School of Information Technology and Engineering at the ADA University

School of Engineering and Applied Science at the George Washington University

Development of Large-scale National Azerbaijan Corpus with UI and Functionality

A Thesis
Presented to the Graduate Program of Computer Science and Data Analytics
of the School of Information Technology and Engineering
ADA University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Computer Science and Data Analytics
ADA University

By
Emin Alizada

April, 2023

THESIS ACCEPTANCE

This Thesis by: Emin Alizada
Entitled: *Development of Large-scale National Azerbaijan Corpus with UI and Functionality*

has been approved as meeting the requirement for the Degree of Master of Science in Computer Science and Data Analytics of the School of Information Technology and Engineering, ADA University.

Approved:

(Adviser)	(Date)
(Program Director)	(Date)
(Dean)	(Date)

ACADEMIC INTEGRITY STATEMENT

"I affirm that this is my own work, I attributed where I used the work of others, I did not facilitate academic dishonesty for myself or others, and I used only authorized resources for my Thesis, per the ADA University Academic Integrity requirements. If I failed to comply with this statement, I understand consequences will follow my actions. Consequences may range from failing the course to expulsion from the program/university and may include a transcript notation."

(Full Name)	(Signature)	(Date: DD.MM.YY)

ABSTRACT

This paper presents the creation of a large-scale Azerbaijani language corpus with more than 50 million tokens, and the development of several functionalities for language analysis and corpus linguistics, including Word Frequency, Ngrams, Concordance, Thesaurus, and Word Sketch. The corpus was collected from various sources, including Azerbaijani books, articles, and websites, and was stored in a relational database. The paper provides a detailed description of the corpus creation process and the database schema used to store the corpus, as well as dives into the creation of each of the functionality of the corpus, and what kind of insights it is possible to get from the given functionality set. Afterwards, the paper analyzes different corpus applications and analyzes their interfaces and user experience provided by the application, before introducing the online application for the Azerbaijani language corpus to make the corpus and its fucntionalities available to the linguists, researchers and language learners. The functionalities were implemented using Python, and the user interface was created using Next.js. The final product is a web application that allows users to access all the functionalities of the corpus easily.

Table of Content

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Explanation
NLP	Natural Language Understanding
ML	Machine Learning
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
CPU	Central Processing Unit
GPU	Graphical Processing Unit
HTML	HyperText Markup Language
DBMS	Database Management System
ORM	Object Relations Mapping
POS	Part of Speech
HMM	Hidden Markov Model
RNN	Recurrent Neural Networks
API	Application Programming Interface
CDN	Content Delivery Network
REST	Representational State Transfer
UI	User Interface
UX	User Experience

1 INTRODUCTION

Language is a fundamental tool for communication, and its role in shaping and reflecting a social, cultural, and political aspects of a society is undeniable. The study of language requires access to a large and diverse language resources, such of corpora, which essentially are collections of texts that are annotated, labeled and searchable for linguistic analysis of linguits. A Corpora have become increasingly important for a language research and an education, as they allow for empirical investigations of a language use and variation, and enable the development and evaluation of the language technologies and applications.

Language analysis has a long history dating back to an ancient times. In the past, the language analysis was primarily focused on the study of the grammar, syntax, and the rhetoric, as these were seen as the essential elements of any of the language. Back then researchers analyzed language as a means of expressing thought and persuasion, and developed theories and methods for analyzing the language use in a different contexts. During the Middle Ages and Renaissance, language analysis were set to further evolve, with a bigger emphasis on the study of the semantics and lexicography. Scholars have researched and developed theories of meaning and reference, and created compiled dictionaries and glossaries of various languages.

In the modern era of time, the language analysis has become more scientific and interdisciplinary, with the development of a linguistic theories and methodologies based on empirical research and data analysis. The emergence of corpus linguistics in the mid-twentieth century revolutionized language analysis, by giving a systematic and quantitative approach to the learning of language use and variation. Corpus linguistics have activated linguists to investigate language patterns and structures in large and diverse fragments of text, and to test hypotheses and theories about language phenomena.

Today, language analysis continues to evolve, driven by advances in technology and interdisciplinary collaborations. Natural language processing (NLP), for instance, combines linguistics with computer science and artificial intelligence to develop systems and applications that can understand, generate, and manipulate human language. The language analysis is also increasingly and sharply applied to social and political issues, such as discourse analysis of media and political texts of any kind, and language variation and change in multilingual societies.

Corpus is one of the most efficient tools for NLP (Natural Language Processing) and computer based linguistics. It is not a novel concept to use collections of texts when studying a language. Making lists of all the words in a book, together with their contexts—a process known today as concordancing—began in the Middle Ages. Other researchers tallied the word frequencies in individual texts or sets of texts and created lists of the most common terms. With one significant exception, they did not utilize computers. Language acquisition, syntax, semantics, and comparative linguistics are a few fields where corpora have been employed. Even if the word "corpus linguistics" was not used, most of the work was comparable to the type of corpus-based research we conduct today. Nowadays, the presence of vast volumes of textual data on the Web stimulated linguists' desire to collect language usage samples from webpages. At the same time, we now have the long-awaited possibility to handle a vast amount of digitally existing data thanks to the dramatic growth in storage capacities of advanced technologies.

Recent advancements in a data science, algorithms, and machine learning (ML) have greatly impacted a lot of industrial fields including the field of corpus linguistics, enabling researchers to build and create better language corpora than created ever before. These advancements have been driven by several factors, including the availability and afforability of more powerful and cheaper computing resources, such as CPUs and GPUs, as well as the decreasing cost of data storage compared with the past.

One key area of advancement has been in the development of machine learning algorithms and techniques for natural language processing. These advancements have activated a more efficient and an accurate processing of the large and diverse language data sets, leading to a more far-reaching and representative language corpora. Furthermore, deep learning techniques, such as CNNs and RNNs, have allowed for the development of a more complex and an accurate language models, improving the quality and accuracy of a language data analysis.

Advancements in computing technology have also played a significant role in the development of better language corpora. The increasing availability of cloud computing services have enabled researchers to be able to process and analyze language data more quickly and efficiently, without requiring thst significant investment in hardware and the infrastructure. In addition to it, the decreasing cost of computing resources, such as CPUs and GPUs, and data storage both physically and a cloud based, has made it a more affordable for researchers to access and use powerful computing resources for creating language corpora.

In Azerbaijan, there is a growing need for a large-scale national corpus that represents the diversity of Azerbaijani language and culture and can serve as a valuable resource for language-related research, teaching, and technology development. However, there is currently no such corpus available that meets the standards of modern corpus linguistics, including systematic and representative sampling, rich and accurate annotation, and user-friendly access and visualization.

This thesis aims to address this gap by presenting the development of a Large-scale National Azerbaijan Corpus with UI and functionality. The Large-scale National Azerbaijan Corpus will be a comprehensive and balanced collection of Azerbaijani texts, covering a wide range of genres, domains, and regions, and will be annotated with various linguistic and metadata information. Moreover, the Large-scale National Azerbaijan Corpus will be designed with a user-friendly interface and advanced functionality, allowing for efficient and flexible exploration, analysis, and visualization of the corpus data.

Overall, this thesis contributes to the development of Azerbaijani language and culture by creating a valuable resource that can benefit various stakeholders, including linguists, educators, language learners, and technology developers. The Large-scale National Azerbaijan Corpus also contributes to the advancement of corpus linguistics and natural language processing by showcasing innovative methods and techniques for corpus creation and exploration.

1.1 Definition of the problem

Despite, the recent great advancements in corpus linguistics, there is a still significant lack of large-skale, high-quality language corpora for a lot of the languages, in the lines of which including Azerbaijani. Right now, there is many corpora for different languages, for English there is American National Corpus (ANC), The British National Corpus, Oxford English Corpus, Corpus of Contemporary American English and many more, for Russian there is Russian National Corpus, and a lot of languages have their own version. For Azerbaijani language I have found a couple of corpora that were mostly done not by Azerbaijani linguists and contain the content only from web resources, also some of them were produced by changing a language that is close by meanings to the Azerbaijani language. As an example, azWaC: Azerbaijani corpus from the web[1] is based on Turkic Web Corpora[2]. Another example would be Azerbaijani news corpus by University of Leipzig[3,4] which crawled all of its data from news and wikipedia, although it contains around 9 million tokens as all of its content is web based this creates a bias and unclearness around the corpus as it doesn't include spoken language or wide variety of genres. As a result although there are several different implementations of corpora they are mostly based on webpage crawling or Turkish language family base, there is a need for a corpus which has a balanced set of tokens and the cooperation with local linguists.This fact that there is a no good quality really large scale corpus that was prepared in cooperation with the local experts and linguists poses a major challenge for researchers and practitioners in the field of natural language processing, as high-quality language corpora are the very essential for developing the accurate and the efficient language tooling and technologies, such as machine translation and different kind of analysis. Moreover, the development of large-scale national language corpora with a user-friendly interfaces and advanced functionalities can enormously benefit the language-related research and applications, enabling a more comprehensive and a representative language analysis. Therefore, there is a need to develop a large-scale national Azerbaijani corpus with advanced set off

functionalities and user-friendly interfaces to address this challenges and pave the way for the new language-related researchs and applications.

1.2 Objective of the Study

The main objective of this study is to develop a large-scale national Azerbaijani corpus that shall address the current gaps in the availability of high-quality language corpora for Azerbaijani. This objective will be achieved through a series of specific goals that include:

1. Corpus Design and Collection: The study will aim to design and collect a corpus that is representative of the Azerbaijani language, includes a balanced set of tokens, and covers a wide variety of genres and domains. To achieve this goal, the study will employ a mixed-methods approach, combining corpus design, collection, and annotation techniques, with software engineering and user-centered design principles. This approach will ensure that the corpus will be really high quality, and will have a clear and a consistent annotation, also it should be very scalable for the future updates and improvements.

2. User Interface and Functionality: The study will provide a user-friendly interface that allows researchers and practitioners to access and analyze the corpus more effectively. The interface will have advanced functionalities such as search, filtering, and visualization tools that enable researchers to analyze the corpus based on various criteria. Additionally, the interface will include tools for corpus exploration, such as concordance, ngrams, and frequency analysis, that will allow researchers to investigate the patterns and features of the Azerbaijani language.

3. Evaluation of Corpus Quality: The study will try to evaluate the qualiti, usability, and the usefulness of the developed corpus with using a various criteria and the metrics. This evaluation will include measures of corpus size, token balance, annotation consistency, and coverage of different genres and domains. The evaluation will try to show the potential of the corpus for various language-related applications and will provide insights into the strengths and limitations of the developed corpus.

4. Advancements in Data Science and Machine Learning: The study will leverage recent advancements in data science, algorithms, and machine learning to ensure that the corpus is of high quality and is representative of various genres and domains.

In summary, the main objective of this study is to develop a large-scale national Azerbaijani corpus that is of high quality, and useful for various language-related applications. The study will employ a mixed-methods approach, leverage recent advancements in data science and machine learning, and provide a user-friendly interface with advanced functionalities. The evaluation of the corpus quality and usefulness will demonstrate the potential of the corpus for various language-related tasks and provide insights into the strengths and limitations of the developed corpus.

1.3 Significance of the Problem

The great significance of the given problem is extremely huge and has few implications for different groups of people, including but not limited to researchers, language enthusiasts, language technology developers, and of course the general public.

For the researchers, a high-quality Azerbaijani language corpus could be a great beginning foundation for a several of linguistic and computational research problems. It also can help the researchers to explore the different aspects of Azerbaijani language, like its syntax, structure, patterns, discourse

structure, and language variation. The researchers can, as well, use the corpus to test various linguistic theories and hypotheses, or to study how the language has changed over periods of time, and to compare Azerbaijani with other close or completely different set of languages. In addition to that, a dependable, trustworthy and representative corpus can also enable researchers to develop computer models that can simulate human language processing and generate natural language outputs. That kind of models can be used in applications like text classification, sentiment analysis, and machine translation.

The big importance of this problem extends beyond the academical insitutes and research communities, it also has implications for language enthusiasts who want to learn and use Azerbaijani in a better way. A high-quality language corpus can provide a rich and diverse source of language material that can help them develop their language skills, expand their vocabulary list, and gain a deeper and full understanding of the language's structure and usage in different circumstances. It can also help preserve the language and its cultural heritage by documenting the diversity of the language and its usage across different domains and genres.

Language technology developers can also benefit from having a reliable and demonstrative language corpus. They can use the created corpus to train, test and publish several language models, as kind of natural language understanding and generation models, and to develop language technologies that can be used in different domains, like education, health, finance, technology and other sectors. Furthermore, the availability of a high-quality corpus can attract more investment and innovation in language technology development, which can create new job opportunities and contribute to the economic growth of the country.

Last but not least, the general public can also benefit from having a high-quality Azerbaijani language corpus. It can help promote the usage and development of the language in different subdomains, like education, media, and communication. It can also help enhance the cultural identity, diversity and multicultural nature of the country and provide a valuable resource for the upcoming future generations to learn about their language history and cultural heritage.

1.4 Assumptions and Limitations

The assumptions and limitations of the Large-scale National Azerbaijan Corpus project are important to consider at the beginning as they can impact the scope and quality of the corpus drastically. As such, in my opinion it is crucial to address and flag these potential limitations to ensure the further validity and the reliability of the whole corpus.

The one significant assumption of the given project might be that there will not be a sufficient amount of language data available for collection and therefore annotation. While there might be numerous various sources of the language data, not every single one of the may be accessible or even suitable for the including in our the corpus. We will need to carefully consider the availability and more importantly the quality of the data sources that we find and decide to use after all, as well as the practicality of collecting and annotating the data itself.

The other possible limitation to take into account might be that the linguists and other team members that are taking part in the project have the required knowledge and expertise to effectively and fully correctly annotate and analyze the language dataset. It is extremely important to ensure that the linguists team are having enough knowledge about the language, underlying culture, and the context of the corpus to minimize the possibility of the errors and inaccuracies in the process of annotating the dataset.

The use of the computational tools and the methods is also important assumption for the project. The project team will have to ensure that the selected set of the tools and the methods are actually appropriate for the data that is being analyzed and that they have the necessary computational resources to carry out the analysis effectively. In addition to that, the computational tools and the methods must be latest available in the industry to make sure that the corpus remains relevant in the extremely rapidly evolving field of natural language processing.

There are also limitations to the project that must be considered. One of such limitation is that the corpus may not be comprehensive enough in the terms of language variation and different domain coverage. It might be difficult to actually capture all of the aspects of the Azerbaijani language and its various domains in the corpus, especially taking into the consideration that one of the main domain as a spoken language data is hard to get.
The subjectivity and the bias inherent in the annotation and the analysis process might be considered as another possible limitation to the corpus creation. Even considering the best efforts to minimize subjectivity and bias, it is impossible to completely eliminate them, as long as human factor is included in work. Therefore, the team will need to carefully consider the theoretical frameworks and assumptions guiding the annotation and analysis process.
The time and the effort required for the linguistic parsing of the collected data is one of the limitations of this study. The process of linguistic parsing is a crucial step at ensuring a quality and an accuracy of the corpus, as it involves the annotation and analysis of the actual linguistic features that are the present in the collected data. However, this process is both time-consuming and drastically resource-intensive, and it actually requires a significant amount of the manual effort from trained linguists. Despite this big limitation, it is genuinely believed that the real benefits of having a high-quality, linguistically-informed corpus by far outweighs the additional time and resources required for its creation. By working with local Azerbaijani linguists, we can ensure that the corpus is representative of the language as it is actually used in everyday life, and that it includes a very diverse range of genres, registers and domains. This, in turn, will enable for the research to be more accurate and meaningful analyses of the language, leading to a deeper understanding of its structure, usage, and evolution over time. It is also worth noting to note that while the linguistic parsing process is highly time-consuming, it is at the same time an essential component of the corpus creation process that can not be omitted. Without this step, the corpus would lack the depth and details that are required for the meaningful language analysis. Therefore, while this limitation is acknowledged, it is not considered a reason to forego the creation of a high-quality Azerbaijani language corpus.
Additionally, the evaluation of the corpus quality, usability, and usefulness may be limited by the choice of evaluation measures and the sample size.
Despite these assumptions and limitations, the development of the Large-scale National Azerbaijan Corpus is a significant contribution to the field of natural language processing and linguistic research. By carefully considering and addressing these potential limitations, it is possible to produce a high-quality and useful language resource for researchers, language educators, industry professionals and the general public.

2 LITERATURE REVIEW

The research for the corpus composition and usage have been carried for many years, since the very first electronic corpus - The brown corpus[8]. The corpus consists of one million words of American English texts printed in 1961. The texts for the corpus were sampled from 15 different text categories to make the corpus a good standard reference. Today, this corpus is considered small, and slightly dated. The corpus is, however, still used.
In more recent times, the Institute of the Czech National Corpus has been updating their corpus with the help of the web content which they described Building a Web Corpus of Czech[9]. The paper describes the process from the motivation till the initial results that were achieved by the team that was working the creation of the corpus. Their motivation was very similar with ours with a slight difference that the already had a corpus, but it was outdated and the recent advancements in the technology and the affordability of the storage have ignited their passion to start the process over to get the modern corpus with all the lates information in it. Their process started from the choosing appropriate web crawler since the biggest chunk of information for any corpus is coming from the web resources having a web crawler adapted to the need of the team is an essential first step for the team that decides to work

on the creation of the web corpus. They didn't want to lose the time by creating their own web crawler from the scratch so instead they listed all available web crawlers for that moment and started to look to their features in terms of speed, accuracy, flexibility to change the crawling technique according to the teams' needs, how the crawler cleans the HTML that it gets from the web source. After the investigation they discovered that the features that they need are not present in an available engines such as near-duplicity elimination and web-page revisiting, so they decided to stop on Egothor engine[10] which simply crawl the webpage and the remaining functionality was added by the team to accommodate the project needs. They further cleaned the data and applied functionality that was available in their previous corpora, as the final results they compared old corpora with the new one which had 50 million tokens now.

The database structure and the reasons to choose the database as a storage for the corpus instead of plain txt or XML file for Corpus of Contemporary American English was described by Mark Davies[5,7]. The modularity of the database structure among with the indexes that are used to make the queries faster are among the major benefits of using database as the storage for the corpus compared to its competitors. In that paper it is also described the user interface part of the application that was presented to the users and the way to interact with the corpus.

The paper[5] by Mark Davies also emphasizes several reasons of having a lot of different domains in the corpus. It is described that, having diverse range of the domains in the corpus allows for a more comprehensive understanding of the language and also its usage across the different contexts. For example, a corpus that includes texts from different domains such as literature, news, social media, and technical writing, provides a more accurate representation of the language and its usage in different fields. It is also mentioned that, having only limited number of genres and domains in the corpus might affect its usages in the future by other researchers, as an example if later on, different set of researchers decide to use the corpus with the limited number of sources or domains in their research related to development of the more robust language models or machine learning algorithms it can have negative impact on the results of the research, as well as tasks like text classification, sentiment analysis, and machine translation based on this might not perform as expected.

Corpus of Contemporary American English had an enormous advantage in the topic of gathering of the diverse domain dataset as it was in English there are plenty of electronic information of any genre in English right now. As it is described in [5], for spoken language they could use CNN transcripts of the shows or the anything that aired on that channel and this set of information is available at least back till 2000s, as in case of written information all the newspapers now also have the electronic distribution as well, so as an example New York Times has an archive of the articles in various genres all available in online to be parsed. Compared to our case, unfortunately our television doesn't keep the transcripts of their shows and it is not publicly accessible to be parsed, so it creates a bit of trouble of getting spoken language data as easy as it was for the Corpus of Contemporary American English.

In another paper by Roland Schäfer, Felix Bildhauer [11], they describe a general way of building a large corpus by using different web techniques. Their approach to collecting the data was only from web and the way they were selecting their data was through the web queries to the search engines, they were using Yahoo API till the end of the free API access and then switched to the Bing with the approximately same logic. Their main challenge was to make sure that they don't include the same document twice in the corpus and the same pieces of the text should also be filtered. As their web crawler they were using Heritrix 1.4 configured as in the paper by Emerson and O'Neil [12], afterwords they were using clean up techniques to get rid of HTML tags in the crawled webpages with simple preprocessing and when light preprocessing was finished, they now needed to make sure to exclude common parts of the websites and keep only the main content, and even this wasn't the end of the processing, the final part of the processing of the crawled and the initially cleaned webpage was making sure that the content is not a duplicate and it is new for document. It was done with the help of the tooling in two steps. The first step was by using texrex tool to remove perfect duplicate removal and then for the near duplicate removal they were using technique of w-shingling described in the paper by Broder et al [13]. Finally, after duplicate removal and making sure that the contents of the document

are new to the corpus it was added to the corpus, and this was the process of creating a corpus that was described in the paper [11].

3 RESEARCH APPROACH AND FUNCTIONS

The primary goal of this thesis is to create a large-scale national corpus of Azerbaijani language that contains around 10 million tokens. The target of 10 million tokens is set to meet the standards of the modern corpuses, that can be used to as a robust of the language and can be considered as a representative sample of the Azerbaijani language. This kind of target is also used considering future research area based on our research which includes but not limited to analyzing contexts of the words, using the created corpus for the sentiment analysis, generating new text, summarizing the text pieces using natural language processing techniques, machine translation, classification of the text. These kind of future opportunities that are born from creating a corpus are the reasons that are validate the ambitious target of having 10 million tokens by the end of the research. Achieving this goal of number of tokens is not the only primary goal, while the goal is to have large corpus is essential, it also has to be of the high quality and in terms of the corpus the quality of the corpus is measured by the number of tokens and the domains that this tokens are collected from, so having balanced token is another primary goal of the research.

3.1 Categories of the data in the corpus

In order to achieve the balance in the domains of the text, we have initially planned the distribution of the documents according to the categories and sub-categories. According to the plan large scale corpus of Azerbaijani language is going to be composed from six different categories and each of the categories should make 16% of the size of the corpus in order to make it well balanced. The categories and the percentage of the corpus is shown in table 1.

Table 1: Categories and the ratios of the corpus

Category	Ratio
Spoken	16%
Fiction	16%
Literature	16%
News	16%
Academic	16%
Web	16%

Each category is carefully revised for its unique features and these features are the reason why we decided to include them in the corpus and give them their respective ratios.

The first category is spoken language, spoken language features different style and uses the phrases which can not be found in any other category, it makes it differ from the written language and sometimes it also features the phrases that are not even in dictionaries, and for these reasons it is extremely important to pay close attention to have this category in any corpus in any language. In terms of the retrieval of the content for this category, requests were send to the local television and radio channels for the manuscripts of their shows which are probably recorded and saved electronically. Also as the other source for the spoken language it might be manuscripts of the live interviews.

The second category is fiction and it will include examples from many genres of fiction, including plays, short stories, and novels. This classification is crucial because fiction frequently employs unusual language patterns and literary elements that are uncommon in other genres. Also it shows the different side of the language as the phrases from this genre are either rarely used or entirely not used by other categories of our corpus, and it makes it important to make sure that this category is having a fair

proportion of the whole data. In terms of the retrieval of this domain, the fiction genre is going to be retrieved from electronic version of fiction books by local authors that are freely available on the internet or from the books in this genre that has been translated to Azerbaijani language by local authors and made available to the public.

The third category is literature, this category will concentrate on non-fiction books including biographies, memoirs, and essays. Because of the fact that it includes a variety of registers and the writing styles that are distinct from those used in fiction, spoken language or academic writing, this category is significant for balancing the corpus and adding the depth of literature phrases to the corpus. In terms of the retrieval of this category, similarly to the retrieval of the fiction category this category is also planned to be retrieved from the web resources that are publicly available and free to use.

The news category, which is the fourth, will include examples from various news sources, including newspapers, news websites, and news broadcasts. The language used in current events is captured in this category, which is significant since it enables examination of the language used in journalism and reporting. Also this category is the easiest to obtain thanks to the overwhelming online resources that are posting the news on different websites and the social networks, as mentioned although it is the easiest source to get the texts from it might be appealing to use it as the primary source of the data, but in order to keep the whole corpus balanced and not make it biased in any category, the ratio of sixteen percent should be followed. It’s also worth to mention that all corpora in Azerbaijani language by foreign creators are using this category as their primary source of the text and this is why usually their corpus is highly unbalanced.

The fifth category of the corpus is academic texts. The corpus's inclusion of academic texts offers an extensive amount of information on the particular vocabulary and grammar used in writing for academic purposes. This category will assist linguists in locating the unique vocabulary and grammatical constructions that frequently appear in writing for academic purposes. Additionally, it will shed light on how academic writing styles and tones can change based on the subject matter. Language students as well as educators who want to develop their abilities in writing and conducting research in academia may find this information particularly helpful. The academic section of the corpus will also enable study on the development of academic language across time, offering insights into how academic language has evolved and developed. The corpus will be able to capture a wide range of academic language by integrating a variety of disciplines within this category, offering a rich and diverse dataset for study. The academic category of the Azerbaijani language corpus will be sourced from digital repositories of local institutions and academic organizations. These sources will provide a diverse range of scholarly materials, such as academic articles, conference papers, and dissertations, across various fields of study. In addition to capturing the language used in academic writing, this category is also valuable for studying the development of specialized terminology in different academic disciplines. By including a substantial portion of academic texts, the corpus will provide insights into the language used in scientific and academic discourse in Azerbaijani, which can contribute to a deeper understanding of the country's intellectual history and academic traditions.

The final category in our corpus is web resources. It seems likely that the "web" category will offer a wide variety of instances from many different websites, covering social media sites, online discussion boards, blog posts, and other internet-based resources. The latter group is essential for capturing the modern vocabulary used in online interactions and shows how language has changed in the age of technology. This category is becoming more crucial for linguists and scholars to investigate language use in a range of circumstances as social media channels as well as online communities are used more frequently. The vocabulary used in online interactions tends to be usually informal and is frequently enhanced with new slang and idiomatic terms, which can offer insightful information on the development of language. Consequently, this kind of information is an essential resource of data for linguists and scholars to examine the present usage of language in digital communication.

The development of this extensive dataset will be a useful tool for academics, for linguist and language experts who seek to research the Azerbaijani language. The corpus will provide a wide range of linguistic variants and structures by drawing from various areas and sources, enabling a greater comprehension of the nuances and complexities of the language. The creation of a superior corpus that is grounded in linguistic knowledge can also help to preserve and promote the Azerbaijani

language and its rich cultural history for a long period of time. Also having this kind of rich context and wide domains of the texts will surely be beneficial for language enthusiasts and the educators.
The spoken genre will be broken down into a number of subcategories, such as TV show programs, local theater scripts, native or foreign movie scripts, and radio show scripts. These sections will each offer insights into the vocabulary used in various spoken communication contexts.
Similarly, the news category will be subdivided into political news, economic and financial news, cultural news, religious news, sports news, and scientific news. This subdivision will enable researchers to analyze language patterns and usage in different news categories and genres, such as journalistic writing, editorial writing, and feature writing.
The academic category will be further divided into scientific papers, technological academic papers, medicinal dissertations, law and political academic publications, philosophical papers, psychological publications, and historical and geographical publications that were published in Azerbaijani language in institutions and academic organizations. This subdivision is crucial for researchers who are interested in analyzing the language used in academic writing across different disciplines.
Finally, the web subcategory of the corpus will include samples from various web pages such as Wikipedia and blog pages. This category will provide an opportunity to study language usage in online communication and social media, which is a rapidly evolving field of linguistics. By including these different subcategories in our corpus, we aim to create a comprehensive and diverse dataset that can be used for various linguistic studies.

3.2 Storing the corpus

Language corpora can be kept in many different ways, from straightforward text files to sophisticated database structures. The selection of storage format is influenced by the corpus's size, the different kinds of data it holds, and the purpose for which it is to be used. The plain text format, which is straightforward and commonly used and can be simply read and edited by a numerous kind of software tools, is one of the typical formats for language corpora. Text analysis, language modeling, and machine learning may all be done with the help of plain text corpora, which are simple to construct and maintain.
Another common format for language corpora is the XML format, which is a flexible and extensible markup language that can be used to encode complex structures and metadata about the corpus. XML corpora are often used for linguistic research, where the annotations and metadata associated with the text are as important as the text itself. XML corpora can be used for tasks such as named entity recognition, part-of-speech tagging, and syntactic parsing.
In addition to plain text and XML, language corpora can also be stored in more specialized formats such as the Corpus Encoding Standard (CES) or the Text Encoding Initiative (TEI). These formats provide more specific guidelines for encoding linguistic data, and they are often used in linguistic research projects. The CES format is designed for storing spoken language corpora, and it provides guidelines for annotating audio and video recordings with linguistic data. The TEI format is designed for encoding literary and historical texts, and it provides guidelines for annotating texts with structural and linguistic data.
The final option is to store language corpora in relational database formats like MySQL. Database corpora can be utilized for tasks like corpus analysis since they make it easy to search and query the corpus data. The setup and upkeep of database corpora can be more difficult than that of plain text or XML corpora, and specific software tools are needed to access and manage the data.
The choice of corpus storage format depends on a variety of factors, including the size and complexity of the corpus, the types of data it contains, and the intended use of the corpus. Database formats provide efficient searching and querying of corpus data but require more work to set up.
After consideration of all of these storing techniques of the corpus in a various manners, the database option is going to be used as the primary way to store the data with the combination of plain text option. Later on, when we will check out the functionality that is available in the corpus both of these choices will be justified.

Storing the Azerbaijani language corpus primarily in a database, has many benefits over alternative storage options. Databases are the best option for managing corpus data since they are made to store enormous amounts of data in an organized and effective way. All of the corpus data can be kept in a database in an organized manner that makes quick retrieval and analysis possible.The decision of maintaining the corpus data in a database management system (DBMS) has its own advantages. The database management system will enable the development of tables that can house the corpus data and offer resources for data querying and manipulation. As a result, researchers will be able to quickly search for and obtain particular corpus items based on their study requirements. Employing a database will also help to enable for the application of numerous security measures to protect the corpus data. To prevent data loss in the event of a system failure or other unanticipated situations, access to the database can be limited to approved users, and backup and recovery methods can be configured. Having a database makes it simple to manage and update the corpus data. It is simple to incorporate new data into the existing database as it is gathered and added to the corpus, rather than doing all the work from the beginning. By doing this, the corpus is kept up to date and relevant and future improvements may be quickly applied without having to completely redo the corpus.
In our database model, initially we were planning to have 5 tables inspired by the architecture described by Davies[5,6]: *mainCorpus*, *dictionary*, *sources*, *author*, *synonyms*. However, later on, we also added *ngrams* table to our database table list.

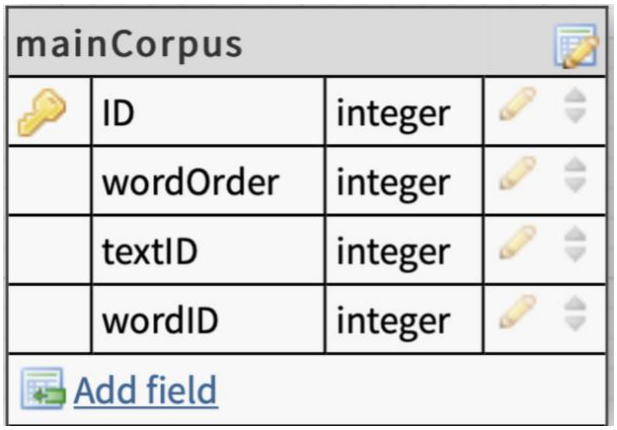

Figure 1: Database table - mainCorpus

The main corpus table structure is illustrated in the Figure 1, and it is going to be the biggest one due to the fact that it is going to have a single row for each of the token in the corpus in sequential order which means that if we succeed with our goal of ten million token corpus then this table is going to contain 10 million rows. The main corpus table is going to be designed in a way that allows easy and efficient querying of the corpus data. The corpus data can be queried for a specific word, phrase, or pattern. The mainCorpus will have an **ID** column as the primary key which demonstrates the order in which each word appears in the corpus as from 1 to overall number of tokens, a **wordOrder** column with the integer value which will demonstrate the order of the word in the text allowing for easy analysis of sentence structure and syntax. This information can be used to identify patterns in the language and to identify grammatical and syntactic errors in the data. Next one is **textID**, which is an integer value which refers to the texts number of the corpus, and last column is a **wordID** is also an integer value which indicates unique words of the whole corpus.

The second table of our data model is dictionary table and it is illustrated in figure 2, this table is designed to keep record of all unique words of the corpus which means that number of records in this table is going to be significantly less than in the mainCorpus table. It consists of 5 columns. The first column of the table is going to be **wordID**, it is an integer value and is also the primary key of the

table, it is going to contain the number of the unique words in the corpus, it will range from 1 to amount of distinct tokens. The second column - **wordFreq** is an integer value and is responsible for keeping track of frequency that the word does appear in the complete corpus. The next column in dictionary table is **wordForm**. It is a string value in which the word does appear in the text without any changes. The other column of this table is **wordLemma**, it is also a string value and is responsible for keeping lemma of a word, basically the base version of the word. We also need to understand this column might have a lot of duplicates cause each lemma might have several word forms. As an example let us review the verb take as a lemma, it has several word forms, to name a few of them: took, taken, takes and etc. The last column of the dictionary table is **wordPos**, this column keeps track of the part of the speech of the word in the given context, to get better results part of speech tagging is planned to be executed in semi automatic way with the help of linguists. As for the logic of the table, each time the new token is going to be inserted into the database, we will first look the word in dictionary table and if it is found in dictionary table we will just increase wordFreq value, and if it is not found and it is the first time we encounter the word in this form the the new record is going to be inserted to the table with wordFreq value of one.

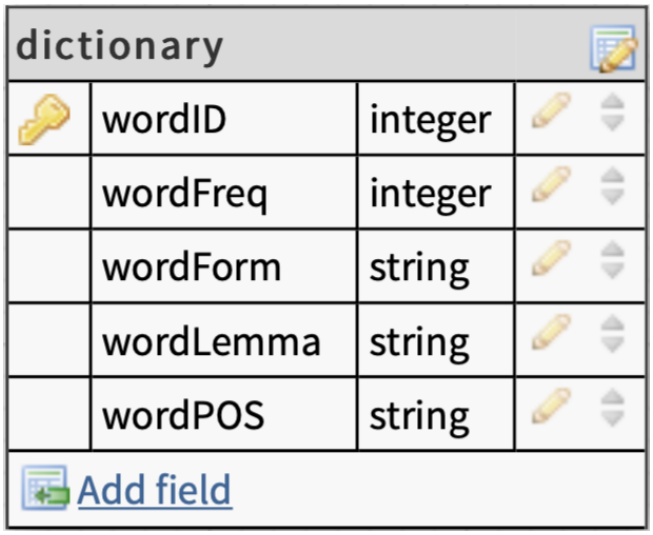

Figure 2: Database table - dictionary

The next table in our data model is sources table and it is illustrated in figure 3, it is responsible for keeping track of all of the sources of our text in the corpus. It keeps track of the text related fields and the metadata that is attached to the text. The sources table has eight columns. The first column of the sources table is **textID**, it represent all distinct text resources in our corpus and keeps the integer value to differentiate them. The second column of the table is **textDate**, the purpose of this column is to keep the date when the text source was first published. It is very important column to have because of the fact that it will help linguists analyze how the language resources are changing over time, to compare the language versions in periods of time and to compare to different languages at the same or different periods, so keeping track of the text publication date is one of the most essential records to keep track of. The third column of the sources table is **textSize**, it is an integer value which identifies the number of tokens that the text does contain. This also might be very useful for specific types of analysis due to the possibility that it can show the relation between the text genres and the text size or how sparse or how rich specific category is. The next column is the **textCat** column, it shows to which category does the text belong to and the other column of textSubCat represents the subcategory of the given category to which the source text belongs to. The following column in the source table is **textTitle**, it will save the title the text piece for future references. **TextMedium** is the next column of

the sources table, and it will keep the data of from which medium the text source is taken from. It might vary from books, radio, tv, call, script, web or other sources. It is also very important to keep this kind of meta information about the text due to the fact that it can show different aspects of each medium. Lastly, **authorID** keeps an integer number of the author who this text belongs to.

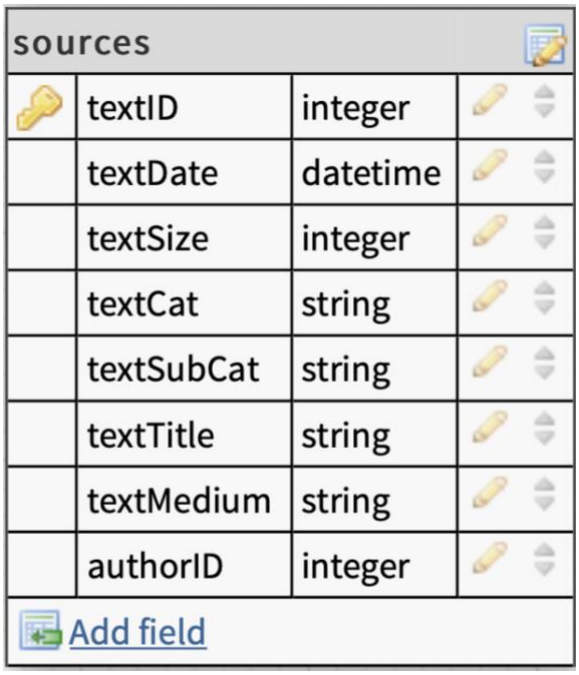

Figure 3: Database table - dictionary

The next table in our data model for the corpus is author table with the structure that is illustrated in figure 4, which consists of following five columns: authorID, authorName, authorType, authorAge, authorGender. The first column of authors table **authorID** is an integer value and primary key of the table that is assigned to each author. The second column is **authorName,** it simply records the names and surnames of the author. The next column is more interesting in terms of analysis - **authorType**, it might be a solo author who produces the text source independently, product of collaboration of multiple authors that worked on the same text or corporate writer. This data might let us compare the work of solo, multiple and corporate authors and understand what kind of effects it has on the produced text. Another interesting metadata is the **authorAge**, similarly for comparison reasons it might be an interesting data to gather. The last column of the authors table is authorGender, identically to previous two columns it will let us compare which words are more popular at the authors of different genders.

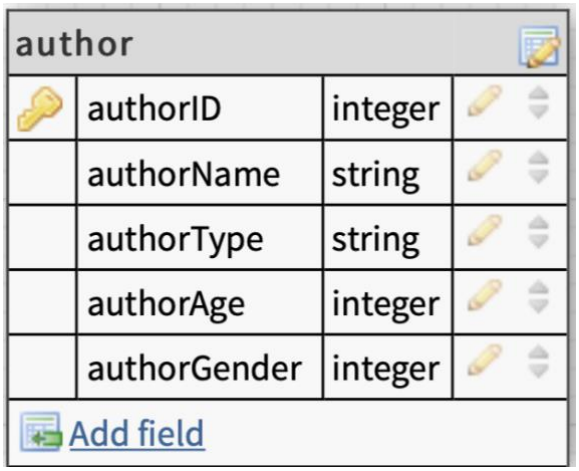

Figure 4: Database table - author

The last table of our initial data model for storing the corpus data is synonyms table which is illustrated in figure 5. This table is pretty straightforward it consists of three columns. The columns are synID, wordID, wordID. The first column is **synID**, which is an integer value and also the primary key of the table. The next two columns are the **wordID** of the words which are synonyms of each other. This table will be very useful when creating thesaurus relations.

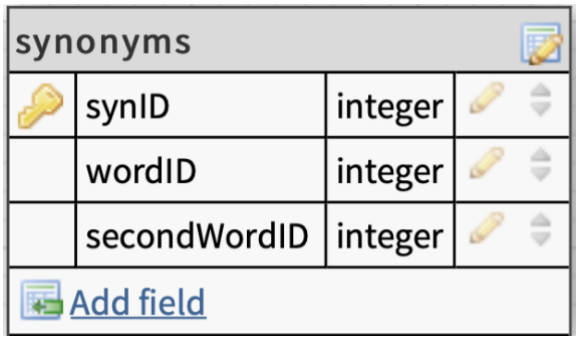

Figure 5: Database table - synonyms

The complete view of the initial data model structure is given in the figure 6, let us dive into the relational details of the model. We can observe that mainCorpus has relations to dictionary table and sources table through the columns of textID and wordID respectively. The sources table is referred from mainCorpus table and the table itself refers to the author table. The dictionary table is the most referred table as it holds the list of all unique words, it is referred from mainCorpus to set the relation for the word and it is twice referred from the synonyms table for both words that are each other's synonym. As mentioned, synonyms table refers to dictionary for the words with their respective wordID. Lastly, author table of our data model is referred by the sources table to get the relationship of the author for each source.

After initial implementation of the database structure, we have noticed that for implementing ngrams functionality, more on the functionality and the ngrams later on, we need a separate table for it. The structure of the ngrams table is given in figure 7.

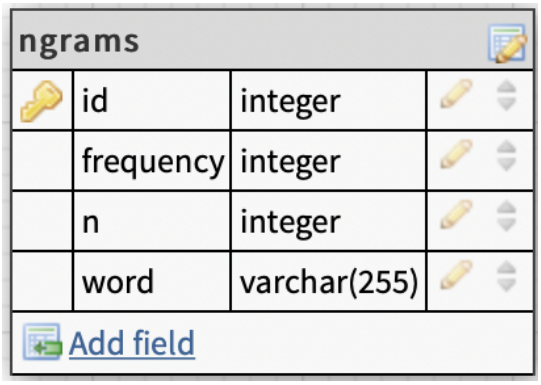

Figure 7: Database table - ngrams

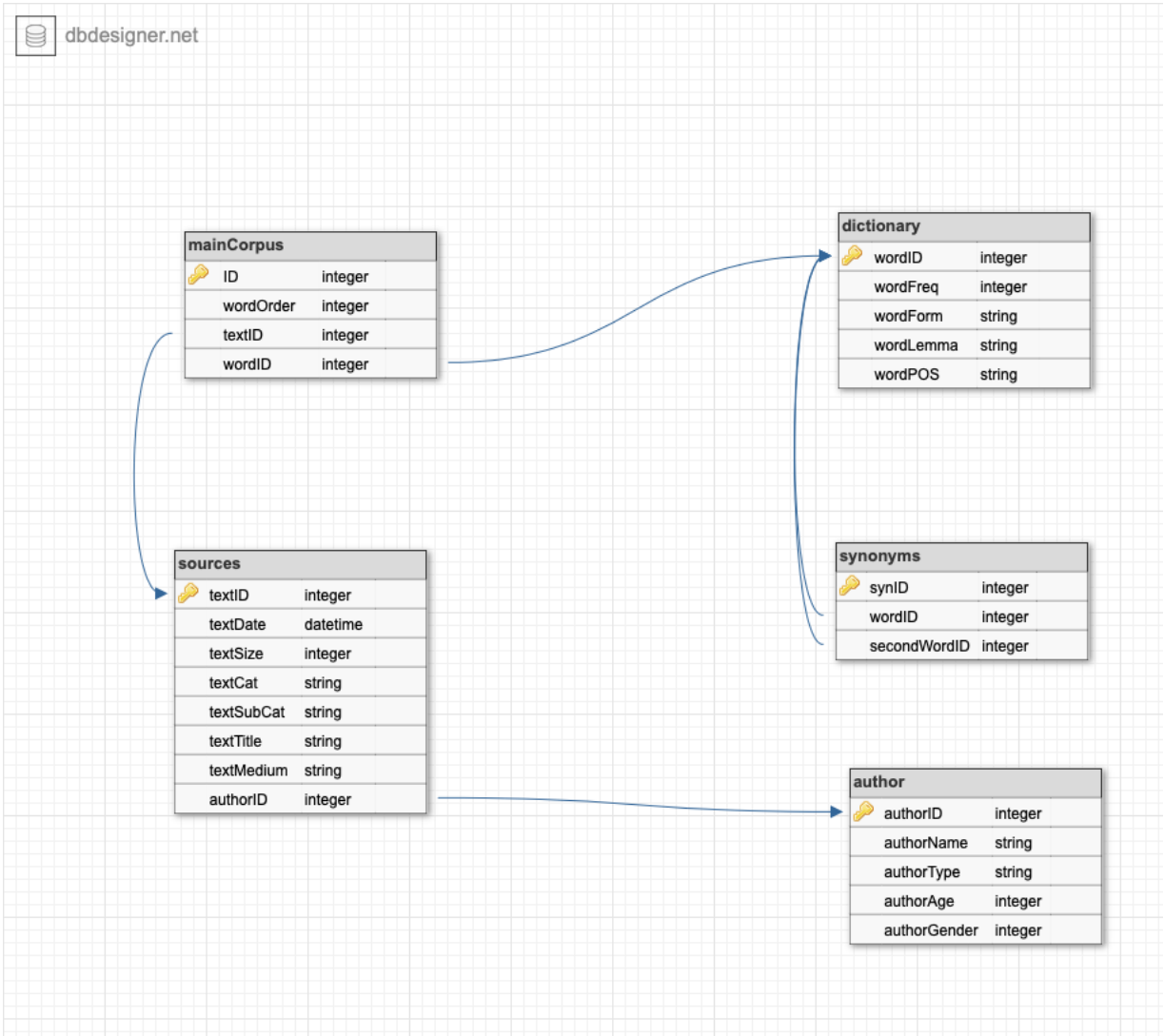

Figure 6: Initial data model

The final table of our database model is the the ngrams table which is illustrated in figure 7, it consists of the **id** column which indicates a singlu unique ngram in our database, and makes querying of it easies. The second column of the table is **frequency** which holds an integer value, this is the column that indicates how frequent this ngram is encountered in our corpus, this column also makes it possible to filter ngrams according to the frequency showing the most frequent ngrams or least frequent ones, also filtering according to the given range. The next column of the ngrams table is **n** which holds an integer value and indicates if the ngram is unigram, bigram, trigram, fourgram, fivegram or sixgram, querying according to this column we will present the results to the user. The final column of the table is word which holds the string value, and indicates the ngram itself.
After addition of the ngrams table final data model has been established, ngrams table is going to play a role of a cache for a quicker access to the ngrams functionality, that is why it doesn't have any relationships with other tables. The final version of the data model is illustrated in figure 8.

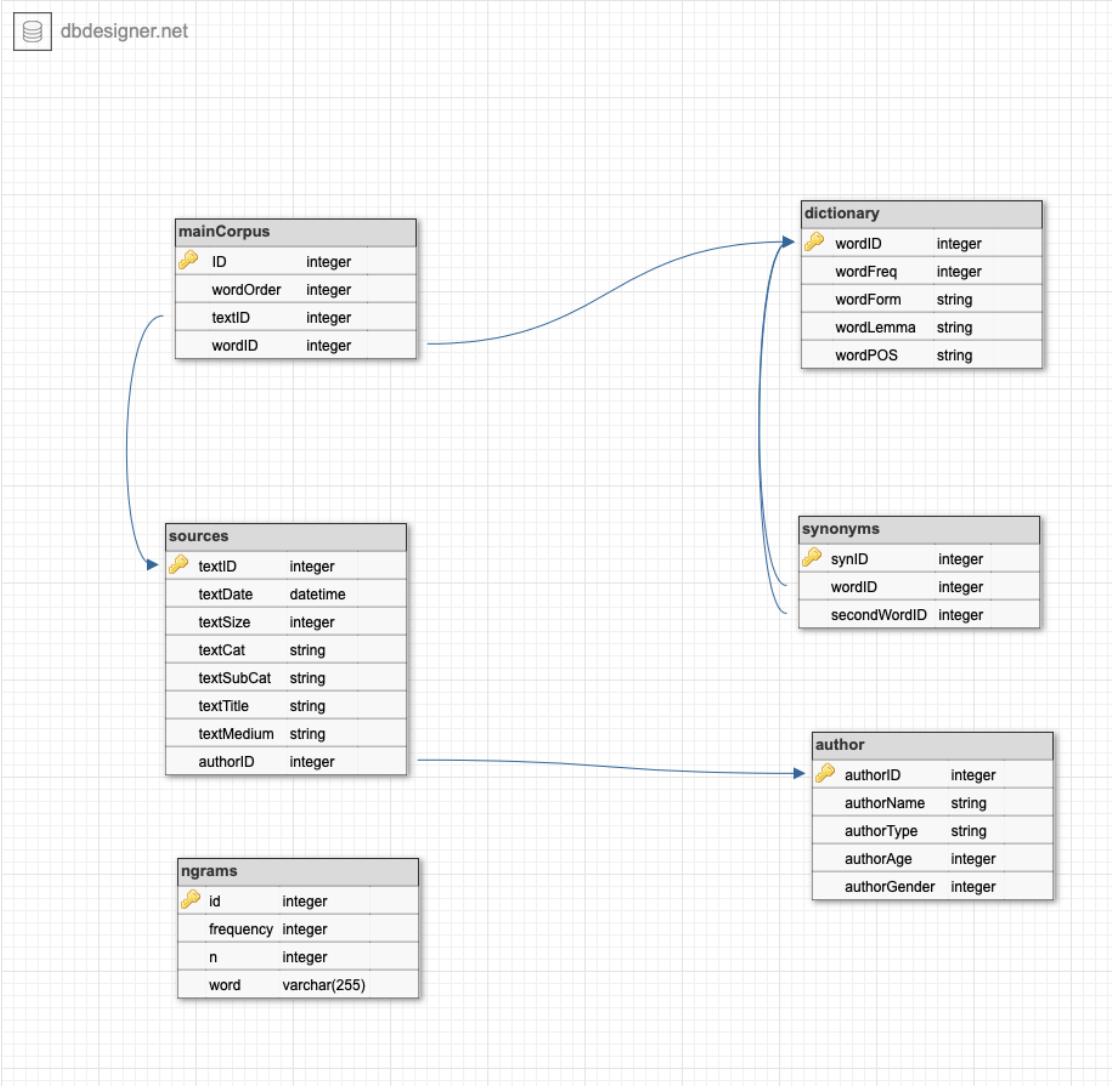

Figure 8: Final data model

We believe that compared to other systems, the relational database architecture offers a number of major benefits. That the very first two are size and speed. Even enormous corpora can be queried quickly since every table is indexed. Although some of indexes are through the use of clustered indexes, it is still going to get the speed advantages compared to other ways of storing the information. To put it another way, the architecture is incredibly adaptable, having very little to no speed loss regardless if we decide to continuously enlarge our corpus from the initial 10 million token corpus to a very large numbers as it is in the biggest corpuses out there[7].
The ability of relational databases about being "modular" structures capable of incorporating a variety of extra feature sets into the design without significantly slowing down the performance is its second major benefit compared to the competition. The "cost" of JOIN queries amongst database tables is quite low because each table has its own unique index that increases the speed of the query dramatically. Consider the sophistication of these kind of corpora operation in the plain txt files where there would be no clear way of querying the data, or within an XML format, where every other aspect is specified inside the document themselves, such as for go the word, part of the speech, lemma form of the word, or the list of synonyms of the word, also imagine the amount of duplication with it. The design of relational databases is exceptionally extensible and suffers very minimum operational damage even when handling extremely big corpora when extra annotation features are introduced via JOINed tables.

3.3 Functionalities of the corpus

This research project's corpus functioning chapter attempts to describe the functions of the Azerbaijani language corpus. The corpus is intended to be a comprehensive dataset for analyzing and researching the Azerbaijani language. It has a wide range of genres and categories that were chosen with care to showcase various facets of the language. The corpus also features a number of activities that give users a variety of ways to engage with the data and derive insightful conclusions.
Word frequency analysis, one of the corpus's main features, is a handy tool for examining the most frequently used Azerbaijani words. The most popular terms and their frequency distribution may be found with this feature, which can be utilized to obtain insight into linguistic usage and patterns.
Ngram analysis, a technique for measuring the frequency of word sequences in a text, is another feature of the corpus. Users can use this tool to investigate the frequency of various word sequences, which can be used to research language structure and patterns.
An additional feature of the corpus is a concordance tool, which enables users to look up certain words or phrases and determine the contexts in which they are used. The language usage and meanings of words in various settings can be studied with this capability.
Furthermore, the corpus has a thesaurus tool that offers users synonyms and terms that are similar to a specific word. The lexical and semantic links between various terms in the Azerbaijani language can be researched using this function.
The corpus also features a word sketch feature that offers users a thorough study of a particular term. This feature provides data about a word's frequency, collocations, and grammatical patterns that can be used to understand how the word is used and how it is understood in various settings.
We are going to dive deeper into each of the key uses of our Azerbaijani language corpus that we briefly mentioned earlier. These features will give linguists and academics a variety of methods for examining and comprehending the linguistic structures and patterns in the corpus. We will go through the main advantages and characteristics of each functionality as well as how to use them in future investigation and analysis. Researchers will be able to learn more about the usage and distribution of words, investigate the connections between words, and spot patterns and trends in the language corpus by making use of these functionalities.

3.3.1 Word Frequency

The word frequency function is one of the fundamental tools of any corpus analysis. It involves counting the number of times a specific word appears in a given corpus. By analyzing the word frequency, we can determine the most commonly used words in a text or collection of texts. This can provide valuable insights into the language and its usage patterns.
In our Azerbaijani language corpus, the word frequency function will be implemented by first preprocessing the raw text data, cleaning the data, removing the stopwords and then counting the occurrence of each word in the corpus. The results will then be stored in database table, where each row corresponds to a unique word in the corpus and the frequency of its occurrence. Later on, the displaying of the results to the user will be implemented using a database query to the table.
Word frequency function can be used to find out a variety of information about the corpus[14], including which words are utilized most often, which words are frequently combined, and which words are more common in particular corpus genres or subcategories. The word frequency function can also be used to compare word frequency across corpora or to study linguistic evolution through time. The outcomes of the word frequency analysis can shed light on the vocabulary and usage trends of the language. The grammar and syntax of a language can be determined by analyzing the most common terms to figure out if they are function words or words of content. The subjects and themes of the language can be determined by examining the frequency of content terms like nouns and verbs. Using the word frequency function, one can also find uncommon or unusual words that would be of interest to academics. These terms may be technical jargon or industry-specific terminology that can shed light on how the language is used in a variety of contexts, including science, technology, and culture. As it is clear, word frequency is a function that gives a lot of insights and can be implemented with any language, that is why most of the corpus are having this feature as a function and our large-scale Azerbaijani language corpus is not an exception.
The user interface for this functionality is given in figure 9.

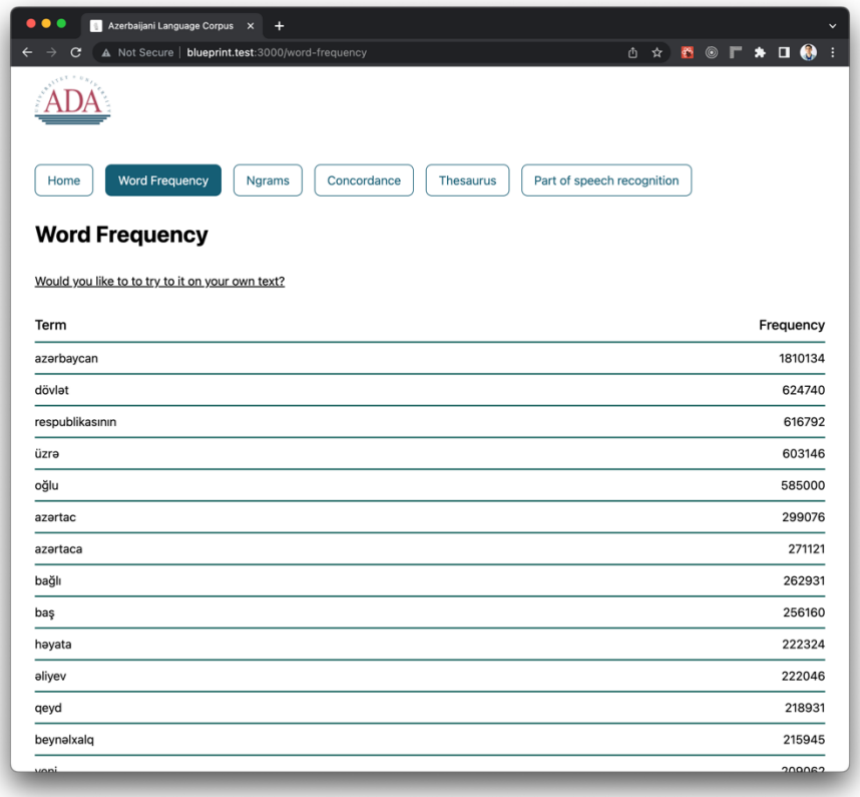

Figure 9: User interface for word frequency

3.3.2 Ngrams

Ngram is a continious sequence of n words in the given sample sentence or the sample of text. Usually n is equal to two, three and for in most analysis and it is called bigram, trigrams and fourgram respectively. In our corpus analysis we will implement from n equal to one till the 6, meaning that we will have unigrams, bigrams, trigrams, fourgrams, fivegrams and sixgrams. To better understand what an ngram is let us examine it on an example sentence in Azerbaijani language. For example, in the given sentence “Mən ADA universitetində təhsil alıram” the twograms would be: “Mən ADA”, “ADA universitetində”, “universitetində təhsil” and “təhsil alıram”. The trigrams would be: “Mən ADA universitetində”, “ADA universitetində təhsil”, “universitetində təhsil alıram”.
Ngrams are a potent tool for studying linguistic information and can give important insights into the organization and patterns of a corpus of texts[18]. Main usage in our application of the ngrams is counting the frequency of each ngram and then letting the users, linguists and the language enthusiasts to see the top used ngrams, or least used ngrams, as well as letting them to put filters like range. Beside this usage ngrams are used in natural language processing.
In natural language processing (NLP), ngrams are primarily used to assess linguistic input. An n-gram is a continuous string of n elements from a specific sample of text, each of which may represent a single character, a word, or a phrase[17]. Therefore, a section of n consecutive components from a lengthier sequence is known as an n-gram.
The n-gram model is used to calculate the likelihood of the next item in a sequence given the first n-1 items in the sequence. Applications of this include machine translation, language modeling, and speech recognition. By looking at the frequency of n-grams within a collection of texts, we can discover more about the linguistic structures and patterns used in that corpus.
Using a sliding window technique, n-grams can be produced by moving a window of size n along a text sample and producing a new n-gram at each step. To create bigrams, for instance, we might move a window of size two along the text while producing a new bigram at each step.
Ngrams in our application are calculated the following way, firstly we get the data, clean the data from the punctuation (code 1), tokenize (code 2) the text and after that we start the calculation of the ngrams(code 3) using python scripts and packages, after that the results were saved as csv files for each n from one to six (table 2), and finally generated csv was parsed and written to the database(code 4).

During the cleaning of the corpus we are substituting according to the regex the string from the corpus text and assigning the result to the clean_corpus variable. (code 1)

Code 1: Corpus cleaning (Python)

```
clean_corpus=[re.sub(r'https?://[^\s\n\r]+', '', corpus_i) for corpus_i in corpus]
clean_corpus=[re.sub(r'[!"#$%&\'()*+,-/:;<=>?@^_`{|}~]+', '', corpus_i)for corpus_i in
clean_corpus]
```

After cleaning the corpus from the punctuation and the links, we are ready to create the tokens from our corpus, and here we are using the package NLKT. The NLTK (Natural Language Toolkit) package is a Python library used for natural language processing (NLP) tasks such as tokenization, part-of-speech tagging, sentiment analysis, named entity recognition, and more. It provides a collection of tools and resources for working with human language data, including corpora (large collections of text), lexical resources (such as dictionaries and word lists), and algorithms for various NLP tasks. NLTK is widely used in academia and industry for research and development in NLP and machine learning, and it is also a popular tool for teaching NLP concepts and techniques[15].

Code 2: Tokenization (Python)

```
clean_tokens =[nltk.word_tokenize(corpus_i.lower()) for corpus_i in clean_corpus ]
```

Ngrams function is also the function that is provided by NLKT package which accepts tokens and the n number to create ngrams.

Code 3: Ngrams calculations (Python)

```
from nltk.util import ngrams


# create ngrams
unigrams=ngrams(clean_tokens, 1)
bigrams = ngrams(clean_tokens, 2)
trigrams=ngrams(clean_tokens,3)
fourgrams=ngrams(clean_tokens,4)
fivegrams=ngrams(clean_tokens,5)
sixgrams=ngrams(clean_tokens,6)
```

After the creation of the ngrams for each n, we are exporting this data to the csv file. First we have tried to make it in JSON format, to make it easier for the NodeJS application to read it, but as the size of the file is extremely big, it wasn't possible to parse the JSON, so as a solution we decided to convert it to the CSV. In the table 2, small sample of the trigram csv is shown.

Table 2: Sample of trigram csv generated file

Trigram1	Trigram2	Trigram3	Frequency
xəbər	verir	ki	80741
mətbuat	xidmətindən	azərtaca	52165
qeyd	edək	ki	22341
milli	elmlər	akademiyası	4196
ilk	dəfə	olaraq	4025
fikir	mübadiləsi	aparılıb	3616
ötən	ilin	müvafiq	2963

Although csv file is not something NodeJS handles out of the box, there are packages that help to work with CSV files in NodeJS as well. In code piece 4, we can see several imports at the beginning. PrismaClient[16] is a package for NodeJS application that works as an Object Relations Mapping (ORM) between the NodeJS application and the relational database, and claims to provide full type safery from the client to the database with the best in class experience. It creates a bridge from application to the database, so we will use it to make the calls to the database. FS is package by NodeJS to access file system of the user, this package is used to get the files from the local machine where the code is run, in our case fs will read locate and read csv files containing ngrams that will be parsed. The next import is a parse function from the csv package, this package contains several functions to work with csv files in NodeJS environment such as csv-parse, csv-generate, csc-stringify, stream-transform. In our use case we are going to use the parse function which reads the csv file parsing it with the delimeter of `,` and starts the reading from the secod line skipping the headers of the csv, and when the line is read and is ready to make manipulations the asynchronous function is going to be called with the row parameter. The last package that is used in this code snippet is zod. Zod is the third party package for NodeJS and javascript runtimes, which provides schema validation, providing the schema which you expect your variable or the oject to be, zod checks given parameter according to the schema and validates that the given input is valid according to the schema, otherwise throws a runtime error preventing malicious variables or objects being used further in the code. In our code, before trying to push the row from csv file, firstly we validate the row with the help validation

schema of tuple containing three strings for the trigram word 1, word 2 and word 3, and an integer value for the frequency. After this we are destructing the row to the variables word1, word2, word3 and frequency, and use prisma ngrams model upsert funcion which corresponds to the database table of ngrams, and upsert function checks if this ngram has already been inserted to the database before, if not it creates a new record concatenating three words of ngram to a single string and giving frequency to it.

Code 4: Parsing the CSV and writing to the database (NodeJS)

```
import { PrismaClient } from '@prisma/client';
import fs from 'fs';
import { parse } from 'csv';
import { z } from 'zod';

const prisma = new PrismaClient();

async function triGram() {
  fs.createReadStream('data/ngrams/trigram.csv')
    .pipe(parse({ delimiter: ',', from_line: 2 }))
    .on('data', async function (row) {
      const rowSchema = z.tuple([
        z.string(),
        z.string(),
        z.string(),
        z.coerce.number().int(),
      ]);

      const [word1, word2, word3, frequency] = rowSchema.parse(row);

      const triGram = await prisma.ngrams.upsert({
        where: { word: `${word1} ${word2} ${word3}` },
        update: {},
        create: {
          word: `${word1} ${word2} ${word3}`,
          frequency,
          n: 3,
        },
      });
    });
}
```

The application contains five more very similar code pieces to parse the unigram, bigram, fourgram, fivegram and sixgram csv files. The main difference in functions is the number of words, their concatenation and validation of the csv row.
The user interface for this functionality is given in figure 10.

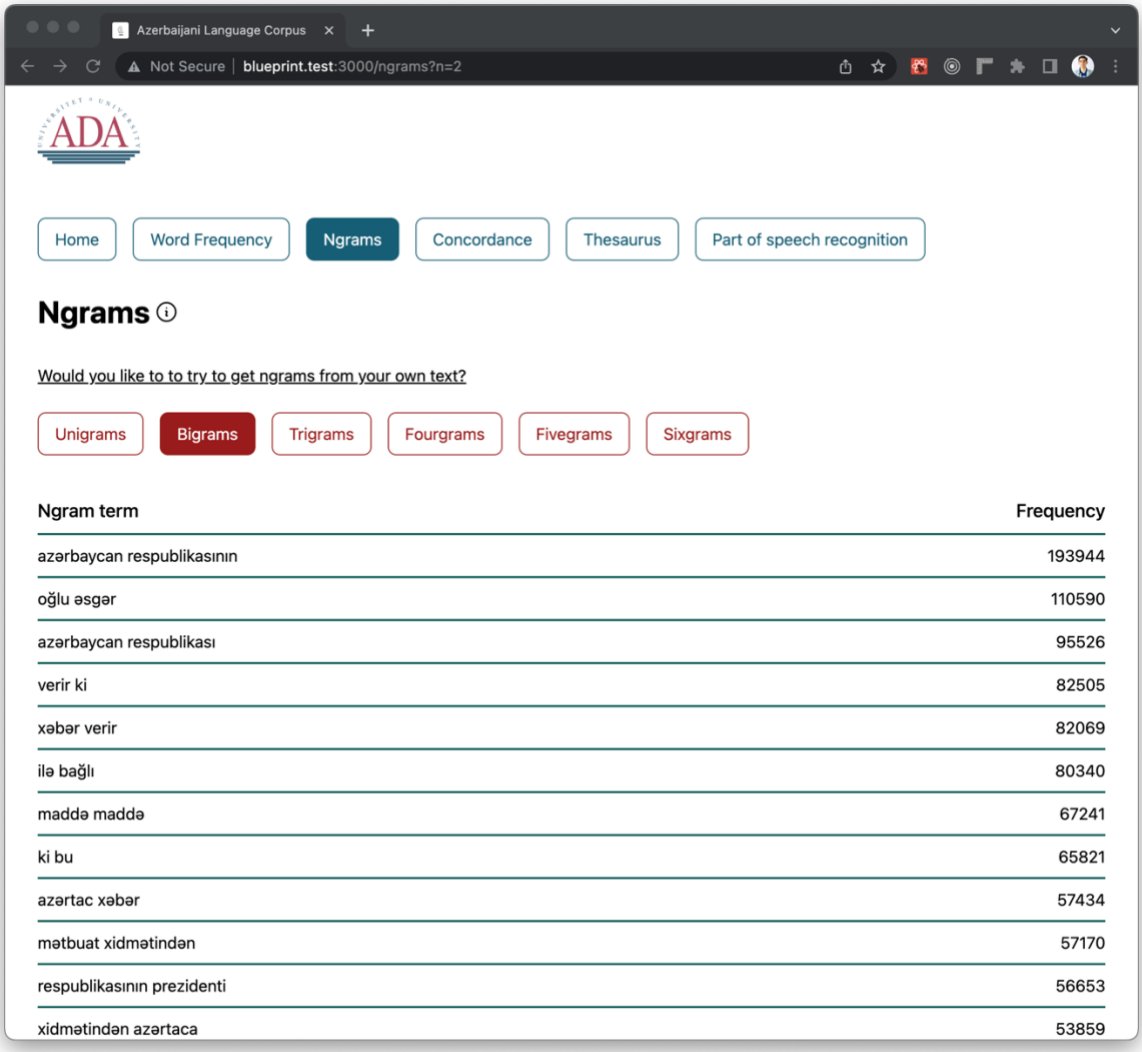

Figure 10: User interface for bigrams

3.3.3 Concordance

Concordance is a technique that is used corpus analysis and corpus linguistics which shows the queried word in all different possible contexts that the given word is used in, and with this it provides impactful insights to the meaning of the word in different contexts, also it helps to show the contexts of the word which might not be so natural with this it helps to reveal the meanings an the contexts of the word which might have been unnoticed without the usage of the concordance.

Concordance in the corpus analysis might also help to discover for the linguists or the language learners the meaning of the word according to different genres, for example some words in the context of official language like literature or academic might have some meaning, but in the spoken usage the people might also use it in different meaning, using concordance of the same word in literature or academic context and the concordance of the very same word in the spoken language might give different contexts, hence different meaning of the same word.

The way concordance works in corpus analysis is that the queried word is searched in the whole corpus, and when the word is found in the corpus, according to the specified width the words surrounding that queried word are also extracted and returned to the user, giving the user the whole context of the word. This allows the linguists to check the surrounding words and how they relate to each other in the context and in the form of the grammatical structure. The concordance is also insightful in case of the learning the language, for instance if somebody is trying to get along with the word, instead of comping up with the usages of the word on their own, they can simply query the word for concordancing and receive the ready results, with the context of the word.

The concordance is very insightful and important feature to use in the text and corpus analysis, as it clears out the usages of the word in different contexts, with different meanings across different domains. Hence, implementing concordance as a function in the corpus is important, so the contexts of the word are searchable.

In our case we will use the function available from the NLKT package, so for this purpose we will first convert our corpus tokens to the format which NLKT package recognizes, this is going to be the Text format, and after that we will implement concordance function to receive concordance examples from our text. This function accepts three parameters, first of which is the word itself that we are looking for concordance, the second parameter is the width which is by default is 80 of the each line where the concordance is returned and the last parameter is the lines parameter which is by default is 25 and it shows how many lines should be returned by the function.
The user interface for this functionality is given in figure 11.

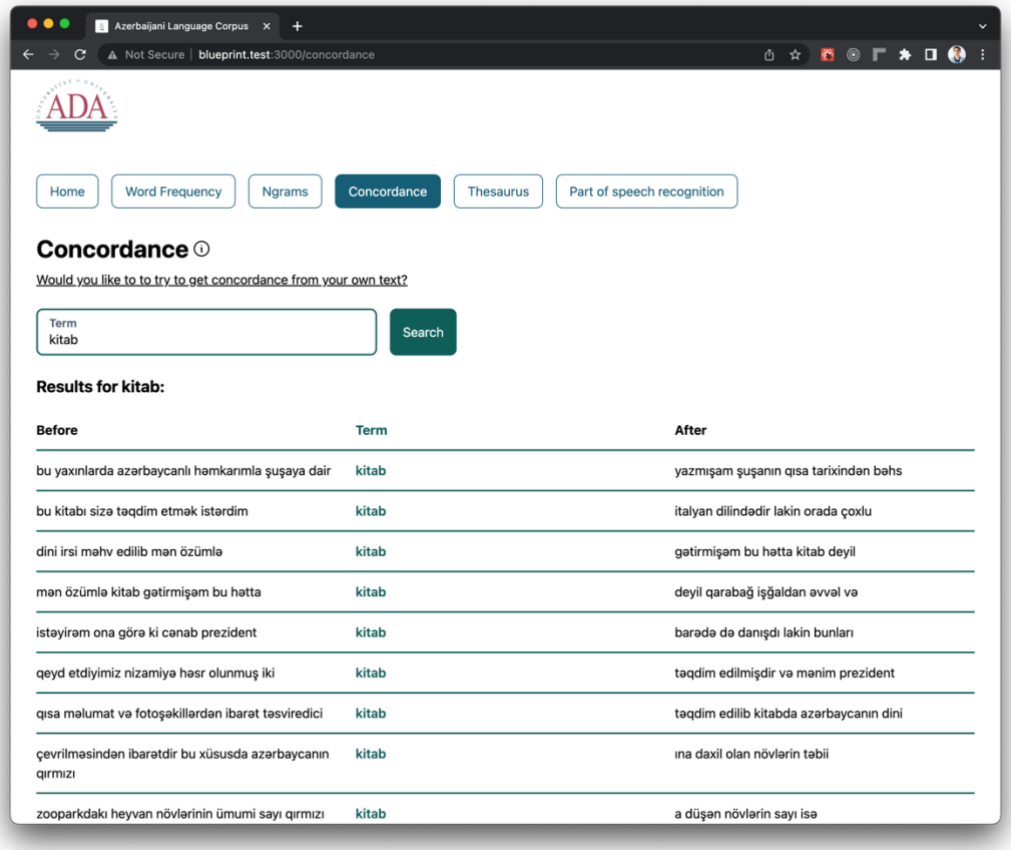

Figure 11: User Interface for Concordance

During the implementation of the function, we discovered that this function doesn't fully cover our requirements, because we needed manually split the returned string to the lines, as well as there was no simple way of dividing the string to the before and after context, so we have started for other function which got this requirements of ours covered. Luckily NLKT package has recently released another function which is called concordance_list which accepts the same set of parameters as it in the plain concordance function, meaning word, width and the number of lines, but instead of returning the plain string this one returns array of lines with properties of left which is an array of words that are before the queried content, the query itself and the right property which is similarly to the left is an array of words that are coming after the queried word. In the storage of the corpus part, it is mentioned that the corpus is stored in database and using plain text files, this is the case why the plain files are also saved, during the concordance functionality we are using plain text files to get the whole text and be able to perform operations in the text file itself. Also, during the architecting the storage versions, it has been decided that although database is giving a lot of advantages there is no harm in keeping the plain text files as well, considering the storage costs in these days, it will not create much of the problem. As a result, we have used plain text files for concordance function.

3.3.4 Thesaurus

A thesaurus is a dictionary or database that lists words' synonyms and antonyms. It is a tool that aids users in discovering words that have meanings similar to a given word or idea. Numerous situations, including as writing, editing, and information retrieval, call for the use of thesauri.
Thesauri are used in linguistics and corpus linguistics to identify and classify the meanings of words as well as to group words together according to their semantic links. Thesauri can assist academics in spotting linguistic trends, such as the words that are frequently used together in a certain context or the terms that are connected to particular topics or domains.
As thesaurus are very important to any tool that are working with the corpus, thesaurus are implemented in our corpus as well. According to the initial plan, we were planning to get the list of synonyms for the word from the linguists team and use the database to retrieve the synonyms for the given word, but unfortunately the time limitation didn't let us use that approach, so we were forced to use data analytics method to come up with the solution to the given problem. The solution we come up with was using Word2Vec model to identify similar word with the similarity rating.
Word2Vec is a method of natural language processing that represents words in a vector space in an effort to capture their semantic and syntactic links. This technology, which is built on neural networks, can figure out if a term will be used given its context and vice versa. The different natural language processing tasks including text categorization, sentiment analysis, and named entity identification may then be carried out using the generated vector space.
The Word2Vec model trains how to depict every phrase in the corpus as a vector in a high-dimensional space from a vast corpus of text as its input. In order to accomplish this, a neural network is trained to anticipate the context of a particular word in a phrase. The group of words that occur around a word within a predetermined window size is referred to as the context of that word. The model learns to recognize the statistical patterns and correlations between words by being trained on a big corpus of text.
One of the key benefits of Word2Vec is its ability to capture the semantic and syntactic relationships between words. For example, the vector for "king" might be close to the vector for "queen" and "prince", while being far from the vector for "car" or "tree". This makes it possible to perform operations such as vector addition and subtraction to capture relationships between words, such as "king" minus "man" plus "woman" equals "queen".

According to this model, we have trained it with our corpus data, saved the model and use it for further analysis. Example usage of this function is shown in code 5, where the model is the Word2Vec generated model and term is the word that we are looking thesaurus for.

Code 5: Thesaurus finding (Python)

```
model.wv.most_similar(term)
```

The results of this model are pretty satisfactory according to our tests, it provides relevant words to the search term and gives the similarity index of the returned term, making it easier for the linguist or the user to understand how similar the words are actually. As an example, here is the sample of the results for the search term `italyan` (table 3).

Table 3: Thesaurus for word `italyan`

Similar word	Similarity rate
ispan	0.884
fransız	0.8637
bolqar	0.8317
polyak	0.822
yapon	0.8182
macar	0.8006
alman	0.7888

The word `italyan` means Italian and the model returns as a result different nationality as Spanish with the highest ratio of 0.884, followed by french, bulgarian, polish and other nationalities.
The user interface for this functionality is given in figure 12.

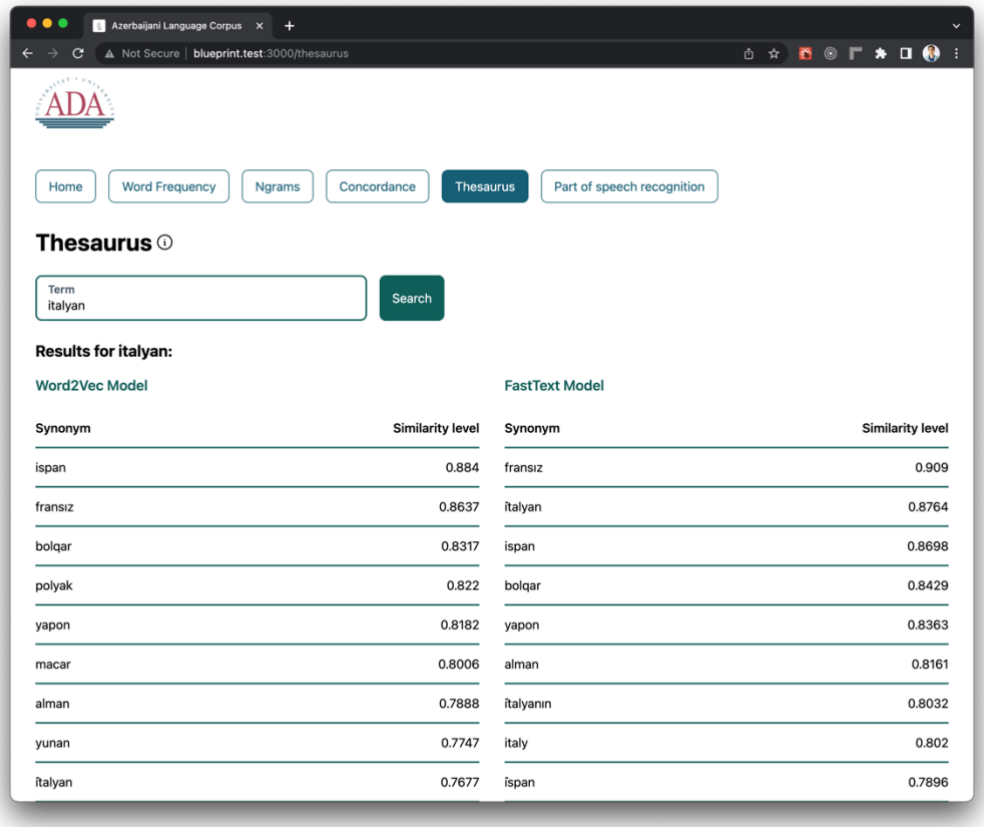

Figure 12: User Interface for Thesaurus

3.3.5 Word-sketch

Word sketch is a functionality set that is similar to concordance with the combination of the ngrams, so the way it works is that the user of our application who might be a linguist or the language learner queries a word, after that we are finding this word in a context with the n words surrounding it, after this process we are counting the frequency of the received string sorting it from the most frequent to the least frequent ones and returning it to the user. This function combines two essential tools that are used during the corpus analysis, therefore providing useful insights in which context the given word is used the word and providing the context at the same time, it will demonstrate the most frequent use cases of the word in the language, opening a wide horizon for the language linguists to analyze the usages of it, at the same time letting the language learner benefit from the main usages of the word in the language.

3.3.6 Part of speech tagging

Part-of-speech (POS) tagging is a process of identifying the grammatical category of each word in a given text corpus. In this process, each word is tagged with a part-of-speech such as noun, verb, adjective, adverb, pronoun, conjunction, preposition, and interjection.
The POS tagging is a crucial task in natural language processing (NLP) as it helps in various applications such as text classification, sentiment analysis, named entity recognition, and machine translation. It is also helpful in disambiguating words with multiple meanings, especially in cases where the meaning of the sentence depends on the POS of certain words.
The POS tagging can be done manually or automatically using machine learning algorithms. The manual POS tagging is a time-consuming and labor-intensive process as it requires human experts to annotate each word with its corresponding POS. On the other hand, the automatic POS tagging relies on statistical models or rule-based methods to predict the POS of each word based on the context of the sentence.
The accuracy of the POS tagging depends on the quality of the training data and the complexity of the language. Some languages have more complex grammatical rules and more ambiguous words, which make the POS tagging more challenging. However, with the recent advancements in machine learning algorithms and the availability of large annotated corpora, the automatic POS tagging has achieved high accuracy levels in various languages. In summary, the POS tagging is a fundamental task in NLP, which helps in various language analysis applications. It can be done manually or automatically using machine learning algorithms, and its accuracy depends on the quality of the training data and the complexity of the language.
In our corpus, initially the plan was to use help of linguists to manually POS tag the data as it was acquired, but soon enough it was realized that linguists will not be able to pos tag the whole corpus as it is very time consuming and manual job and it would take forever to POS tag the whole corpus, so instead it was decided to use mixed approach. The approach that it has been decided to implement is that linguists are pos tagging a big chunk of text which can later be used as an input for the machine learning algorithms, so we the rest of the corpus might be POS tagged automatically. Initial attempts to do this were not so successful, our first model that was using Hidden Markov Model trainer from NLTK package. Hidden Markov Model (HMM) is a statistical model widely used in natural language processing for Part-of-Speech (POS) tagging. It helps to identify the grammatical category of words in a sentence such as noun, verb, adjective, etc.
HMM is a probabilistic model that consists of a set of states and a set of observations. In POS tagging, the states represent the possible POS tags of a word, while the observations represent the actual words in the sentence. The probability of a particular POS tag for a word is estimated based on the probability of the previous tag and the probability of the current word given the tag.
The HMM-based POS tagging process begins by training the model on a large corpus of labeled data, where each word is annotated with its corresponding POS tag. The model then uses this information to predict the POS tag of a word in an unseen text.
During the tagging process, the HMM starts with an initial state and predicts the POS tag for the first word. It then moves to the next state and predicts the POS tag for the second word, based on the

probability of the previous tag and the current word. This process continues until the end of the sentence.
This model did not perform well and was only showing 45% of accuracy which is obviously is not an acceptable result. As a result, we have started looking for alternatives that could perform better and decided to try Recurrent Neural Networks (RNN). RNNs are a type of neural network that are commonly used in natural language processing tasks, including part-of-speech tagging. RNNs can take into account the sequence of words in a sentence and use context to predict the part of speech for each word.
The basic idea behind RNNs is that they maintain a hidden state that captures information from all previous words in the sequence. At each step, the hidden state is updated based on the current word, and this updated hidden state is used to predict the part of speech for the current word.
To train an RNN for part-of-speech tagging, a large annotated corpus of text is typically used. The network is trained to predict the part of speech for each word in the training set, and the weights of the network are adjusted using backpropagation to minimize the difference between the predicted and actual part-of-speech tags.
Once the RNN is trained, it can be used to tag the part of speech for words in new sentences. The input to the network is the sequence of words in the sentence, and the output is a sequence of predicted part-of-speech tags.
After applying the Recurrent Neural Networks (RNN) model instead of Hidden Markov Model (HMM) with a few tweaks in the parameters we managed to increase the performance of the model up to 97%. This result is now acceptable and can be used for further usage, so we saved this model and decided that we will use this model for part of speech tagging. The model returns the tags for the sentence, the following is the example of how it performs on the example sentence:

Biz Sizinlə fəxr edir, bundan sonra da yeni-yeni qələbələr diləyirik.

- Biz - əvəzlik-şəxs
- Sizinlə - feil-sifət
- fəxr - b/feil
- edir, - e/feil
- bundan - əvəzlik
- sonra - qoş-zaman
- da - bağ-iştirak
- yeni-yeni - sifət
- qələbələr - sifət
- diləyirik. - isim

The user interface for this functionality is given in figure 13 and 14.

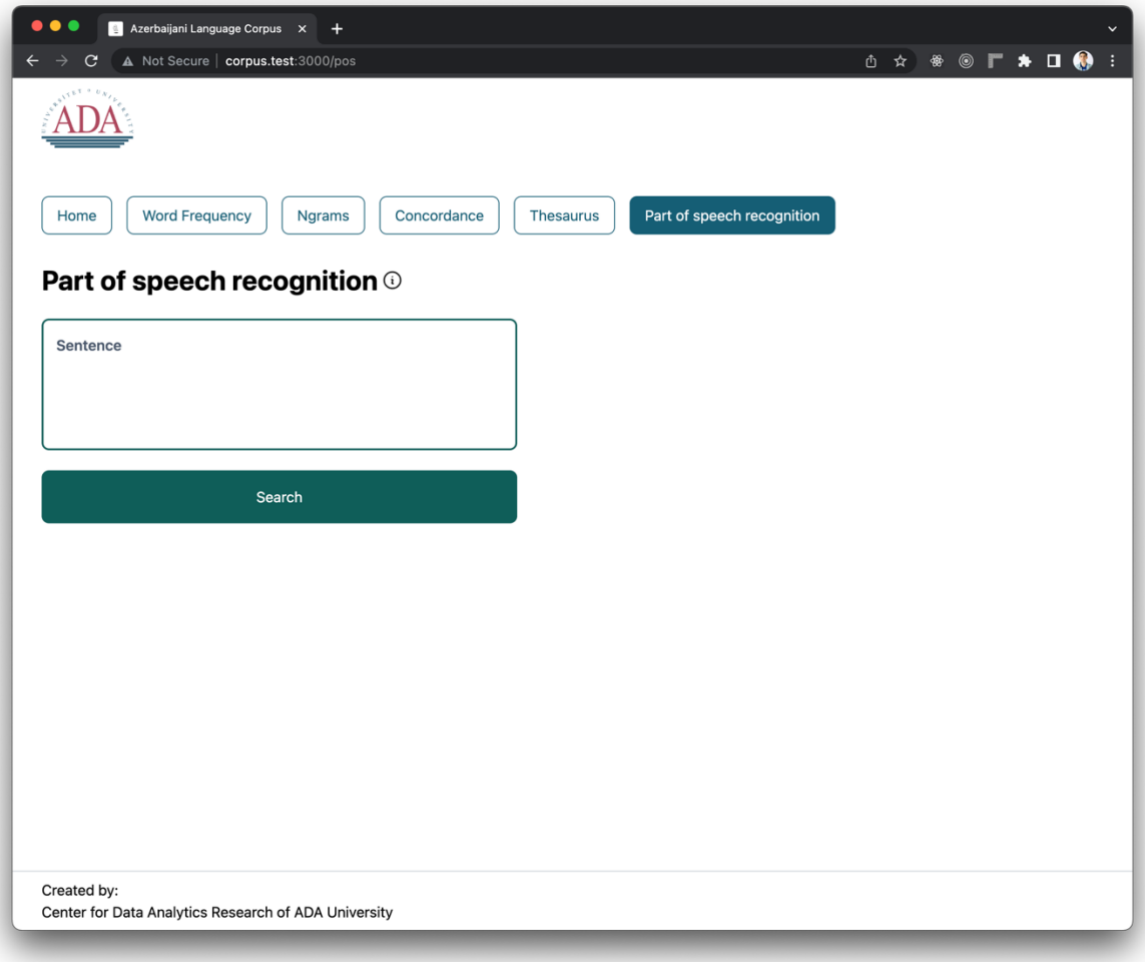

Figure 13: User Interface for Part of Speech tagging before user input

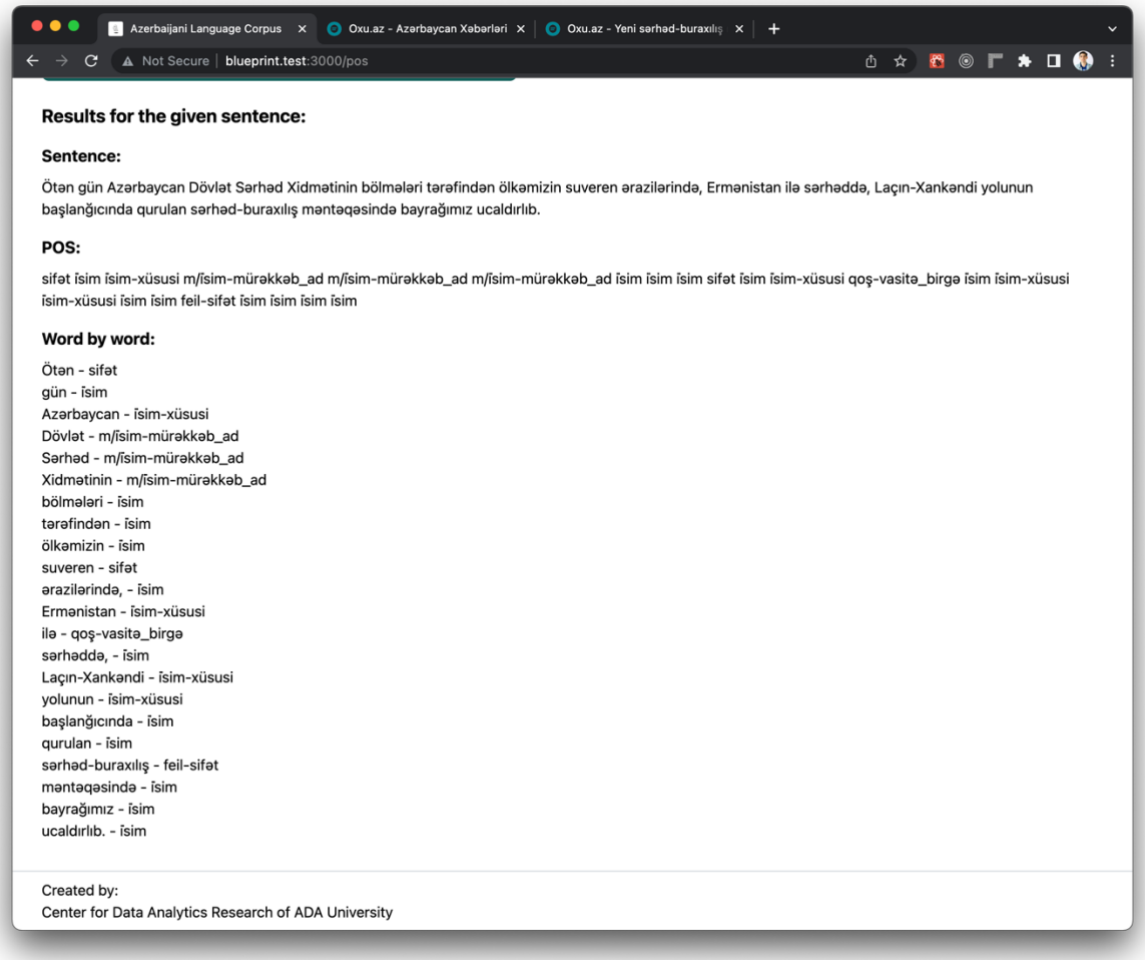

Figure 14: User Interface for Part of Speech tagging after user entered the text

3.4 Application and User Interface and User Experience

Sketchengine UI, our UI, weird problem of ngrams in the database utf8mb4_bin

The next stage is to make these resources accessible to a larger audience once we created a thorough corpus of Azerbaijani language and put different features in place to study the language. In order to accomplish this, we will create a web application that gives users simple access to the corpus and its features. A user-friendly interface for interacting with the corpus and performing linguistic analytics will be provided through the online application, to make the access to it easier and available worldwide.

We intend to encourage the use of Azerbaijani language resources in scholarly study, language instruction, and language technology development by making the corpus accessible through a web application. Users of the program will be able to browse word frequency, look for specific words or phrases in the corpus, and examine linguistic trends.

The web application will provide users with a platform to access to different functionalities like word frequency, ngrams, concordance, thesaurus, and word sketch. By using the web application, users can analyze the Azerbaijani language and gain insights into its usage, structure, and patterns.

To make the application the fastest and the most secure possible, modern technologies were analyzed and it has been decided to use the template provided by the T3 stack[21]. T3 stack includes in it the most modern set of tools and technologies and is considered to be the full stack framework template, using the NextJS as the UI element for the website which is the framework of ReactJS.

Next.js is an open-source JavaScript framework that enables developers to build web applications with server-side rendering and automatic code splitting. It is built on top of React, one of the most popular JavaScript libraries for building user interfaces. One of the main benefits of using Next.js is its ability to handle server-side rendering, which means that the initial HTML is generated on the server before being sent to the client. This allows for faster initial load times, improved SEO, and a better user experience.

Next.js also provides automatic code splitting, which means that the JavaScript code is split into smaller chunks that are loaded as needed. This helps to improve the performance of the application and reduce the time it takes to load. Other features of Next.js include support for static site generation, dynamic routing, and API routes. It also has a wide range of plugins and packages that can be used to add additional functionality to the application.

T3 stack makes use of typescript as the programming language instead of usual javascript. TypeScript is a programming language that is a superset of JavaScript. It adds static type-checking capabilities to JavaScript and supports modern JavaScript features such as classes, interfaces, and modules. It was developed and is maintained by Microsoft.

The main advantage of TypeScript over JavaScript is its type system, which helps prevent common coding errors by catching them during compilation. TypeScript allows developers to declare variable types, function return types, and interfaces, providing a layer of safety that can improve the maintainability and scalability of large codebases. Additionally, TypeScript's strong typing can help IDEs and editors provide better code completion and navigation support.

TypeScript code is compiled into JavaScript, which means that it can be run in any browser or environment that supports JavaScript. It can be used for both front-end and back-end development and has become a popular choice for web development frameworks such as React and others.

In terms styling of the website T3 suggests the usage of the TailwindCSS. Tailwind CSS is a utility-first CSS framework that provides a set of pre-defined classes to quickly and efficiently style web pages. Unlike traditional CSS frameworks, which have pre-designed components with specific styles, Tailwind CSS provides utility classes that can be combined to create custom designs without writing custom CSS. This approach allows developers to build custom designs quickly and efficiently without the need for additional CSS code.

Tailwind CSS offers a large set of pre-defined classes for styling elements such as typography, colors, spacing, borders, shadows, and more. It also includes a comprehensive documentation website with examples and explanations for all of its classes. This makes it easy for developers to get started with the framework and to quickly find the classes they need.

Another benefit of Tailwind CSS is its ability to customize and extend the framework by configuring the theme, adding new classes, or overriding existing ones. This allows developers to create unique designs that match their brand or style. Tailwind CSS can be used with any JavaScript framework or library and can be installed via a package manager or included via a CDN.

As the backend service our application has two services one of which is Next JS backend which is run in NodeJS runtime to query the database and Fast API to run python scripts. In Node JS runtime Prisma client is used to make calls to the database. Prisma is an open-source database toolkit that simplifies database access and management for developers. It provides a type-safe and intuitive way to interact with databases using a set of APIs and a query language called Prisma Client. With Prisma, developers can easily create database schema, perform CRUD operations, and manage database relationships, all with automatic migration support.

Prisma supports multiple databases, including MySQL, PostgreSQL, SQLite, and Microsoft SQL Server. It generates code for the database models and types, which eliminates the need for manual coding and reduces the risk of errors. Prisma's schema designer also provides an intuitive graphical user interface that allows developers to create and modify database schema visually.

One of the main benefits of using Prisma is its security. It automatically validates and sanitizes user input, which helps prevent SQL injection attacks and other security vulnerabilities. It also enforces data validation rules at the database level, which ensures that only valid data is stored in the database. Prisma also provides real-time data synchronization and subscription support through its Prisma Client API. This allows developers to build real-time applications and features without having to write complex code or use third-party libraries.

In order to use the python scripts that are required for some functionality pieces, we have also introduced Fast API in python. FastAPI is a modern, fast (high-performance), web framework for building APIs with Python 3.7+ based on standard Python type hints. It is designed to be easy to use and to provide high performance while still being very expressive. The framework is built on top of Starlette for the web parts and Pydantic for data parts, providing a seamless integration between them. FastAPI is built on top of asynchronous I/O, allowing for highly performant, scalable and concurrent applications. It offers built-in support for HTTP features like WebSocket, GraphQL, and OAuth, and also comes with automatic interactive API documentation that can be used via a web browser. The framework offers a clean, easy-to-use API that makes it simple to write complex and powerful applications. It also has a very low overhead, making it ideal for building high-performance APIs. FastAPI is designed to be developer-friendly, providing code that is easy to read and maintain. Additionally, FastAPI offers an integrated dependency injection system that makes it easy to manage and share resources across different parts of an application. It also has strong typing support that helps in building reliable and scalable applications.

The data exchange from the frontend services and the backend services is going to be provided in two ways, with the FastAPI backend service the information exchange is going to be handled in a tradional REST API manner, while Next JS front end will use tRPC to query Next JS backend services.

REST (Representational State Transfer) is a common architectural style used for building web applications. It defines a set of guidelines for how data should be transferred between the client (usually a frontend application) and the server (backend application). In a RESTful architecture, the client sends requests to the server to retrieve or modify data. These requests are made using HTTP methods such as GET, POST, PUT, and DELETE. The server then sends back responses to the client in the form of JSON or XML data.

The second type of the communication is going to be based on tRPC, tRPC is an advanced tool designed to create fully typesafe APIs, enabling seamless communication between clients and servers in a type-safe manner. This powerful framework leverages TypeScript's robust typing system, ensuring that both the client and server sides of an application are aware of the types they exchange, thus reducing the likelihood of runtime errors due to type mismatches. In this academic exploration, we will delve into the key features of tRPC, elucidate its benefits, and provide examples to illustrate its practical application.The following are the key features of the tRPC. End-to-end type safety: tRPC ensures that types are consistently maintained between the client and server, eliminating the need for manual type-checking and reducing the possibility of errors. Auto-generated client libraries: tRPC can

generate client libraries based on server-side APIs, simplifying the client-side code and maintaining type consistency.Seamless integration: tRPC can be easily integrated with popular frameworks like Next.js, React, and Express.js, facilitating seamless adoption into existing applications. It also has numerous benefits compared to the standard way of communication from frontend to the backend. Improved reliability: With tRPC, both client and server codebases have accurate knowledge of data types, leading to more reliable and maintainable applications. Enhanced developer experience: By automatically generating client libraries and providing type safety, tRPC streamlines the development process and minimizes errors. Accelerated development: With its ability to integrate with various frameworks, tRPC speeds up development, allowing developers to focus on implementing core application logic.

As a choice of the database in our application we have decided to go with MariaDB. MariaDB is a popular open-source relational database management system (RDBMS) that was developed as a fork of MySQL. It was created as a community-driven alternative to MySQL, which was acquired by Oracle Corporation in 2010. MariaDB is designed to be a drop-in replacement for MySQL, which means it can be used as a direct replacement for MySQL without any modifications to existing applications or databases.

MariaDB supports a wide range of features, including multiple storage engines, high availability, replication, and clustering. It also provides support for JSON and GIS data types, as well as a number of plugins and extensions that can be used to enhance functionality. One of the key benefits of MariaDB is its strong focus on performance and scalability. It is optimized for high-performance environments and can handle large volumes of data with ease. Additionally, it is designed to be highly scalable, with support for sharding and other advanced features that allow it to scale horizontally.

Transitioning from frontend and backend technologies to the user interface (UI). User interface is a crucial step in developing a web application. The UI is where the users interact with the application, so it should be designed to be intuitive, responsive, and visually appealing.

A user-centered design process is used to create UIs, with the goal of addressing users' demands in the process. This involves taking into account elements including the target audience, their objectives, and their technological familiarity. User testing and feedback are crucial for fine-tuning and enhancing the user interface.

The first stage in creating a user interface (UI) is to investigate the demands of the users and collect suggestions for a design. Examining existing ready-made UI solutions on the market is one approach to achieve this. This makes it easier to get ideas, evaluate the features and design, and spot typical user interface trends.

The goal, target market, and functionality of a ready-made UI product should all be considered while evaluating it. This aids in focusing the search on goods that are pertinent to and helpful for the endeavor.

As a team we decided that to start designing our user interface we can explore ready made products – corpus analysing websites available to the public. After looking through and testing several applications that are providing different functionalities to work on the corpus, we have decided to look up to the two of the applications that were having more user friendly user interface and an overall better user experience.

The first one is the online application of the Corpus of Contemporary American English[22]. It provides a simple and intuitive design for the user with some instructions on the right side, to access this website, there is no registration required, user can jump on and start using it, which makes it super simple to start with. The welcoming user interface is displayed on the figure 10.

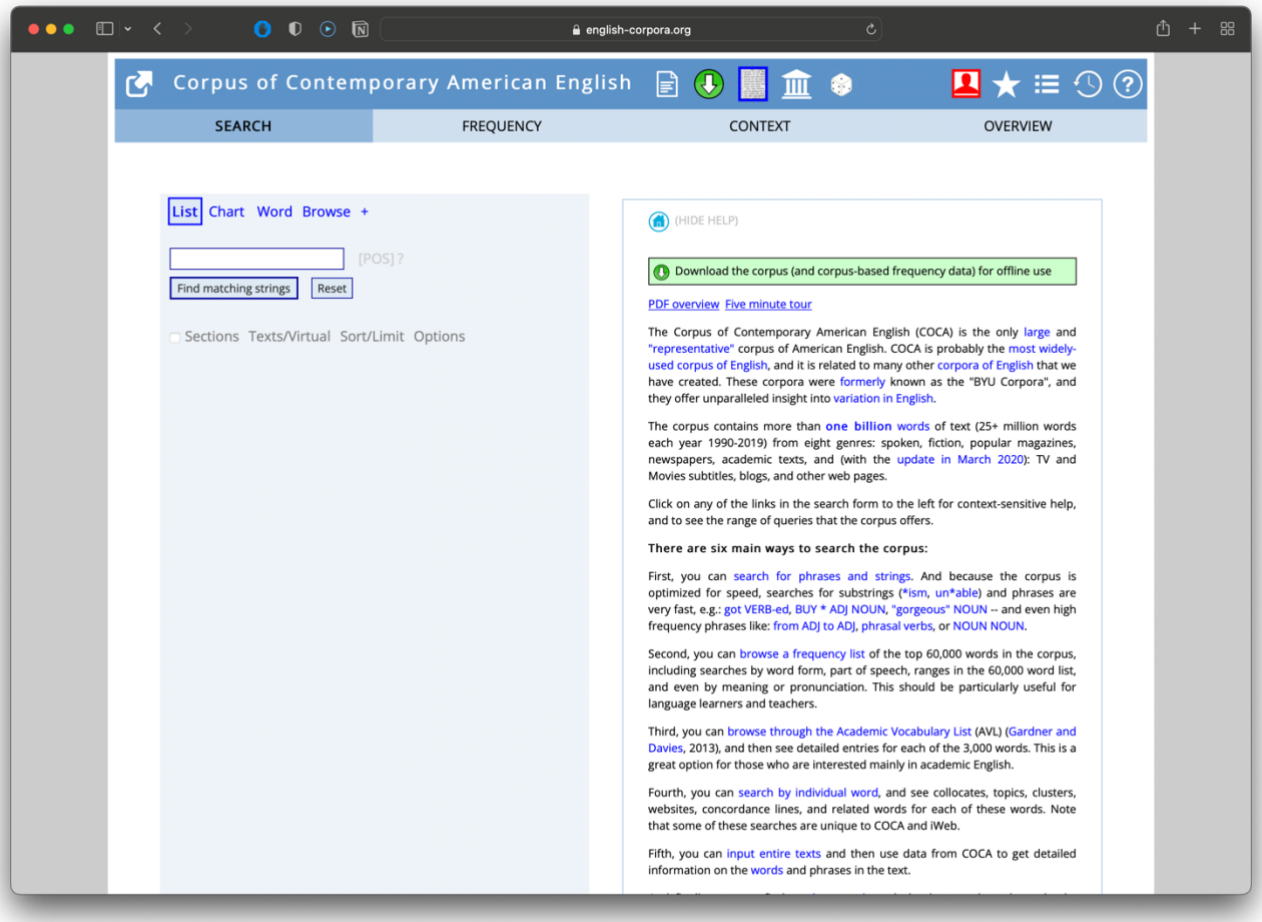

Figure 15: The user interface of the Corpus of Contemporary American English

If we analyze the user interface from figure 10, we can see the navigation bar including functionality pieces available in the application like search, frequency, context and overview, and by default we are in search bar, letting user start using the corpus right away.

The second online application that lets users to work with the corpuses is sketch engine [23]. Sketch engine compared to the previous website requires registration and gives a free trial of thirty days for the user to decide whether they want to switch to the paid subscription or require institutional login to use the platform. Another difference is that compared to the COCA corpus online application, sketch engine lets user work with multiple corpuses in different languages like English, Arabic, Russian, Dutch and many more. The first thing that user will need to do after successful registration is to choose the language that it wants to work with, and after that user will get the interface shown in figure 11.

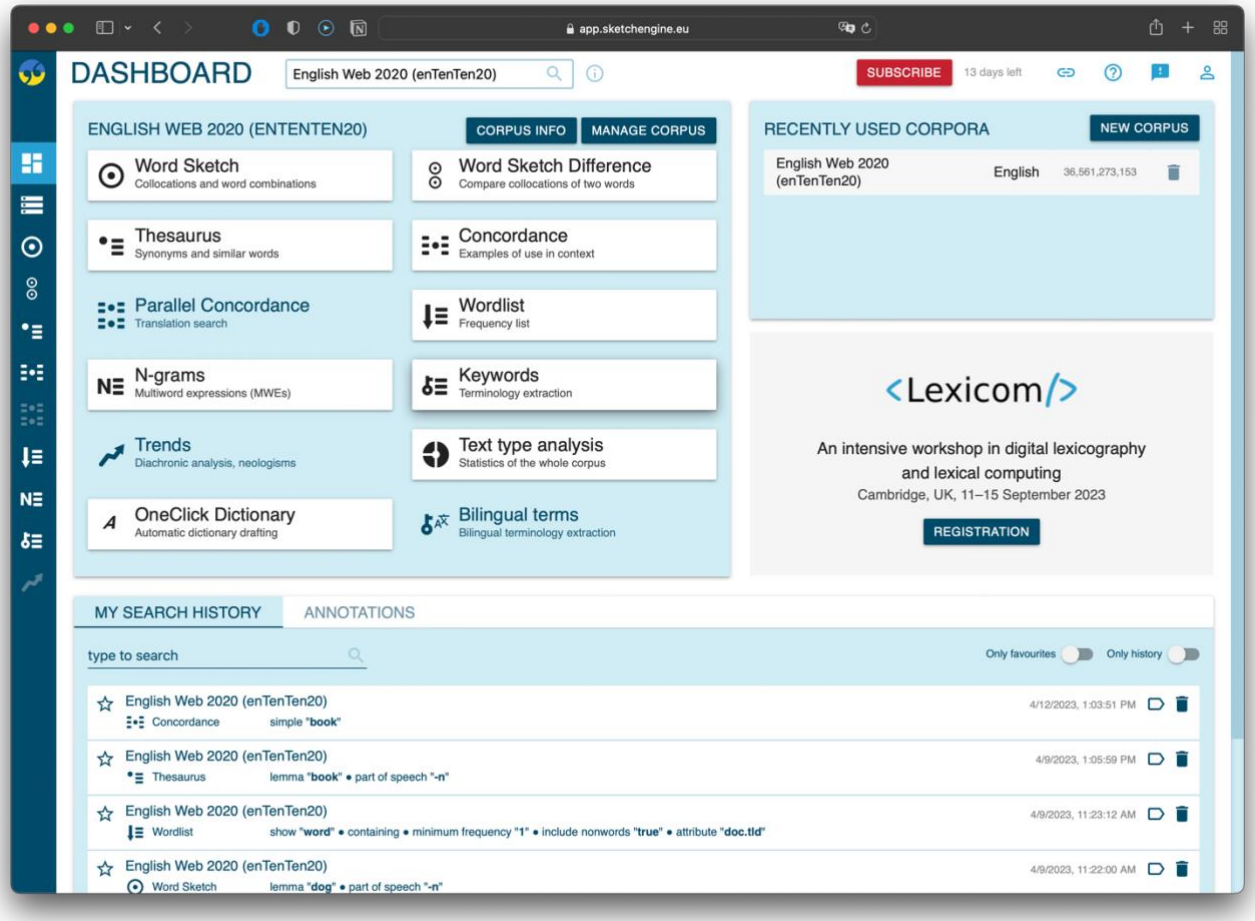

Figure 16: The user interface of Sketch Engine

Instantly we can see that compared to the COCA application this interface is more populated, hence it might be hard for some users to get started with the functionality itself, but after spending a bit of time user will realize that on the left side there are icons which represents navigation tab and serves as a quick links to different tabs of the application. On the bottom part of the website we can see the search history of the user and in which corpus this search was performed in. The top row of the application serves as a place where you can see which corpus user is working on right now and can switch the corpus to a different one, at the same time it shows if the subscription is active and suggests to extend the subscription period. Finally, the most important part is left top part of the user interface shows us list of functionalities that are available in the current corpus. Again, as there are way more functionality pieces, it might be hard for the first-time user to get along with the system. After getting over of each functionality and interface piece available in several websites and more specifically in COCA application and sketch engine, we come up with our own user interface, the homepage is illustrated in figure 12.

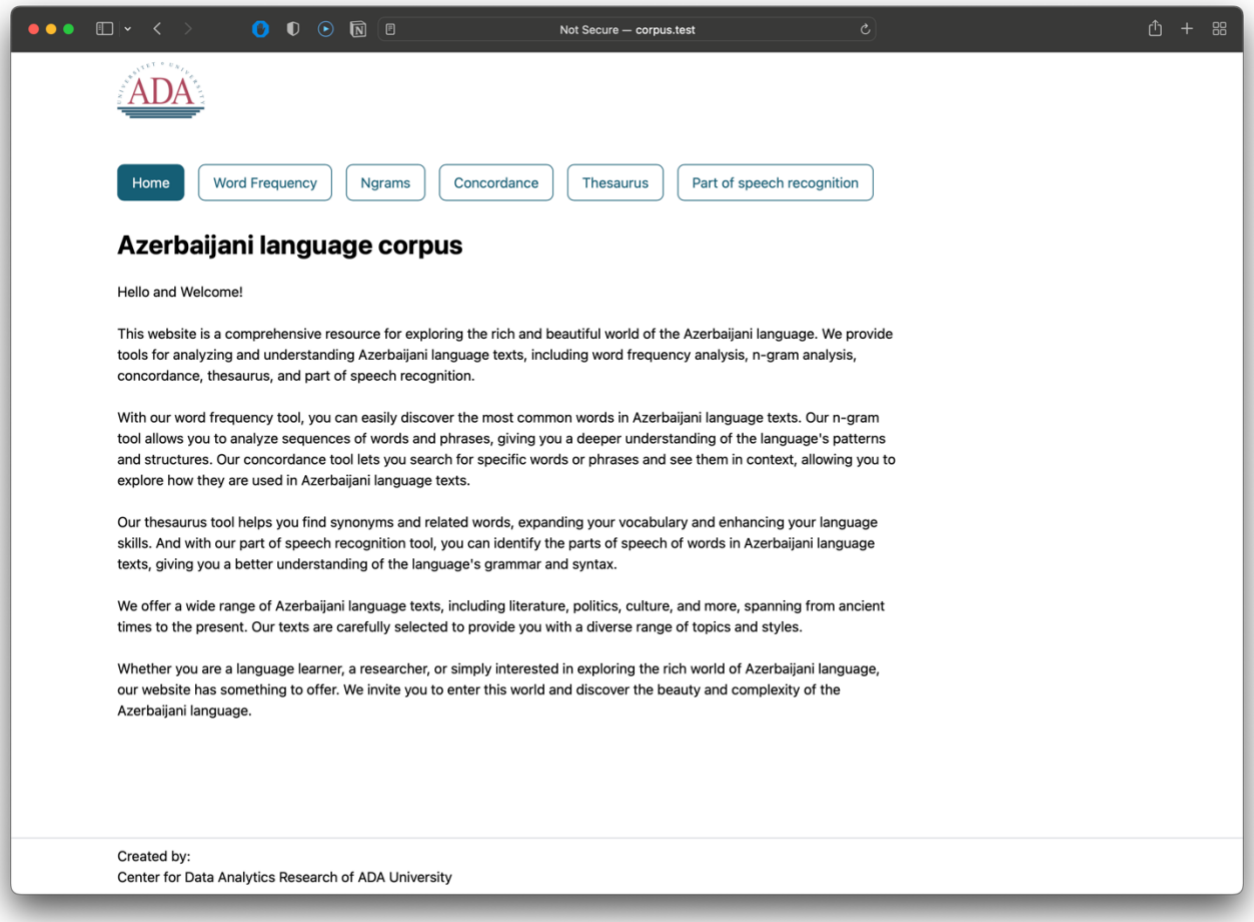

Figure 17: The user interface of the homepage of Azerbaijani language corpus application

The homepage is giving the initial information about the corpus and lets user to get started with the functionality pieces available in the application. On the top of the page we can see the logo od the ADA university, right under it we have the navigation bar with all the functionality tabs, the current tab will always be selected in different color to give a user an instant feedback on which page the user is located right now. After there is an information about the corpus and functionality, and finally, as the footer there is small note that it is created by Center for Data Analytics of ADA University. The general color theme is also adopted from the official website of the ADA university to give a single themed experience for the user.

4 RESEARCH RESULTS

After conducting thorough research and analysis, we have developed a large-scale Azerbaijani language corpus with a wide range of functionalities. It has the following features such as Word Frequency, Ngrams, Concordance, Thesaurus, and Word Sketch to enable a comprehensive language analysis. The development process also included creating an online application where users can test all of the functionalities of the corpus.

The Word Frequency function allows users to analyze the frequency of words in the corpus, giving them a better understanding of the most commonly used words in the Azerbaijani language. Ngrams, on the other hand, allows users to study the frequency of word combinations, providing a deeper analysis of the Azerbaijani language's syntax and structure. The Concordance function enables users to examine how a particular word is used in the corpus, providing insight into the word's meaning and usage.

The Thesaurus function helps users find synonyms and related words in the Azerbaijani language, making it easier to expand their vocabulary and use more nuanced language. Finally, the Word Sketch function provides a function of most frequent usage of the word in the context, enabling users to understand the word's usage in a more meaningful way.

As a result of the research, we have also developed a user-friendly online application to access all of these functionalities in one place. The application is designed to be easy to use, allowing users to access the corpus's features without any technical expertise or training. With the online application, users can analyze the Azerbaijani language in a variety of ways, from examining specific words to studying the language's overall structure and syntax.

The Azerbaijani language corpus created in this research is an extensive collection of texts comprising more than 50 million tokens. This makes it a significant resource for studying the Azerbaijani language and its characteristics. With such a large number of tokens, the corpus offers a rich and diverse dataset for various linguistic studies and analyses.

The size of the corpus enables researchers to study language patterns and tendencies in the Azerbaijani language in much greater depth and detail. The vastness of the corpus allows for a broader range of language usage, including both formal and informal language, dialects, and variations, all of which can be analyzed for linguistic features and trends.

5 SUMMARY AND FUTURE WORK

In summary, this project aimed to create a large-scale Azerbaijani language corpus and develop several functionalities such as word frequency, n-grams, concordance, thesaurus, and word sketch. The corpus was constructed by collecting text from news articles resulting in more than 50 million tokens. It is planned to be extended by adding different categories in the future. The corpus was stored in a MariaDB database and accessed through a web application built with Next.js and FastAPI.

The developed functionalities allow users to perform various analyses on the corpus, such as identifying frequently used words, extracting commonly occurring phrases, finding contextual examples of a word, and exploring synonyms and related words. The web application provides an easy-to-use interface for users to interact with these functionalities and get insights into the Azerbaijani language.

As for future work, there are several areas of improvement that can be explored. One suggestion is to develop more advanced natural language processing (NLP) functionalities such as sentiment analysis and named entity recognition. Another suggestion is to expand the corpus by collecting more diverse sources of text and incorporating spoken language data. Additionally, integrating machine learning

models such as word2vec and deep learning models for part-of-speech tagging can also be explored to enhance the accuracy of the developed functionalities.
Overall, this project contributes to the development of Azerbaijani language resources and provides a foundation for further research in the field of natural language processing and corpus linguistics.

6 REFERENCES

[1] "AzWaC – Azerbaijani Corpus from the Web." Sketch Engine, 20 Feb. 2018, https://www.sketchengine.eu/azwac-azerbaijani-corpus/

[2] Baisa, Vit, and Vit Suchomel. *Large Corpora for Turkic Languages and Unsupervised Morphological Analysis*. https://www.sketchengine.eu/wp-content/uploads/Large_Corpora_for_turkic_2012.pdf

[3] Eckart, T., Quasthoff, U. (2013). Statistical Corpus and Language Comparison on Comparable Corpora. In: Sharoff, S., Rapp, R., Zweigenbaum, P., Fung, P. (eds) Building and Using Comparable Corpora. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-20128-8_8

[4] *Leipzig Corpora Collection:* Azerbaijani news corpus based on material crawled in 2011. Leipzig Corpora Collection. Dataset. https://corpora.uni-leipzig.de?corpusId=aze_newscrawl_2011

[5] Davies, M. 2005. "The advantage of using relational databases for large corpora: Speed, advanced queries, and unlimited annotation". International Journal of Corpus Linguistics, 10 (3), 307–334.

[6] Davies, M. 2009. "Word frequency in context: Alternative architectures for examining related words, register variation and historical change". In D. Archer (Ed.), What's in a Word- list? Investigating Word Frequency and Keyword Extraction. London: Ashgate, 53–68.

[7] Davies, Mark. "The 385+ million word Corpus of Contemporary American English (1990—2008+): Design, architecture, and linguistic insights." International Journal of Corpus Linguistics 14 (2009): 159-190.

[8] W.N. Francis and H. Kucera, Brown University, Providence, RI. "The Brown Corpus." *University of Essex*, https://www1.essex.ac.uk/linguistics/external/clmt/w3c/corpus_ling/content/corpora/list/private/brown/brown.html

[9] Drahomíra „johanka" Spoustová, Miroslav Spousta, and Pavel Pecina. 2010. Building a Web Corpus of Czech. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

[10] Leo Galambosˇ. 2006. Egothor, full-featured text search en- gine written entirely in java. http://www.egothor.org/.

[11] Building Large Corpora from the Web Using a New Efficient Tool Chain (Schäfer & Bildhauer, LREC 2012), http://www.lrec-conf.org/proceedings/lrec2012/pdf/834_Paper.pdf

[12] Thomas Emerson and John O'Neil. 2006. Experience building a large corpus for chinese lexicon construc- tion. In Marco Baroni and Silvia Bernardini, editors, *Wacky! Working papers on the Web as Corpus*. GEDIT, Bologna.

[13] Andrei Z. Broder, Steven C. Glassman, Mark S. Man- asse, and Geoffrey Zweig. 1997. Syntactic clustering of the web. Technical Note 1997-115, SRC, Palo Alto, July 25.

[14] "Word Frequencies in Written and Spoken English: based on the British National Corpus." (Geoffrey Leech, Paul Rayson, Andrew Wilson), ISBN 0582-32007-0

[15] "NLTK :: Natural Language Toolkit", [Online]. Available: https://www.nltk.org. [Accessed: 6-Apr.-2023].

[16] T. Hutchinson, "Prisma", *Prisma*. [Online]. Available: https://www.prisma.io. [Accessed: 7-Apr.-2023].

[17] Esmaeilzadeh, A., Cacho, J.R.F., Taghva, K., Kambar, M.E.Z.N., Hajiali, M. (2022). Building Wikipedia N-grams with Apache Spark. In: Arai, K. (eds) Intelligent Computing. SAI 2022. Lecture Notes in Networks and Systems, vol 507. Springer, Cham. https://doi.org/10.1007/978-3-031-10464-0_45

[18] Avasthi, S., Chauhan, R., Acharjya, D.P. (2021). Processing Large Text Corpus Using N-Gram Language Modeling and Smoothing. In: Goyal, D., Gupta, A.K., Piuri, V., Ganzha, M., Paprzycki, M. (eds) Proceedings of the Second International Conference on Information Management and Machine Intelligence. Lecture Notes in Networks and Systems, vol 166. Springer, Singapore. https://doi.org/10.1007/978-981-15-9689-6_3

[19] Chodorow M, Ravin Y, Sachar HE. A tool for investigating the synonymy relation in a sense disambiguated thesaurus. Proceedings of the Second Conference on Applied Natural Language Processing; 1988. p. 144–151.

[20] Alrasheed H (2021) Word synonym relationships for text analysis: A graph-based approach. PLoS ONE 16(7): e0255127. https://doi.org/10.1371/journal.pone.0255127

[21] "Create T3 App", *Create T3 App*. [Online]. Available: https://create.t3.gg. [Accessed: 11-Apr.-2023].

[22] "English-Corpora: COCA", [Online]. Available: https://www.english-corpora.org/coca/. [Accessed: 12-Apr.-2023].

[23] "Sketch Engine", [Online]. Available: https://app.sketchengine.eu/. [Accessed: 12-Apr.-2023].