmatization is the technique of using of vocabulary and morphological analysis of words, to remove the inflectional suffixes from the given word and only return the base form of a word also known as the normalized form of a word. This base form is referred to as the lemma. To make matters clearer, consider the words "görür", "gördü" and "gördüm". These are the same words but in a different form due to the inflectional suffixes of "ür", "ü" and "üm". Lemmatization is the technique of removing these and other inflectional suffixes to obtain a normalized form of "gör". To achieve this task, first the list of inflectional suffixes in Azerbaijani were compiled. There are certain exceptional derivational suffixes that affect the process of lemmatization. To account for those cases, the collection of those suffixes was assembled. To obtain the most accurate result during the lemmatization process, the input text and words in the dictionary were all lowercased. To further better the algorithm, a noise removal tool was created to remove all characters except letters and numbers from a given word. Finally, a stop word removal tool formulated to remove words that did not borne meaning specific to a category. Initial version of the list of such words was taken from Natural Language Toolkit (NLTK) library, however, many additions were made to improve the quality and quantity of the words. In the end the list contained 184 stop words.

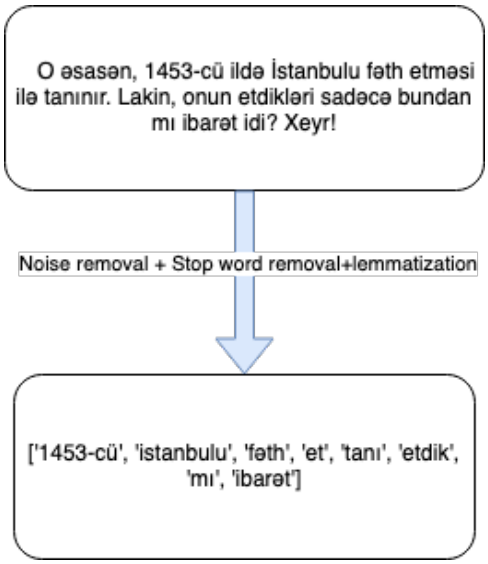


Figure 16: Lemmatization tool in action

The above picture demonstrates the end result achieved by the created lemmatization tool. The rest of this section will discuss the creation process of the tool in detail.

**A.2.1** ***Letter conversion issue*** An issue faced with this task was that Python's built-in `lower()` function does not work well with letters "I" (uppercase ı) and "i" of the Azerbaijani language. This irregularity is demonstrated in the below picture.

```
>>> "i".upper()
'I'
```

Figure 17: Default behavior

Due to this faulty conversion, we came up with our own letter map devised for the Azerbaijani language to correct the issue. The result is presented below.

```
>>> "i".translate(upper_map)
'İ'
```

Figure 18: Using custom letter-map

Similar behaviour was present with the letter "i" as well. The behaviors of default `upper()` function and our letter map is illustrated in the below pictures.

```
>>> "i".upper()
'I'
```

Figure 19: Default behavior

```
>>> "i".translate(upper_map)
'İ'
```

Figure 20: Using custom letter-map behavior

**A.2.2** ***Noise removal*** After solving the letter conversion issue, we created a Python program to remove noise from a given string. In this context noise is referred to any character that is not alphanumeric that is neither a number nor a letter. However, it was configured that only characters that appeared in front or end of the word would be removed. For example the word "zirzibil" would stay intact, however, in the word "-Xeyr" the dash symbol would be removed. An example of the usage is shown below.

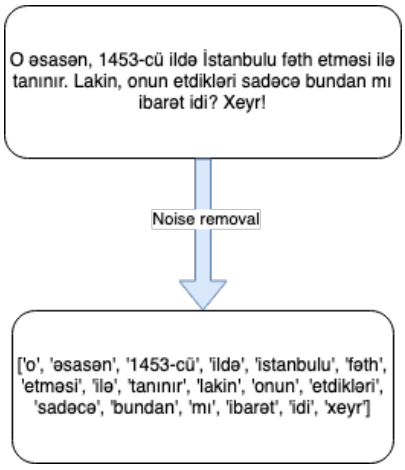


Figure 21: Noise removal tool in action

**A.2.3** ***Stop word removal*** The penultimate stage in the lemmatization was removing the stop words from a given sentence. Stop words are the words that although carry a meaning in a sentence and appear a lot, do not necessarily belong to any category which is detrimental to the text classification algorithms. Therefore, a list of 184 stop words of which some were taken from the NLTK library was created to be used in the project. The results are illustrated in the below image.

The list of the stopwords are show in in the Appendix A.

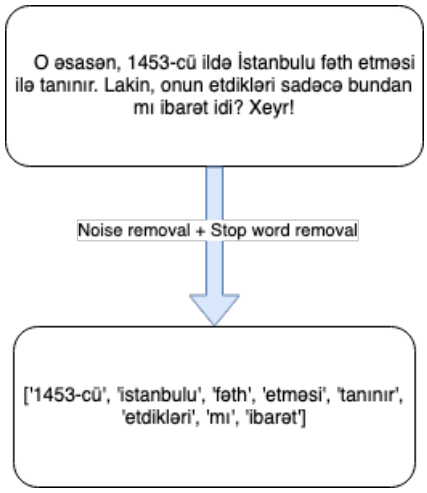


Figure 22: Noise removal and stop word removal tools in action

**A.2.4** ***Lemmatization tool*** The final piece of the "Lemmatizer" was the actual lemmatization tool. The lemmatization tool takes an array of words as an input then removes each inflectional suffix from the word in an iterative manner. It takes a suffix from the lists of inflectional suffixes checks if the word ends with that suffix, if yes it removes and continues the process recursively. If no, then it moves on to the next suffix. If the word ended with none of the suffixes from the list, then the word itself is considered to be the lemma. Obviously, this tool is used in combination with the noise and stop word removal tools to achieve the maximum effectiveness. The final result of using these three tools together is shown below.

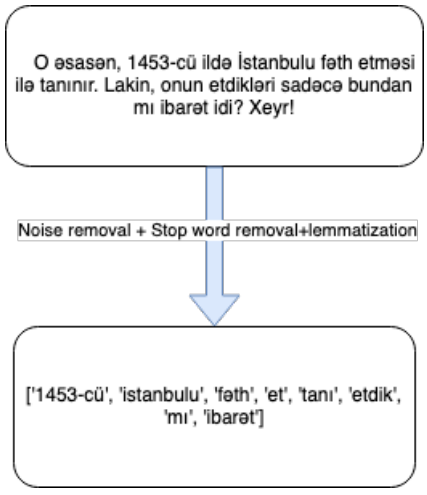


Figure 23: Lemmatization tool in action

As it can be seen from the above diagram, the

sentence is converted into an array form, lowercased, stripped from unnecessary characters, have its stop words removed and its words are transformed to their base form only. The resulting words are then fed into the classification algorithms.

### A.3 Text classification algorithms

Four text classification algorithms were used to achieve the project goal — Naive Bayes, Support Vector Machine (SVM), Random Forest, Recurrent Neural Networks (RNN). Rest of this section will talk about their implementation in detail.

**A.3.1** ***Naive Bayes*** The Naive Bayes classifier was implemented using the NLTK library. Each category is divided to a separate plain-text files. The naming convention used for the files is *category_name*.txt. Sentences from each text file is extracted, cleaned, lemmatized, its stop words are removed and finally they are divided into tokens. Then, each token is labelled with a certain category. Then, the labelled tokens are combined into a unified data set which itself is shuffled. 70% of the data set is used for training the model while the remaining 30% is left for the testing. When training is done, the model outputs most informative 20 features from each category as well as the accuracy score. Finally, the training result is pickled to a file. In the end the accuracy score was about 75%.

**A.3.2** ***Support Vector Machine (SVM)*** The Support Vector Machine classifier is implemented using the scikit learn library. First the data is read from the .csv file with Tab (\t) as the delimiter. In the .csv file the first column describes the sentence while the second one the category of the sentence appearing in the first column. The extracted sentences are first cleaned, lemmatized, cleansed of stop words. Then, the 70% of the data is divided for testing and 30% for training. Further, the label data is encoded from string form into a numerical form. Then, the using the TF-IDF vectorizer, the data is converted to vectors. The idea behind the TF-IDF is the following. Term Frequency summarizes how often a given word appears within a document, while Inverse Document Frequency down scales words that appear a lot across documents. In other words, TF-IDF tries to find words that appear a lot in a given file, but less frequently so across the files. Additionally, we tested several SVM kernel types, however, decided to settle with Linear kernel because we obtained the most accuracy from it. In linear kernel the decision boundary is a straight line (or a hyperplane in higher dimensions). We ended up with an accuracy score of about 69%.

**A.3.3** ***Random Forest*** The implementation of Random Forest was identical to that of SVM. Here we also used the same .csv file and vectorizer (TF-IDF). After several trials, we decided to set the number of estimators (the number of decision trees to be used in the estimation process) to 200. The final accuracy score was around 60%.

**A.3.4** ***Recurrent Neural Network (RNN)*** Due to the RNN's vanishing gradient problem we implemented LSTM version of it. We defined the epoch size to be 10 and batch size 64. In one epoch the entire data set passes forward and backward through the neural network only once. The model was reading the data from a .csv file.

80% of the data was alloted for training while the remaning 20% was left for testing. Tokenizer class from keras was used to convert the given textual data into numeric form. We set the dropout rate to 50%. Dropout prevents the model from overfitting. The learning rate was set to 0.001 and Adam optimizer was used. Optimizers are needed to control the attributes such as weights and learning rate of a given neuural network to reduce losses. Finally, the number of epochs was set to 10.

### A.4 Website

The project we worked on needs to reach out to the general audience. We see behind the scenes - just a bunch of code and data that we know how to put together and make the system work. However, for general usage, people need an interface through which they can easily benefit the system. We came up with a user-friendly, fully responsive website with a colorful interface that briefly gives information about the project itself, authors, resources, and mentors who helped a lot throughout the project's lifecycle for usage and presentation purposes. On the website, the functionality of the analysis tools is relatively straightforward: a user needs to insert a text with some word limitation into the input box, decide on the analysis tool (text classification or lemmatization), and if the choice of analysis tools is text classification, then choose the desired algorithm. For all four (naive bayes, neural networks, random forest, support vector machine (SVM)) algorithms that are given as an option to a user, the input and output of the tool are the same, whereas their working principle is different and has a different

level of accuracy. If the tool choice is text classification, the output will be just one word - category name of the inserted text. In the lemmatization tool case, the result will be the exact inserted text that contains words with just base (suffixes are removed). One of the superior sides of the website is that it is available in English as well as the Azerbaijani language. Since the principal purpose behind the project is to improve the lexical and morphological analysis system in Azerbaijani and the audience who will benefit from it will be Azerbaijanis, the idea of making the website available in both languages made much sense to us. Talking about the technical stuff about the website, we used HTML5 and CSS3, also the bootstrap (v.4.6) framework for easy layout and responsiveness for the front-end side of it. For the functionality of the tools - backend, we used one of the essential python frameworks, Flask. We were willing to use the framework Flask because it provides simplicity, flexibility, fast debugging, and leaves the freedom of implementation and architecture to the developer. Also, Flask is one of the latest frameworks, and with it, things get easier and faster; it is proven that this framework is the best for small web projects.

### A.5 Timeline and Gantt chart

Since the project consisted of two phases, researching and implementing, we came up with two Gannt charts which are described in the next page.

The project lifecycle is indicated in the timeline as a Gantt chart. Firstly, we did our best to choose the most compatible and needy topic to fulfill it properly. What we had in mind was something local and national; since our country is in the phase of improvements in IT, the related IT projects are essential for us as a nation and country. Therefore, we decided to deal with our language and researched the area to know how and where to start. The research phase lasts in 3 weeks because we had problems with the resources; they were mostly about other various languages. We needed to figure out the project, work principle, tools and methods to use, and so on so forth. After having enough information, with the guide of our project mentors, we spent about one week to know which technologies (e.g., NLP) to use for the project. Some research and understanding of these technologies were also essential; we spent a reasonable amount of time reading papers, researching the indicated techs, discussing them with our mentors. What we needed later is a massive amount of data; this was the challenging part. It was easy to find any amount of data from the internet in English, but we needed particular sources in Azerbaijani. Although some of them were provided by our mentors (presidential library), we found additional sources(ebooks.az, bookleaks.org, bookmate.com) that we can use in the phase of data collection. We needed a kick-off meeting before starting the data collection because it would be a very long process, and some job distribution was required. The data collection phase was the never-ending part of the project; we spent almost 12 weeks on it. This phase contains finding, downloading books, parsing, and pre-processing them according to the categories. We automated the process; using scripts written in Python, all the process could be done with just a couple of compiling. Issues we confronted with here were some books were ruined by unknown signs while parsing. Some of them cannot be cleaned with just automated scripts because exceptional cases appeared, and we needed manual work to do on the data cleaning. For the other feature of our project (lemmatization), we needed data from dictionaries, and the process was straightforward; it lasted one month, and after some data collection for clean sentences, we started this process and did it along with sentence cleaning, otherwise we could not have managed our time. We ended our SDP1 with data cleaning and sufficient research on the methods, tools, techs, etc. Since we already had considerable data in our hands, we began to implement the algorithms needed for the project starting from the beginning of SDP2. The algorithm implementation part took almost two months, and the main reason behind that was data being exceptionally huge. It was tough to train the amount of data that the process was extending; therefore, we worked on training and analyzing data throughout the implementation of algorithms. While implementing the algorithms, since we were close to the end, we started to work on the website for usage and presentation purposes. We created a site map for the website to know what information we will present. Along with creating the website prototype, we started to test our system on the actual data. We constantly meet, discuss and analyze the final product the result with our mentors during that period. The last step was developing the website and integrating our project into it to let the general audience be able to use tools efficiently.

### A.6 Testing and Validation of results

**A.6.1 Data set** Since the data was collected across many sources, to make them consistent we developed several regular expressions to replace

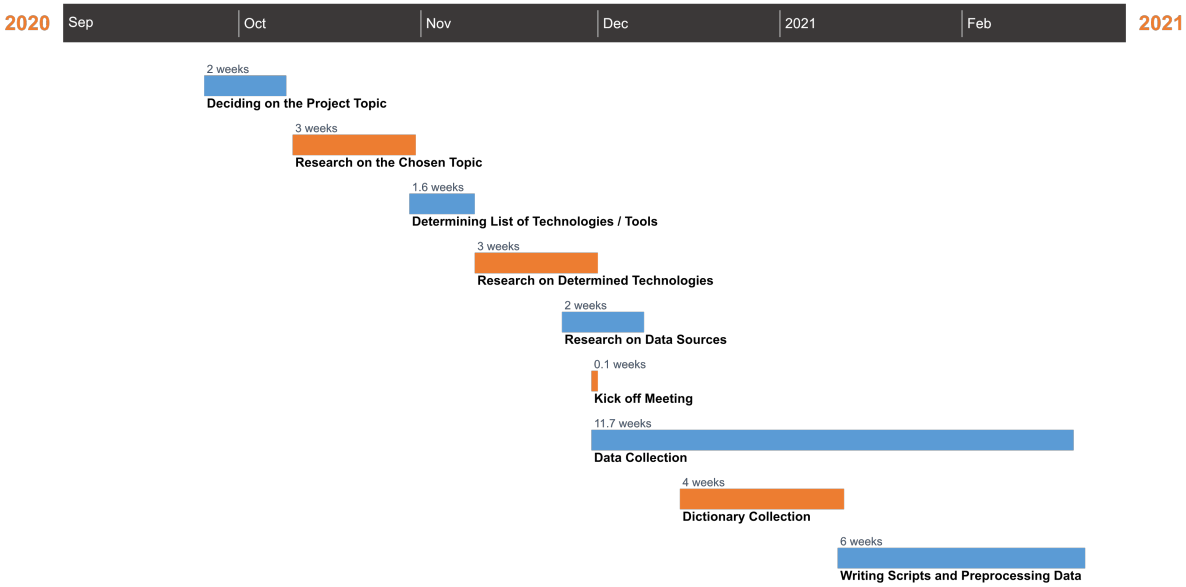


Figure 24: Gannt chart for SDP I

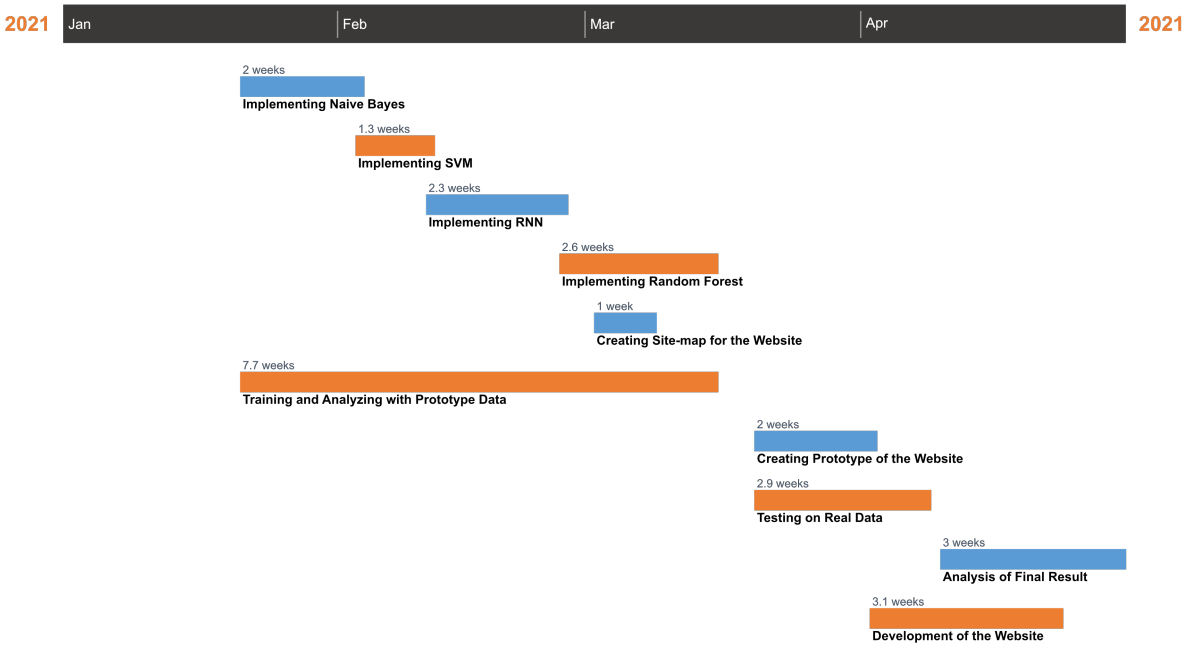


Figure 25: Gannt chart for SDP II

or remove certain characters. Furthermore, we filtered out all the books that were in French, English, Turkish, Russian and Cyrillic Azerbaijani. In the end we ended up removing about 300 books from our database before arriving to the total of about 3090 books.

**A.6.2 Classification Algorithms** Initially, we trained and tested our data on a 2017 model Dell Inspiron 15-7579 machine with Manjaro OS (KDE Plasma 5.19.3). The machine had 2.70 GHz, 2 core Intel i7-7500U CPU with 4 threads and 12 GB DDR4 RAM. For the initial testing we used 2000 sentences from each category (44000 sentences in total). To ensure the validity of the results the tests were conducted multiple times and our results varied by around 1%. After ensuring that the algorithms were working, we loaded our entire database. However, the problem we encountered was that the machine was not powerful enough to process the data in a sensible amount of time for all algorithms and it did not have the required RAM capacity for Random Forest, SVM and RNN algorithms. Therefore, we opted for migrating our data to Google Compute Cloud as they offered a credit of $300 for new users. We created 4 VMs of N2D series. One VM had 32GB RAM, while the other 3 were equipped with 64GB. The disk size was 32 GB for all VMs and they all had Debian as the OS. Furthermore, each VM had 8 vCPU because 8 was the default vCPU quota assigned by Google.

Once the VMs were ready and data transfer completed, we started the Python codes on each VM and let them run. Naive Bayes completed in just 2 hours. We re-ran the test to ensure that it was not due to some bug and we more or less got the same training time. The final result showed 60% of accuracy. Random Forest algorithm completed after approximately 5 hours of training and outputted 56% of accuracy. RNN with LSTM training which was configured with 10 epochs and 64 layers lasted for about 9.5 hours before printing a 67% accuracy result. Finally, Support Vector Machine algorithm took about 10 hours to complete and had 58% of accuracy score.

## VI. Conclusion

### A. Discussion of results

When we trained the data on a small sample where each category had 2000 sentences, the accuracy score we got from each algorithm, except for Random Forest which stayed the same, was higher than when we used the final data set. We believe that the main reason behind this is the disparity between the amount of data from every category. Some categories, for example architecture have "only" around 4900 sentences which pales in comparison to the categories such as history or literature which have over 1.2m. and 2.3m sentences respectively.

### B. Future work

Even though we spent a reasonable amount of time on data collection and pre-processing, some points for future work should be stressed out: new words could be inserted into Azerbaijani language vocabulary, the spelling of words could change, and dialects should be taken into consideration. It would be nice to work on data sources and make the project more flexible, considering future changes/additions for vocabulary: this could be done with another field of machine learning - word prediction and embedding techniques. Furthermore, usage of synonyms of words could be subject to maximize text classification. Also that would be interesting to see new categories and subcategories (for particular fields) added to the project to widen the usage areas. While all categories are at a good state, we deducted from our data analyses that the category of healthcare could be improved in terms of content. Currently, the data in the category mostly focuses on ophthalmology. Aside from data sources, the accuracy levels of algorithms that we have used to get results could also be improved. If we increase the number of epochs and layers, we probably would have seen improvements in inaccuracy. However, due to lack of time, we only could reach the maximum 67% accuracy level with the Recurrent Neural Network algorithm. While LSTM provided fairly good amount of accuracy of 67%, the other algorithms were equal or below 60%. We believe that this accuracy could be improved significantly if some categories of texts were to be merged. For example, categories such as architecture and culture talk about more or less the same field. Similarly, categories of military and history have close resemblance to one another too. Merging these categories with one another could prove to have several benefits. Firstly, the number of categories that algorithms have to choose from would decrease. Secondly, the categories that have limited amount of data to train from would be eliminated. However, it should be bore in mind that in doing so, the categories would end up containing slightly unrelated sentences.

References

[1] H. L. M. Lan, S.Y. Sung and C. Tan, "A comparative study on term weighting schemes for text categorization," 2005.

[2] M. Sauban and B. Pfahringer, "Text categorization using document profiling," 2003.

[3] E. Leopold and J. Kingermann, "Text categorization with support vector machines: How to represent text in input space?" 2003.

[4] F. S. F. Debole, "Supervised Term Weighting for Automated Text Categorization," 2002.

[5] R. Yasotha and E. Y. A. Charles, "Automated text document categorization," 2015.

[6] C. Jian-fang and W. Hong-bin, "Text categorization algorithms representations based on inductive learning," 2010.

[7] U. R. Erlin and Rahmiati, "Text message categorization of collaborative learning skills in online discussion using support vector +machine," 2013.

[8] C.-H. Lee and H.-C. Yang, "Text mining of multilingual corpora via computing semantic relatedness," 2002.

[9] "Reuters-21578 collection." [Online]. Available: http://www.research.att.com/~lewis/reuters21578.htm

[10] L. Lei and G. Qiao, "Text categorization using SVM with exponent weighted ACO," 2012.

[11] T. M. Lengyel, "ICT as an education support system quantitative content analysis based on articles published in EMI," 2013.

[12] M. Thangaraj and M. Sivakami, "Text classification techniques," 2018.

[13] R. T. Michal Tomana and K. Jezek, "Influence of Word Normalization on Text Classification," 2006.

[14] E. Frank1 and R. R. Bouckaert, "Naive Bayes for Text Classification with Unbalanced Classes," 2006.

[15] L. Ge and T.-S. Moh, "Improving Text Classification with Word Embedding," 2017.

[16] C. Olah, "Understanding LSTM Networks," 2015. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[17] "Presidential Library of Azerbaijan." [Online]. Available: https://ebooks.az/

[18] "Bookleaks." [Online]. Available: https://bookleaks.org/az

[19] "Bookmate." [Online]. Available: https://az.bookmate.com/

[20] "E-Derslik." [Online]. Available: https://www.e-derslik.edu.az

[1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20]

## VII. Appendices

Appendix A - List of stop words

1. əfsus
2. əgər
3. əlavə
4. əlbəttə
5. əlqərəz
6. ən
7. əsl
8. əsla
9. əsəsan
10. əvvəl
11. əvvəla
12. şübhəsiz
13. şəksiz
14. a
15. ah
16. aid
17. amma
18. ancaq
19. artıq
20. axı
21. axır
22. ay
23. ay
24. ay
25. aycan-aycan
26. barı
27. barədə
28. bax
29. başqa
30. belə

31. beləliklə
32. bil
33. bir
34. bircə
35. bizcə
36. bizə
37. buna
38. bəh-bəh
39. bəli
40. bəlkə
41. bəri
42. bəs
43. bəyəm
44. ca
45. can
46. cümlədən
47. cə
48. cən
49. da
50. daha
51. dair
52. demə
53. deyilənə
54. deyəsən
55. di
56. doğru
57. doğrudan
58. doğrudur
59. doğrusu
60. düz
61. düzdür
62. düzü
63. də
64. dək
65. eh
66. ehey
67. ehtimal
68. elə
69. ey
70. fəqət
71. gah
72. gör
73. görsənə
74. görünür
75. görə
76. görək
77. görəsən
78. güman
79. güya
80. gəl
81. gəlin
82. gəlsənə
83. gərək
84. ha
85. habelə
86. halbuki
87. halda
88. haqda
89. hey
90. heyif
91. heç
92. hə
93. həm
94. həmçinin
95. həqiqətən
96. hərgah
97. hərçənd
98. hətta
99. ilə
100. indi
101. istər
102. istərsə
103. isə
104. kaş
105. ki
106. kimi
107. lakin
108. lap
109. madam
110. mi
111. müxtəsər
112. məgər
113. məhz
114. məncə
115. mənə
116. məsələn

117. məxsus
118. naminə
119. necə
120. nə
121. nəhayət
122. nəinki
123. o
124. of
125. olduqca
126. olmaya
127. olsun
128. ona
129. ondan
130. onun
131. ox
132. oxqay
133. oy
134. paho
135. qabaq
136. qarşı
137. qeyri
138. qoy
139. qısası
140. qədər
141. qərəz
142. sana
143. sanki
144. sanız
145. sarı
146. savayı
147. sonra
148. sözsüz
149. sıradan
150. səncə
151. səniz
152. sənə
153. tfu
154. təbii
155. tək
156. təkcə
157. təki
158. tərəf
159. təəssüf
160. uf
161. ura
162. uğrunda
163. vay-vay
164. və
165. xa-xa-xa
166. xas
167. xeyr
168. xox
169. xux
170. xülasə
171. ya
172. yalnız
173. yaxud
174. yaxşı
175. yox
176. yoxsa
177. yəni
178. zənnimcə
179. çünki
180. ötrü
181. özgə
182. ümumiyyətlə
183. üstündə
184. üçün

## **VIII.** Acknowledgement

We would like to express our deep and sincere gratitude to our research supervisors Dr. Abzatdin Adamov and Dr. Samir Rustamov who are the professors of School of Information Technologies and Engineering at ADA University for their great contributions and invaluable guidance throughout the SDP project on the topic "Development of text classification for Azerbaijani Language". The dynamism, vision, sincerity and motivation have deeply inspired all of us, as a group member. The Professors taught us the methodology and techniques to carry out the research and to present the work as clearly as possible. It was a great privilege and honor to work and study under their guidance. In any meeting, their constructive suggestions pushed us to sharpen our analysis and elevate our work to a higher degree. Mentors have presented us with the resources we needed to make the best decisions and finish the project successfully. We would also like to express our appreciation and heartfelt thanks to the anonymous reviewers whose insightful feedback aided in the creation of this final draft.