Addresses Standardization and Geocoding using Natural Language Processing

by

Meiirgali Mussylmanbay

Submitted to the Department of Data Science
in partial fulfillment of the requirements for the degree of

Master of Science in Data Science

at the

NAZARBAYEV UNIVERSITY

July 2022

Author .....
Department of Data Science
July 27, 2022

Certified by .....
Dr. Adnan Yazici
Professor
Thesis Supervisor

Certified by .....
Enver Ever
Associate Professor
Thesis Committee

Certified by .....
Askar Boranbayev
Assistant Professor
Thesis Committee

Accepted by .....
Vassilios D. Tourassis
Dean, School of Engineering and Digital Sciences

Addresses Standardization and Geocoding using Natural Language Processing

by

Meiirgali Mussylmanbay

Submitted to the Department of Data Science
on July 27, 2022, in partial fulfillment of the
requirements for the degree of
Master of Science in Data Science

Abstract

Geocoding, the process of converting the textual addresses into a pair of coordinates, is a preliminary step in spatial analysis. However, converting addresses into latitude and longitude is not a trivial task as they are presented as arbitrary text, mostly lacking completeness, and do not follow a concrete fixed structure. Therefore, the thesis discusses the theoretical fundamentals of textual data normalization and standardization techniques and presents adequate practical approaches to how addresses written in various ways can be brought to a single standard.

For binding the textual addresses with their appropriate geocodes, we have conducted practical experiments using the data collected from 5 publicly available sources and such tools as Elasticsearch, including its built-in BM25 similarity algorithm, as well as a state-of-the-art algorithm - BERT. Also, we have admitted Open Street Map address structure as a golden standard and cosine similarity algorithm as a text similarity algorithm. The practical outcomes of the models were verified on randomly chosen 100 records. The results were visualized on the map to illustrate the applicable cases of geocoding usage.

Further, the raw address data and address standardization results serve as train and test data to predict the closest address and adequate geocodes for given arbitrary address representations. For the thesis, we used models based on Transformer architecture, namely T5 and BART, for predicting 'correct' addresses. In addition, BLEU was used as a reference metric to compare the models' accuracy.

Overall, the thesis can boast rich theoretical background information and be a practical reference to how clean addresses can be revealed using state-of-the-art models given non-standard addresses.

Thesis Supervisor: Dr. Adnan Yazici
Title: Professor

Acknowledgments

This research would not have been possible without the support and assistance from my adviser and colleagues. I thank my adviser, professor Adnan Yazici, for his indispensable guidance and supervision. I am grateful to my professor for his patience in letting me work at my tempo, his sense of humor, and his insightful support whenever I needed it.

I am also indebted to my colleagues, namely data scientist Mukhmedzhan Karatayev and data analyst Dias Kalkamanov. They gave me a hand at setting up the environment, collecting data, and running the algorithms for address standardization task. Moreover, they were eager to share their knowledge and expertise whenever I had a practical obstacle or technical challenge.

Contents

Chapter 1

Introduction

The background information required to comprehend the thesis is supplied in this chapter. The background of address standardization and geocoding, as well as their significance and relevance, are covered in the first section. The remaining parts describe the motivation for the thesis, the purpose of the thesis, the problem statement, the status of related research, and the main content and organization of the thesis.

1.1 Background

With the upcoming era of big data and the rapid development and broad use of Geographical Information Systems (GIS), volumes of diverse data are being collected across several areas (Coetzee & Judith, 2009). The ability to assign coordinates to these data and the continuation of research into big data from a spatial perspective appears to be essential requirements (Jing et al., 2021). A crucial area of study is the spatial information-containing words known as addresses, which might be the primary means by which residents communicate geographical information or location (Tian et al., 2016).

The address is one of the most used methods to define an indirect geographical location. The indirect reference uses geographical codes or names about a known location without precise coordinates. In a broader sense, an address is a geographical description for all types of service delivery, including physical services like mail

delivery, commodities delivery, utility services, and emergency dispatch, as well as more abstract services like a credit application, tax collecting, customer relationship management, and land administration (Cetl et al., 2018).

Addresses by using a structured composition of geographical names and identifiers identify the permanent location of a property, such as a parcel of land, building, section of a building, or other structure. Cetl et al. (2018) stated that, although all national and local address systems share similar concepts and general properties, differences still exist in the formal and informal standards, rules, schemas, and data models worldwide. Addressing cannot be standardized internationally because addresses have a solid cultural connotation and also because addressing is governed by the laws of a particular country.

To reliably identify an address in a broader context, it must be coupled with several address components that specify its placement within a certain geographic area. Each component of an address indicates a spatial identifier, such as the name of a street, city, district, postcode, region, , municipality, or country. An address's geographic position or spatial location is represented by a spatial point including information on its origins. An address identifies a location through a geospatial reference system (GRS), and there are three categories of GRS:

(i) A geographic identifier reference system identifies the location by a label or code.

(ii) The linear reference system determines the location concerning a segment of a linear geographical object and the distance along this segment from a given point.

(iii) The location is specified by a coordinate reference system concerning a geodetic datum.

Geocoding is generally understood to translate a locational description, such as an address, into a geographic representation, such as geographic coordinates (latitude and longitude). It may be applied in a wide variety of domains such as health care and emergency crime analysis, route planning, political science, localization of

administrative and social governmental services, and computer science (Goldberg et al., 2007). Moreover, this technique is essential in several scientific fields since it is often the initial stage in creating geographic data utilized in later spatial analysis. Therefore, geocoded data's precision, granularity, and dependability are of the utmost relevance in studies that utilize address data as their underlying geographic data source. Goldberg et al. (2007) gives a general overview of the concept of geocoding, including its historical developments and future challenges where geocoding is defined as finding a geographically referenced code representing an input address determined by a processing algorithm. The processing algorithm consists of general of the following steps (Goldberg et al., 2007):

(i) Normalization and standardization. In this step, the input address is transformed into a standard form, for instance, by removing punctuation and recognizing abbreviations. The words in the input string (or substrings of the input string) are also tokenized (i.e., classified as belonging to a specific input type).

(ii) Weighting attributes. The attributes obtained from tokenizing are weighted to measure the similarity of the input address with the addresses in the reference data set.

(iii) Search the reference set for a match and output the geographical coordinates.

Address standardization, which may be summed up as the translation of an address from one format to another, is a crucial stage in the geocoding process and several other disciplines that require precise addresses. These other disciplines include health, natural and manufactured risks, law enforcement, the environment and agriculture, land-use planning, and education. The objectives of these disciplines intersect when they need the construction of substantial (i.e., enormous volume) and exact locational data sets to support their respective endeavors, sometimes in the form of massive databases. Since a considerable portion of the acquired address information is incomplete or unclear, address standardization is vital in compiling such resources. (Goldberg et al., 2014).

Standardizing addresses is one of the steps that comprise the data cleansing process. The degree of "cleanliness" of input data may be one of the most significant determinants of a geocodes success or failure (Goldberg, 2008). Address information is regarded as "dirty" for various reasons, including using non-standard acronyms, attribute orderings, and basic data input errors. In addition, to address standardization, the input data cleansing procedures consist of address normalization and parsing. Address normalization identifies the component components of an address. In contrast, parsing is typically regarded as the portion of the normalization algorithm that seeks to determine the most probable address attribute to connect with each component of the input address.

1.2 Motivation for this Thesis

Address information is essential for a large number of industries, including both the government and business. According to Meticulous Research (2021), the geospatial analytics market is expected to reach 256 billion US dollars by 2028. We should clarify that when people refer to geospatial data, they are often describing address-related data (a specific address, ZIP code, a specific latitude, and longitude which have been matched to a relevant address written in a text format, Global positioning system data, remote sensing data). Thus, conventional address information that human beings are used to exploit is also the type of geospatial data. Remarkably, geospatial analytics based on conventional addresses is characterized by extra steps, including address standardization and geocoding processes.

A few cases may be outlined as major ones among the cases of geospatial analytics. The most obvious one is market segmentation, where the goal is to divide customers into groups with common characteristics. The customer data augmented with geocoded data from ZIP codes associated with a concrete customer can contribute to the geographical visualization distribution of the customer segments, which in turn assists in promotional activity. Another case is asset management. Physical asset management refers to the systematic monitoring of equipment, buildings, and other

physical property. It is known that geospatial data has been used by law enforcement agencies to predict and prevent crimes in a number of ways, including mapping crime data, issuing alerts on events of interest, and predicting future crime figures based on historical data and known patterns. Geospatial analytics can contribute to transportation enhancement and logistics planning. In this case, based mostly on GPS data, logistics companies can re-plan and improve their ordinary routes. Geospatial analytics may be directly used in location planning. Companies face numerous considerations when they are deciding where to locate a new store, restaurant, healthcare facility, or other commercial property. A very common example is how geospatial analytics can be used in deciding where a new restaurant should be located, including market targeting, analysis of competitors in an area, and understanding the cost associated with operations (mainly supply).

These cases demonstrate how address information can be used in practice. On the other hand, an initial analysis of open databases used in Kazakhstan that contain address information related to persons and assets illustrates that addresses are stored in unstructured formats. Unstructured textual address data result in multiple representations of the same entities. These addresses obviously need to be cleaned before they can be used in the analysis; otherwise, it will yield a weak and wrong analysis.

Address Register intends to produce, accumulate, and process data on addresses in Kazakhstan. A unique code of real property has been developed within Address Register; it is called address registration code, and consists of 16 digits (gov.kz (Unified Platform of Internet Resources of State Bodies), 2022). Although setting up an Address Register was one of the steps toward address standardization, due to its lacking geocoding information, namely latitude and longitude, the address information stored in the Address Register may hardly be used in geospatial analysis.

Given the fact that the integration of Address Register with the information systems of public agencies was started in 2013; and currently, 14 governmental bodies are connected to Address Register, there are still databases in which addresses are stored in a form different from the standard set in Address Register (nitec.kz (National Information Technologies), 2022). Thus, these addresses need to be standardized first.

Postal codes may also be used to address standardization. Generally, postal codes are assigned to geographical areas, and sometimes they may also identify individual addresses. In 2016, Kazakhstan switched to a new system of postal codes consisting of 7 symbols. Within a new system of postal codes, each object has its own unique postal code. It was planned that the deployment of the new system would be finished by 2016 (bizgid.kz (Postal Codes of Kazakhstan), 2022). The postal codes used before 2016 identify the only common geographical area and fail to identify individual objects. In fact, currently, both systems are in use. Thus, postal codes of the new format cannot be used to extract standardized address information as they are not spread broadly.

As it is stated earlier, in a number of databases, address information has been stored as text chunks. In some cases, this information is presented in the form of a semi-structured text, but there are also cases when addresses are stored as arbitrary text data. Also, it should be noted that addresses are gathered from human beings, mostly at service provisioning, and can be characterized as one of the samples of natural language. Given the fact that the conversion of conventional addresses into geospatial information requires a unification of different forms of a single factual address, in practice, address standardization can be considered as the task of text similarity, which can be solved by using Natural Language Processing (NLP). In fact, being a subfield of artificial intelligence, NLP has progressed a lot due to innovations recently made by giant tech companies. Recent NLP advances promise a bright and solid solution for the task of address standardization and geocoding.

The addresses unattached to a person are open data that can be used for scientific and applied tasks. Given the fact that these addresses can be more or less easily accessed, data scarcity is not considered an obstacle to this thesis.

Data privacy issues

The Kazakh legislation defines personal data as data recorded on an electronic, paper, and (or) other material carriers which belong to a concrete person or can be used to identify a concrete person.

Given the definition of personal data, we presume that the addresses which are

not attached to concrete persons cannot be qualified as personal data. First, there is no information to whom these addresses belong. Second, compared to personal data such as full name, registration number, and others alike, there are almost zero chances to identify the person based on pure address information.

In addition, according to article 6 of the Law of the Republic of Kazakhstan, "About personal data and their protection" there are two types of personal data: publicly available personal data and the data of restricted access (or confidential data) (Shapovalova, 2022). There are a number of sources that might be classified as the sources of publicly available data, to name a few, bibliographic guides, telephone books, and address books. A small search over the Internet resulted in finding the sources, mostly online address books, which list addresses of various caliber. For instance, an online Phonebook of Nur-Sultan (n.d.) and Phonebook of Almaty (n.d.).

On the other hand, the legislation of the Republic of Kazakhstan also defines confidential data, which should be protected and cannot be revealed to the public (Shapovalova, 2022). The following secrets are certainly defined as the confidential data by the legislation of the Republic of Kazakhstan:

- personal and family secrets
- commercial or entrepreneurial secrets, adoption secrets
- secrecy of correspondence, telephone conversations, postal, telegraph and other messages
- medical secrecy and professional secrecy of healthcare professionals
- the secrecy of retirement savings, insurance, and vote
- tax, attorney, bank, and investigation secrecy
- court judgement, juror‘s secrecy, notary secrecy

After scanning the various laws and codes on the subject of what each secrecy covers, we have drawn the conclusion that pure addresses collected from different public sources without attachment to any entity or person and taken out of context cannot reveal any confidential data.

1.3 Purpose of the Thesis

The major purpose of the thesis is to present a practical solution for address standardization and binding geodata to the non-standard address written in an arbitrary form in Russian and Kazakh languages using advanced NLP methods and demonstrate the proposed solution's efficiency. To achieve the major purpose, I plan to investigate the address structure set in the Address Register. Besides the national standard, I also intend to research other available open source and proprietary standards. As I achieve an initial goal within the thesis, I plan to develop a model which will predict a standardized address to a given non-standard address written in the arbitrary form in Russian or Kazakh languages. Figure 1-1 shows a flow chart that explains the overall project workflow.

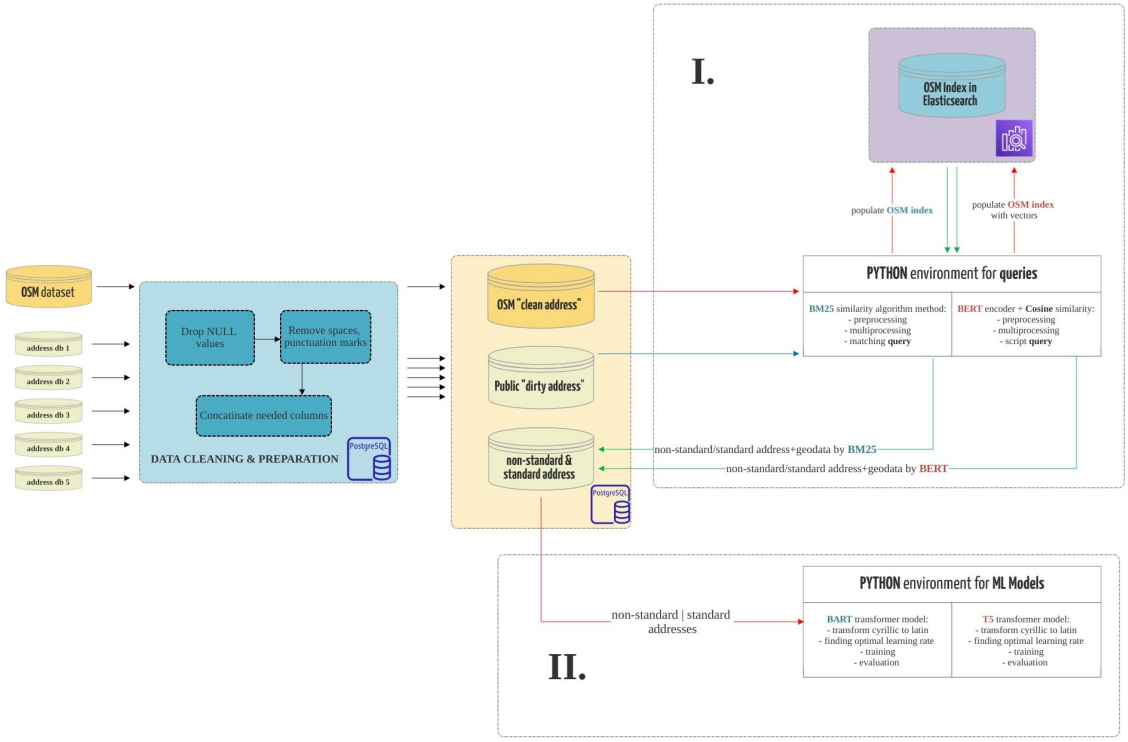

Figure 1-1: Flow chart of this work

1.4 Related Works

Currently, different researchers from all over the world have adopted different methods and approaches to standardize addresses with a correct outcome, that is, with a high degree of accuracy and a low probability of address loss and errors. In this section, we present previously conducted studies and examples of their methods and techniques for address standardization and geocoding.

Abbasi (2005) stated that address standardization is a subcategory of information extraction (IE) from semi-structured records. Information extraction is concerned with identifying predefined types of information from text. Abbasi (2005) has analyzed numerous information extraction techniques and discussed how rule-based information extraction techniques, such as RAPIER (Robotic Automated Production of Information Extraction Rules) and GRID (Global Rule Induction for text Documents), can be utilized to address standardization in the same manner as the HMM (Hidden Markov Model) model.

In the past several decades, a substantial amount of Computer Science research has been devoted to tackling the parsing difficulty. According to Christen and Belacic (2015), the most prevalent method for address standardization is the manual specification of transformation and parsing rules, and AutoStan is a well-known example of this methodology in biomedical research. AutoStan first analyzes the input string into individual words, and with the help of a re-entrant regular expression analyzer, each word is then mapped to a token of a certain class. Moreover, Christen and Belacic (2015) proposes an alternative to these rule-based deterministic approaches, a technique based on HMM to clean and standardize addresses and reach a 72.87% address level matching using Geocoded National Address File (GNAF) data.

According to Lu et al. (2019) the current address standardization methods are mainly divided into two types: address matching based address normalization methods and natural language processing-based address standardization methods. The method of address standardization based on address matching classifies the address before obtaining a standardized address based on the matching mechanism. Kang et

al. (2015) establishes a tree model for the address and improves the address matching rate by creating the spatial constraint link between the address components. Y. Wang et al. (2016) proposes a standard processing approach based on address tree layer-by-layer matching concurrently considering the address location connection. This approach employs geographic location information to increase the accuracy of address standardization by facilitating matching throughout the standardization process. Based on the dictionary matching technique, the address normalization method provided by Ying et al. (2017) has greater operational efficiency and particular address text parsing value. However, this solution depends too heavily on the standard address database and does not effectively manage element-missing and address-error issues.

The NLP-based address standardization technique accomplishes high-precision address matching by constructing a natural language model directly. On the basis of the Bayes probability model and cosine similarity, Yu and Zhang (2019) presents an address normalization approach that uses a finite state automaton to modify the levels of the address. Address normalization performed by Higuera and Oncina (2013) using a probabilistic finite automaton, which can finish the matching task in a polynomial amount of time but has the drawback of being heavily data dependent. In contrast, to address matching algorithms, algorithms based on natural language models demand models and data of a higher quality.

Tian et al. (2016),in his paper, proposes an optimal approach for Chinese geocoding address matching with three components: address modeling, address standardization, and address matching. The suggested model is structured around a standardization process based on an address tree model proposed in a Kang et al. (2015)'s study. This paradigm parses and organizes the address string into a collection of address elements X and a semantic collection S. Create the root node and retrieve address element X1. Finally, all the semantic components in S1 related to X1 are traversed to generate semantic address nodes, which are then linked to the root node (Matci & Avdan, 2018).

1.5 Main content and organization of the Thesis

The thesis is organized and contains as follows. After chapter 1, Introduction, comes chapter 2, Theory, where the relevant theory that is needed to understand the basic concepts of the NLP methods used in the framework of a research question is given. In chapter 3, Methodology, the data sets, text similarity methods, and ML models are presented. Chapter 4, Experiments and Results, describes different experiments on training models and evaluation. Moreover, the discussion part is included. In chapter 5, Conclusion, the conclusion of the thesis is presented.

Chapter 2

Theory

This chapter presents the general theoretical foundation and the main references for the thesis. The first part focuses on the levels of Address Standardization. After, general information about Geographical Information Systems is presented. The remaining parts contain the theory of implementing Sentence embedding methods, Text similarity metrics, and the Elasticsearch search engine.

2.1 Different Levels of Address Standardization

Standardization of addresses is the process of checking and correcting address records to a standard format, according to an authoritative database, consisting of several stages. In our case, referring to Goldberg et al. (2014)'s numerous studies, we consider this process on three main levels:

The input address data

Diverse fields collect locational input data in several formats and with differing degrees of precision. One of the significant obstacles faced by individuals who want correct address data is the presence of low-quality or partial input data. Consequently, the main objective of address collecting systems should be to reduce the uncertainties produced when the fundamental parts or qualities of address data are frequently omitted by accident. The ideal situation is to gather the most comprehensive and

accurate information possible at the start of any address data gathering operation. According to Goldberg (2008), the desired level of completeness in address data is referred to as the “gold standard”.

Address normalization

Address normalization might be regarded as a preparatory step to effective address standardization. This phase in the process involves, as a general rule, identifying the component elements or features of an address so that such address can afterward be translated into another format of the user’s choosing. A normalization algorithm must find the "gold standard" address characteristic to correlate with each input address component. Consequently, address normalization is essential to the address cleaning procedure. Without defining which text relates to which address characteristic, it is not easy to translate the text among standard formats or utilize it for feature matching. In geocoding, various normalizing strategies may be employed, including replacement, context, and probability-based normalization.

Address standardization

In the most narrow sense, address standardization is the process of changing an address from one normalized format to another. It is closely related to normalization and depends significantly on how well the normalization process works. In a nutshell, standardization is transforming normalized data into the format required by later address processing system components like a geocoder. A shortage of a standard format for addresses can be used for various reasons and by various organizations, making it difficult for data to be shared between them. An agreement to implement a standard format is necessary for interoperability. In order to develop an address standardization system, it is necessary to overcome the fact that different organizations (government, academia, etc.) may demand or utilize more than one address standard for various purposes, including those not related to geocoding. As a result, translation between shared address standards may be necessary following attribute identification and normalization.

2.2 Geographical Information Systems

Geographic Information System (GIS) integrates hardware and software to store, analyze and map spatial data (Elwood, 2006). GIS allows users to visualize (i.e., map) geographic aspects of data including locations or spatial concentrations of phenomena of interest.

Geographic information science, often known as geospatial information studies, is a significant subfield within the broader academic topic of geoinformatics, commonly abbreviated as GIS ("Geographic information system (GIS)", 2021). Many diverse procedures, technologies, and methodologies fall under the umbrella name of GIS. Engineering, planning, logistics, insurance, telecommunications, and other business-related fields can benefit from its use. Since location-enabled services rely on spatial analysis and visualization, GIS and location intelligence technologies provide the core of these services.

Improved functionalities of GIS have been developed due to recent advancements in computer technology, remote sensing, GPS, and communication technology. ArcGIS, QGIS, OpenStreetMap, and GoogleMaps are some of the many available commercial and open source GIS software applications. These applications allow users to manipulate and analyze geographic data by visualizing statistics such as climate data and trade flows on maps constructed from layer-building techniques. Moreover, many academic organizations and departments in the humanities and the sciences employ GIS software for revolutionary research purposes.

2.3 Word and Sentence Embedding

For machine learning algorithms to function with textual input, the data must first be converted into numerical data that an algorithm can comprehend. In other words, the words must be given meaning for machine learning and deep learning algorithms. This is not as simple as with graphics or images, which can be represented simply by numbers. In most cases, word embedding methods convert text into vectors. These

are models that have been trained to map words or phrases into a real-valued vector of a fixed size in a way that meets specific semantic and linguistic requirements (Vusak et al., 2021).

Word embedding algorithms are the standard way to improve the performance of models in many NLP tasks. The wide range of applications have reported improvements upon integrating word embedding, including machine translation (Zou et al., 2013), syntactic parsing (Weiss et al., 2015), text classification (Kim, 2014) and question answering (Bordes et al., 2014).

In contrast to word embedding, sentence embedding is proposed to provide distributed representations for texts of various lengths, from single phrases to full publications. In addition to text categorization, information retrieval, question answering, and query rewriting, it has been utilized extensively and efficiently in various domains (Xing et al., 2018). Sentence embedding approaches portray sentences as continuous vectors in a low dimensional space, representing the connections among many words and phrases in a single vector. The vector of a phrase is traditionally constructed by weighing and averaging the word vectors of its elements (Lamsiyah et al., 2019). Recently, more elaborated architectures have been introduced to construct more viable sentence representations, such as Doc2vec, SentenceBERT, InferSent, and Universal Sentence Encoder.

2.4 Text Similarity Metrics in NLP

It is becoming increasingly important to use text similarity measures in tasks such as information retrieval, text classification, and document clustering to detect and track topics of interest, as well as for question-answering, text summarization, and machine translation (Gomaa & Fahmy, 2013). Finding similarities between words is a crucial aspect of text similarity, which is subsequently utilized as a foundation for identifying similarities across sentences, paragraphs, and documents.

There are four major categories of text similarities measures (Alqahtani et al., 2021). The string-based similarity measure is one of the oldest methods. The main

forms of string-based similarity comprise both character-based similarity and token-based similarity functions (Kuipers, 2013). The corpus-based similarity follows semantic approaches. The methodology aids in the determination of similarity between two concepts in terms of information from the respective corpora, which is a collection of electronic, spoken, or written text. These methods enshrine some predefined set of sentences coupled with their translation in other dialects, with the intention of matching the input text within the corpus and the final translations (Kuipers, 2013). The knowledge-based similarity measures are defined as a series of semantic measures enlisted for information adopted from semantic networks. The aim of such information is geared towards the identification of the extent of similarity within words. Knowledge-based similarity consists of semantic relatedness and semantic similarity. Hybrid classification similarity measures do not form a distinct group. They are combinations of the previous approaches in an effort to attain their merits (Kuipers, 2013). These approaches work on recursive approaches to tackle limitations of the other measures.

Many experiments were conducted with different similarity learning algorithms over various datasets. According to Liu et al. (2014); Mohammadi and Khasteh (2020) the Cosine text similarity algorithm is more accurate than other similar algorithms such as the Euclidean distance, Pearson correlation coefficient (PCC), Jaccard, and Mean Square Difference (MSD) in terms of measuring the similarity or difference between texts. Moreover, traditionally the Cosine measure has been shown to perform well for the textual datasets (Qamar, 2010). All these metrics have their specification to measure the similarity between two queries.

2.5 Search engine: Elasticsearch

Elasticsearch is an open-source distributed database system capable of real-time full-text search and analytics and has been successfully used in conjunction with a gazetteer to identify street addresses and assign latitude and longitude coordinates. It is implemented in Java via the Apache Lucene library (Kumar et al., 2018). Ac-

cording to the book "Elasticsearch Server" it is "widely utilized in many common or lesser-known search and data analysis platforms" today (Kuc & Rogozinski, 2016). It was initially released in 2010, and its popularity has grown steadily over the years afterward.

Designed for managing Big data, Elasticsearch is document-oriented instrument. The information in Elasticsearch is stored as JSON documents. All of a document's fields are searchable and indexed by default (Figure 2-1). Elasticsearch's near real-time search is made possible by its default behavior of refreshing the index once every two seconds (Sonawane et al., 2018). Items fields that are stored as JSON documents are mapped, and all items within an index will have the same field mapping. Field mapping options include string, boolean, data, geo-point, and geo-shape and can be explicitly set by the user or dynamically mapped by Elasticsearch. Items can be bulk uploaded, and each item in an index is assigned a unique identifier. The aggregation capabilities of Elasticsearch allow it to execute complicated analyses on the stored data (Divya & Goyal, 2013).

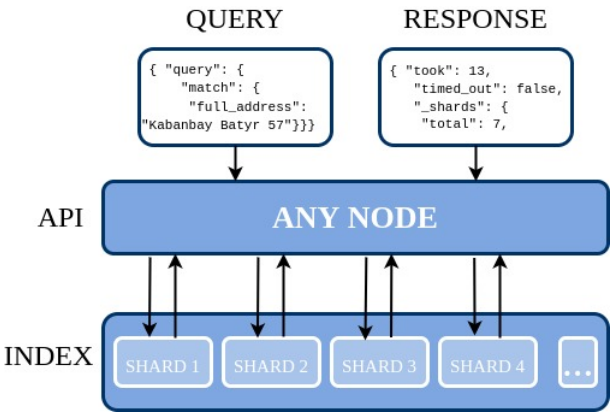

Figure 2-1: Simple Elasticsearch query works

A working Elasticsearch server may be launched immediately after installation, even with the default configuration, due to the system's ease of installation and configuration. The primary benefits of an Elasticsearch architecture are horizontal scalability and effective load balancing (Zamfir et al., 2019).

According to Bartlett (2019), Elasticsearch was selected for geographic informa-

tion retrieval and storage for the collection due to its relatively simple structure that permitted the creation of one index to store lexicons and a second index to house Collection texts. The lexicons use the geo-point and geo-shape field types to store geographic coordinates of the locations being sought; thus, the lexicon index functions as both lexicon and gazetteer.

Elasticsearch default document sorting uses the TF-IDF model to calculate the score of text similarity. In the TF-IDF model, there are two factors that affect the ordering of correlation: Term Frequency (TF) and Inverse Document Frequency (IDF), which respectively represent the occurrence frequency of a certain word in the document and the number of documents in the document set . The higher the word frequency, the higher the relevance. The more documents, the lower the relevance. The equation 2.1 shows the equation for score:

$$score(q,d) = \sum_{t}^{q} \sqrt{TF_{td}} * lDF_{td}^{2} * norm(t,d) * boost(t)2^{-n} \tag{2.1}$$

where *Score* is a relevancy score. The *norm* is a regular value for the length of a field; the shorter the field, the higher the weight. *Boost* silent is 1, so the user can assign weights.

Chapter 3

Methodology

3.1 Datasets

In this work, in order to implement the address standardization task, we first needed to designate the data as a database of dirty addresses and clean addresses. *Dirty addresses* are addresses from different public sources that we use as input data that will need to be standardized and geolocations defined for each address. Meanwhile, *clean addresses* are an Open Street Map (OSM) database standardized addresses with geolocation, which we will take as the "gold standard", as noted in the section 2.1.

For storing, processing, and searching data, the PostgreSQL (v.14), which is a free and open-source relational database management system emphasizing extensibility and SQL compliance, and the Elasticsearch (v.8.2.3) search engine (section 2.5) were used.

3.1.1 Data collection

We have collected one database of dirty addresses from 5 different open public sources. Each of the tables of dirty address source consists of six columns (full_address, region, district, city, street, housenumber) and contains about 300,000 records, for a total of 1,500,000 addresses (Table 3.1).

In our work, several different sources with addresses were considered clean ad-

full_address	region	district	city	street	housenumber
МАНГИСТАУСКАЯ АКТАУ АКТАУ 4 3	МАНГИСТАУСКАЯ	АКТАУ	АКТАУ	4	3
АКТЮБИНСКАЯ УИЛСКИЙ ОЙЫЛ ЛЕНИНА 32	АКТЮБИНСКАЯ	УИЛСКИЙ	ОЙЫЛ	ЛЕНИНА	32
ЖАМБЫЛСКАЯ ШУСКИЙ ШУ КИРГИЗБАЕВА 22	ЖАМБЫЛСКАЯ	ШУСКИЙ	ШУ	ШКИРГИЗБАЕВА	22
АЛМАТИНСКАЯ АКСУСКИЙ КЕҢЖЫРА ОРТАЛЫҚ 3	АЛМАТИНСКАЯ	АКСУСКИЙ	КЕҢЖЫРА	ОРТАЛЫҚ	3
...	...	...	...	...	...
...	...	...	...	...	...
АТЫРАУСКАЯ АТЫРАУ АТЫРАУ СОВЕТСКИЙ 123	АТЫРАУСКАЯ	АТЫРАУ	АТЫРАУ	СОВЕТСКИЙ	123

Table 3.1: "Dirty_address" database‘s table contents

dresses: the Address Register of the Republic of Kazakhstan, as well as other databases with addresses from GIS applications like Google Maps, ArcGIS, and OSM. The Address Register of the Republic of Kazakhstan (AR), which is a unified register of addresses of real estate objects on the territory of the Republic of Kazakhstan, including administrative-territorial objects from regions and cities to houses and apartment numbers, was very suitable as a "gold standard". This database is presented as an open-source; however, no data on the geolocation of localities is presented. In the case of Google Maps and Arcgis, addresses are represented with geo data; however, to access this data, a license purchase is needed, that is, an issue of high cost. Therefore, we settled on the address database provided by OSM, which is available in open form and, in addition, contains addresses geodata.

OSM is an open-source dataset with map data from all over the world (“OSM map features”, 2018). The data has been collected by crowdsourcing, which means that the content is collected from a great number of people. Since crowdsourcing never stops working, OSM has a chance of being updated as fast or faster and as accurate or more accurate than leading licensed map services. It uses the open data under the Open Database License (ODbL) published by Open Data Commons, part of the Open Knowledge Foundation. The license allows anyone to copy, distribute and present the data as long as OSM is credited with "© OpenStreetMap contributors" (OpenStreetMap, 2018).

The data with Kazakhstani addresses were downloaded in *.osm.pbf* format from the *www.geofabric.de* website, which provides raw data from the OSM dataset. This file was uploaded to PostgreSQL using the *osm2pgsql* command-line tool, which is

used to import OSM data into a PostgreSQL database for rendering into maps and many other uses. In the end, we received an OSM database of Kazakhstani addresses with geodata consisting of 8 tables. (Table 3.2):

table_schema	table_name	rows_n
osm_kz.public	planet_osm_nodes	20268350
osm_kz.public	planet_osm_ways	2325858
osm_kz.public	planet_osm_polygon	1843927
osm_kz.public	planet_osm_line	476440
osm_kz.public	planet_osm_point	385523
osm_kz.public	planet_osm_roads	47093
osm_kz.public	planet_osm_rels	11014
osm_kz.public	spatial_ref_sys	8500

Table 3.2: Contents of the "osm_kz" database with number of rows in each table

3.1.2 Data cleaning and preparation

In the process of data preparation, that is, in the process of cleaning and converting raw data before processing and analysis, we paid special attention to the records with empty data that we discovered during the general analysis of the whole data. It is important to note that the data preparation process is an important step before processing and often involves making corrections to the data, reformatting the data, and combining datasets to enrich the data.

In the case of the dirty_address database, we needed to clear the cells of each column with a NULL value. As we know in practice, records with empty values have a negative impact on data analysis and processing. Due to the fact that data with dirty addresses from different sources were manually entered by people, in addition to empty values, it was also revealed that some columns have incorrect values like punctuation marks("!!", "*"), spaces ("_", "-"), and others ("no address", "###"). Such cases as lack of records or spaces instead of records would mean that the address would be incomplete, and this would affect the poor outcome in the results of the work.

Accordingly, we have deleted records with similar incorrect values. Moreover, using the concatenation function, we added new columns like *region_ district*, *city_ street*, which are necessary for further use. Also, a *city_ street_ vectors* column with empty cells was created, where during the experiments, vector values (formed using the word embedding encoders) for each entry of city_street will be recorded. With the removal of empty values from each column, there was a slight decrease in data. Finally, we received a dataset with five tables (a separate table for each source) consisting of columns: *full_ address, region, district, city, street, housenumber, region_ district, city_ street, city_ street_ housenumber, city_ street_ vector*. A description of the final prepared dataset with dirty addresses is given in Table 3.3.

table_schema	table_name	rows_n
dirty_address.address	source1_300k	294967
dirty_address.address	source2_300k	292816
dirty_address.address	source3_300k	290352
dirty_address.address	source4_300k	278585
dirty_address.address	source5_300k	205010

Table 3.3: Contents of the "dirty_address" database with number of rows in each table

In the process of analyzing the "osm_kz" database, we found that not all addresses are complete. That is, points and numbers are represented without a city or street. Hence we used the PostGIS‘s (an open source software program, that adds support for geographic objects to the PostgreSQL object-relational database) *st_ contains* and *st_ distance* functions to get the rest of the data from spatial relationships. In other words, the necessary data was collected from each table into one table. Consequently, also deleting records with empty values and concatenating some columns, we prepared a dataset of enriched clean data with *osm_ id, full_ address, region, district, city, street, housenumber, region_ district, city_ street, city_ street_ housenumber, location (longitude-latitude)* columns consisting of a total of 663 679 addresses (Table 3.4).

In conclusion, we have prepared datasets like dirty_address - consisting of 5 tables

table_schema	table_name	rows_n
osm_kz.address	osm_700k	663679

Table 3.4: Contents of the "osm_kz" database with number of rows in a table

and defined in our work as dirty addresses source, and osm_kz - consisting of one table and defined as clean addresses source.

3.2 Methods of binding geodata to addresses

In accordance with section 3.1.2, we have prepared a dataset of dirty addresses that need to be brought into a standard form and also define geodata for them. This is one of the main tasks of this stage. In addition, at the end of these works, we will receive ready-made data with dirty and corresponding clean addresses for use in our proposed machine learning model.

Elasticsearch itself, which is the fastest search engine (Section 2.5), is used as a storage for a clean address, where we query for each dirty address its clean appearance. Since we are talking about searching and matching addresses, that is, working with data in text form, we have tried out methods of text similarity. Thus, we propose two different methods that have coped very well with this task. The first is the BM25 similarity algorithm in Elasticsearch, and the second is the Sentence embedding method. All tests, comparisons, and relevant analyses and conclusions regarding these methods are more detailed in the 4.2 section.

3.2.1 BM25 similarity algorithm in Elasticsearch

When it comes to information retrieval, the Okapi BM25 similarity algorithm is a ranking function that search engines as Elasticsearch employ to determine the relevance of documents to a particular search query. BM is an acronym that stands for best matching. Okapi BM25 similarity (BM25) is the configuration that is now used by default in Elasticsearch. It is a TF-IDF-based similarity that includes built-in "tf" normalization and is thought to be more effective for use with shorter fields (elastic.co,

2022). Although it is not the most recent model in natural language processing for document retrieval to be considered state-of-the-art, it is a highly common trade-off between accuracy and scalability.

The BM25 function assigns a score to each document in a corpus that is based on how relevant the content is to a particular text query. It is a probabilistic model that compares the frequency with which it appears in the provided text for each word of the query to the frequency with which it appears in the whole corpus. Suppose a term from the query frequently appears in the document but also frequently in many other documents. In that case, it is likely to be considered less essential than a word frequently appearing in the document but seldom appearing throughout the corpus.

For a query Q, with terms q_1,...,q_n , the BM25 score for document D is shown in the equation 3.1.

$$BM25(D,Q) = \sum_{i=1}^{n} IDF(q_i, D) \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{d_{avg}})} \tag{3.1}$$

where:

- $f(q_i, D)$ is the number of times term q_i occurs in document D.
- $|D|$ (length of the document) is the number of words in document D.
- d_{avg} is the average number of words per document.
- b and k_1 are hyperparameters for BM25. k_1 - controls non-linear term frequency normalization (saturation), the default value is 1.2. b - controls to what degree document length normalizes "tf" values, the default value is 0.75.(elastic.co, 2022)

For example, in the query “Mangylik-el street Baikonyr district”, words “street” and "district", which should be present in all documents, whereas “Mangylik-el” or “Baikonyr” should be more specific of a small subset of documents. Hence those latter words will be weighted more in the BM25 score of each documents.

3.2.2 BERT encoder and Cosine similarity in ElasticSearch

BERT encoder

Along with the BM25 similarity algorithm, for determining the binding of geodata to dirty addresses, we used the Sentence embedding technique BERT, which, in preliminary comparison with other techniques like GloVe and Universal Sentence Embedding, turned out to be better (see the section RESULTS).

In the area of deep learning, natural language representations are commonly employed as features for machine learning tasks or pre-training. Devlin et al. (2018) proposed a new pre-trained language representation model BERT (Bidirectional Encoder Representations from Transformers), which has set an excellent performance on various sentence-pair similarity comparison tasks. BERT consists of several transformer encoder layers that enable models using it to extract deep language features on both token-level and sentence-level (T. Wang et al., 2022). BERT pushes the state of the art in NLP by combining two powerful technologies:

→ It is based on a deep Transformer encoder network, a type of network that can process long text efficiently by using self-attention.

→ It is bidirectional, meaning that it uses the whole text passage to understand the meaning of each word.

The fundamental component of the BERT network (figure 3-1) is the BERT Encoder block, which is responsible for its implementation. It is constructed out of 12 layers of consecutive transformers, each of which has 12 attention heads. It is estimated that there are 110 million parameters in all. Every token that is sent into the block is first inserted into a learned embedding vector that is 768 positions long. After that, each embedding vector undergoes a progressive transformation each time it travels through one of the BERT Encoder layers:

→ Through linear projections, every embedding vector creates a triplet of 64-long vectors, called the key, query, and value vectors.

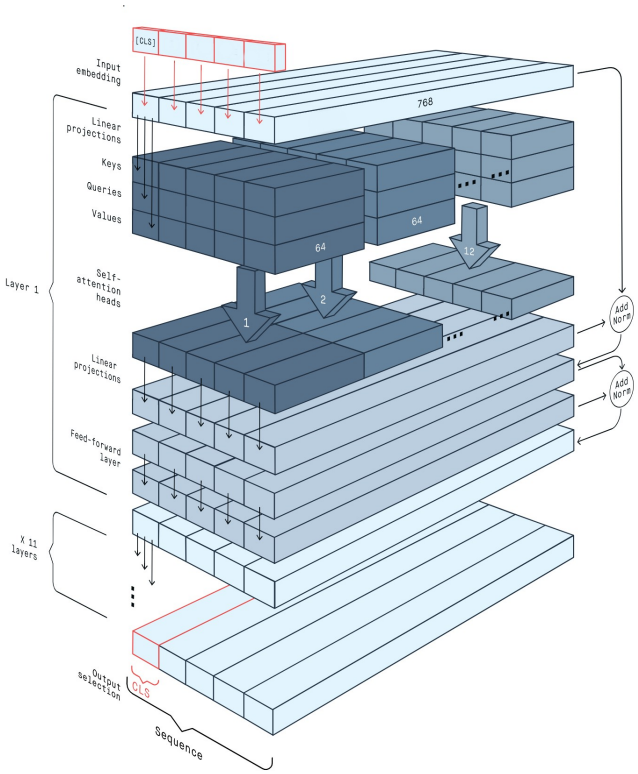

Figure 3-1: The BERT base architecture (Devlin et al., 2018)

→ The key, query, and value vectors from each of the embeddings are fed into a self-attention head, which generates a single vector that is 64 vectors long for each input triplet. Because each and every output vector from the self-attention head of BERT is a function of the whole input sequence, it is possible to say that BERT is aware of its context.

→ A single embedding vector utilizes 12 distinct triplets of key, query, and value vectors, all of which pass through their individual self-attention heads. These triplets are created using separate linear projections. Because of this, each self-attention head can concentrate on a distinct facet of the tokens' dynamic interaction with one another.

→ First, the output from each of the self-attention heads is concatenated together; then, after passing through a further linear projection and a feed-forward layer, both aid in leveraging deep non-linearity, the concatenated output is used. To further strengthen the robustness of the system, residual connections from ear-

lier states are utilized.

The outcome is a sequence of changed embedding vectors that are delivered 11 more times via the same layer structure. After the twelfth encoding layer, the embedding vectors include more precise information about each token. When doing classification tasks, "CLS" (the default) will only deliver the converted embedding of the first token; this is often the most effective method. The result of calling "Sequence" is the transformed embedding for each of the input tokens. As an aggregate sequence representation for NSP (next sentence prediction), the BERT network structure carries a unique classification token denoted "CLS". In order to convert "Sequence" into one vector, a mean pooling operation is used. Each of these tokens within "Sequence" has a value of 768. This pooling method will take the average of all token embeddings and compress them into a single 768 vector space, so producing a 'sentence vector'.

SentenceTransformers is a Python framework for state-of-the-art text and sentence embeddings. Each sub neural network takes in data, maps it to a high-dimensional feature space, and then outputs the resulting representation. By computing the distance between the two representations, such as cosine distance, researchers can compare the similarity of the two input sentences.

Cosine similarity

Cosine similarity evaluates the similarity between two inner product space vectors. It is determined by the cosine of the angle that exists between two vectors, and it reveals whether or not the vectors point in a direction that is nearly identical to one another (Han et al., 2012). In text analysis, it is frequently utilized as a measurement tool for document similarity. It is useful to use it to compare texts or, for instance, to provide a rating of documents in relation to a specified vector of query terms. Let **x** and **y** be two vectors for comparison. Using the cosine measure as a similarity function, we have

$$\mathbf{sim(x,y)} = \frac{x \cdot y}{||x||||y||} \tag{3.2}$$

where ||x|| is the Euclidean norm of vector $\mathbf{x} = (x_1, x_2, ..., x_p)$,
defined as $\sqrt{x_1^2 + x_2^2 + ... + x_p^2}$.
Conceptually, it is the length of the vector. Similarly, ||y|| is the Euclidean norm of vector **y**. The measure computes the cosine of the angle between vectors x and y. A cosine value of 0 means that the two vectors are at 90 degrees to each other (orthogonal) and have no match. The closer the cosine value to 1, the smaller the angle and the greater the match between vectors. As a example, suppose that **x** and **y** are the vectors, where x=(0,0,2,0,0,2,0,3,0,5) and y=(1,0,1,0,1,1,0,2,0,3), how similar are **x** and **y**:

$\mathbf{x} \cdot \mathbf{y} = 0 \times 1 + 0 \times 0 + 2 \times 1 + 0 \times 0 + 0 \times 1 + 2 \times 1 + 0 \times 0 + 3 \times 2 + 0 \times 0 + 5 \times 3 = 25$

$||x|| = \sqrt{0^2 + 0^2 + 2^2 + 0^2 + 0^2 + 2^2 + 0^2 + 3^2 + 0^2 + 5^2} = 6.48$

$||y|| = \sqrt{1^2 + 0^2 + 1^2 + 0^2 + 1^2 + 1^2 + 0^2 + 2^2 + 0^2 + 3^2} = 4.12$

$\mathbf{sim(x,y)} = 0.94$

Cosine similarity is superior to Euclidean distance because even if two text documents are separated by a vast Euclidean distance, there is a possibility that they are contextually near (Kanani, 2019).

3.3 SequenceToSequence Models

SequenceToSequence (Seq2Seq) model can be utilized for a variety of sequence translation-related tasks.Translating an input sequence into an output sequence of any length (independent of the input sequence length) is known as Sequence Translation (figure 3-2). SequenceToSequence model which provided by the ArcGIS Developers (2022) is built on top of Hugging Face (2022c) transformers (state-of-the-art Machine Learning for PyTorch, TensorFlow) library. A wide variety of transformer architectures can be accessed through this library. Early studies on transfer learning for NLP utilized recurrent neural networks (Peters et al., 2018), but "Transformer" architecture-based models have recently become more prevalent. The Transformer was initially demonstrated to be useful for machine translation, but it has since been implemented in several NLP contexts (Devlin et al., 2018). All of the models we explore are based

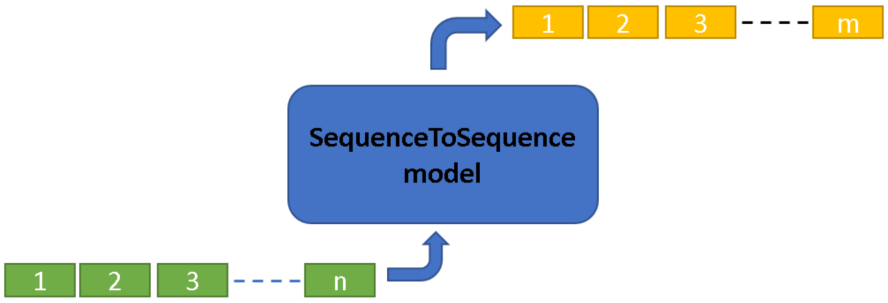

Figure 3-2: High-level view of sequence translation. (ArcGIS Developers, 2022)

on the Transformer design due to its growing ubiquity.

The APIs provided by Transformers make it easy to obtain and train state-of-the-art models. Using pre-trained models saves time, cost, and resources by reducing the need to start from scratch with a new model. The transformer architecture, as proposed in the Vaswani et al. (2017), consists of an encoder block and a decoder block. ALBERT, BERT, and other recent transformer-based architectures rely solely on the transformer's encoder. While other models, such as GPT, GPT-2, and others, only use the decoder part. For Seq2Seq tasks, however, both the encoder and decoder are required. T5, mBart, and Bart are examples of architectures that maintain both the encoder and decoder.

3.3.1 The "Transformer" model

The original Transformer is a six-layered Encoder-Decoder architecture that generates a target sequence from a source sequence using the Decoder. At a high level, the Encoder and Decoder are composed of a self-attention layer and a feed-forward layer. An additional attention layer enables the Decoder to map its necessary tokens to the Encoder for translation. Self Attention permits the search of remaining inputs at various positions to identify the significance of the currently processed input. This is conducted on all input words to achieve more excellent encoding and contextual comprehension of all words.

Transformer architecture was developed to induce parallelism in LSTM, and RNN's sequential data, where input tokens are delivered instantly and matching embedding

are formed concurrently via the Encoder. For instance, a pre-trained embedding space like GloVe, which maps a word to a vector that can be trained on the fly, can be utilized to save time. Nonetheless, the same tokens in different sequences may have various meanings resolved by a positional encoder that generates context-based word information about its position. As a result, attention vectors are generated that identify the significance of the i^{th} word in a sequence concerning other words, which is put back into the attention layer to increase contextualization further. These attention vectors are then supplied into the feed-forward Neural Network. For the following 'Encoder' or Decoder's 'Encoder-Decoder Attention' block, they are changed in to a more digestible form.

The latter receives Encoder output and Decoder input embedding, which executes attention between them. This provides the importance of the Transformer's input tokens in relation to its destination tokens since the decoder establishes the actual vector representation between source and target mapping. The decoder predicts the following word using softmax, which is carried out across many time steps until the token representing the completion of the sentence is generated. There are residual connections followed by a layer normalization (Ba et al., 2021) step at each Transformer layer to accelerate training during backpropagation. All of the transformer architectural details are demonstrated in Figure3-3.

Encoder and Decoder Stacks

Encoder: Six $\mathbf{N = 6}$ identical layers make up the encoder's structure. There are two sub-layers under each of the main layers. The first is a multi-head self-attention mechanism, while the second is a straightforward, position-wise, completely linked feed-forward network. We apply a residual connection (He et al., 2016) around each of the two sublayers, followed by layer normalization (Ba et al., 2021). Therefore, *LayerNorm(x + Sublayer(x))* is the output of each sub-layer, where the function implemented by the sub-layer is *Sublayer(x)*. To support these residual connections, all model sublayers and embedding layers provide outputs with the dimension d_{model} = 512.

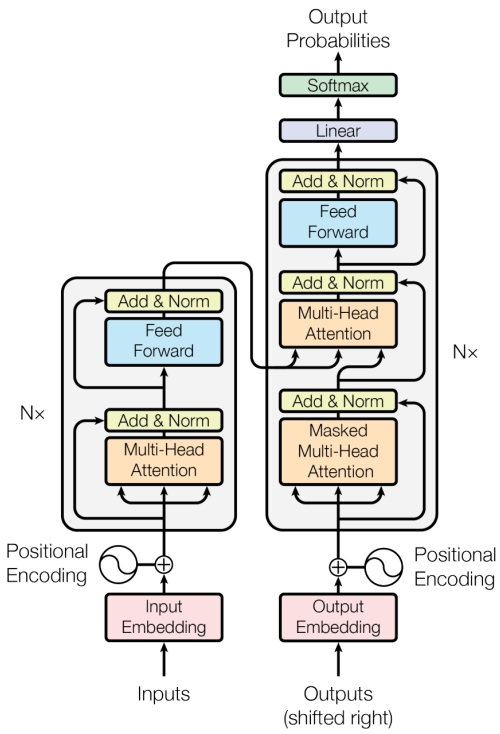

Figure 3-3: The Transformer model architecture. (Vaswani et al., 2017)

Decoder: The decoder is also N = 6 identical layers stack. The decoder adds the third sublayer (Encoder-Decoder Attention) to each encoder layer in addition to the two already there, allowing it to execute multi-head attention on the encoder stack's output. Similar to the encoder, residual connections, followed by layer normalization, are utilized around each sublayer. A further modification is made to the decoder stack's self-attention sublayer in order to stop positions from paying attention to following positions. This masking, along with the fact that the output embeddings are off by one position, makes sure that the predictions for position i can only be based on the known outputs at positions before i.

Attention

A mapping between a query and a collection of key-value pairs to output is an attention function, where the query, output, and the key-value pairs are vectors. The output is generated as a weighted sum of the values, with the weight allocated to each value being determined by a compatibility function between the query and the

relevant key.

Multi-head attention: MHA increases the ability of the model to emphasize different positions of sequence markers by implementing attention in parallel several times. Using a linear layer into the expected dimensions, the resulting individual output data or attention heads are combined and transformed. Each of the numerous heads permits observing the sequence components from a different perspective while giving equivalent token representations. Due to the high dot product, each token's self-attention vector may weigh the word it represents more heavily than other words. This is ineffective, given that the objective is to achieve identically evaluated interactions with all tokens. Therefore, self-attention is computed eight times, yielding eight distinct attention vectors for each token, which are utilized to generate the final attention vector for each token through a weighted sum of all eight vectors. The resulting multi-headed attention vectors are calculated in parallel and sent to the feed-forward layer.

Masking: Each consecutive target token T_{t+1} is created using the same number of source tokens $(S_0, ..., S_{t+n})$ in the encoder. However, in an autoregressive decoder, only prior time-stepped target tokens $(T_0, ..., T_t)$ are used for future target prediction purposes, a phenomenon known as causal masking. This is supplied to maximize the learning of the target tokens that will be subsequently translated. Consequently, during parallelization through matrix operations, the forthcoming target words are masked to zero so that the attention network cannot look into the future. The above-described Transformer resulted in substantial enhancements to the NLP area. This results in an abundance of high-performance designs, which are described in the following sections.

Queries, Keys, and Values

Attention mechanism input consists of target token Query vector Q, its associated source token Key vector K, and embedding matrices Values V (Singh & Mahmood, 2021). Using inner dot product, mapping source and target tokens in machine translation may be quantified based on the sequence similarity of each token. In order to

ensure correct translation, the key must have a high dot product value with its related query. Assume $Q \in \{L_Q, D\}$ and $K \in \{L_K, D\}$, where L_Q, L_K represent target and source lengths, and D signifies the dimensionality of the word embedding. Softmax is employed to generate a probability distribution in which all Query, Key similarities add up to one, hence directing greater attention to the most closely matched keys.

$$W_{SM} = softmax(Q.K^T) \text{ where } W_{SM} \in \{L_Q, L_K\} \tag{3.3}$$

Assuming that values and keys are generally similar, a query assigns a probability of a match to the key, therefore:

$$Z_Att = Attention(Q, K, V) = softmax(Q.K^T).V = W_{SM}.V \tag{3.4}$$

Position-wise Feed-Forward Networks

Additional attention sub-layers are also included in our encoder and decoder's feed-forward networks. These feed-forward networks apply to each position individually and identically. A ReLU activation is sandwiched between two linear transformations.

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{3.5}$$

Despite using the same parameters in different positions, linear transformations employ distinct parameters for each layer. Another approach to describe this is as two convolution layers with a kernel size of 1. Input and output dimensions are $d_{model} = 512$, but the dimension of the inner layer is $d_{ff} = 2048$.

Embeddings and Softmax

Similar to other sequence transduction models, we transform input and output tokens to vectors of dimension d_{model} using learned embedding. In addition, we employ the conventional linear transformation and softmax function to turn the decoder output into projected next-token probabilities. Similar to (Press & Wolf, 2016), in our approach, the two embedding layers and the pre-softmax linear transformation

share the same weight matrix. In the embedding layers, those weights are multiplied by $\sqrt{d_{model}}$.

Positional Encoding

Since our model does not feature recursion or convolution, we must inject information about the relative or absolute location of the tokens in the sequence for the model to make use of the sequence's order. In order to accomplish this, "positional encodings" are added to the input embeddings at the bottom of the encoder and decoder stacks. The positional encodings and embeddings have the same dimension model, allowing the two to be added. There are learned and fixed many choices of positional encodings (Gehring et al., 2017).

In this work, we use sine and cosine functions of different frequencies:

$$\begin{aligned} PE_{pos,2i} &= sin(pos/10000^{2i/d_{model}}) \\ PE_{pos,2i+1} &= cos(pos/10000^{2i/d_{model}}) \end{aligned} \tag{3.6}$$

where *pos* is the position and i is the dimension. That is, each dimension of the positional encoding corresponds to a sinusoid.

3.3.2 BART - Transformer based Generative Model

Bidirectional and Autoregressive Transformer (BART) is a denoising autoencoder constructed with a sequence-to-sequence model that is suitable for a wide variety of tasks (Singh & Mahmood, 2021). Two levels of pre-training exist: where the further text is distorted by an arbitrary noising function; in a latter, a Seq2Seq model is learned to restore the original text (Lewis et al., 2019). The greatest advantage of the approach where random adjustments not restricted to length changes are applied to the original text is noise flexibility. Two noise variations that stand out are shuffling the original sentence's order randomly and using a filling scheme in which texts of any length are randomly replaced by a single masked token. BART deploys all possible document corruption schemes as shown below in figure 3-4.

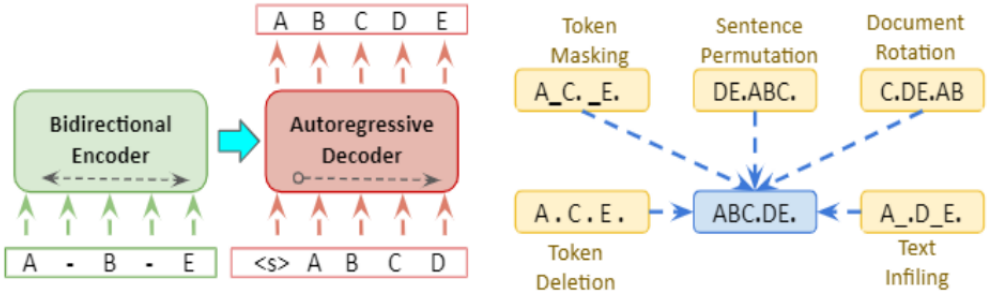

Figure 3-4: Denoised BART model and its noising schemes. (Singh & Mahmood, 2021)

BART uses the standard Seq2Seq2 Transformer architecture from Vaswani et al. (2017) which, despite its simplicity, can be seen as generalizing BERT (due to the bidirectional encoder), GPT (with the left-to-right decoder), and many other more recent pre-training schemes.

The Encoder part of BART uses BERT's bi-directional encoder to find the best way to represent the sequence it is given. The BERT encoder gives out an embedding vector for each token in each text sequence it receives, as well as another vector with information about the whole sentence. Thus, the decoder can train for both token-level and sentence-level tasks, providing a solid foundation for any future fine-tuning tasks. Figure 3-5 depicts how masked sequences are utilized for pre-training. As in the transformer sequence-to-sequence model, each layer of the BART decoder also pays attention to the last hidden layer of the encoder. However, unlike a BERT, BART doesn't use an extra feed-forward network before word prediction. To put it another way, BART has about 10% more parameters than the BERT model (Lewis et al., 2019).

A decoder must read the token- and sentence-level representations of an input text sequence in order to translate them to the output target. However, when utilizing a similarly built decoder, tasks such as predicting the next sentence or tokens may perform badly since the model requires a full input prompt. In these instances, we want model architectures that can be trained to generate the next word in a sequence by examining only the preceding words. Consequently, a causal or autoregressive

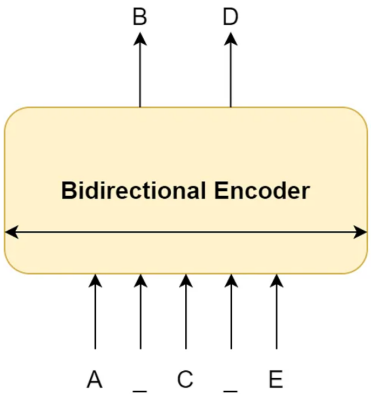

Figure 3-5: Bidirectional Encoder

model that uses just historical data to forecast the future is useful.

The GPT1 model utilized a similar design to the decoder portion of the original Transformers. GPT progressively stacks 12 such decoders such that only prior tokens can influence the current token computation(Lewis et al., 2019). The architecture is shown in Figure 3-6. Moreover, GPT-1 task-based architecture and magnified views of transformer-based decoder figure are included in the paper's Appendix B-1. Similar to the original Transformer decoder, The GPT decoder also employs a masked multi-headed self-attention block and a feed-forward layer.

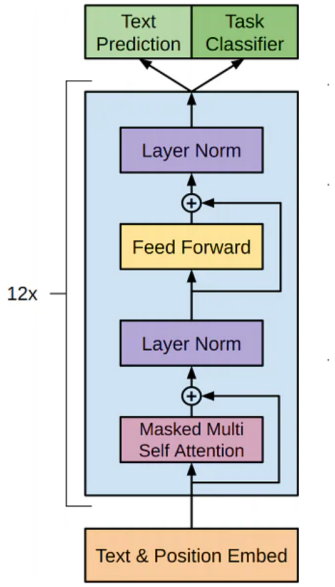

Figure 3-6: GPT decoder

Despite separating the decoder from the encoder, the input to the decoder would still be a learned representation (or embedding) of the initial text sequence. Therefore, BART connects the bi-directional encoder to the autoregressive decoder in order to produce a denoising auto-encoder architecture. Based on these two components, the complete BART model would appear, as seen in Figure 3-4.

The BART model was trained on many different datasets and tasks as SQuAD (Rajpurkar et al., 2016), MNLI (Williams et al., 2017), XSum (Narayan et al., 2018), and comes in different sizes: BART-base (140m paramteres, 6 layers), BART-large (400m parameters, 12 layers) and etc. (Hugging Face, 2022a).

3.3.3 T5 - Transformer based Generative Model

Text-to-text Transfer Transformer (T5) (Raffel et al., 2020) is an encoder-decoder model pre-trained on a multi-task mixture of unsupervised and supervised tasks and for which each task is converted into a text-to-text format whose basic architecture is Transformer, which was proposed by Vaswani et al. (2017). The core principle behind T5's encoder-decoder architecture is to receive a text as input and generate new text as output. It achieves state-of-the-art on multiple tasks, which shows the power of the large pre-training model and Seq2Seq paradigm.

In general, the T5 implementation roughly adheres to its suggested form Vaswani et al. (2017). First, an input token sequence is translated to an embedding sequence, which is subsequently supplied to the encoder. Each "block" in the encoder consists of two subcomponents: a self-attention layer and a tiny feed-forward network. Layer normalization (Ba et al., 2021) is performed to each subcomponent's input. Activations are just rescaled in a reduced variant of layer normalization. After layer normalization, a residual skip connection connects each subcomponent's input to its output (He et al., 2016). Within the feed-forward network, on the skip connection, on the attention weights, and at the input and output of the entire stack, dropout (Srivastava et al., 2014) is implemented. The structure of the decoder is similar to that of the encoder, except that it includes a standard attention mechanism after each self-attention layer that attends to the encoder's output. The decoder's self-attention

mechanism also employs an autoregressive or causal version of self-attention, which only allows the model to focus on previous outputs. The final decoder block's output is fed into a dense layer with a softmax output, the weights of which are shared with the input embedding matrix. Before being further processed, the outputs of all Transformer attention mechanisms are concatenated. T5 uses a simpler form of position embeddings in which each "embedding" is merely a scalar added to the associated logit used to compute the attention weights, while the original Transformer employed a sinusoidal position signal or learned position embeddings. To sum up, T5 is basically comparable to the original Transformer provided by Vaswani et al. (2017) with the difference of removing the Layer Norm bias, relocating the layer normalization outside the residual route, and employing a new position embedding strategy.

The T5 model was trained on unlabeled data created with a more refined version of Common Crowl (an organization that crawls the web and freely provides its archives and datasets to the public (Xia, 2014)) - Colossal Clean Crawled Corpus(C4), and it comes in different sizes: T5-small (60m parameters, 6 layers), T5-base (220 m parameters, 12 layers), T5-large (770m parameters, 24 layers) and etc. (Hugging Face, 2022a).

3.3.4 Model Evaluation Metric

We require measurements to validate transformed texts' quality and accuracy to assess our models. We utilize the Bilingual Evaluation Understudy (BLEU) to evaluate the T5 and BART Transformer models. BLEU is an algorithm to assess machine-translated text quality and is knowns as an inexpensive and automated metric (Hugging Face, 2022b). It was one of the first metrics to correlate highly with how people evaluated quality.

Individual translated parts, often sentences, are scored by comparing them to a collection of high-quality reference translations. These ratings are then averaged over the whole corpus to determine the overall quality of the translation. Neither comprehensibility nor grammatical accuracy is considered.

In BLEU score calculation we have the target sentence and the predicted sentence.

The first step is to compute Precision scores (number of corrected predicted and total predicted) for 1-grams through 4-grams and combine them using the formula below 3.7:

$$GeometricAveragePrecision(N) = (p_1)^{\frac{1}{4}} \cdot (p_2)^{\frac{1}{4}} \cdot (p_3)^{\frac{1}{4}} \cdot (p_4)^{\frac{1}{4}} \tag{3.7}$$

In order to offset that predicted sentence could come as one-word Brevity Penalty is calculated by the formula 3.8:

$$\begin{cases} 1, if c > r \\ e^{1-r/c}, if c <= r \end{cases} \tag{3.8}$$

Where c is predicted length = number of words in the predicted sentence and r is target length = number of words in the target sentence Finally, to calculate the Bleu Score, the Brevity Penalty with the Geometric Average of the Precision Scores are multiplied as shown in 3.9:

$$BLEU(N) = BrevityPenalty \cdot GeometricAveragePrecisionScores(N) \tag{3.9}$$

Chapter 4

Experiments and Results

Our study can be divided into two large parts; in the first, we use the BM25 and BERT text similarity methods for binding geodata to dirty addresses, while we get a dataset, which we use in the last to compare the transformer models T5 and BART. As hardware, we used NVIDIA DGX A100, which is a universal system for the entire artificial intelligence infrastructure, consisting of 8 GPUs of 40 GB each, as well as a CPU of 256 cores. These impressive characteristics of this equipment immensely helped us to save time in the process of completing tasks in both parts.

4.1 Experimental Setup

Working with ElasticSearch, we used Python with the Elasticsearch library. First of all, we indexed the OSM dataset (section3.1.2) to Elasticsearch as described in 2.5 section, and created a *query* for finding similar addresses of a dirty dataset from the OSM index (clean dataset). Queries for matching the full address (*full_ address* column) did not produce the appropriate results since there are many similarities in the region and district in the full addresses, and the text similarity methods allocate much attention to numbers rather than texts. In this regard, to improve the accuracy of the query, we added a *filter*, where before searching for an address, it is first compared to the region-district and house number, where if the house numbers do not match the requested address, they are skipped. Furthermore, the matching was carried out for

the values of the city and street only. For these purposes, we used the appropriate columns(*region_ district, housenumber, city_ street*), that we created during the data preparation process (section 3.1.2). Using the *iteration* function, we queried each address of the dirty dataset. In order to ensure or enhance performance while querying, the dirty dataset data was preprocessed. Moreover, we used *multiprocessing*, which enabled the DGX A 100 to utilize multiple cores of a CPU to run tasks and processes in parallel. This parallelization leaded to significant speedup in tasks that involve much computation. In our case , we have reduced the processing and execution time of the request to 1.5 minutes. Whereas without multiprocessing, it took 3 hours to process each table with 300,000 dirty dataset addresses. For "city_street" *matching* (Figure 4-1), the BM25 similarity method was used, which is built-in and used by default in Elasticsearch and presented as *baseline* for binding geodata to addresses in our study. The query process is as follows: The address from the dirty dataset

```
"query": {"bool": {
    "must": [
        {"match": {"osm_city_street": {"query": query_city_street}}}],
    "filter": [
        {"term": {"osm_region": query_region}},
        {"term": {"osm_district": query_district}}],
    "should": [
          {"match": {'osm_housenumber': {"query": query_housenumber}}}]}}
```

Figure 4-1: Matching query in Elasticsearch

is matched with each address of the OSM index, where the "city_street" similarity score (described in 3.2.2) will be formed. The OSM index address with the highest score will be the most similar for the queried dirty address. The higher the score, the more similar the addresses. Each dirty address queried is recorded in a new data frame with a similar clean address (OSM) defined with the corresponding geodata and similarity score. This created a dataset with standard (clean) and non-standard (dirty) addresses used as input data for training/validation of Transformer models.

In our *proposed* Sentence embedding method, we use the BERT (described in 3.2.2) encoder to vectorize "city_street" of clean (OSM) and dirty datasets, where then CosineSimilarity (described in 3.2), comparing the similarity of vectors, reveals

the most similar addresses by the *score* value (Figure 4-2). To sum up, the querying

```
"script_score": {
    "query": {"bool": {
        "filter": [
            {"term": {"osm_region": query_region}},
            {"term": {"osm_district": query_district}}],
        "should": [
            {"match": {"osm_housenumber": {"query": query_housenumber}}}]}},
    "script": {
        "source": "cosineSimilarity(params.query_vector, 'osm_city_street_vector') + 1.0",
        "params": {"query_vector": query_city_street_vector}}}
```

Figure 4-2: Cosine Similarity query in Elasticsearch

process of the proposed method is similar to the BM25 method that we described above, with the exception of two cases; the former is we create a new OSM index, where during the indexing process, a vector will be created for each "city_street" value using BERT encoder and written to the field *city_street_vector*. In the case of dirty addresses, its "city_street" is vectorized during the querying process. Furthermore, the latter is that Cosine similarity script score query is implemented.

In our study we use the Seq2Seq (section 3.3) model proposed by ArcGIS Developers (2022) whose architecture is based on the original Transformer (Vaswani et al., 2017). SequenceToSequence class constructor accepts two named arguments *data* and *backbone*. Where data is prepared dataset with two columns: standard (clean_address) and non-standard (dirty_address), and backbone is a pretrained transformer model based on the transformer architecture of choice: T5 and Bart (section 3.3.1). All work with the models was performed using the *arcgis* library for Python in the NVIDIA DGX A100 device with eight 40GB GPUs.

First of all, we imported 200,000 addresses (standard & non-standard) from the prepared dataset (1,500,000 addresses) where the similarity score is greater than the overall mean. After by *cyrtranslit* Russian and Kazakh letters of addresses are transformed into Latin and saved as .csv format for further use. As mentioned above, training data must have two columns, one for input text and the other for translated output text. In the above example, non-standard_address is the input text column, which is dirty_address of our dataset, and standard_address is the output text col-

umn which is the clean_address (table 4.1). In this step, data preparation includes dividing the data into training and test sets, creating the necessary data structures to load data into the model, and so on. The function *prepare_textdata* of the SequenceToSequence class allowed to directly read training samples in the above format and automate the entire process. When calling this function, we provided the following arguments: train_file as the .csv dataset; batch_size as 16; task as the sequence translation; text_columns as dirty_address; and label_columns as clean_address. Thus, we have prepared the data for the training.

non-standard_addresses (dirty_address)	standard_addresses (clean_address)
juzhno-kazahstanskaja, enbekshinskij, shymkent, mikrorajon sever, 7	shymkent, shymkent, enbekshinskij rajon, mikrorajon teriskej (sever), 7
akmolinskaja, esil'skij, esil', ulica gagarina, 11	akmolinskaja oblast', esil', esil'skij rajon, ulica gagarina, 11
almaty, turksibskij, turksibskij, ulica fuchika, 100	almaty, almaty, turksibskij rajon, ulica fuchika, 100
akmolinskaja, akkol'skij, akkol', tihaja, 20	akmolinskaja oblast', akkol', akkol'skij rajon, ulica tihaja, 22
aktjubinskaja, aktobe, aktobe, zarechnyj-3, 266	aktjubinskaja oblast', aktobe, aktobe g.a., zarechnyj-3, 674
almatinskaja, alakol'skij, usharal, sh. ajmanova, 4	almatinskaja oblast', usharal, alakol'skij rajon, ajmanova, 4
karagandinskaja, karaganda, imeni kazybek bi, pereulok ryleeva, 16	karagandinskaja oblast', karaganda, karaganda g.a., ryleeva pereulok, 16

Table 4.1: Standard and non-standard addresses for training

The next step was to define the backbone for the Seq2Seq model; whereas the baseline, we chose the BART transformer model presented by ArcGIS Developers (2022) and our proposed transformer model T5. Before training models, we found optimum learning rates accessible through the model's *lr_find()* method included in the ArcGIS library. In machine learning (ML), the learning rate (LR) (Murphy, 2012) is a tuning parameter that defines the step size at each iteration while approaching a loss function's minimum. It indicates how quickly ML models "learn." When the learning rate is low, model training will take a long time since the steps toward the minimal loss function are so few, and if it is high, training may not converge or even diverge (ArcGIS Developers, 2022). We trained the model using *fit()* method till the validation loss (or error rate) continued to go down with each training pass also known as *epoch*. This is indicative of the model learning the task. By default, the earlier layers of the model (i.e., the backbone) are frozen. Once the last layers have been sufficiently trained, the earlier layers are unfrozen (by calling *unfreeze()* method of the class) to further fine-tune the model. After unfreezing, we re-found the optimal learning rate and also continued to train. In order to get desired results,

we assigned epochs in different sizes for the T5 model: 5-3, 7-5, and for Bart: 5-3, 7-5, respectively.

4.2 Results

As a result of working with the BM25 and Sentence Embedding (BERT text similarity + Cosine Similarity) methods, we received two datasets with dirty addresses, clean addresses determined by these methods with their corresponding geodata, as well as a similarity score. BM25 gives a score from 0 to 48, and Sentence embedding (BERT + Cosine Similarity) gives a score from 1 to 2 (tables 4.2 and 4.3). We do

dirty_address	clean_address_OSM	lat_lon	score
жамбылская, тараз, тараз, салтанат м. а. , 29	Жамбылская область, Тараз, Тараз Г.А., Салтанат микрорайон, 29	71.351527375185 01,42.88895107260998	17.21896
павлодарская, павлодар, павлодарское, переулок северный, 8	Павлодарская область, Павлодарское, Павлодар Г.А., Северный переулок, 8	76.86426195000921,52.38685989912765	21.316845
алматинская, карасайский, каскелен, карасай батыра, 36	Алматинская область, Каскелен, Карасайский район, улица Карасай Батыра, 36	76.62107168298155,43.20093370074606	19.726704
южно-казахстанская, сайрамский, жібек-жолы, жібек-жолы, 15	Туркестанская область, Жибек жолы, Сайрамский район, улица Жибек Жолы, 26	69.93299310000229,42.47665725054343	13.012959
шымкент, аль-фарабийский, аль-фарабийский, улица свободы, 42а	Шымкент, Шымкент, Аль-Фарабийский район, улица Свободы, 42А	69.59270460576687,42.31279637937487	16.803616

Table 4.2: "BM25 similarity method results"

dirty_address	clean_address_OSM	lat_lon	score
акмолинская, аршалынский, аршалы, джолдаспаева, 9	Акмолинская область, Аршалы, Аршалынский район, улица Джолдоспаева, 9	72.175889966177 35,50.82065860577256	1.9884919
акмолинская, кокшетау, кокшетау, село красный яр геологов, 22	Акмолинская область, Красный Яр, Кокшетау Г.А., Геологов улица, 2	69.255283818959 31,53.31730849349184	1.9660609
костанайская, костанайский, затобольск, ленина, 10	Костанайская область, Заречное, Костанайский район, Ленина, 11	63.740391570058 66,53.23506154796411	1.9639688
кызылординская, байконыр, байконыр, глушко, 7	Кызылординская область, Байконур, Байконыр Г.А., улица Глушко, 7	63.314581933869 63,45.62650587562673	1.9775691
алматинская, сарканд ский, сарқан, ватутина, 29	Алматинская область, Сарканд, Саркандский район, Ватутина, 47	79.9288626082061,45.402840603183	1.9456124

Table 4.3: "Sentence Embedding method results"

not have an exact formula for comparing such different types of scores. Therefore, to determine which of the above methods coped with the task of similarity more correctly and efficiently, we used the manual method. That is, 100 addresses randomly selected from each dataset were compared visually. As a result of our comparison, the BM25 method coped with 82 addresses correctly, whereas the Sentence embedding with 76. Accordingly, we can conclude that within the framework of our work, the default Elasticsearch BM25 method is 8% better than Sentence embedding. Dataset with BM25 similarity method results was chosen for the use in model training.

The figure 4-3 illustrates an example of "Heat mapping" created by the identified coordinates (lat and long) while using similarity methods for dirty addresses. In this

way, it is possible to visualize the country's statistics according to fertility, death, divorce, crime, and the others on a map.

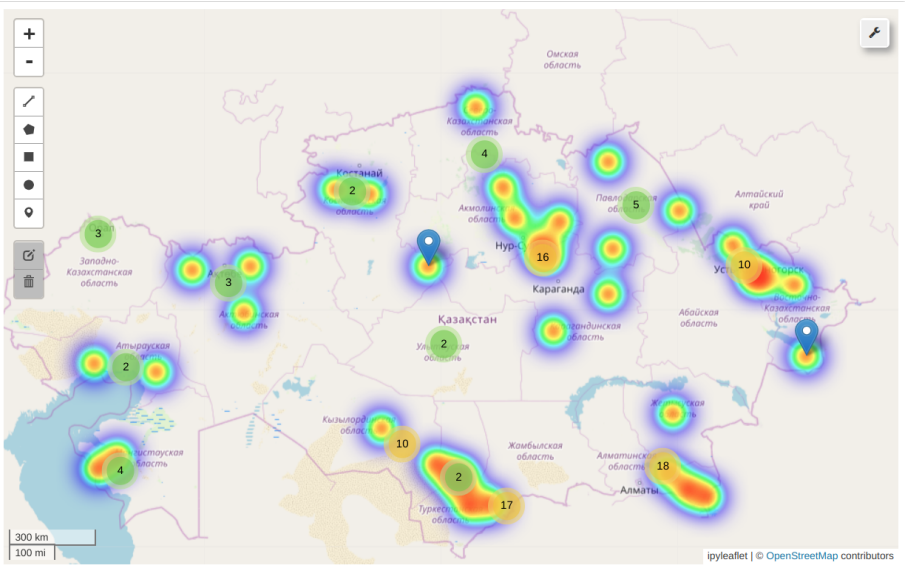

Figure 4-3: Drawing points on the map (heatmap) by coordinates (lat and lon)

Baseline model BART and our proposed T5 were trained in 2 steps due to the need for unfreezing the model, as noted in section 4.2. After unfreezing, new learning rates for T5 and BART (Figure 4-4) were determined. The learning rate in each model changes dramatically. Concerning the BART model, the low learning rate is determined after five epochs, and in the T5 model after seven epochs.

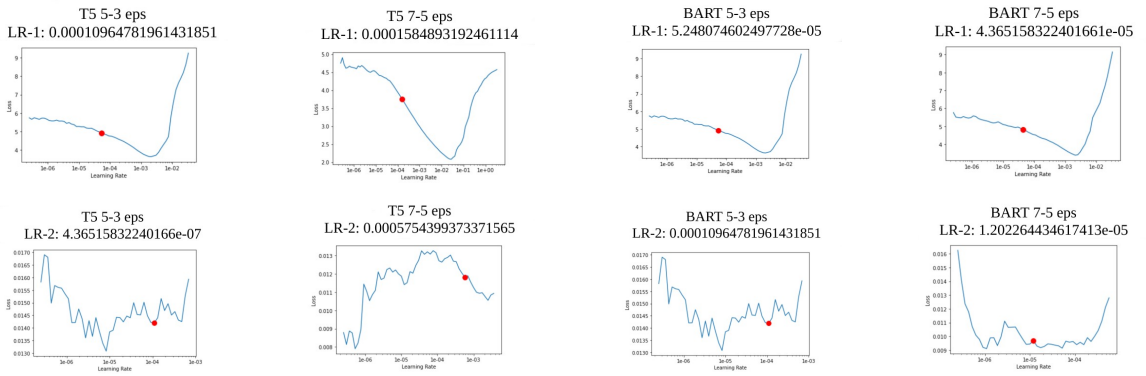

Figure 4-4: BART and T5 models learning rates

Figure 4-5 demonstrates validation and training losses for each model, where we see the lowest validation loss in the T5 model with 7-5 parameters. The T5 and Bart

models with parameters in 5-3 epochs have a not low but decreasing validation loss, in contrast to the Bart model with a parameter in 7-5 epochs, which almost has a stable loss. As noted in section 4.1, validation loss must steadily decline for proper training.

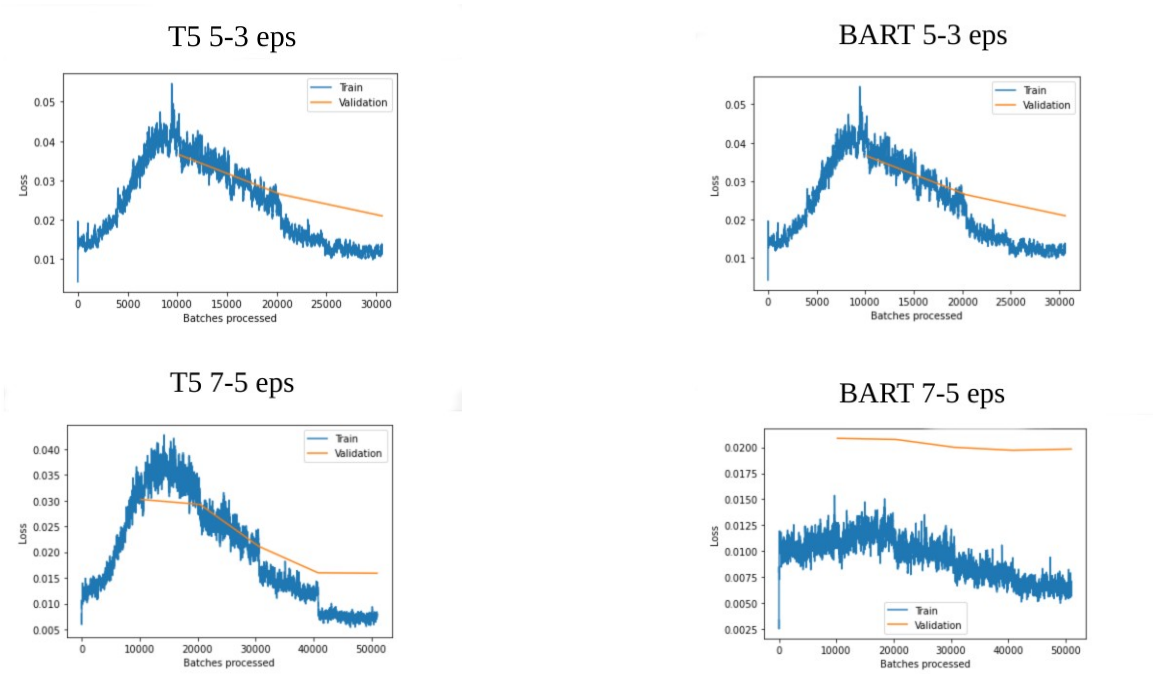

Figure 4-5: Validation losses for T5 and BART 5-3 and 7-5 epochs

It is important to note that when we assigned parameters for the T5 model as 10-5 epochs and 20-10 epochs (Figure 4-6), we found that validation loss did not change in any way in the first and even began to increase in the latter. Thus, we decided that for training transformer models with the sequence translation task, the size of the epochs should not be significant, that is, up to 7 epochs.

Table 4.4 illustrates the results of the training of transformer models. As we can see in BART, there is a slight difference between 5-3 and 7-5 parameters. BART with parameters in 7-5 has good results even though it had an almost stationary validation loss. In contrast, in T5, we can see a big difference in validation loss, where parameters 7-5 is 0.015 and 5-3 is 0.024. Also, our proposed model T5 with parameters 7-5 reaches a maximum BLEU value of 0.9925 compared to BART, where BLEU is 0.9910. Moreover, the T5 and BART models trained on our data were evaluated

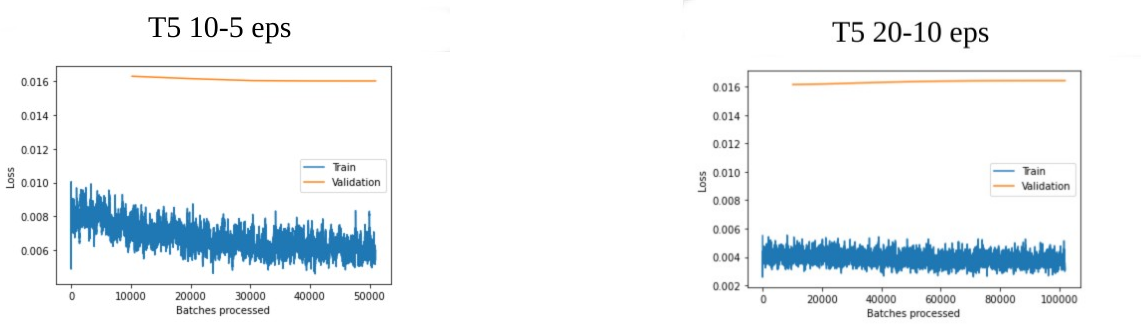

Figure 4-6: Validation losses for T5 10-5 and 20-10 epochs

Model	Training pass	Training loss	Validation loss	Seq2Seq Accuracy	BLEU
T5-base	5epochs-3epochs	0.021941	0.024103	0.994782	0.987827
	7epochs-5epochs	**0.007877**	**0.015900**	**0.996827**	**0.992522**
BART-base	5epochs-3epochs	0.012364	0.020994	0.995775	0.989808
	7epochs-5epochs	0.007027	0.019840	0.996297	0.991038

Table 4.4: T5 and BART models results

on randomly selected 100 non-standard addresses (not in the training dataset) to predict its standard version. The results are demonstrated in Table 4.5 and A.1, where, according to the BLEU evaluation metric, T5 with parameters of 7-5 epochs turned out to be the best.

Model	seq2seq_ccuracy	BLEU
BART 5-3 epochs	0.9958	0.9898
BART 7-5 epochs	0.9963	0.9910
T5 5-3 epochs	0.9948	0.9878
T5 7-5 epochs	**0.9968**	**0.9925**

Table 4.5: T5 and BART evaluation results

BLEU is an important evaluation metric (model's *get_model_metrics()* method) in our study for comparing models, as noted in section 3.3.4; therefore, we can conclude that the proposed T-5 copes with translation tasks better than baseline BART.

Chapter 5

Conclusion

Within the thesis, we have aimed to solve the task of converting arbitrary written primarily by humans addresses into a standard format and adequate geocoding using state-of-the-art NLP algorithms and ML models. To reach the goal, we have collected the so-called dirty addresses mainly submitted by the citizens of the Republic of Kazakhstan when receiving various services. Overall, our dataset contains 1,5 million records from 5 publicly available sources.

The first part of the research is related to normalization and standardization procedures. Before the standardization, we had pre-processed the raw data by applying data cleansing and enrichment techniques; then, we processed the records using the BM25 ranking function and the Sentecn Embedding algorithm consisting of BART text similarity and cosine similarity algorithms. Finally, we admitted the OSM index consisting of approximately 700 thousand records as a golden standard reference data as it was open and available and contained geocodes and textual address representations. It should be noted that the very first approach was to use a national address registry as the golden standard. However, the original idea was not feasible due to the registry lacking geocodes.

As a result, each normalized textual record containing address information was bound with adequate geocodes. However, one of our discoveries at this step is the overall poor performance of address matching when comparing the complete addresses—unlikely, filtering out the addresses by district and house number before

matching displayed better results. We found this happened as there were significant similarities in the region and district names in the complete addresses—also, the text similarity methods allocated much attention to numbers rather than texts. The practical outcomes of the address standardization were assessed manually by comparing the results on randomly selected 100 records. We conclude that the BM25 method is 8% better than the Sentence embedding combined method.

The second part of the research used the dataset with dirty, and 'clean' records obtained earlier at the address standardization step. In the research, we admitted the BART model as a baseline and ran the proposed T5 model with various settings on epoch numbers. According to BLEU metric, an algorithm for evaluating the quality of machine - translated text, T5 (0.992522) outperformed BART (0.991038) at the setting of 7-5 epochs.

The thesis results demonstrate that using the performance of the T5 model to predict addresses given arbitrary written non-standard incomplete addresses is a reasonable approach for implementing the solutions to solve live practical cases.

Appendix A

Tables

dirty "non-standard" addresses	predicted "standard" addresses
карагандинская, абайский, карабас, улица қазақстан, 7	карагандинская область, абай, абайский район, улица казахстан, 7
мангистауская, актау, актау, 5, 3	мангистауская область, актау, актау г.а., 5 микрорайон, 3
северо-казахстанская, кызылжарский, бескөл, советская, 103	северо-казахстанская область, бесколь, кызылжарский район, советская улица, 103
алматинская, жамбылский, каргалы, алатау, 77	алматинская область, каргалы, жамбылский район, алатау улица, 77
нур-султан, алматы, микрорайон юго восток, пер. аккол, 7	нур-султан, нур-султан, район алматы, переулок аккол, 7
актюбинская, актобе, алматы, вокзальная, 64	актюбинская область, актобе, актобе г.а., вокзальная улица, 64
актюбинская, актобе, нур-султан, абулхаир хана, 27	актюбинская область, актобе, актобе г.а., проспект абилкайыр хана, 27/1
акмолинская, бурабайский, щучинск, улица темирязева, 7	акмолинская область, щучинск, бурабайский район, улица темирязева, 7
костанайская, костанай, костанай, киевская, 18	костанайская область, костанай, костанай г.а., киевская улица, 5
алматы, бостандыкский, алматы, гагарина, 181	алматы, алматы, бостандыкский район, проспект гагарина, 181
акмолинская, ерейментауский, ерейментау, улица богенбая, 67	акмолинская область, ерейментау, ерейментауский район, богенбая улица, 67
актюбинская, актобе, ақтөбе, московская, 154	актюбинская область, актобе, актобе г.а., московская улица, 154
туркестанская область , шардаринский район, шардара, улица аймахан таутанов, 5	туркестанская область, шардара, шардаринский район, таутанов, 5
актюбинская, шалкарский, шалкар, пушкина, 75	актюбинская область, шалкар, шалкарский район, улица пушкина, 75
южно-казахстанская, сайрамский, аксукент, балуан шолак, 70	туркестанская область, аксукент, сайрамский район, улица калдаякова, 70
павлодарская, лебяжинский, аққу, пушкина, 46	павлодарская область, шарбакты, щербактинский район, пушкина улица, 46
кызылординская, кызылорда, кызылорда, акмечеть, 26	кызылординская область, кызылорда, кызылорда г.а., акмечеть, 26
акмолинская, степногорск, аксу, улица габита мусрепова, 29	акмолинская область, степногорск, степногорск г.а., улица габита мусрепова, 29
мангистауская, бейнеуский, бейнеу, улица и. сужеубаев, 3	мангистауская область, бейнеу, бейнеуский район, ивана сужеубаев, 3
костанайская, костанайский, заречное, северный, 426	костанайская область, заречное, костанайский район, северный переулок, 426
южно-казахстанская, сайрамский, карасу, школьная, 25	туркестанская область, карасу, сайрамский район, улица школьная, 23
костанайская, сарыкольский, тагильский, тагильский, улица ленина, 11	костанайская область, сарыколь, сарыкольский район, улица ленина, 10
алматинская, карасайский, райымбекский, абай, улица д қонаев, 115	алматинская область, абай, карасайский район, улица бейбитшилик, 115
атырауская, атырау, атырау, пр. балхаш, 26	атырауская область, атырау, атырау г.а., балхаш, 26
туркестанская область , жетысайский район, жетысай, улица казибек би, 17	туркестанская область, сарягаш, сарягашский район, казибек би улица, 17
кызылординская, шиелийский, шиели, улица досбол датка, 13	кызылординская область, аральск, аральский район, досбол датка, 13
восточно-казахстанская, курчумский, курчум, малдыбаева, 73	восточно-казахстанская область, курчум, курчумский район, улица малдыбаева, 69
кызылординская, казалинский, арандинский, кожабакы, улица бейбитшилик, 27	кызылординская область, казалинск, казалинский район, улица бейбитшилик, 27
карагандинская, караганда, октябрьский, улица лермонтова, 188	карагандинская область, караганда, караганда г.а., лермонтова, 188

Table A.1: Predicted addresses by proposed T5 Transformer model (7-5eps)

Appendix B

Figures

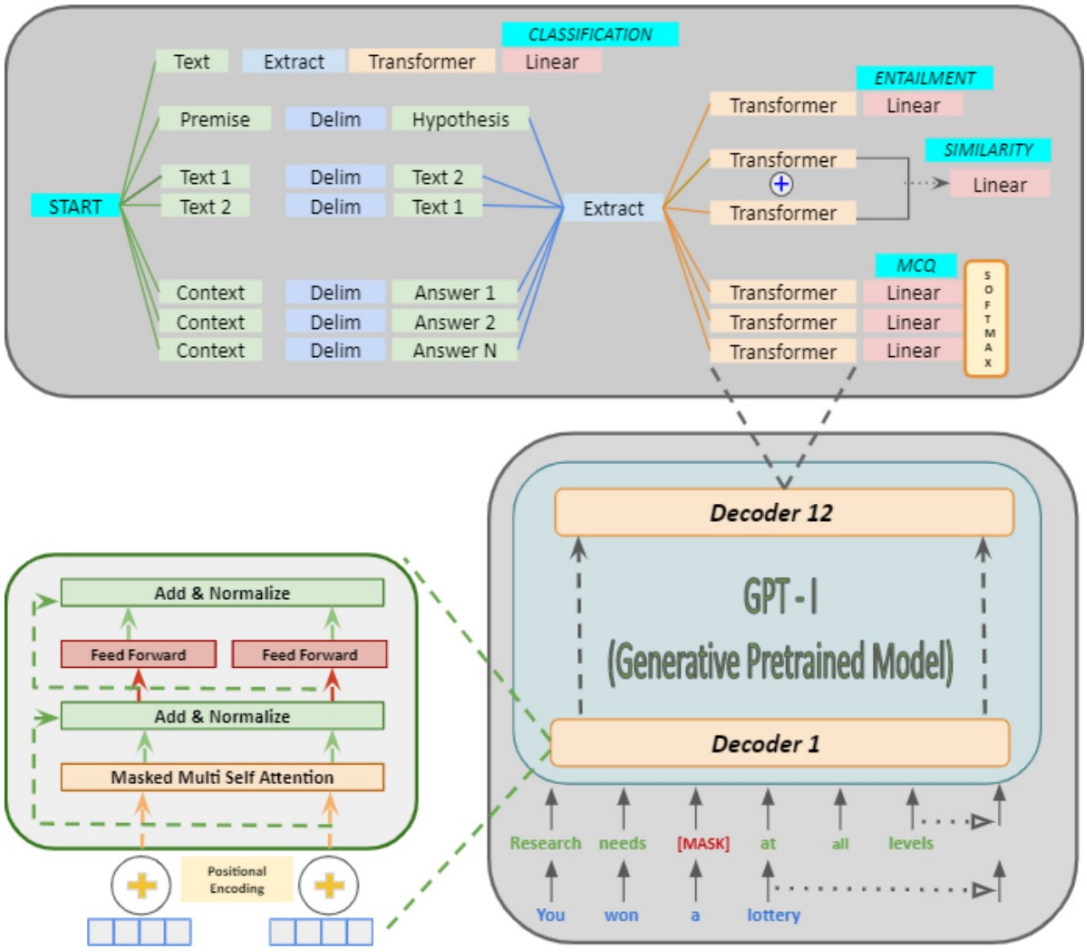

Figure B-1: GPT-1 task-based architecture (top) and magnified views of transformer based decoder (bottom). (Singh & Mahmood, 2021)

References

Abbasi, R. (2005). Information extraction techniques for postal address standardization. *2005 Pakistan Section Multitopic Conference, 9th International Multitopic Conference, IEEE INMIC 2005*, 1-6. doi: 10.1109/INMIC.2005.334455

Alqahtani, A., Alsubait, T., Alhakami, H., & Baz, A. (2021). A survey of text matching techniques. *Engineering, Technology and Applied Science Research*, *11*(1), 6656-6661. doi: 10.48084/etasr.3968

ArcGIS Developers. (2022). *How sequencetosequence works.* Retrieved from `https://developers.arcgis.com/python/guide/how-sequencetosequence-works/#References`

Ba, J. L., Kiros, J. R., & Hinton, G. E. (2021). Layer normalization. *arXiv preprint*. Retrieved from `https://arxiv.org/pdf/1607.06450.pdf`

Bartlett, R. (2019). Local geographic information storing and querying using elasticsearch. *ACM International Conference Proceeding Series*. doi: 10.1145/3371140.3371144

bizgid.kz (Postal Codes of Kazakhstan). (2022). *Postal codes of kazakhstan in a new format.* Retrieved from `https://bizgid.kz/postal/`

Bordes, A., Chopra, S., & Weston, J. (2014). Question answering with subgraph embeddings. *EMNLP*. Retrieved from `http://arxiv.org/abs/1406.3676`

Cetl, V., Kliment, T., & Jogun, T. (2018). A comparison of address geocoding techniques–case study of the city of zagreb, croatia. *Survey Review*, *50*(359), 97-106. doi: 10.1080/00396265.2016.1252517

Christen, P., & Belacic, D. (2015). Automated probabilistic address standardisation and verification. *Proceedings of the 4th Australasian Data Mining Conference*. Retrieved from `http://hdl.handle.net/1885/83649`

Coetzee, S., & Judith, B. (2009). Address databases for national sdi: Comparing the novel data grid approach to data harvesting and federated databases. *International Journal of Geographical Information Science*, *23*(9), 1179-1209. doi: 10.1080/13658810802084806

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint*. Retrieved from `https://arxiv.org/pdf/1810.04805.pdf&usg=ALkJrhhzxlCL6yTht2BRmH9atgvKFxHsxQ`

Divya, M. S., & Goyal, S. K. (2013). Elasticsearch: An advanced and quick search technique to handle voluminous data. *Compusoft*, *2*(6), 171-178.

elastic.co. (2022). *Similarity module.* Retrieved from `https://www.elastic.co/`

guide/en/elasticsearch/reference/current/index-modules-similarity.html

Elwood, S. (2006). Critical issues in participatory gis: Deconstructions, reconstructions, and new research directions. *Transactions in GIS*, *10*(5), 693-708. Retrieved from https://doi.org/10.1111/j.1467-9671.2006.01023.x

Gehring, J., Auli, M., Grangier, D., Yarats, D., & Dauphin, Y. (2017). Convolutional sequence to sequence learning. *In International conference on machine learning*, 1243-1252. Retrieved from http://proceedings.mlr.press/v70/gehring17a/gehring17a.pdf

Geographic information system (gis). (2021). *Journal of Remote Sensing and GIS*. Retrieved from https://www.walshmedicalmedia.com/scholarly/geographic-information-system-gis-journals-articles-ppts-list-4051.html

Goldberg, D. (2008). A geocoding best practices guide. *North American Association of Cancer Registries*. Retrieved from http://www.naaccr.org/filesystem/pdf/Geocoding_Best_Practices.pdf

Goldberg, D., Wilson, J., & Knoblock, C. (2007). From text to geographic coordinates: The current state of geocoding. *URISA Journal*, *19*(1), 33-46.

Goldberg, D., Wilson, J., & Swift, J. (2014). Address standardization. *Technical report 12. Los Angeles, CA: GIS Research Laboratory, University of Southern California*. Retrieved from https://spatial.usc.edu/wp-content/uploads/2014/03/gislabtr12.pdf

Gomaa, W. H., & Fahmy, A. A. (2013). A survey of text similarity approaches. *International journal of Computer Applications*, *68*(13), 13-18. Retrieved from https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.403.5446&rep=rep1&type=pdf

gov.kz (Unified Platform of Internet Resources of State Bodies). (2022). *Issuance of abstract on allocation of real property address on the republic of kazakhstan territory.* Retrieved from https://www.gov.kz/services/3690?lang=en

Han, J., Kamber, M., & Pei, J. (2012). Getting to know your data. *The Morgan Kaufmann Series in Data Management Systems. Data Mining*, 39-82. doi: 10.1016/B978-0-12-381479-1.00002-2

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, 770-778. Retrieved from https://openaccess.thecvf.com/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf

Higuera, C., & Oncina, J. (2013, july). Computing the most probable string with a probabilistic finite state machine. *FSMNLP*, 1-8. Retrieved from https://aclanthology.org/W13-18.pdf#page=13

Hugging Face. (2022a). *Datasets.* Retrieved from https://huggingface.co/datasets

Hugging Face. (2022b). *Metric: Blue.* Retrieved from https://huggingface.co/spaces/evaluate-metric/bleu

Hugging Face. (2022c). *Transformers.* Retrieved from https://huggingface.co/

docs/transformers/index

Jing, Y., Wei, J., & Rumin, C. (2021). Research on offline reverse geocoding algorithm based on k-d tree. *International Conference on Computer Information Science and Artificial Intelligence (CISAI), Computer Information Science and Artificial Intelligence (CISAI), 2021 International Conference on, CISAI*, 548-552. doi: 10.1109/CISAI54367.2021.00111

Kanani, B. (2019). Cosine similarity – text similarity metric. *Natural Language Processing*. Retrieved from https://studymachinelearning.com/cosine-similarity-text-similarity-metric/

Kang, M., Du, Q., & Wang, M. (2015). A new method of chinese address extraction based on address tree model. *Cehui Xuebao/Acta Geodaetica et Cartographica Sinica*, *44*(1), 99-107. doi: 10.11947/j.AGCS.2015.20130205

Kim, Y. (2014). Convolutional neural networns for sentence classification. *In Proceedings of EMNLP*, 1746-1751.

Kuc, A., & Rogozinski, M. (2016). Elasticsearch server. *Packt Publishing Ltd*.

Kuipers, P. (2013). Empowerment in community-based rehabilitation and disability-inclusive development. *Disability, CBR and Inclusive Development*, *24*(4), 24-42. doi: 10.5463/DCID.v24i4.274

Kumar, A., Bandyopadhyay, A., Bhoomika, H., Singhania, I., & Shah, K. (2018). Analysis of network traffic and security through log aggregation. *International Journal of Computer Science and Information Security (IJCSIS)*, *16*(6).

Lamsiyah, S., Mandaouy, A. E., Alaoui, S. O., & Espinasse, B. (2019). A supervised method for extractive single document summarization based on sentence embeddings and neural networks. *In International Conference on Advanced Intelligent Systems for Sustainable Development*, 75-88. Retrieved from https://hal.archives-ouvertes.fr/hal-02433565/file/AI2SD_2019_Lamsiyah_Salima-final.pdf

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., ... Zettlemoyer, L. (2019). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint*. Retrieved from http://arxiv.org/abs/1910.13461

Liu, H., Hu, Z., Mian, A., Tian, H., & Zhu, X. (2014). A new user similarity model to improve the accuracy of collaborative filtering. *Knowledge-Based Systems*, *56*, 156-166. doi: 10.1016/j.knosys.2013.11.006

Lu, Y., Liu, H., & Zhou, Y. (2019). Chinese address standardization based on seq2seq model. *ACM International Conference Proceeding Series*, 1-5. doi: 10.1145/3372422.3372441

Matci, D. K., & Avdan, U. (2018). Address standardization using the natural language process for improving geocoding results. *Computers, Environment and Urban Systems*, *70*, 1-8. doi: 10.1016/j.compenvurbsys.2018.01.009

Meticulous Research. (2021, September). *Geospatial analytics market.* Retrieved from https://www.meticulousresearch.com/product/geospatial-analytics-market-5161#description

Mohammadi, H., & Khasteh, S. H. (2020). A fast text similarity measure for large document collections using multireference cosine and genetic algorithm. *Turkish*

Journal of Electrical Engineering and Computer Sciences, *28*(2), 999-1013. doi: 10.3906/elk-1906-30

Murphy, K. P. (2012). Machine learning: a probabilistic perspective. *MIT press, book*.

Narayan, S., Cohen, S., & Lapata, M. (2018). Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *arXiv preprint*. Retrieved from https://arxiv.org/pdf/1808.08745.pdf

nitec.kz (National Information Technologies). (2022). *Residential address registry information system.* Retrieved from https://www.nitec.kz/en/proekty/residential-address-registry-information-system?q=/ru/proekty/informacionnaya-sistema-adresnyy-registr

OpenStreetMap. (2018). Open database license. *online*. Retrieved from https://www.openstreetmap.org/copyright/en

Osm map features. (2018). *online*. Retrieved from https://wiki.openstreetmap.org/wiki/Map_Features

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint*. Retrieved from https://arxiv.org/pdf/1802.05365.pdf

Phonebook of Almaty. (n.d.). *Telephone directory of kazakhstan.* Retrieved from http://adresok.net/Almaty

Phonebook of Nur-Sultan. (n.d.). *Telephone directory of kazakhstan.* Retrieved from http://adresok.net/Astana

Press, O., & Wolf, L. (2016). Using the output embedding to improve language models. *arXiv preprint*. Retrieved from https://arxiv.org/pdf/1608.05859.pdf

Qamar, A. M. (2010). Generalized cosine and similarity metrics: A supervised learning approach based on nearest neighbors (phd-thesis). *Computer Science. Université de Grenoble*. Retrieved from https://tel.archives-ouvertes.fr/tel-00591988

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *JOURNAL OF MACHINE LEARNING RESEARCH*, *21*, 67. Retrieved from https://arxiv.org/pdf/1910.10683.pdf

Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint*. Retrieved from https://arxiv.org/pdf/1606.05250.pdf

Shapovalova, N. (2022). *Personal data law: first results.* Retrieved from https://online.zakon.kz/Document/?doc_id=32677053&pos=6;-109#pos=6;-109

Singh, S., & Mahmood, A. (2021). The nlp cookbook: Modern recipes for transformer based deep learning architectures. *IEEE Access*, *9*, 68675-68702. doi: 10.1109/ACCESS.2021.3077350

Sonawane, P., Pingale, K., & Gawali, M. (2018). Proactive montoring of server logs to prevent instant shutdown of the server using elasticsearch. *In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929-1658. Retrieved from https://www.jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf?utm_content=buffer79b43&utm_medium=social&utm_source=twitter.com&utm_campaign=buffer,

Tian, Q., Ren, F., Li, R., Du, Q., Hu, T., & Liu, J. (2016). Using an optimized chinese address matching method to develop a geocoding service: A case study of shenzhen, china. *ISPRS International Journal of Geo-Information*, *5*(5). doi: 10.3390/ijgi5050065

Vaswani, A., N.Shazeer, N.Parmar, Uszkoreit, J., Jones, L., Gomez, A., ... Polosukhin, I. (2017). Attention is all we need. *arxiv print*. Retrieved from https://arxiv.org/pdf/1706.03762.pdf

Vusak, E., Kuzina, V., & Jovic, A. (2021). A survey of word embedding algorithms for textual data information extraction. *44th International Convention on Information, Communication and Electronic Technology (MIPRO)*, 181-186. doi: 10.23919/MIPRO52101.2021.9597076

Wang, T., Shi, H., Liu, W., & Yan, X. (2022). A joint framenet and element focusing sentence-bert method of sentence similarity computation. *Expert Systems with Applications*, *200*. doi: 10.1016/j.eswa.2022.117084

Wang, Y., Guo, Q., Liu, J., & Luo, A. (2016). The standardization method of address information for pois from internet based on positional relation. *Cehui Xuebao/Acta Geodaetica et Cartographica Sinica*, *45*(5), 623-630. doi: 10.11947/j.AGCS.2016.20150618

Weiss, D., Alberti, C., Colins, M., & Petrov, S. (2015). Structured training for neural network transition-based parsing. *arXiv:1506.06158.*. Retrieved from https://arxiv.org/abs/1506.06158

Williams, A., Nangia, N., & Bowman, S. R. (2017). A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint*. Retrieved from https://arxiv.org/pdf/1704.05426.pdf

Xia, R. (2014). Tech entrepreneur gil elbaz made it big in l.a. *Los Angeles Times*. Retrieved from https://www.latimes.com/business/la-xpm-2012-feb-05-la-fi-himi-elbaz-20120205-story.html

Xing, W., Yuan, X., Li, L., Hu, L., & Peng, J. (2018). Phenotype extraction based on word embedding to sentence embedding cascaded approach. *IEEE transactions on nanobioscience*, *17*(3), 172-180. doi: 10.1109/TNB.2018.2838137

Ying, S., Li, W., He, B., Wang, W., & Zhao, C. (2017). Address text matching method based on city address tree. *Geomatics World, In Chinese*(6).

Yu, L. I. U., & Zhang, J.-H. (2019). Address standardization algorithm based on aho-corasick automaton and address probability model. *Computer and Modernization*(12), 45. doi: 10.3969/j.issn.1006-2475.2018.12.009.

Zamfir, V., Carabas, M., Carabas, C., & Tapus, N. (2019). Systems monitoring and big data analysis using the elasticsearch system. *22nd International Conference on Control Systems and Computer Science (CSCS)*, 188-193. doi: 10.1109/CSCS.2019.00039

Zou, W. Y., Socher, R., Cer, D., & Manning, C. D. (2013). Bilingual word embeddings for phrase-based machine translation. *In Proceedings of the 2013 conference on empirical methods in natural language processing*, 1393-1398. Retrieved from `https://aclanthology.org/D13-1141.pdf`