School of IT and Engineering

Senior Design Project

FINAL REPORT

Development of Computer System for Lexical and Morphological Analysis of Azerbaijani

Authors:

1.	Jalal Rasulzada	BSCE
2.	Nicat Mursali	BSCS
3.	Fidan Aliyeva	BSIT
4.	Nergiz Huseynova	BSCS

Project Advisor:

Samir Rustamov PhD.

Baku, April 2019

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Explanation
HTTP	HyperText Transfer Protocol
CSS	Cascading Style Sheets
RDBMS	Relational Database Management System
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
POS	Part of Speech
NEWSDB	Cleaned data taken from more than 30 thousand news
DBW	Words combined and labeled from Obastan and orthographical dictionary

Abstract

The report describes the building process and development phases of our project. During the lifetime of our project fundamental objective that we had to complete was conducting a deep research and study of complex and rich Azerbaijani language. The most significant component of our project was the precise and clean vocabulary data, which contains all the initial word forms that exist in our language. This was the first challenge that we have faced, as there are very poor available sources in our language, where we could obtain this information. That is why we had to clean, modify and transform the obtained data into the ready-to-use form. Eventually our gathered data contained 77.740 words excluding names (which technically do, but logically does not take most of the grammatical suffixes) from different sources and our program generates 27.799.420 different word forms with all the possible grammatical suffix combinations. We have investigated lots of different algorithms including Machine Learning; however, with our resources the most suitable algorithm was Rule-based Algorithm. As Azerbaijani language is based on complex and structured rules, rule-based algorithm was the most efficient and easy to apply for us. The next step of implementation of our algorithm was making the statistical analysis of Azerbaijani words by dividing them into POS and checking which POS are used more. Due to having lack of resources, we acquired the data from 85,000 Azerbaijani news and made data cleaning which brought us more than 3,500,000 sentences. After acquiring the data, we made statistical analysis about the implementation of words and how many of them we have in our dataset which doesn't include several types of words such as names. With advice of our faculty members, we have compared our software with last year Senior Design Project, which almost performs the same tasks with our algorithm, but with different approach. The advantage of our algorithm over the other one was working without any errors and gave the better result. The outcome of our project is a web-interface where the users can enter the words and texts and then check the stemmed version of text and analysis of the words. The user can also pick an algorithm for checking stemming algorithms that has been developed in English language.

Introduction

Definition

Our project mainly covers two proximately cognate subdomains in the field of computational linguists. The first and the most consequential one is lexical analysis which is the first sequence of the compiler. The main conception of this analysis is that it takes the sentence (also word) and breaks it into tokens (that's why this is called tokenization) by just simply deleting any comment in the code itself. Tokens are strings with an assigned meaning thus making it more fathomable for computers. Such programs that analyze given input lexically are conventionally termed a lexer, tokenizer, or scanner. Morphological analysis, on the other hand, is the study of word forms, how in a certain language words are composed (derived or inflected), and how those words are cognate to other words in that language. It analyses the structure of words and components of words, such as stems, root words, prefixes, and suffixes. One of the main resources in such analysis is morphological dictionary which contains correspondences between surface form (found in any text) and lexical forms lemma followed by grammatical information (for example, the component of verbalization, gender and number). Different methodologies for morphological analysis have been studied and after deep analysis of each of these methodologies, Rule-Predicated methodology has been considered to be the most decent for implementation.

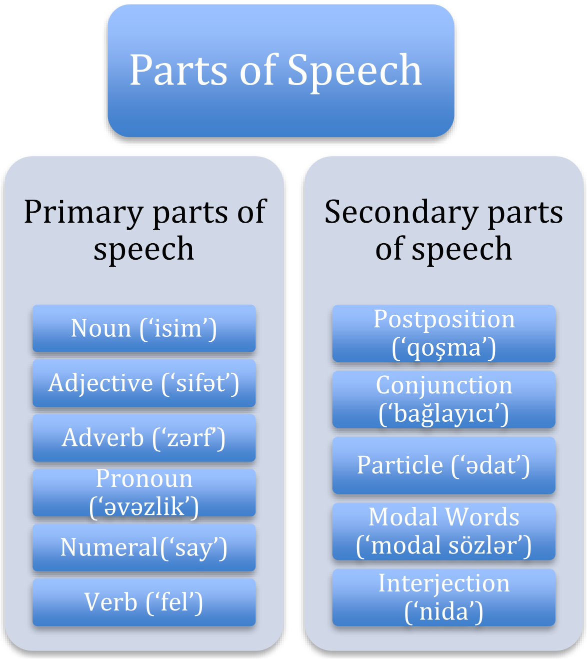
One of the main components of linguistics is grammar which learns the way words can transform in different forms and differentiates how the words can be composed with their meaning. Generally, grammar is considered to be the structure and system of a language and consist of two main components: morphology and syntax. The word morphology denotes (in Greek, morfos designates form and logos designates inculcation and science) discovering and developing of words by their forms/ shapes. Fundamentally, in morphology is the study of parts of speech, the structure of them and the rules of how parts of speech can transmute.

In fact, in linguistics, the words are broken into several components according to their grammatical meanings. For instance, home, book, pen, and school define the names of objects, things; consequently, these kinds of words are grouped by one of the parts of speech called noun. Another part of speech is adjective that combines words that describe the qualities or states of nouns and things such as tall, fascinating, thick, blue and big are some examples of adjectives. In addition, in morphology, there are action words in a sentence that describe what the subject is doing, and where and when this action happens is described by the part of speech named adverb etc.

In Azerbaijan language, parts of speech are divided into two parts considering their important indications: primary parts of speech and secondary part of speech. Primary parts of speech have 2 main features:

- They have independent lexical meaning and indicate the name, quantity, quality, action and etc. of objects.
- They own a role as of part of sentence.

On the other hand, secondary parts of speech have none of above qualities and they only have grammatical meaning. In Table 1 you can see these groups of parts of speech:



Parts of Speech

Primary parts of speech	Secondary parts of speech
Noun ('isim')	Postposition ('qoşma')
Adjective ('sifət')	Conjunction ('bağlayıcı')
Adverb ('zərf')	Particle ('ədat')
Pronoun ('əvəzlik')	Modal Words ('modal sözlər')
Numeral('say')	Interjection ('nida')
Verb ('fel')	

Table 1. Parts-of-speech in Azerbaijani language

Moreover, in Azerbaijani language, there are suffixes (joining to the end of the word) and prefixes (joining to the beginning of the word); however, in the comparison to suffixes, prefixes are very few and there are no grammatic prefixes at all. Generally, suffixes are divided into three main categories according to its features: static suffixes (-m, -n etc.), suffixes that can be added in two different forms (for ex; -a^2 it can be either 'a' or 'ə'), suffixes that can be added in 4 different forms (for ex: '-ı^4' it can be either ı, i, u, or ü). The change of depends on the characteristics of the last vowel of the given word. In our language, vowels can be bold, thin, open and closed. According to the harmony of vowels in a word, the appropriate suffixes are added.

Another important factor of suffixes is that there exists specific rule of addition of a 'connecting letter' between the initial form of the word and suffix. Mostly, this specific consonant is '-n' '-m', but of course, in our language there are several exceptions that you are going to get familiar in the following paragraphs.

In addition, syllable is one of the necessary concepts in morphological and lexical analysis system. In Azerbaijani language the number of the syllables are equal to the number of vowels. The use of this rule in implementation of division of syllables in the given word will be discussed in following the paragraphs as well.

Another essential part of the analysis is the stemming which define the stem of the given word. In the Azerbaijani language, the grammatical suffixes are not considered in the stem; only lexical suffixes are included in it.

Purpose

The reason why we have chosen this project is that the language is one of the most important attributes of a country and for every citizen, the language is very valuable, and the development of our national language analysis system is our duty as citizens of Azerbaijani Republic. If we have a look through history, we can observe that Azerbaijan alphabet was accepted in 1991, which means it is not a very long period and our language is spoken only in Azerbaijan. Therefore, so far, this kind of language analysis system has not been created and did not have chance to be developed and advanced by several nations like English language. That is why we as a team decided to work on this kind of project to express our respect and love to mother language.

In our generation, there are several technological advances that help us to create and develop different kinds of machine learning algorithms in different areas in order to make our life easier. For instance, we can find lots of algorithms that are already made by developers to perform the morphological and lexical analysis in different languages; however, such an algorithm has not been created for our national language. The problematic issue is the lack of the morphological analysis of Azerbaijani linguistics currently and there is not such an available source which can include the morphological analysis of every word in Azerbaijani language in order to eliminate that problem. There are lots of advantages of our algorithm in order to solve many problems; therefore, our team has decided to make the vision on this topic in order to perform the tasks for Azerbaijani languages. Thus, since the morphological analysis of the Azerbaijan language has not been completed by any other source including Microsoft Word with its editing function or other applications due to not considering every single word according to its right spelling, our basic objective is to radically solve this issue.

Thanks to advanced software for word correction in modern times writing in almost all different languages is made very easy especially for those who are writing long academic papers and articles. However, as mentioned before this is almost impossible in our national language and the available software is far behind the perfect. Unfortunately, in this modern world, people writing papers in the Azerbaijan language have to use old-fashioned ways such as books and dictionaries to improve their writings. Our project aims to solve this problem and take Azerbaijani language to the same level as others. One of the outstanding potential applications is word correction which will do its job very well by our entered algorithms and satisfy the potential users of the application. The target segmentation can be demonstrated as the essay and article writers, journalists or anyone else that are involved such kind of academic issues whom our website will help.

In other words, the purpose is to contribute our work to Azerbaijan Republic to develop the Azerbaijani language in terms of its functionality. Additionally, a research paper is prepared to show all the algorithms and techniques that we used.

Project Objectives, Significance, Novelty

For the project, we needed knowledge of two fundamental fields, especially, linguistics and computer science. To implement our idea, coding skills with programming languages (in our case, Python), ability to choose and perform suitable algorithms, as well as knowing important grammatical rules of Azerbaijani have been necessity.

In today's world there are several tools and algorithms that were made by individuals to develop and enhance the language and make our life easy. One of such algorithms is to demonstrate the possible options for any word and check the words in the sentence; however, such an algorithm has not been developed for the Azerbaijani language. There's one tool for checking the words only, however, that algorithm is not able to check it for the sentences. There are several problems that can be solved by implementing this algorithm. First of all, we were able to contribute to the linguistics by implementing this algorithm, by showing possible ways to generate different words for any input. Therefore, our team has decided to implement this algorithm within Azerbaijani language. Here are the main approaches that we put to practice: checking the possible cases for every word, checking the sentence and dividing it into parts for every word. Here are the main project objectives:

1. Getting and adding the databases of information related to Azerbaijani language
2. Checking the database of words every time
3. Applying our algorithm for every word such as noun, verb, adjective and pronoun
4. Evaluating our algorithm with other algorithms by checking the similarities and differences
5. Designing the website for our software in order to make in available to public use.

Our team checked the possible algorithms that had been developed in English and Azerbaijani language. Our job was applying these approaches into Azerbaijani language and get help from linguistics to improve them. Our team made a deep research related to this issue and studied several algorithms that helped us in accomplishment of this project.

The significance of our project is that we have contributed to the language and linguistics by applying this algorithm and there's lack of tools that check the words and sentences. In development of such an important software of analysis of Azerbaijani language we got significant help and advice from our project advisor Dr. Samir Rustamov and SDP coordinator Dr. Abzatdin Adamov. As a team, we created an Azerbaijan language library that evaluates morphological analysis. Basically, it consists of all parts of speech including primary and secondary ones. In this project, you can access all Azerbaijan words and their possible suffixes.

Problem Statement

Eventually, after discussing all members, we came up with many ideas of how to implement our idea. Firstly, we got dictionary of Azerbaijani language and added to our database. However, when we entered an input, it sometimes threw error because of spelling specific letters such as 'ə', 'ö', 'ğ', 'ş', 'ç', 'ü'.

All these letters are not supported in UTF-8, it means it does not support Python 2.7. Therefore, it led to create several problems when it did not support. So, we had to work with UTF-8, which is supported by Python 3.0 version.

Another problem happened because of some exceptions in our linguistics, sometimes outputs were incorrect due to this drawback. In other words, as a methodology we used Rule-Based.

Even though this method is great when we implement all possible conditions, the Azerbaijani language has numerous exceptions that developers might forget to program all of them and that required a lot of, time, work and responsibility to implement all of them.

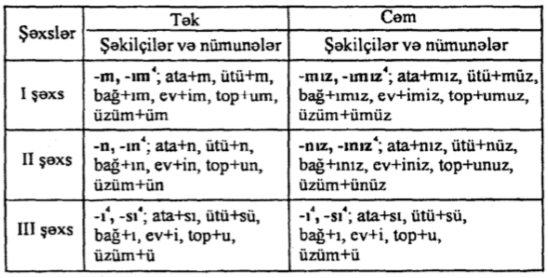
Furthermore, redundant storage of data and data inconsistencies could occur due to lack of functionality. All these technological factors can lead to delay of our system or even crash when it waits for a long time.

Literature review

Buludxan Xelilov's 'Azerbaycan dilin morfologiyası' explain everything related to Azerbaijani morphology in details. To illustrate, due to grammatical meaning of noun, it is studied name of the objects, humans, places and other things and its questions of What? Who? (sometimes Where?) according to Azerbaijan linguistic. The most important feature of noun is being singular or plural, change according to its belonging and modify due to 6 different forms/cases of noun.

In our language, when nouns become plural, the majority of them add same kind of suffixes like 'lar' or 'lər' (in English, suffix is 's'). However, there are exceptions that some words cannot add these suffixes because they mean plural by default. Buludxan Xelilov's book illustrates the words 'camaat' and 'əhali' (in English, 'people') that mean plural noun and it is impossible to add 'lar' or 'lər' to their endings.

In particular, nouns are modified because of their possession (to whom or to what). The specific suffixes can identify the object is belonging to exactly which person. In Azerbaijani, possessive suffixes (In Azerbaijani, it is called ‘mənsubiyyət şəkilçisi’) define object’s owner is whether first, secondary, or third person (‘I şəxs’, ‘II şəxs’ and ‘III şəxs’). The below graph illustrates all possessive suffixes according to its owners and examples:



Şəxslər	Tək	Cəm
	Şəkilçilər və nümunələr	Şəkilçilər və nümunələr
I şəxs	**-m, -ım^4**; ata+m, ütü+m, bağ+ım, ev+im, top+um, üzüm+üm	-mız, -ımız^4; ata+mız, ütü+müz, bağ+ımız, ev+imiz, top+umuz, üzüm+ümüz
II şəxs	**-n, -ın^4**; ata+n, ütü+n, bağ+ın, ev+in, top+un, üzüm+ün	**-nız, -ınız^4**; ata+nız, ütü+nüz, bağ+ınız, ev+iniz, top+unuz, üzüm+ünüz
III şəxs	**-ı^4, -sı^4**; ata+sı, ütü+sü, bağ+ı, ev+i, top+u, üzüm+ü	**-ı^4, -sı^4**; ata+sı, ütü+sü, bağ+ı, ev+i, top+u, üzüm+ü

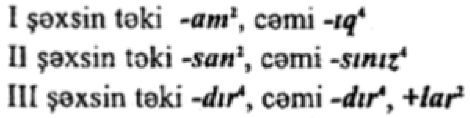
Fig 1. Suffixes for Owners

As you can see from the table, bold letters are suffixes and same words are used by their personal feature (in graph, ‘I şəxs’, ‘II şəxs’ and ‘III şəxs’). Generally, suffixes are ‘-m’ or ‘-ım^4’ for 1st singular, ‘-n,’ or ‘-ın^4’ for 2nd singular, ‘-ı^4’ or ‘-sı^4’ for 3rd singular person. Similarly, ‘-mız^4’ or ‘-ımız^4’ for 1st plural, ‘-nız^4’ or ‘ınız^4’ for 2nd plural and the last ‘-ı^4’ or ‘-sı^4’ for 3rd plural person.

The reason why it sometimes adds –m (-n, -ı^4, -mız^4 -nız^4) suffixes or –ım^4 (–ın^4 ,-sı^4, -ımız^4,-ınız^4) is that if the noun ends with vowel, it should add first category as a suffixes of ‘mensubiyyet’; on the other hand, those words who ends with consonant, suffixes should begin with ‘-ı^4’+first category of the suffixes of ‘mensubiyyet’. The meaning of ‘-ı^4’ is that it can be ‘ı’,’i’,’u’, or ‘ü’ as suffixes in Azerbaijani. The decision of which one to use is related to characteristics of last vowel in the noun dependent of whether vowel is bold, thin, open or closed. In the above table, another interesting fact is that 3rd person’s singular and plural version has the same suffixes.

Besides that, ‘Müasir Azərbaycan dili’ (Modern Azerbaijani Language) by Gulara Abdullayeva gives explanations of tons of exceptions which break the rule mentioned above. For example, there are several loanwords, especially which ends with vowel in our language, but do not obey the above rule. For instance, these loanwords: ‘tale’, ‘mövqe’, ‘mənşə’, ‘mənafe’ and ‘mənbə’ does not add these suffixes: -m, -n, -sı^4, -mız^4, -sı^4. Instead, they use additional letter ‘y’ as a connecting letter: Tale-**yim** tale-**yin**, tale-**yimiz**, tale-**yi**. The original Azerbaijan words of ‘su’ and ‘nə’ also included in this exception category of rule. They also modify by its belonging with connection letter ‘y’: su-**yum**, su-**yunuz**, nə-**yim**, nə-**yin**.

Moreover, noun can be changed due to its owner (‘şəxs’):



I şəxsin təki *-am*^{2}, cəmi *-ıq*^{4}
II şəxsin təki *-san*^{2}, cəmi *-sınız*^{4}
III şəxsin təki *-dır*^{4}, cəmi *-dır*^{4}, *+lar*^{2}

Table 2. Cases for Noun

‘Y’ connecting letter is also used when the noun is ended by vowel and it is 1st singular or plural person. To illustrate, ‘Mən balıqçı-**yam**.’, ‘Biz danizçi-**yik**’.

According to another literature, Mohsun Nagisoylu’s and Mehman Zeynalli’s ‘Azerbaycan dili’ (Azerbaijani Language), other important thing is that nouns can be modified by adding grammatical suffixes due to its dependency with other words in the sentence. This action is called as a case (In Azerbaijani, it is ‘hallanmaq’). In our language, there are 6 different cases of nouns and all of them have different features. The first case is called as ‘Adlıq hal’. In this case, nouns answer the questions of ‘who?’ ‘what?’ and ‘where?’. Student (who?), book (what?), Baku (where?) are several examples of this case. ‘Adlıq hal’ has no specific suffixes, so without any suffixes, by default nouns are in this case. However, a noun with ‘Adlıq hal’ can have suffixes of plural, ‘mensubiyyet’ and ‘xeberlik’. For example: ev-**lər**, kitab-**ım**, qələm-**dir**.

The second case of noun is called as ‘Yiyelik hal’. In this case, it answers the question of whose? What? and it adds suffixes of –ın^4 (-nın^4) to the end of nouns. For instance: ‘ağac-**ın**’, ‘qapı-**nın**’. The reason why it sometimes adds –ın^4 suffixes or –nın^4 is that if the noun ends with vowel, it must add -nın^4 as suffixes of ‘yiyelik hal’. On the other hand, if the noun ends with consonant, the suffixes must be –ın^4.

The third case of noun is called ‘Yonluk hal’ which answers the question of ‘to whom?’, ‘to what?’ or ‘to where?’. As suffixes, it uses –a^2. This suffixes means that it can be ‘-a’ or ‘-ə’ depending on last vowel of the noun. If it is bold, then ‘-a’ must be used, otherwise, ‘-ə’ must be added to the end of noun as a suffix of ‘Yonluk hal’.

The fourth case of noun is called as ‘Təsirlik hal’. It has ‘-ı^4’ or ‘-nı^4’ as its specific suffixes. Again, if the noun ends with vowel, it should add ‘n’ as a connecting letter, otherwise, it is added plainly without any connecting letter as ‘-ı^4’. The second decision of whether to use ‘ı’,’i’,’u’, or ‘ü’ as suffixes depend on characteristics of last vowel in the noun dependent of whether vowel is bold, thin, open or closed.

Other case of noun is called as ‘Yerlik hal’ which adds ‘-da^2’ to the given noun and has no complications as previous three. For instance: kitab-**da**, bağ-**da**. It is very similar to the English language notion of ‘in the’ or ‘at the’.

Last case of noun is called ‘Çıxışlıq hal’ that basically answers the question of ‘from whom?’ ‘from what?’ or ‘from where?’ (for ex; məktəb-**dən**). For instance: kitab-**dan**, bağ-**dan**. It is very similar to the English language notion of ‘from the’.

In Azerbaijani language, there are several exceptions which are related to the dropping of last vowels of word in two-syllables nouns when some specific suffixes are added. Some words are mostly loanwords that will drop its last vowels when any kind of suffixes are added (either ‘hal’ or ‘mənsubiyyət’ suffixes). These words are ‘qədir’, ‘qisim’, ‘zehin’, ‘eyib’, ‘ətir’, ‘isim’, ‘izin’, ‘nəsil’, ‘qədir’, ‘ömür’, ‘səbir’, ‘sətir’, ‘sinif’, ‘fəsil’, ‘fikir’, ‘cisim’, ‘şəkil’, and example of dropping its last vowels of ‘sinif’ are represented as follows: ‘sinfin’,’sinfə’, ‘sinfimiz’. Furthermore, there are native Azerbaijani words which also drop its last vowels if and only if it consists of ‘mənsubiyyət’ suffixes. These words are ‘ağız’, ‘alın’, ‘beyin’, ‘boyun’, ‘burun’, ‘ovuc’, ‘qarın’, ‘qoyun’, ‘oğul’, ‘çiyin’. It is also necessary to say that noun mostly used as a subject (‘mübtəda’) in the sentence.

Adjective mainly defines the object’s colours and qualties. In Azerbaijani, adjective has no grammatical suffixes, but if it is being substantive, then it will add suffixes of noun. Adjective is the only part of speech that is attribute (in Azerbaijani, ‘təyin’). Similarly, adverb (‘zərf’) pronoun (‘əvəzlik’), quantifier (‘say’) also has no specific grammatical suffixes.

Last part of speech, verb defines the action of an object in the Azerbaijan language. Generally, verbs answer the question of ‘what is does?’ ‘what it did?’, ‘what it will do’ etc. Like nouns and adjectives, verbs also occupy a huge segment of Azerbaijani dictionary and is frequently used in our language. Unlike other parts of speech, verbs have specific features such as being affirmative or negative (in Azerbaijani, it is called ‘təsdiqdə və inkarda olmaq’), define the tense, being transitive or intransitive (in Azerbaijani, ‘təsirli və təsirsiz olmaq’).

To tell in detail, affirmative verbs have no specific suffixes, for instance: ‘yaz’, ‘aldı’, ‘gəlirəm’, ‘baxacaq’. On the other hand, negative verbs have ‘-ma^{2}’ suffix, such as ‘yazma’, ‘getmə’, ‘almadım’, ‘baxmayacaq’. Undoubtedly, this rule also has several exceptions. To illustrate, it is true to say that, in present and uncertain future tense, the ‘-ma^{2}’ suffix drops the vowel and transforms to ‘m’. In other words, vowel of this suffix will drop if the verb is in present tense or uncertain future tense. For instance,

oxu+yur → oxu+m+ur,
yaz+ar+am → yaz+m+ar+am.

Besides that, the verbs in uncertain future tense in 3rd person both singular and plural forms, ‘r’ letters from ‘-ar^{2}’ suffix modify to ‘z’. For example,

gəl+ər → gəl+m+əz
oxu+yar+lar → oxu+m+az+lar

Furthermore, verbs are divided into transitive or intransitive (In Azerbaijani, ‘təsirli’ or ‘təsirsiz’). For instance, ‘gördüm’,’yazır’,’içirsən’ verbs are included to the transitive verbs; however, ‘baxıram’,’yaşayırıq’ and ‘gəlirsən’ verbs are intransitive verbs. Sometimes intransitive verbs can change themselves to transitive verbs. In this case, ‘-dır^{4}’ and ‘ır^{4}’ (occasionally, ‘-t’,’-ar^{2}’,’-ıt^{4}’) suffixes helps this change to occur. For example:

Inanmaq → inandırmaq
Çatmaq → çatdırmaq
Bişmək → bişirmək
Qopmaq → qoparmaq

Similarly, sometimes it is possible to change verbs from transitive form into intransitive. In order to do this, ‘-ıl^{4}’ ‘ın^{4}’ ‘ış^{4}’ suffixes are added. To illustrate:

Çəkmək → çəkilmək
Almaq → alınmaq
Yazmaq → yazışmaq

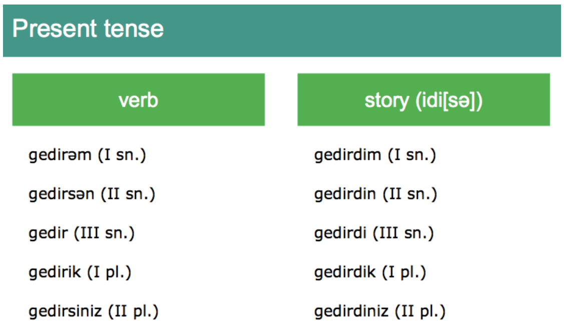
Moreover, verbs have grammatical meanings according to the relationship between person, who acts and the objects which are used. For that purpose, verbs can be active, passive, reflective, coexistence, and causative forms. Correspondingly, suffixes are demonstrated on the following table:

Active Voice (‘Məlum növ’)	no suffixes
Passive Voice (‘Məchul növ’)	‘-ıl^{4}’, ‘-ın^{4} (-n)’
Reflective (‘Qayıdış növ’)	‘-ıl^{4}’, ‘-ın^{4} (-n)’
Coexistence (‘Qarşılıq-birgəlik’)	‘-ış^{4} (-ş)’
Causative (‘İcbar növ’)	‘-dır^{4}’,’ -t’

Table 4. Suffixes for Verb

In addition, there are several categories that verbs include to those groups such as ‘təsriflənən’ and ‘təsriflənməyən’ forms. Basically, it means that if the verb can change according to tense forms, person and objects, then it belongs to the first category. Then, verbs in this category also divide into 6 forms: Command form (In Azerbaijani ‘Əmr şəkli’), reporting form (In Azerbaijani ‘Xəbər şəkli’), wishing form (In Azerbaijani ‘Arzu şəkli’), urgent form (In Azerbaijani ‘Vacib şəkli’), need form (In Azerbaijani ‘Lazım şəkli’), and condition form (In Azerbaijani ‘Şərt şəkli’).

Moreover, Azerbaijani Natural Language Processing (Azeri NLP) were the first system that initiated Azerbaijani stemming. However, in this service, it is only concentrated on verbs, not other parts of speech. As an input, you can enter the verb and the system will return all the verb forms as output (http://nlp.jsoft.ws/verb). For example, you can see the specific verb with all tense forms including past tense, present tense and future tense. In addition, Azeri NLP is also considered on several conditions and moods of given verb including singular or plural cases of 1st, 2nd and 3rd person forms and positive or negative forms of verbs:



Present tense

verb	story (idi[sə])
gedirəm (I sn.)	gedirdim (I sn.)
gedirsən (II sn.)	gedirdin (II sn.)
gedir (III sn.)	gedirdi (III sn.)
gedirik (I pl.)	gedirdik (I pl.)
gedirsiniz (II pl.)	gedirdiniz (II pl.)

Table 3. Present tense for Verb

Besides that, there are several parts in that system which indicate other important information of given input. In this case, input can be either verb or any other part of speech. To illustrate, there exists ‘Native Azerbaijan words finder’ which indicates whether the word is native word or not.

Furthermore, another section is called ‘Word analysis’, which basically analyzes the given word. After entering the word ‘Fidan’, it returns letter count: 5; vowel count: 2; vowels: a, i; bold vowels: a; thin vowels: i; open vowels: a; closed vowels: i; nonlip vowels: a, i; consonants count: 3; consonants: f, d, n; voices consonants: d (unvoiced counterpart is t), n; Unvoiced consonant: f (voiced counterpart is v); Comments: Word does not satisfy the harmony rule.

One of the interesting sections of Azeri NLP is ‘Digitizer’ part. It is said write any number with words and shows blank space in order to give input. To test this, the word ‘altı’(In English, it is six) is written, then, it really transforms the word into number in this section of the system, as a result, it shows 6 in the system.

Stemming is the area where the user can input the text and see the result of all stemmed words.

Text Analysis is another section where you can input the text, but in this case, results will be related to number of characteristics (with or without space), words count, keywords that was repeated in the text and another essential factors.

Hyphenation is also included in Azeri NLP which divides the given word(s) into the syllables correctly.

The last interesting area is called as Compare texts where you can compare two texts according to its words.

3 DESIGN CONCEPTS

3.1.1 Approaches

It’s been centuries that several researchers have been making lots of investigations about the morphological and syntactical analysis in different languages and most of them have been made in English. For creating our own algorithms, we had to make a huge amount of research to understand most of the created applications regarding the morphological and syntactical analysis. While making research, we were able to understand the main idea of the topic, how those algorithms work, strengths and weaknesses of those algorithms, which helped us to create and develop better algorithm for Azerbaijani language. For Azerbaijani language, we have searched quite a lot about other algorithms that can be applicable to our application as well, however, during the research we have found several problems with other algorithms that cannot be applicable to our language. The problem with our application was to implement the characters such as ‘ə’ in Python programming language. We have found the solution for this challenge by upgrading Python environment and adding our own Python script inside our application. Second problem was with implementing and adding the dataset into our application because it contained lots of unnecessary things such as JavaScript codes, words in English, Arabic, Russian and etc. We have cleared the dataset by implementing Python script to get rid of the problems that are mentioned above. Another problem we met was to have and replace some characters in Azerbaijani language such as using ‘I’ instead of ‘İ’ and we have solved this issue by replacing ‘İ’ with ‘i’ in Python programming environment.

3.1.2 Detailed Description of Solutions

As mentioned above, we have made a lot of research about other algorithms by checking every single important detail and we have chosen the best 3 algorithms that made us realize the weaknesses of our algorithm. The names of these algorithms are Polysyllabic, Morphology and Implications for English Language Teaching, Acquiring Lexical Semantics, Machine Learning Techniques for Stemming and Lemmatization and etc. We have made research about implementing them into our application by checking which application works much better than the other all.

3.1.2.2 Acquiring Lexical Semantics

The main idea of this algorithm is to detect the meaning of words by relating them to semantic representations that has been derived from perceptual or cognitive processing. For instance, as an example we can say that one person says ‘Tutu topu dostuna atır’. In this example, the girl whose name is ‘Tutu’ understands what exactly ‘top’ and ‘atmaq’ mean. This is the approach of two researchers named Grimshaw and Pinker, who tried to implement this approach during 20th century. However, the problem was not having enough Artificial Intelligence system that could handle it. Other researchers tried to implement this approach; however, using NLP systems were not able to be provided during that time.

Another approach that has been included to lexical semantics was actually not using semantic information related to the input. However, it uses different variations of words and classifies them into different classes. For instance, we can use several words in different lexicological meanings such as ‘yaz’, ‘oxu’, ‘aş’, ‘at’ and other kinds of these words can be used as noun and verb at the same time in Azerbaijani language. In lexical semantics this way of classifying the words is called surface cueing.

There are both advantages and disadvantages of applying this approach. The main advantage of this algorithm is that it can detect the syntactic structure of any word in the sentence by diving it into parts which is called part-of-speech tagged. In following paragraphs, you will get familiar with it in brief. However, the main disadvantage of this algorithm is that it’s not always easy to find or uncover the lexical semantics inside the sentence.

3.1.2.3 Machine Learning Techniques

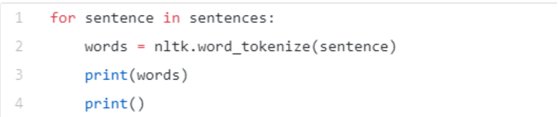
Natural Language Processing

Before explaining our approach, first we have to give brief information about the Natural Language Processing (NLP) which is one of the subfields of computer science and machine learning and its work is based on detecting the interactions between the human and computer languages. It’s one of the widely used machine learning algorithms for text and speech. NLP has been used for huge translation systems such as Yandex and Google, autocomplete systems and etc.

Usage of NLP is getting wider because of the algorithms that it provides and becoming the leading platform with human language data. In the following paragraphs, we will give brief information about the algorithms that NLP uses for text.

First of all, word tokenization is one of the techniques that is widely used which divides the whole string into words by checking the space between them.

Example. For the illustration purposes, we can say that the input sentence for this algorithm is ‘Əli bu gün gecəyə kimi dərs oxudu.’. We can use ‘*nltk.word_tokenize’* function to tokenize this example.



```
for sentence in sentences:
    words = nltk.word_tokenize(sentence)
    print(words)
    print()
```

Fig 2. Usage of NLP

The output we get from this example is

[‘Əli’, ‘bugün’,’gecəyə’,’kimi’,’dərs’,’oxudu’,’.’]

The advantage of this algorithm is that we can divide all the words from the sentence and add them into one new array in order to use afterwards.

Sentence tokenization is similar to word tokenization and it divides the string into component sentences by checking the punctuation mark which is ‘.’ in the end of each sentence. The usage is also similar to word tokenization; however, we change the *nltk.word_tokenize* into *nltk.sent_tokenize.*

Example. Let’s say our example for the input is as following:

‘Bizim komandanın uşaqları hər şeyi vaxtında edib bitirməyə çalışırdılar. Son gün yaxınlaşırdı.’

As an output, we get 2 component sentences separately.

Decision Tree

Besides of those algorithms that could be useful for implementing our own algorithm, we tried to implement our own algorithm with machine learning techniques. In the following paragraph we will describe some of them in detail.

Decision tree is a model that looks like tree of decisions and the consequences that come with those decisions. There are several ways to graph the conditional control statements and decision trees are one of those algorithms. Decision trees has two elements which are classification and regression trees. Classification trees are the targets to the discrete variable features; however, regression trees are represented in the manner of continuous features. For our approach, we needed decision trees for showing the tree of characteristics of any word by diving it into the nodes that have heads (we can also say the root) and tails. A tree has to be divided into several parts by splitting the word into subsets based on the morphemes which are mainly about morphological units of the language. It’s always repeated process, so it’s *recursive approach* until we get the desired result which is the collection of word and morphemes.

There are both advantages and disadvantages of using decision trees in machine learning. In following paragraphs, we will be giving brief information about the strengths and weaknesses of decision trees.

First fundamental strength of decision trees is that decision trees are understandable and easy to implement. As we have showed example above, it takes the word and divides into the parts we call nodes. The second strength of decision trees is that decision trees don’t require much computation because it’s light-weight. From this point of perspective, it’s easy to implement classification to decision trees. Thirdly, decision trees can be helpful with implementing a clear understanding of fields that are important for prediction and classification.

Despite all the strengths, there are weak sides of decision trees as well. First of all, as we have mentioned decision trees are good to implement on classification, however, it’s less appropriate for implementing on continuous features. Secondly, decision trees may not be helpful if the data is too huge because it can bring out the errors in classification problems. The last but not least weakness is that decision trees are very expensive to train. Decision trees always increase by power of two, so for this reason it gets more problematic when the tree grows larger.

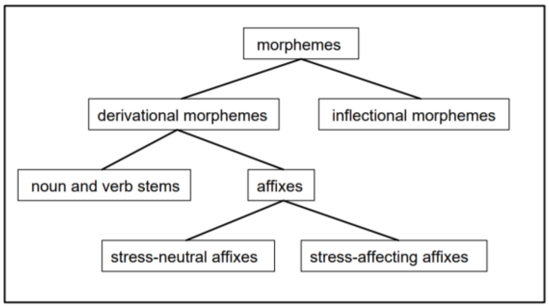


Fig 3. Decision trees for Suffixes

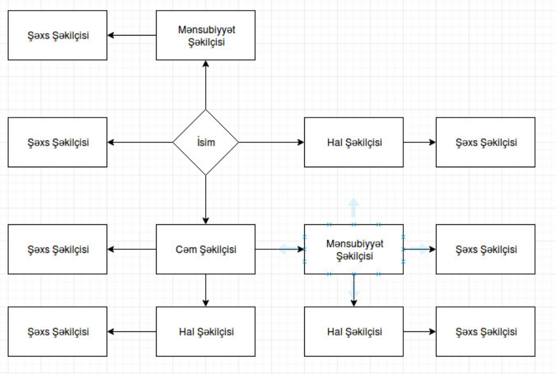
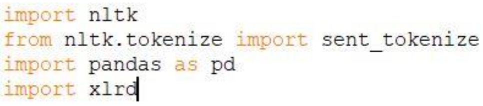


Fig 4. Tree for our algorithm

Natural Language Toolkit

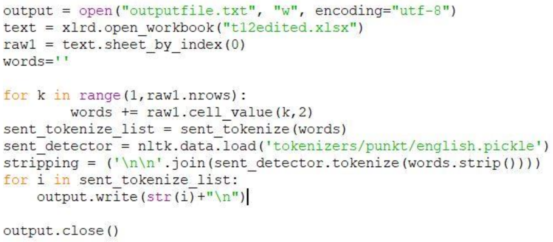
Natural Language Toolkit (NLTK) is one of the famous platforms for creating and developing Python applications for being able to work with human language data. It’s possible to implement stemming, tokenization and tagging by using NLTK library.

The reason why we used Natural Language Toolkit was because of dividing the excel files into the combination of sentences that output to the text file. The first we needed to include the library by adding following commands:



```
import nltk
from nltk.tokenize import sent_tokenize
import pandas as pd
import xlrd
```

Figure 5. Importing NLTK



```
output = open("outputfile.txt", "w", encoding="utf-8")
text = xlrd.open_workbook("t12edited.xlsx")
raw1 = text.sheet_by_index(0)
words=''

for k in range(1,raw1.nrows):
        words += raw1.cell_value(k,2)
sent_tokenize_list = sent_tokenize(words)
sent_detector = nltk.data.load('tokenizers/punkt/english.pickle')
stripping = ('\n\n'.join(sent_detector.tokenize(words.strip())))
for i in sent_tokenize_list:
    output.write(str(i)+"\n")

output.close()
```

Figure 6. The main part of using NLTK

After including the required library, we declared the desired output text file and the excel file that should be opened for reading the data. Our dataset had only one index, so it was not necessary to declare it. Then we initialized the array of characters in order to put all the tokenized sentences inside the array. At the end, we wrote all the converted data using NLTK to the text file.

3.1.2.4 Stemming and Lemmatization

Stemming is one of the most popular ways that has been used in linguistics by reducing the words into their initial form which is also called base form. There are several algorithms that has been created that stems the words. For instance, in English every word can be stemmed such as ‘arguing’, ‘argues’, ‘argued’ into the base form of ‘argue’. A number of approaches have been reported by several researchers for several languages.

Stemming and lemmatization are used together most of the time, however, there are subtle differences between them. Stemming is the way of cutting the end or beginning of the word by taking the prefixes and suffixes into account. However, lemmatization takes the whole sentence and does morphological analysis. For instance, we can show an example in Azerbaijani language such as

Real: ‘Əli dərsini oxuyur’.
Lemmatization: ‘Əli dərs oxu’.

which means it reduces the words inside the sentence into the initial forms if exists. However, in stemming the example can be

‘oxudu’, ‘oxuyur’, ‘oxuma’, ‘oxu’

Despite all the advantages of using stemming, unfortunately there are two fundamental errors in it. The first error is called over stemming which happens when two words that have different stems are stemmed into the same initial form. This is called one of the Machine Learning concepts which is false positive. Second error with stemming is called under stemming which happens when two words should be stemmed into the unit root, however they don’t actually stem. There are several algorithms that has been created for stemming in English language and in the following paragraphs we will be talking about those algorithms in brief.

Lovins Stemmer

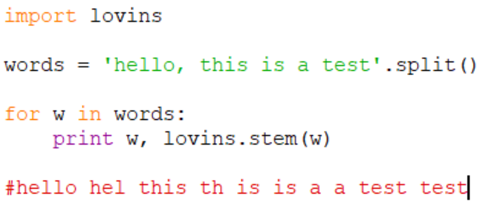
This is one of the oldest and popular algorithms that has been proposed in twentieth century by famous researched whose name was Lovins. There's a table of all the suffixes and prefixes that performs the transformation rules for the words and this method have been called also the longest match principle. In this algorithm, the main idea is to remove the longest suffix or prefix from the word and try to convert the stems into valid words by removing maximum of one suffix or prefix from any word.

There are both positive and negative sides of using this stemming algorithm. Firstly, the negative side is that this algorithm uses lots of data in the memory which takes a lot of time to read. Secondly, another drawback is that this algorithm doesn't include all the suffixes or prefixes in the table which can be hard to detect and stem any word. However, despite all the drawbacks, there are also the essential advantages of using this algorithm. Firstly, this algorithm can be beneficial for removing two suffixes at the same time which reduces the time.

If we describe the algorithm itself more, we have to mention all the conditions on the suffix's removal.

1. Minimum size of a stemmed word is increased by following ending's removal.
2. Keeping the endings when certain letters are present in the remaining stem.
3. Combination of the statements above.

Algorithm



```
import lovins

words = 'hello, this is a test'.split()

for w in words:
    print w, lovins.stem(w)

#hello hel this th is is a a test test
```

Fig 6. Lovins Algorithm

Porters Stemmer

Porters stemming algorithm is also one of the oldest and most popular algorithms that has been created during the twentieth century. The fundamental concept of this algorithm is that in English language most of the suffixes are made up by the combination of smaller suffixes or prefixes. In Porters stemming algorithm, there are main steps or as called *phases* that rules are applied for every phase.

The rule is similar to

<condition> <suffix> ⟶ <new suffix>

which means there is a condition that checks the suffix and detects if the new suffix can be made or not. For instance, if we have EED -> EE which simply means that if any word ends with 'eed' then change the ending to just 'ee' which stems the word. As an example, we can say 'agreed' stems into 'agree' by replacing 'eed' with 'ee'. There are many other examples like this in English language.

This algorithm is much better than Lovins algorithm because in this algorithm there are less errors than in Lovins algorithm. However, this algorithm takes much more memory than Lovins because Lovins algorithm is more light-weight than Porter's algorithm.

There are several implementations of this algorithm in different languages such as Russian, Greek, Turkish etc.; however, there's no any implementation of this algorithm into the Azerbaijani language.

Implementation of Stemming and Lemmatization in Python

The fundamental concept is that stemming, and lemmatization are parts of *text normalization* techniques that is described in the field of **Natural Language Processing** that we have mentioned in previous paragraphs. For instance, we can show examples such ,

playing ⟶ play
plays ⟶ play
played ⟶ play

which stems the words into the initial or root form. For showing the normalization of sentences with stemming and lemmatization, we can show examples.

Example. Let's take one sentence and try to apply stemming into that.

Firstly, the algorithm applies stemmer to the sentence and in the return it brings the distinct words of the sentence.

- 'the girl's toys looked differently'
- 'the girl toy look differ'

As shown in the example, the algorithm removes all the suffixes from the sentence and gives the initial form of the words as output.

NLTK library is the best for stemming the words and sentences in English language by eliminating the suffixes and prefixes as well.

For implementing the stemmer in Python programming language by using *nltk* library we must include the following

```
from nltk.stem import PorterStemmer
porter = PorterStemmer()
print(porter.stem('cats'))
```

The code above will give the output of ‘cat’ because the PorterStemmer uses suffix stripping as we mentioned in Porters stemming.

However, in Azerbaijani language we couldn’t use any of these algorithms because these algorithms are not available for our language (only *nltk* could be used for tokenizing the sentences for our dataset). Instead of that, we accomplished our own algorithms that works as same as Porters stemmer which takes the suffixes and stems the words and sentences into the initial forms.

Data Cleaning

Data cleaning is the way of finding and deleting unnecessary and inaccurate data from the data set such as files, databases, tables and etc. Data cleaning should be done in the first part of the project because no any researcher would like to have messy data set and implement it. There are several ways that the data can have errors in it such as irrelevant data, duplicate data, type conversion and syntax errors. However, first of all, we have to find and detect the unexpected, incorrect and noncompatible data. By doing this we can understand and detect what we have to remove from the data set. In the following paragraphs, we will be giving more information about cleaning process of the data set.

First of all, the fundamental idea for cleaning the data is inspecting the data itself. For instance, several things such as detecting unnecessary, incorrect and inconsistent data should be found and noted. For inspecting the data there are several algorithms that have already been made, but we have created our own script for it.

Secondly, after noting and inspecting the data, we have to check which data is irrelevant and messy. In our dataset, we had several problems with the data such as having JavaScript files, Arabic, English and Russian sentences, unnecessary CSS scripts and so on. This category of data is called irrelevant data because we needed sentences in our language. In our script, we have initialized the character arrays with the unnecessary and irrelevant data in order to detect and replace them with empty space. Another problem in our data set was having the duplicated sentences inside the excel file. The reason behind this issue was that we have combined all the data sets that we have got from different news into one big file. The solution for this was to extract the sentences from excel file into the text file and adding our function in order to check if there are duplicate sentences or not. If the duplicated sentences are found, the script automatically deleted all of them by saving only one sentence. Another problem was with syntax that created problems in our dataset. Those problems were white spaces, sentences that started with ‘*’, ‘- ‘, ‘@’, ‘`’. Our solution for this was to add all of those characters into one array, detect and delete all of them automatically.

3.1.4 Research Methodologies and Techniques

For our project, we have used interview technique which is part of qualitative research methods. The fundamental reason why we chose interview approach was because interviews enable face to face discussion between people lps to get more precise information. We could also use other approaches such as questionnaires, experiments or surveys, but the truth is that interviews are more beneficial for getting the exact and possibilities of collecting detailed information and answers about the research questions.

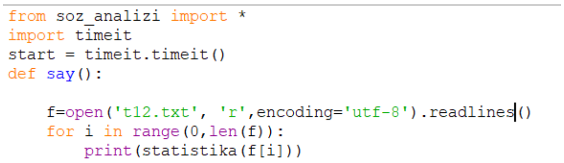
One of the most crucial things for interviews is to choose the right people and asking both open and closed questions should also be chosen. Despite the fact it’s good choice to use closed questions, it’s much more beneficial to ask open questions to get the exact related answers. Closed questions, however, do not require speculations and they only require short answers which was not beneficial for our approach.

Most of the people that we used in our interview was the master’s degree students that study linguistics and researchers in this fields as well when we first started our project. The main questions were about the linguistics itself and the way of using it in Azerbaijani language. We tried to collect sufficient amount of data during this interview. First of all, we asked about using forms in Azerbaijani language and how we can combine the words into one sentence and apply our algorithm. We also tried to get information about usage of words and how one word can have several ways of expression such as adding suffixes and prefixes.

Statistics with Cleaned Data

Before mentioning the statistics we have created, first we have to describe the steps that we made for this process. First and most important phase of cleaning the data was screening phase that we have mentioned in previous paragraph and in this stage we have detected all the problematic strings that could make problem when we run our statistical algorithm. Second stage for our statistical data was called treatment stage which comes after identifying the errors and cleaning all the problems in our data.

For our statistical data, first we have created the Python script in order to open the file which was our dataset with the size of 500 MBs and we tested our dataset in our program.

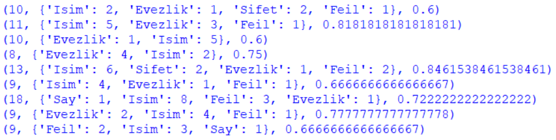


```
from soz_analizi import *
import timeit
start = timeit.timeit()
def say():

    f=open('t12.txt', 'r',encoding='utf-8').readlines()
    for i in range(0,len(f)):
        print(statistika(f[i]))
```

Fig 8.1. Implementation for Data

What this script simply does is that it checks our algorithm and method called “statistika” and checks the text file line by line and prints the statistics that is shown in Figure.



```
(10, {'Isim': 2, 'Evezlik': 1, 'Sifet': 2, 'Feil': 1}, 0.6)
(11, {'Isim': 5, 'Evezlik': 3, 'Feil': 1}, 0.8181818181818181)
(10, {'Evezlik': 1, 'Isim': 5}, 0.6)
(8, {'Evezlik': 4, 'Isim': 2}, 0.75)
(13, {'Isim': 6, 'Sifet': 2, 'Evezlik': 1, 'Feil': 2}, 0.8461538461538461)
(9, {'Isim': 4, 'Evezlik': 1, 'Feil': 1}, 0.6666666666666667)
(18, {'Say': 1, 'Isim': 8, 'Feil': 3, 'Evezlik': 1}, 0.7222222222222222)
(9, {'Evezlik': 2, 'Isim': 4, 'Feil': 1}, 0.7777777777777778)
(9, {'Feil': 2, 'Isim': 3, 'Say': 1}, 0.6666666666666667)
```

Fig 8.2. Statistical Data

As seen from the Figure.8. our algorithm gets more than 70% of accuracy with the dataset of 1 million lines of sentences. As a consequence, we have got 13 million words in overall with 6 million nouns, 1 million adjectives, 1 million verbs and others.

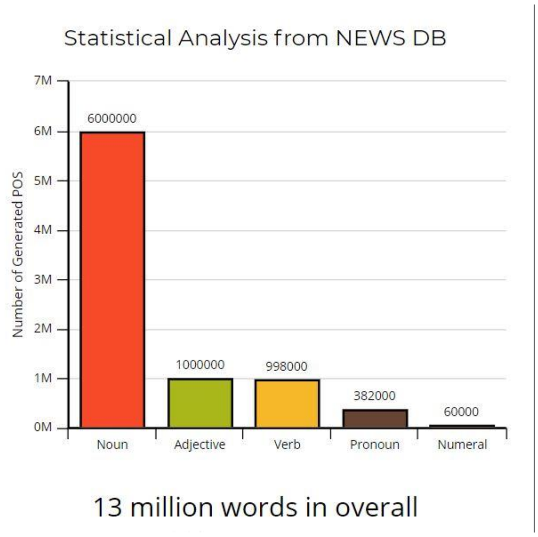
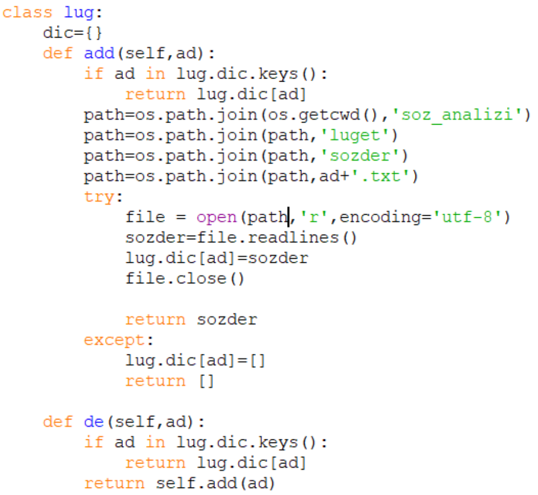


Table 5. Statistical Data

In our algorithm, we created one class to speed up the process for statistical analysis which checks the file line by line and adds it to the dictionary of arrays and returns it. This method was helpful because it printed out 1000 lines of text file just in 30 seconds. The performance has been done on laptop with speculations of Intel Core i5, 8GBs of RAM and Ubuntu operating system. The implementation of the class for speeding up the process is shown in Figure 8.3.



```
class lug:
    dic={}
    def add(self,ad):
        if ad in lug.dic.keys():
            return lug.dic[ad]
        path=os.path.join(os.getcwd(),'soz_analizi')
        path=os.path.join(path,'luget')
        path=os.path.join(path,'sozder')
        path=os.path.join(path,ad+'.txt')
        try:
            file = open(path,'r',encoding='utf-8')
            sozder=file.readlines()
            lug.dic[ad]=sozder
            file.close()

            return sozder
        except:
            lug.dic[ad]=[]
            return []

    def de(self,ad):
        if ad in lug.dic.keys():
            return lug.dic[ad]
        return self.add(ad)
```

Fig 8.3 Implementation of Speeding Class

3.1.5 Architecture, Model, Diagram Description

The final approach that we came up with is to add all the required information by implementing our own algorithm. First of all, we have implemented everything in Python programming language. Python's popularity has increased over the years because of the tasks that it can easily implement. One of the greatest programmers in history named Guido van Russom invented Python programming language back to 1991 and he wasn't even aware that this programming language could become this popular one day. There are several reasons why we have chosen this programming language. First of all, Python is one of the best programming languages because of its usage in Machine Learning, Artificial Intelligence and other subjects. The main reason that Python is good with these subjects is because it contains extensive selection of libraries and frameworks that can implement anything. For instance, usage of NumPy, SciPy, NLTK, Tensorflow has been increased over the years because of facilitating coding and it also saves a lot of development time during implementation. Secondly, Python is one of the best languages because of its simplicity in a sense of declaring the readable codes even for the beginners. These are the two fundamental reason why we chose this programming language; however, there are a lot more to describe.

3.1.7 Social Impact

It's not hard to see the impact of technology in our universe and there's no doubt that technology is one of the causes for social change. Inventions of lots of algorithms, technological tools can be examples for seeing the beneficial side of technology. Invention or implementation of algorithms assist people benefit the advances of technology in the society. From our point of view, the effect of our algorithm in Azerbaijani language will be huge in a sense that several researchers will benefit.

As we have already mentioned above, there are no any application for morphological and syntactical analysis in Azerbaijani language. From our point of view, this application will benefit people, especially researchers, schools and universities in a sense of helping them to get different versions for the words and realize to which class each word belongs. For instance, let say one researcher wants to get all the possible cases for each word. We can show 'qələm' as an example which it creates more than four hundred possible new words from exact word. Despite to all the bugs and problems, our application works perfectly, and it defines all the included cases for each word. In addition, researchers will also be able to clean their dataset with our readymade script in Python language which possibly removes all the unnecessary things such as Javascript codes, words in other languages and etc. from the dataset.

4. IMPEMENTATION

Hardware Design

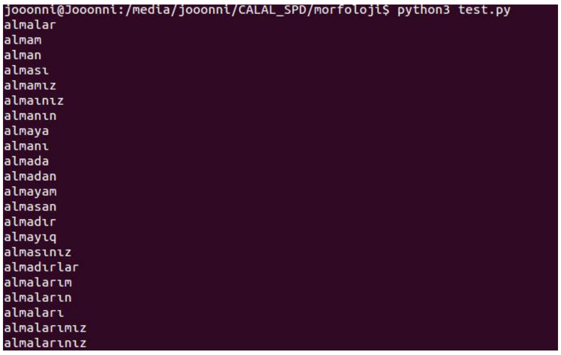
Even though our project did not require hardware implementation, we had necessity to have a server that supports utf-8 and python 3. 8GB of Ram and intel core i5 was enough for completing the testing phase.

Software Design

Using our library, users will be able to do following tasks: divide word into syllables, generate all possible forms of the given word, stem text, and determine part of speech, base word, and all suffix types, which are clearly explained below.

For creating all possible forms of the given word our algorithm needs to know to which part of the speech does this word belong. After this algorithm picks up suffixes from Appendix A and adds them to the given word. Furthermore, each word that contains any suffix also can contain additional suffixes which are mentioned in Appendix B. While adding suffixes decision making about suffix type is made according to rule mentioned in p2.3.

The first feature we had been working on generates all the possible word for the suffixes that can be added to the initial form of the word. All the suffixes are implemented into code according the rules of Azerbaijani language mentioned above. As Azerbaijani language has several complex rules such as the change of the initial form when suffixes are added (letter changes or drop) the program takes this change into consideration as well. Also, each suffix has its order that it can be added to the word and using this feature of Azerbaijani language we constructed a tree structure of suffix order. As detailly described above the algorithm that we had written identifies which suffixes can be added to initial form of the word and each suffix has a list of other suffixes which can follow up.



```
jooonni@Jooonni:/media/jooonni/CALAL_SPD/morfoloji$ python3 test.py
almalar
almam
alman
alması
almamız
almaınız
almanın
almaya
almanı
almada
almadan
almayam
almasan
almadır
almayıq
almasınız
almadırlar
almalarım
almaların
almaları
almalarımız
almalarınız
```

Fig 9. Testing for algorithm

The second feature breaks down the inputted word into initial form and suffixes and tells which part of speech the word belongs to and identifies the types of added suffixes. For completing this task algorithm looks for all possible words, which could be the initial form of this word. Considering the fact that in Azerbaijani language there no grammatical prefixes we are looking for words, which first two or three letters are identical and Levenshtein distance is less or equal to two. For instance, it the inputted word is 'almalar' (apples) the program takes first two letters of the word and looks for corresponding words. For instance, it finds word 'al' (buy) and tries to construct the given word. Until it constructs the given word, it looks for every possible solution and finally returns all the matching words to our input.

Levenshtein Distance:

Mathematically, the Levenshein distance between two strings a,b (of length |a| and |b| respectively) is given by $\mathrm{lev}_{a,b}(|a|,|b|)$ where

$$lev_{a,b} = (i,j) = \begin{cases} \max(i,j) & if\ \min(i,j) = 0 \\ \min \begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i,j-1)+1 \\ lev_{a,b}(i-1,j-1)+1_{a_i \neq b_i} \end{cases} & otherwise. \end{cases}$$

Where $1_{ai\neq bi}$ is the indicator function equal to 0 when $a_i \neq b_i$ and equal to 1 otherwise, and $\mathrm{lev}_{a,b}(i,j)$ is the distance between first i characters of a and the first j characters of b.

The third feature breaks down the input to the syllable level. In Azerbaijani language each syllable must contain only one vowel. All consonants in the beginning of the word belongs to first syllable, and all consonants in the end belongs to the last syllable. In the case when there are multiple consonants between two vowels first consonant belongs to the first syllables and all others to the second. However, when there is only one consonant between two vowels it belongs to the second one. For example: The word 'Elektronik' has 4 vowels: E-lek-tro-nik.

The last but not least function of our library is stemming the text. This functions work principle is the same as the second feature's which is mentioned before. It eliminates all the grammatical suffixes and punctuation marks from the texts and returns stemmed text to the user.

Instead of storing all words in the unit database, we decided to break them down into various folders. The words which consist of three or more letters are divided into groups by their first three letters. Parts of speech of all words are specified in those text files.

In are source code there are two main classes called 'soz' and 'shekilchi' (word and suffix). The 'shekilchi' class consists of five main components:

1. Self.sonra - the list of suffixes that can follow up
2. Self.forma– which type of suffix is the form of the suffix (all the possible forms of the given suffix)
3. Self.qn – the number of possible forms (1, 2 or 4)
4. Self.stat - is there connecting letter inside the word or not
5. Self.adi – the label of the suffix

When the word is inputted into the system it goes through three different stages. In fist stage according to the vowel of last syllable, the system decides which kind of suffix is going to be added. In second phase program decides is there going to be a connecting letter or not. In the final phase the program checks if there is going to be a difference in the word itself, as in our language depending on suffixes the initial form of the word can be modified.

For instance, if the verb 'telefon' is our input, the program identifies that the '-yin4' suffix can be added to that word. When the vowel of last syllable is 'o' or 'u' the '-yun' form is chosen. In second stage, program checks the last letter of the word. As in given situation the last letter is constant, the connecting letter 'y' is dropped. In third stage program checks all the change conditions and as in this case our input is not related to any of them, the suffix is finally added to the word. All the suffixes and their order is stored in arrays.

In the 'soz' class there are six main components:

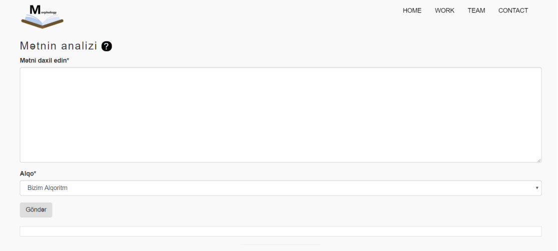
1. Self.kok – the initial form of the word
2. Self.ozu – the full version of the word (with suffixes)
3. Self.hecalar – the syllables of the word
4. Self.saitler – the massive if the vowels in the word
5. Self.nitq – the part of speech of the word
6. Self.sonra – the list of suffixes that can be added to the word

In this class the parts of speech are defined, the letters can be turned into capital letters or vice versa, and we can identify initial form of the word and its suffixes.

Implementation of the Website

Our solution for the website consists of a front-end (web-interface) and the back-end side of the application. The front-end has been written by using HTML, CSS, Bootstrap and Javascript and the back-end side of the application has been developed using python programming language with version 3.6 and Django framework. Django is a free framework which is open source that is written in Python programming language. The fundamental logic for using Django framework was because it's light-weight, fast to read and it's easy to handle requests from the server.

In the web-interface, there are several fields for entering the pieces of text for the input such as analyzing the text with several sentences and words, option tag for choosing the part of speech (POS) for the desired result and the submit button in order to click which the application will send the input texts to the backside of the application via POST request which will detect the word and sentence initial form and produce the result. After the user presses the submit button for the text analysis, stemmed version of the words have to be printed out into the screen. If the user presses to button for the word analysis, the result for the stemmed version and the details for the suffixes should be printed out. You can see the user interface of the morphological analysis website in Figure.



Mətnin analizi

Mətni daxil edin*

Alqo*

Bizim Alqoritm

Fig 10. User interface for the Website

In addition, we have added two more pages for the information about our team and our contacts. In our team page, we have added the members of our team with the links to our social media profiles. Another page for our website is contact page which gives brief information about our team and contact numbers. Additionally, we have added mailing system with Bootstrap and Django for backend in order to send mails if desired.

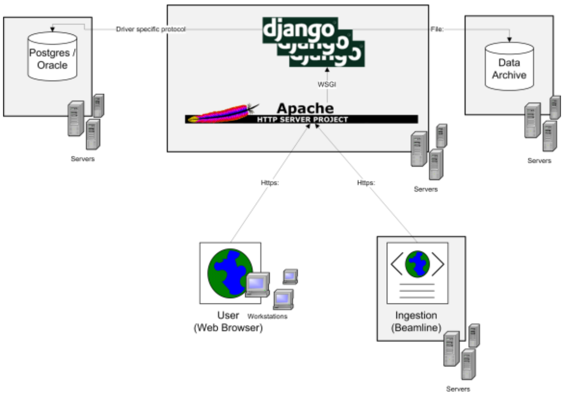
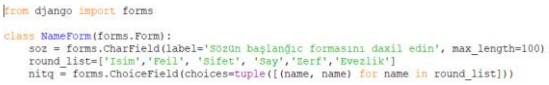


Fig 11. Server side of Django

The server side of our algorithm which is shown in Figure. Consists of a Python Django framework application which uses several libraries of Django and libraries of Python such as numpy, pandas, sklearn and matpotlib to process all the information and return the result. We have published our page on the internet which accepts the port for HTTP which is 80 and accepts the requests such as POST and GET. The index page of our application is the first page that user sees and interacts while opening the website with GET request. However, when the user presses ‘Göndər’ button which is for submitting the request, the web-server with Django accepts it a POST request and sends the forms that are shown in the Figure 11.

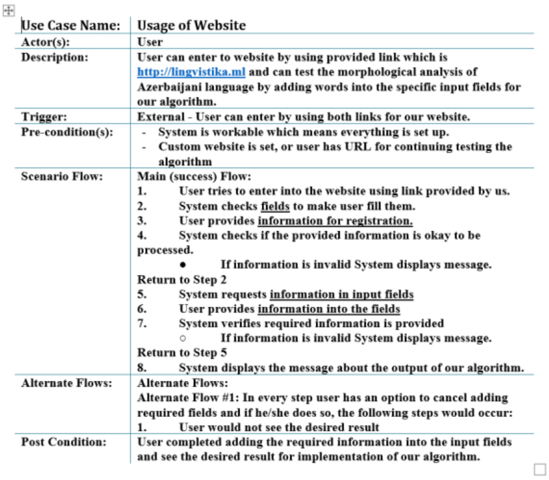


```
from django import forms

class NameForm(forms.Form):
    soz = forms.CharField(label='Sözün başlanğıc formasını daxil edin', max_length=100)
    round_list=['Isim','Feil', 'Sifet', 'Say','Zerf','Evezlik']
    nitq = forms.ChoiceField(choices=tuple([(name, name) for name in round_list]))
```

Fig 12. Usage of Forms in Django

After that the back-end application starts the necessary methods, checks algorithm and posts the necessary output for the website back to the user after acquiring the desired result and user sees the result.



Use Case Name:	**Usage of Website**
Actor(s):	**User**
Description:	**User can enter to website by using provided link which is http://lingvistika.ml and can test the morphological analysis of Azerbaijani language by adding words into the specific input fields for our algorithm.**
Trigger:	**External - User can enter by using both links for our website.**
Pre-condition(s):	**- System is workable which means everything is set up.** **- Custom website is set, or user has URL for continuing testing the algorithm**
Scenario Flow:	**Main (success) Flow:** **1. User tries to enter into the website using link provided by us.** **2. System checks fields to make user fill them.** **3. User provides information for registration.** **4. System checks if the provided information is okay to be processed.** **● If information is invalid System displays message.** **Return to Step 2** **5. System requests information in input fields** **6. User provides information into the fields** **7. System verifies required information is provided** **○ If information is invalid System displays message.** **Return to Step 5** **8. System displays the message about the output of our algorithm.**
Alternate Flows:	**Alternate Flows:** **Alternate Flow #1: In every step user has an option to cancel adding required fields and if he/she does so, the following steps would occur:** **1. User would not see the desired result**
Post Condition:	**User completed adding the required information into the input fields and see the desired result for implementation of our algorithm.**

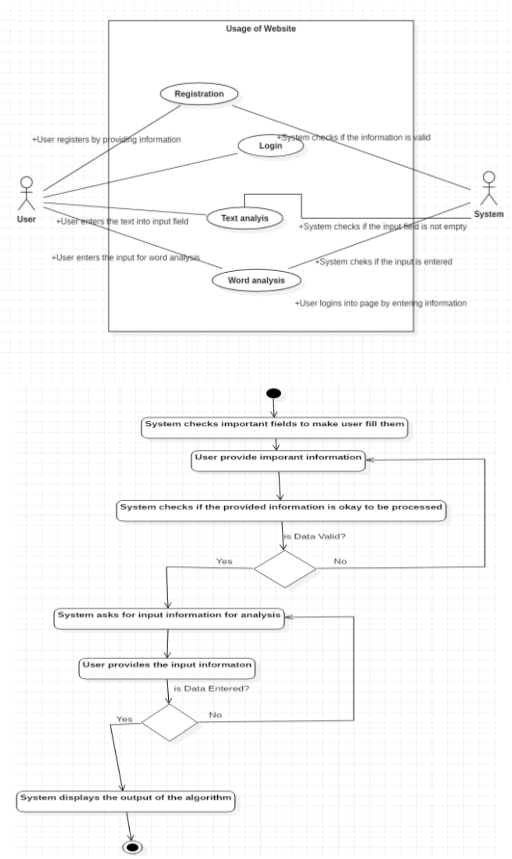

Usage of Website
Registration
Login
Text analyis
Word analysis
User
System
+User registers by providing information
+System checks if the information is valid
+User enters the text into input field
+System checks if the input field is not empty
+User enters the input for word analysis
+System cheks if the input is entered
+User logins into page by entering information
System checks important fields to make user fill them
User provide imporant information
System checks if the provided information is okay to be processed
is Data Valid?
Yes
No
System asks for input information for analysis
User provides the input informaton
is Data Entered?
Yes
No
System displays the output of the algorithm

Fig 13. Usage of Website

You can simply find the source code for our website on the link below: https://github.com/nijatmursali/SDPIIWebsite

The website for checking the functionality of the algorithm is: http://nijatmursali.pythonanywhere.com/

Essential Components of the Project

When the users input word, they have an option to specify its part of speech, but even though they do not the program does it itself. The most essential part of our program is word class, which contains the list of different types of words: initial form of the word (String), the word itself (String), and list of suffixes of the given word (array of suffix class). During the new instance of word is created program checks this word with the list of exceptions (as mentioned before, such as ‘camaat’ or ‘əhali’ words, which do not take plural suffixes even though all the other nouns do) and assigns the list corresponding suffixes to the array and saves them inside the word instance. In this feature as next step the program decides to which list does the word belong and creates new instance of the word class.

Timeline or Grant Chart

Initially, the Azerbaijani words were classified, and dictionaries were improved. Gradually, noun and verb compositions were developed and prepared to be included. Then the algorithms for further implementations are ready. Firstly, we developed algorithms and gathered significant data about morphological and lexical structure of Azerbaijani language. During our researches, we observed that noun and verb have the most complex and challenging structure, therefore we decided to start with them. We gathered and prepared data about the noun and verb and made them ready for implementation. Also, we improved the dictionary and broke it down into categories to boost the performance of our software. After implementation of noun and verb we tested and debugged the written code.

During January and February, we did deep research and gathered information about all primary and secondary parts of speech – adjective, pronoun, numeral and adverb and etc., implemented them and developed a new interface. On March we have devoted to our time for testing and debugging of new features. By the end of April, we are planning to develop a website for our software and make it available for public usage. Moreover, we have got help from scholars from higher institutions such as Azerbaijan University of Languages to make our work more precise and accurate on our project.

Testing/Verification/Validation of results

We selected the random words for testing and algorithm formed all possible versions. Then all the results realized by the program have been manually checked by group members Fidan and Nargiz. All the detected bugs and errors have been corrected and eliminated respectfully. Furthermore, Nargiz and Fidan have chosen some kind of news, tested them and structured through the program functions in order to define the omissions and rectify them. Consequently, all problematic nuances have been amended and improved.

Statistics

We have used 'Obastan' e-dictionary as our primary and Azerbaijani language orographic e-dictionary as secondary data source for our project. Our database automatically derives words from Obastan e-dictionary, which currently contains over forty-five thousand words and their parts of speech. Commonly our library contains 77,740 words, which we have extracted from Obastan.com and Azerbaijani language orographic e-dictionary. Primary parts of speech – noun, adjective, verb, adverb, numeral and pronoun occupy the largest portion of word segment, containing 37,216, 15,639, 17,452, 60, 89, and 2,628 words respectively. In Azerbaijani language there are many words which belong to two or more parts of speech at once, which are known as homonyms. There are 2385 words of this kind, which are counted separately in our library as well. Secondary parts of speech contain postposition, conjunction, particle, modal words, and interjection correspondingly containing 25, 51, 57, 18, and 113 words. Our program generates in total 27,799,420 different forms of words. Respectively noun, adjective, verb, adverb, and numeral generate 12,270,893, 68,39,009, 8,592,435 , 3,694, and 18,709 unique word forms.

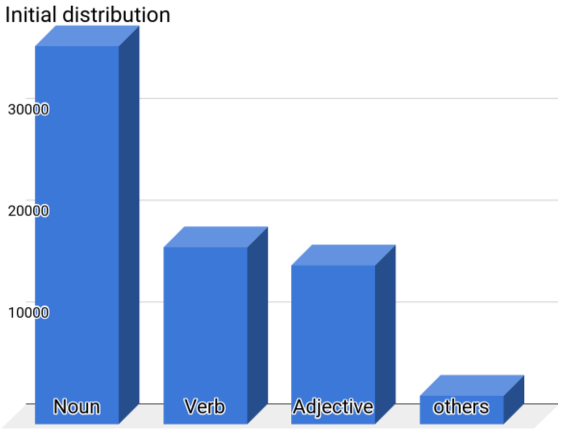


Table 5. Parts of speech in DBW

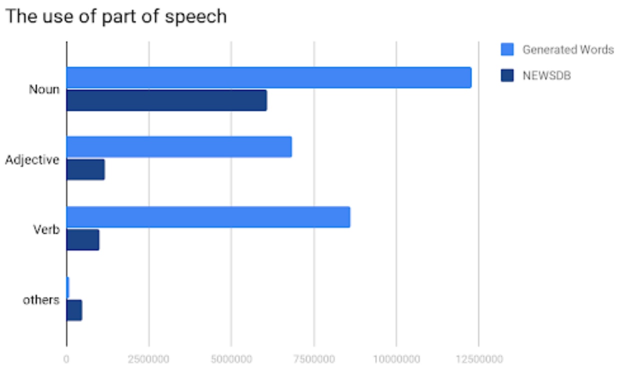


Table 6. Generated Data & NEWSDB

5 Conclusion

5.1 Contribution of the Work

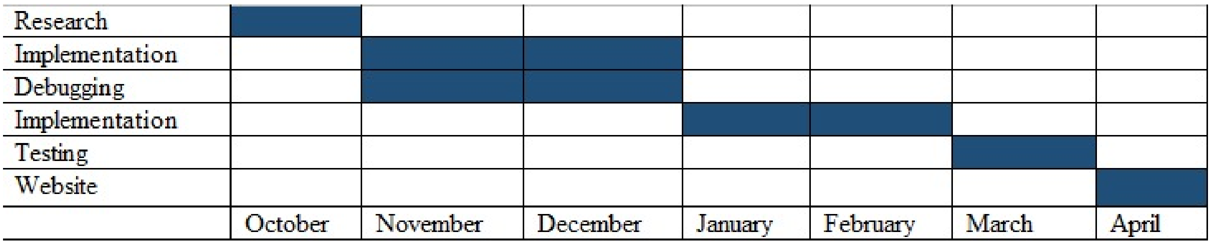
This paper has been created and exhaustive research was done on morphological and lexical analysis of Azerbaijani language by implementing our own algorithm. The accuracy of our algorithm is higher than 70% by implementing it with more than forty thousand sentences. Moreover, we have made a lot of research on several popular algorithms that has been made till now for morphological analysis and tried to implement some of them into our algorithm by also mentioning the positive and negative sides of them and how we can actually develop them more for Azerbaijani language. Although there were lots of algorithms that have been made by several researchers till now, we feel that there are still problems that have to be solved in English and Azerbaijani language as well. We presented several algorithms for checking the morphological and lexical analysis by comparing and mentioning the advantages and disadvantages of them.

Another work we have done was to clean the data by using our own Python scripts and bringing the clean data for our project in order to check how accurate our algorithm works. For cleaning we have also used *nltk* and our sorting approach in order to get rid of simple and messy sentences. In our data, there were sentences in different languages such as English, Arabic, Russian and etc. For solving this issue, we have created the script that contained only Azerbaijani characters and other special characters such as slash, comma and etc. Then we have checked if there's no any sentence with those characters, our algorithm automatically deleted the sentences.

5.2 Future Work

This project is created for the thesis purpose and it's in starting point in the development cycle for morphological and lexical analysis in Azerbaijani language. Although it works as expected, however, it needs to be improved by time by fixing all the current problems that we are facing during this project. During this time, we have analyzed and found several problems in our algorithm. The fundamental problem with our algorithm was that not every algorithm in English was applicable to our language and we had to develop our own algorithm in order to achieve desired result. Another future work for this project will be developing our dataset with Azerbaijani words and applying to the Microsoft, Google and other famous companies in order to develop auto-correction system in Azerbaijani language.

Appendices



Research	■						
Implementation		■	■				
Debugging		■	■				
Implementation				■	■		
Testing						■	
Website							■
	October	November	December	January	February	March	April

Figure 1. Grant Chart

Appendix A.

Part of Speech	Suffix type
Noun (İsim)	Cəm
Noun (İsim)	Hal
Noun (İsim)	Mənsubiyyən
Noun (İsim)	Şəxs

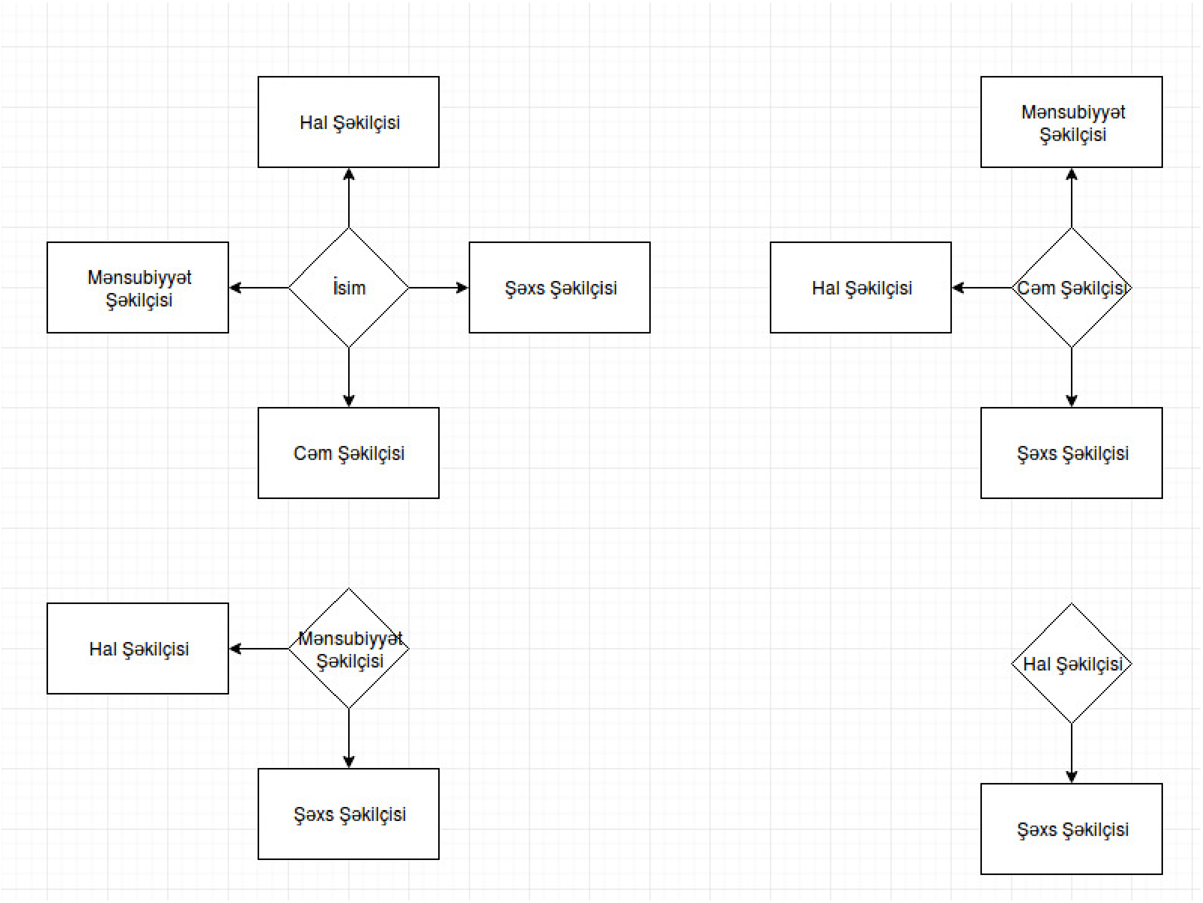
Appendix B.

Suffix	Type	Mode	Used in	Possible suffixes types to be joined (Özündən sonra gələ bilər)
-lar^{2}	Cəm	Cəm	Noun (İsim)	Hal, Cəm, Mənsubiyyət
-nın^{4}	Hal	Yiyelik	Noun (İsim)	Cəm, Şəxs
-ya^{2}	Hal	Yonluk	Noun (İsim)	Cəm, Şəxs(III tək, III cəm)
-nı^{4}	Hal	Tesirlik	Noun (İsim)	Cəm, Şəxs
-da^{2}	Hal	Yerlik	Noun (İsim)	Cəm, Şəxs
-dan^{2}	Hal	Cixisliq	Noun (İsim)	Cəm, Şəxs
-ım^{4}	Mənsubiyyət	I tək	Noun (İsim)	Cəm, Hal, Şəxs
-ın^{4}	Mənsubiyyət	II tək	Noun (İsim)	Cəm, Hal, Şəxs
-sı^{4}	Mənsubiyyət	III	Noun (İsim)	Cəm, Hal, Şəxs
-ımız^{4}	Mənsubiyyət	I cəm	Noun (İsim)	Cəm, Hal, Şəxs
-ınız^{4}	Mənsubiyyət	II cəm	Noun (İsim)	Cəm, Hal, Şəxs
-yam^{2}	Şəxs	I tək	Noun (İsim)	
-san^{2}	Şəxs	II tək	Noun (İsim)	
-dır^{4}	Şəxs	III	Noun (İsim)	
-yıq^{4}	Şəxs	I cəm	Noun (İsim)	
-sınız^{4}	Şəxs	II cəm	Noun (İsim)	
-dırlar^{4}	Şəxs	III cəm	Noun (İsim)	

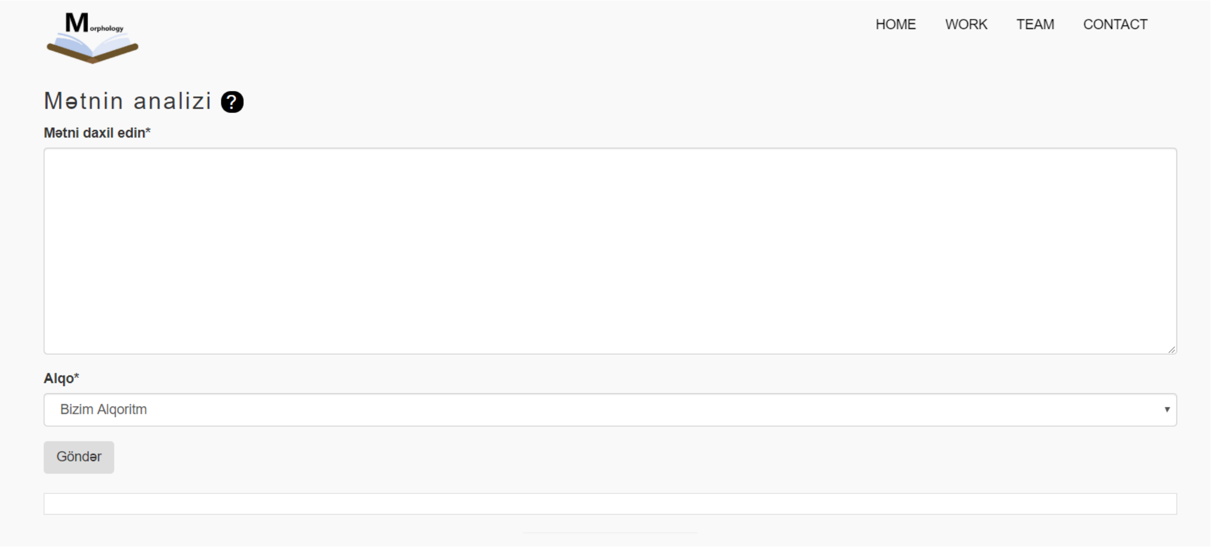
Reference

- http://buludxan-xelilov.com/pdf/azerb-dilininin-morfologiyasi-I-hisse.pdf
- http://www.anl.az/el/Kitab/252289.pdf
- http://www.buludxan-xelilov.com/pdf/sintaksise%20girish%2003.07.2017-A4.pdf
- http://elibrary.bsu.az/books/N_2246.pdf
- http://nlp.jsoft.ws
- https://aclweb.org/anthology/P99-1037

Flowcharts


Hal Şəkilçisi
Mənsubiyyət Şəkilçisi
İsim
Şəxs Şəkilçisi
Cəm Şəkilçisi
Mənsubiyyət Şəkilçisi
Hal Şəkilçisi
Cəm Şəkilçisi
Şəxs Şəkilçisi
Hal Şəkilçisi
Mənsubiyyət Şəkilçisi
Şəxs Şəkilçisi
Hal Şəkilçisi
Şəxs Şəkilçisi

Images of the final product

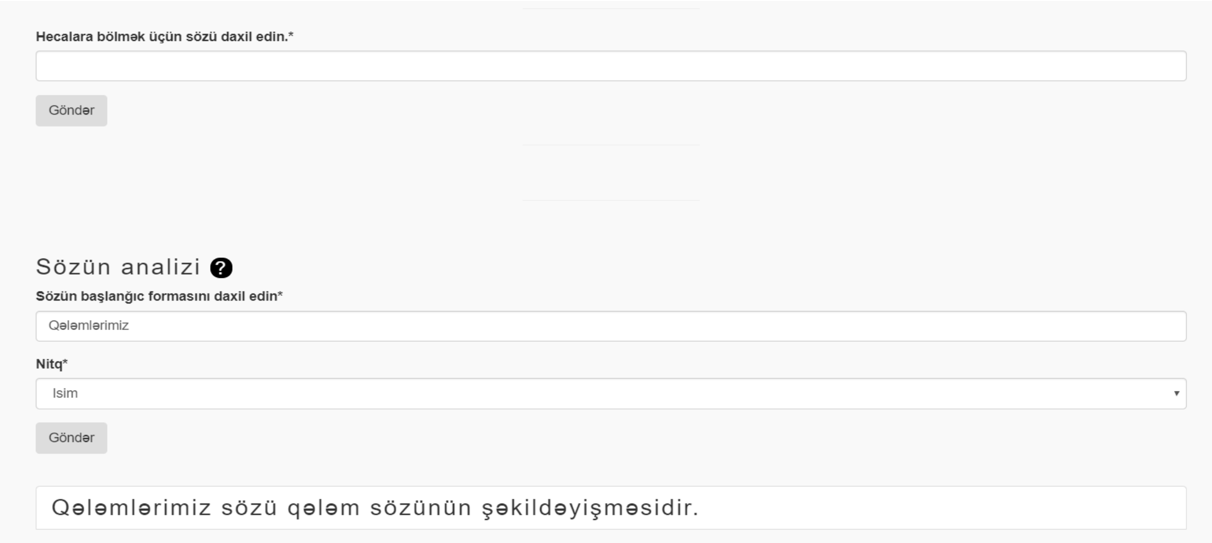


Mətnin analizi ?

Mətni daxil edin*

Alqo*

Bizim Alqoritm



Hecalara bölmək üçün sözü daxil edin.*

Sözün analizi ?

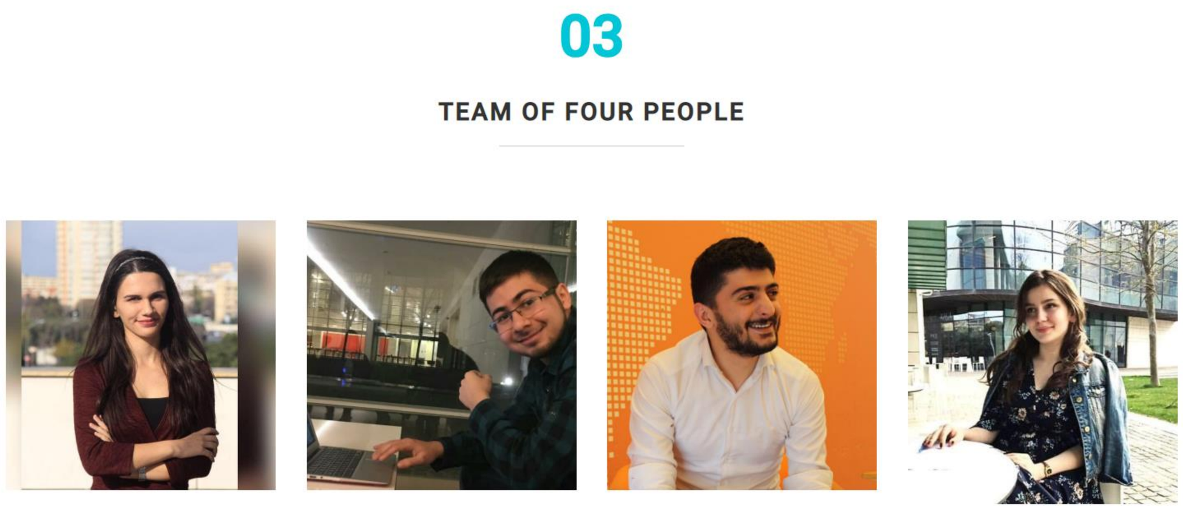
Sözün başlanğıc formasını daxil edin*

Qələmlərimiz

Nitq*

Isim

Qələmlərimiz sözü qələm sözünün şəkildəyişməsidir.



03

TEAM OF FOUR PEOPLE