Deep Time Series Model on Translation and Forecasting

by

Jingjing Li

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
December 11, 2021

Keywords: Deep learning, Natural Language Processing, Database, Time Series Model

Abstract

Deep learning techniques have acquired much attention and have been shown to outperform previous state-of-the-art methods in plenty of fields over the last years. This dissertation delves further into the deep time series model and its application to a variety of datasets, including natural language sequence datasets and real estate-related datasets, with a deeper insight using a comprehensive set of analytical methods and algorithms.

The first part is a study of the deep learning techniques for processing natural language data. A spatial translation interface is proposed that focuses on the spatial domain vocabularies and translates the natural language questions to structured queries executable by database management systems (DBMS). Inspired by the deep comprehension model, we propose a natural language interface(NLI) with the spatial comprehension model that is able to recognize the meaning of spatial entities based on the semantics of the context. Our system could support a flexible back-end of multiple database query languages, such as SQL and Prolog, which are all supported based on our effective strategy. A transfer learning strategy is also presented to deal with the challenge of translating spatial language into database queries. A basic model is trained on one sort of database query before being fine-tuned to work with another. The models are verified using the Geoquery dataset, and the performance is demonstrated to outperform conventional approaches.

The second portion covers the application of time series models to real estate forecasting. To complete the real-estate price prediction task, a large-scale real estate-related dataset is constructed, encompassing both static and dynamic features, combining the numerical real estate price history data from Zillow and the survey data from the Census Bureau public dataset. A carefully designed Transformer-based forecasting model which could capture the change of real estate prices and predict hotspot areas for investment in real estate is proposed based on this time series dataset. The model is designed to embed the data with sequential temporal features and combine them with non-temporal features for subsequent prediction tasks. The results of the experiment reveal that our suggested model has a high level of accuracy and surpasses all baseline models.

Acknowledgments

Throughout the completion of this dissertation, and more broadly, my entire Ph.D path, I have received enormous counsel and support from many people in a variety of ways. It is a great pleasure to thank everyone who helped me complete my degree. The words I have used here cannot express how grateful I am.

First and foremost, I would like to give my sincere gratitude to my advisor, Dr. Wei-Shinn Ku, for his expert effort and support in the past five years. He always gives me plenty of room to develop as an independent researcher. I could not earn my Ph.D degree without his generous guidance and continuous encouragement. Also, I would like to thank all my committee members: Dr. Cheryl Seals, Dr. Anh Nguyen, Dr. Bo Liu and Dr. Thaddeus Roppel, for reviewing my dissertation and providing useful feedback.

Besides, I must give my most profound appreciation to my labmate, Dr. Wenlu Wang, for her insightful suggestion and instruction on my research. She gave me great assistance throughout my entire dissertation work. My sincere gratitude also goes to my other fellow labmates, Chen Jiang, Bo Hui, Ai-Te Kuo, Zhitao Gong and Ting Shen, for their helpful discussion and collaboration.

In addition, I would like to acknowledge my two internship managers, Christian West in Apple and Shuting Wang in Facebook. These two incredible internships taught me how to apply what I learned in school in a real-world business environment, considerably enhancing my software programming skills and widening my grasp of research technique.

My best friend, Zijie Zhang, deserves praise. I consider myself quite fortunate to have a friend with such advanced programming abilities and in-depth comprehension of the research. The discussion with him enriches my knowledge and broadens my perspective of the academic issues. I am also grateful to my best friend, Muzi Li. I will never forget the moment when we laughed together. For all my other friends, Chang Ren, Dongji Feng, Yuanzhi Yao, Yanbo Gong, Boning Liang, Yuqiao Zhang, Ziqi Zhou, Tianhang Lan, Junwei Ma, Huanyi Zhou, Jueting Liu, Jinyan Cui, Zhitao Yu, Chao Yang, Jing Wu, Ya-Chi Kuo, Jiayi Xu, Chi Xu, Zhaohui Jin, Minghong Jian, Lin Lu, Yu Wang, Wei Huang, Chengfei Wang, Kenan Xiao, Yankun He, Dehua Li, Xing Wang, etc. The time, memories, and friendship that we shared will be treasured for the rest of my life.

Last but not least, I want to give special thanks to my parents and family for their continuous support in assisting me in obtaining my degree. They never fail to inspire and empower me whenever I am perplexed or frustrated. They have always been my source of strength in overcoming problems and challenges throughout my life.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

In mathematics, a time series is a sequence of data points listed or indexed in time order. Examples of time series include weather records, economic indicators and stock prices forecasting. Traditionally, time series forecasting has been dominated by the statistical methods to extract meaningful characteristics of the data and predict future values based on previously observed values. Deep learning techniques have acquired much attention and have outperformed previous state-of-the-art methods in plenty of fields over the last years. It shows deep learning methods could offer a lot of promise for time series forecasting and are able to automatically learn complex mappings from inputs to outputs and support multiple inputs and outputs. Neural networks like Multilayer Perceptrons(MLP) [1, 2], Recurrent Neural Network (RNN), Long Short-Term Memory Network (LSTM) [7, 8], Gated Recurrent Unit(GRU) [9, 45] and Transformer model [25] are all proposed to deal with the time-series data, including the natural language sequences and real estate datasets.

In this thesis, my work will focus on the deep time series model for two tasks: (1) the translation of the natural language sequences, and (2) the prediction of the real estate data. Specifically, as the Figure 1.1 shows, in the first part of this thesis, my work will focus on the deep time series model for processing natural language sequence data. In the second part of this thesis, time series models are designed for the prediction of the real estate-related data.

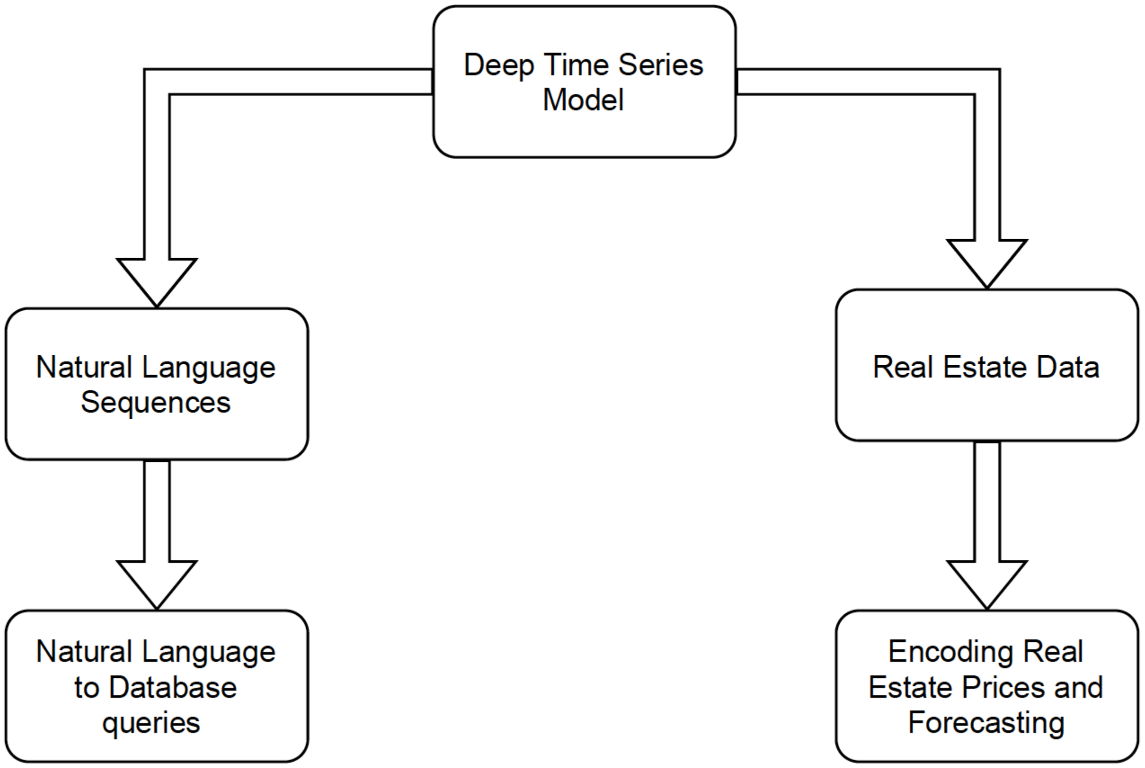

Deep Time Series Model
Natural Language Sequences
Real Estate Data
Natural Language to Database queries
Encoding Real Estate Prices and Forecasting

Figure 1.1: Thesis Overview.

1.2 Deep Natural Language Translation

The first part is a study of a spatial translation interface that focus on the spatial domain vocabularies and translates the natural language questions to a structured queries that is executable by database management systems (DBMS). A natural language interface(NLI) that is trained in the general domain is hard to apply in the spatial domain due to the idiosyncrasy and expressiveness of the spatial questions. Inspired by the deep comprehension model, we propose a spatial comprehension model that is able to recognize the meaning of spatial entities based on the semantics of the context. The spatial semantics learned from the spatial comprehension model is then injected to the natural language question to ease the burden of capturing the spatial-specific semantics.

In the second part of this section, we propose to not only address the spatial domain generalization challenge, but also support spatial flexible back-end. Multiple database query languages, such as SQL and Prolog, are all supported based on our effective strategy. We propose to add a prefix symbol to support the flexible back-end. Comparisons against other State-of-the-art approaches are demonstrated in the experiments. With our spatial comprehension model and information injection, our NLI for the spatial domain is able to capture the semantic structure of the question and translate it to the corresponding syntax of an executable query accurately.

In the third part of this section, A transfer learning approach is proposed to address the problem of translation from spatial language to database queries. We first train a base model from one natural language to database query dataset and then fine-tune the model applying to another type of database queries. The models are validated with the Geoquery dataset and shown to achieve a superior transferability performance compared with traditional methods.

1.3 Deep Real Estate Forecasting

There are several ways in which deep learning has benefited modern society. A good example is how precise economic forecasting may assist individuals in more effectively managing and distributing their resources. This year's COVID-19 epidemic has led to an increase in U.S. housing prices of 13.3 percent, compared to this time in 2019. Real estate market forecasting is vital for home buyers and investors to make well-informed decisions about their purchases and investments. Accurate property price assessments are sometimes even more important than ever in preventing financial blunders. This section offers a large-scale real estate-related dataset that may be used to aid in the prediction of property values. It is constructed with numerical historical data on real estate prices obtained from Zillow[1] and survey data from the Census Bureau's public dataset. Our objective is to simulate the dynamics of real estate using both temporal and non-temporal data from a range of sources in order to better understand the market. For the purpose of subsequent prediction, we suggest the use of a Transformer to integrate sequential temporal data and concatenate them with

[1]https://www.zillow.com/

non-temporal information. We evaluate this approach using a variety of number of classes $L \in \{2, 3, 4, 5\}$. With our proposed model, we were able to achieve 93.5 percent prediction accuracy when $L = 2$, 90.1 percent prediction accuracy when $L = 3$. The recommended model appears to outperform all of the other models that were assessed in the study.

Chapter 2

Related Work

2.1 Deep Time Series Model

The Sequence-to-sequence (Seq2Seq) [9, 45, 44] method is one of the training models to natual language processing, solving complex Language problems like machine translation, question answering, text summarization, etc... It does so by use of a RNN (recurrent neural network) or more often LSTM (Long short-term memory) [7, 8] or GRU (gated recurrent unit) [9, 45].

The RNN is an artificial neural networks where connections between nodes form a directed graph along a temporal sequence. RNNs can use the internal memory to process variable length sequences of inputs. LSTM A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate.

The most common architecture used to build Seq2Seq models is Encoder-Decoder architecture. The encoder reads the input sequence and summarizes the information into a context vector. This vector aims to encapsulate the information for all input elements in order to help the decoder make accurate predictions. The decoder is used to generate the output sequence, using the vector from encoder as the input. One of the drawbacks of the Seq2seq model is that the output sequence relies heavily on the context defined by the hidden state in the final output of the encoder, making it challenging for the model to deal with long sentences. In the case of long sequences, there is a high probability that the initial context has been lost by the end of the sequence. One of the most important optimization for the Seq2seq model is the attention mechanism [25], which help to solve this problem by allowing the decoder to look at the input sequence selectively.

The Transformer [25] model is designed to process the input sequence data for tasks such as translation and text summarization. Unlike RNN(recurrent neural network) that handle the input tokens in order, the Transformer adopts the mechanism of attention, that identifies the context that confers meaning to each word in the sentence. The Transformer uses the encoder-decoder architecture. The encoder is to encode the input data and the decoder is used to generate incorporated sequence using the encoding information. It shows the attention layer is powerful compared to the model before. The Transformer allows the training on larger datasets and has lead to the development of the pretrained models such as BERT [26] and GPT series models.

Then the pre-trained models become popular these days. For this kind of pre-trained models, the model is first trained by an extremely large amount of natural language dataset. Later on, to work the model on a specific problem or area, the model fine-tuning will be done on a specific small dataset. With finetuning, the same model could be repurposed to perform different NLP functions on new dataset. By doing this, a lot of time and computational resources could be saved when building a new complex language model.

BERT (Bidirectional Encoder Representations from Transformers) is such kind of pre-trained model developed by Google. Based on the Transformer we mentioned before, BERT is designed to pretrain deep bidirectional representations of input language sequences from both left and right side, with only the encoder part of the Transformer model. As a result, the pre-trained BERT model can be finetuned with just one additional layer to work for other tasks. Since it only includes the encoder part of the Transformer, the BERT works well for problems such as word embedding, question answering... But does not work for tasks such as machine translation.

Another famous pretrained model these days is the GPT2 (Generative Pre-trained Transformer 2) [24], a successor to GPT [27], developed by OpenAI. It has 1.5 billion parameters, trained on a dataset of 8 million web pages. Same as the BERT, it is a Transformer based language model. But different with the BERT, which uses a bidirectional Transformer, the GPT2 uses a left-to-right Transformer. The left-to-right Transformer works well for sentence-level tasks. But it is harmful when applying fine-tuning to token-level tasks such as question answering which incorporate context from both directions.

The third-generation language prediction model in the GPT-n series, GPT3 [29], is introduced by OpenAI in May 2020. It has a capacity of 175 billion parameters in the model. GPT3 provides a text-in, text-out interface via the Python. With its 'text in, text out' API, the user can input the text and the model will generate the text completion to match the input context. The GPT3 performs very well on the translation, question-answering and text generation tasks. The quality of the text generated by GPT3 is very high.

2.2 Time Series Forecasting

There are several diverse approaches to time series forecasting. When the input data has a strong linear connection with the goal value, traditional statistical models such as Exponential Smoothing, Vector Autoregression (VAR), Auto-Regressive Integrated Moving Average (ARIMA) [72] [79] have all achieved high results. A linear model, on the other hand, is incapable of capturing non-linear patterns in time series data. To analyze non-linear relationships in time series data, the Threshold Autoregressive (TAR) and Autoregressive Conditional Heteroscedastic (ARCH) [97] models have been proposed. Because the specialized rules generated in these models could not be extended to other prediction tasks, these models could not be extensively employed in general prediction issues.

Machine learning methodologies were employed to construct an adaptable model for general prediction tasks. [93] and [69] employed the Support Vector Machine (SVM) to solve financial time series forecasting issues, and both of the suggested models have achieved excellent accuracy. But their dataset in the experiments are relatively small. Kayacan et al. [81] evaluate the effectiveness of several modified grey models in predicting foreign currency exchange rates (FX rates); the findings indicate that the grey model is superior in terms of model fitting and forecasting.

Neural networks gained popularity as a technique for solving issues involving time series prediction. At the moment, Recurrent Neural Networks (RNNs), and particularly the long-short term memory network(LSTM) [7, 70], are widely employed in time series forecasting. The RNN and LSTM have the connections between nodes in the time order, which allows it to have the temporal dynamic behavior and use the features from the previous neuron to informa the later ones. It could help to understand the data following the time order. [88] did a prediction of the Bitcoin price using the RNN and LSTM network. Convolutional Neural Networks (CNN) are also a viable alternative for sequence prediction, since its calculations, unlike those of RNN, can be parallelized. [67].

2.3 Deep Linguistic Processing

Word embedding

A word embedding is a learned representation for the words in the text where the words having similar meaning will have more similar vector representations.

Word2vec is a group of word embedding models published in 2013. With the input text, the Word2vec is trained to learn word embeedings from large dataset where similar words are closer to each other. Embeddings learned through Word2Vec have been proven to succeed on a variety of natural language processing works. GloVe [59] is an open sourced word embedding model launched by Stanford in 2014, which uses an unsupervised learning algorithm to obtain the word representations. But there are some problems for the Word2vec and GloVe. Since the vector representation remains same for one word in Word2vec and Glove, they are not effective for words with the same spelling but different meanings.

Then models such as ELMo [31] and BERT start to be used in word embedding. Unlike previous models, the ELMo is a word embedding method that is based on the bi-directional LSTM, which could produce different representations for the same word. Also, BERT take into the account of the whole context and could provide different word embedding based on different context and sentence. It shows the BERT could provide a more powerful word embedding in the natural language processing tasks compared with Word2vec and GloVe.

Natural language and semantic parsing

In recent years, a line of works has been focusing on semantic parsing, which aims at converting natural language utterances to formal meaning representations. ZC05 [30], ZC07 [32], UBL [33] and FUBL [34] induce the specific grammars to make the translation, which defines the meanings of individual words and phrases. KCAZ13 [35] and [12] use ontologies to help form the grammar. [36] uses domain-independent facts to make the translation and ZH15 [37] builds the grammar based on the specific entity type of words. DCS+L [38] and [6] introduce tree structure for input natural language to solve this problem. However, most of the conventional methods rely on predefined templates or manually designed features to complete the translation, which is not comparable to ours, as we avoid using such lexicon mapping and predefined templates of prior knowledge in our system.

Spatial natural language interface to database

Also, some of the work focuses on the Natural Language Interface to Databases (NLIDB) for users to interact with the database without acknowledging the grammar of structured queries executed by the database engine. [3] first explores this issue with a specific database and concrete examples. [39, 11, 40, 41, 42] also work on this issue depending on grammars and processes the semantic grammars manually for each individual database. [4] and [5] work on the NLIDB systems requiring large sets of natural language and SQL pairs. [10] and [13] present an interface with the help of the feedback from users and PEK03 [11] also defines the coverage of the NLIDB system, which is certainly not suitable for all databases. The problem is that the NLIDB study mentioned above is all designed for the general domain and is hard to apply to spatial natural language queries directly without loss of accuracy.

Now deep neural network models have been applied successfully to semantic parsing to exploit the sequential structure on both input and output side. One of them is the Encoder-Decoder model [43, 44, 45]. FKZ18 [46] works on translating the input to SQL queries based on the Encoder-Decoder model. TRANX [22] and ASN [47] construct Decoder-Encoder models with the tree structure. SQL2TREE [48] proposes a seq2seq model based on the Encoder-Decoder architecture and JL16 [49] enhances the performance of seq2seq by adding attention-based copying in the output and implementing data augmentation. [50, 14] work on an Encoder-Decoder based transfer learning for semantic parsing. [51] trains one model that is able to parse natural language sentences from multiple different languages and [52] exploits the Encoder-Decoder model in different domains. [53] introduces a framework with reinforcement learning to generate SQL queries. Here, we introduce on seq2seq model in our system. Compared with SQL2TREE [48] and JL16 [49], our model solves the spatial ambiguity problem for the input natural language more efficiently.

Spatial natural language processing

Natural language process for the spatial domain has been observed in literature. [54] annotates the spatial relation in natural language based on the specific annotation schema. [55] focuses on spatial ontologies to process the input spatial natural language queries. [56] maps the objects and spatial relations to formal linguistic terms, which disambiguate the spatial meanings of objects. [57] uses a form of symbolic expressions to extract spatial terms from natural language descriptions to represent spatial features and relations between them. All of them limit the query in the fixed form and have difficulty dealing with different kinds of spatial complex queries. For [58], it introduces a system that is capable of capturing the semantics of spatial relations in natural language using the neural network. But none of the above gives users an interface to interact with the database.

2.4 Deep Real Estate Prediction

Numerous studies have been conducted in recent years that have focused on real estate-related issues. Deep neural networks, hybrid modeling, and linear regression are just a few of the approaches that have been developed over the last several years to address real estate value prediction difficulties.

Numerous indications based on the Hedonic Model [91] are identified and discussed, as well as their application to house price prediction. By applying an error correction technique, Hall et al. [78] estimate real property values in the United Kingdom. The authors' findings demonstrate the parameters' instability as a result of economic fluctuations and the magnitude of the divergence from long-run equilibrium.

Wang and Peng [94] predicted the monthly Shanghai housing price index using a wavelet neural network. The Grey-Markov model and n-polynomial model [96] are used to forecast the annual average price of housing. Fu et al. [73] [74] evaluated real estate based on the location of the point of interest, the popularity of the property, and customer ratings.

Yan et al. [95] predicted seasonal housing prices using regression and grey models, which they combined with wavelet neural network based on TEI@I methodology to forecast housing price. They also specified contributing indicators that would assist improve the prediction result's accuracy. The TEI@I approach was initially presented for the purpose of forecasting oil prices.

Manjula et al. [87] proposed an Artificial Neural Network (ANN) model for forecasting for forecasting land values in the Chennai Metropolitan Area, India, utilizing data gathered from interviews, newspapers, magazines, and the Internet. Guirguis et al. [77] proposed the Generalized Autoregressive Conditional Heteroscedastic (GARCH) model and Kalman Filter with Autoregressive Presentation (KAR) tmodels to simulate the housing market in the United States. By applying univariate time series models such as the ARIMA, GARCH, and regime-switching, Crawford et al. [71] were able to predict home price increases in states such as California, Florida, Massachusetts, Ohio, and Texas from January 1979 to April 2001.

They discover that regime-switching models outperform ARIMA and GARCH in in-sample forecasting while ARIMA and GARCH outperform ARIMA and GARCH in out-of-sample forecasting.

Chapter 3

Deep Linguistic Translation

3.1 Spatial Natural Language Interface

3.1.1 Problem Overview

Many business applications rely on data-warehousing. To facilitate the usage of database management systems to the public, NLI to databases has been extensively studied [3, 4, 5, 6, 10, 11, 12, 13, 14, 15, 16]. Spatial Domain NLI to databases has drawn great attention due to the popularity of spatial applications [17, 18, 19, 20]. An intuitive solution is to adopt existing NLI in general databases to the spatial domain. However, due to the idiosyncrasy and expressiveness of the spatial semantics, it is unfeasible to adopt general NLI for the spatial domain directly. The challenge of adopting the existing general domain NLI to spatial domain lies to harnessing the expressiveness of spatial semantics. The expressiveness of spatial semantics can be justified based on the following observations [21]:

The meaning of spatial phrase "*Mississippi*"	
How many rivers does ***Mississippi*** *have*?	State
How many cities does ***Mississippi*** *traverse*?	River
The meaning of spatial phrase "*over*"	
How many people walked ***over*** *the bridge*?	On
How many birds flew ***over*** *the bridge*?	Above
The meaning of spatial phrase "*at the back of*"	
How many trees are ***at the back of*** *the building*?	Exterior
How many rooms are ***at the back of*** *the building*?	Interior

Figure 3.1: Spatial Semantics Is Encyclopedic.

The examples as mentioned in Figure 3.1 show that the same spatial phrase in different questions embodies divergent senses expressing divergent query intentions. In the first two questions, "*Mississippi*" as a name can refer to either a state or a river, depending on the context where it is mentioned. In this example, the type of word "*Mississippi*" depends on the verb located after the name ("*have*" or "*traverse*"). In the second two questions, the preposition "*over*" means either a superior position or on the surface. Its spatial meaning depends on the verb located before the preposition ("*walk*" or "*fly*"). In the last two questions, the prepositional phrase "*at the back of*" means either outside the building or inside the building, which depends on the noun before the prepositional phrase ("*tree*" or "*room*"). Such contextually dependent spatial semantics raises serious challenges for NLI to spatial domain databases. For instance, in the third example, if there are two spatial tables (one for the interior architecture of a building and one for the surroundings of a building), a wrongly comprehended spatial semantics would cause the NLI to query a wrong table. In general, spatial semantic understanding relies heavily on its contextual interpretation.

Taking Prolog query as an example, existing works of NLI rely on conventional grammar-based methods or neural network-based methods. The former line of existing work uses predefined templates or manually designed features, which has the lower-transfer ability, thus confined in its specific dataset. The latter line of existing work uses grammar embedded neural networks. Embedding grammar into a model relies on converting the process of generating a sequence of tokens to the task of generating a sequence of actions that expands a syntax tree. Converting word space to action space will inevitably introduce transformation error, which can not guarantee overall accuracy. To the best of our knowledge, the state-of-the-art Syntax-based method TRANX [22] achieves an accuracy of 88.2 percent which is 2.1 percent lower than our accuracy.

The aforementioned observations and survey inspired us to propose a Spatial Domain NLI that is able to support the idiosyncrasy of spatial semantics. Inspired by the NLI in [14, 16], we propose a strategy to address the ambiguity of spatial meaning (mentioned in Figure 3.1) and data sparsity problem by feeding necessary spatial semantics to the deep model. Here ambiguous spatial phrases are those that can not be uniquely identified by the schema. By feeding external spatial semantics, our NLI is able to support various spatial questions even when it has not seen similar semantics in the training set. The extra spatial semantics is recognized by our external spatial comprehension model, whose functionality is to recognize pre-defined spatial semantics.

We propose to capture spatial semantics using an external spatial comprehension model, where the interpretation of each word is based on the attentive combination of the context. We then complete our NLI model using a sequence-to-sequence (seq2seq) translation, which is not only able to achieve grammar correctness but also robust with data sparsity problem. Our fundamental strategy is to separate the tasks of NLI to (1) learning semantic structure of a natural language question, and (2) learning the spatial semantics of a spatial question.

The necessity of the external spatial comprehension model is due to the seq2seq translation model's failure to capture all the spatial semantics while learning the structure of the question. In our design, Task (1) is assigned to the seq2seq model, while an external spatial comprehension model is in charge of Task (2). We propose our spatial comprehension model as a bi-directional attentive workflow [23, 28].

Our strategy is a general-purpose automatic solution that only relies on database content, spatial comprehension model, a seq2seq model, and a minimum amount of human knowledge. To the best of our knowledge, we are the first to use an external spatial semantic understanding model to enhance the performance of the main seq2seq model. Our solution not only addresses the problem of data sparsity but also introduces minimum error since the spatial comprehension model achieves an accuracy of 98 percent for Geoquery dataset.

Our contribution are described as follows

- We propose a spatial comprehension model that is able to recognize the meaning (e.g., POI type) of an ambiguous spatial phrase (e.g., POI name) based on contextual interpretation.

- After injecting spatial semantics learned from spatial comprehension into the question, our model outperforms the state-of-the-art.
- We propose a simple but effective strategy to support multiple query languages (e.g., both Prolog and SQL) in the back-end of databases.
- We evaluate our strategies systematically and show that our spatial comprehension model and injection format perform well as expected.

3.1.2 Challenges

I. Sparse training data. Even though data augmentation is a feasible solution to sparse training data, it is likely that the deep model will be forced to handle unseen questions that are not covered by data augmentation, and required to support transfer learning.

II. Spatial semantics ambiguity. A unique feature of spatial questions is its expressiveness in the spatial domain, and a spatial phrase often has an ambiguous meaning. For example, "*Mississippi*" could be either a state or a river, "*New York*" could be either a city or a state (taking Figure 4.2 as examples).

	Question
State	*How many people live in Mississippi?*
River	*How many states does Mississippi traverse?*
City	*Is New York or London bigger?*
State	*What is the capital of New York ?*

Figure 3.2: Spatial POI Ambiguity

Theoretically, a powerful data augmentation should be able to address the first challenge; however, such data augmentation strategy is rare in practice. Moreover, a seq2seq model is designed to translate a sentence, it is reasonable that it fails to capture the context precisely and infer the correct spatial semantics. For example, in the question "*How many rivers does*

Mississippi have ?" (shown in Figure 3.3), a seq2seq model should be able to understand the context and infer "*Mississippi*" as a state. However, since the word "*rivers*" appears in the question and precedes the word "*Mississippi*", which means it has a major impact on the prediction when attentive on "*Mississippi*", in that case, it is highly possible that `riverid` will be inferred instead of `stateid`.

Question	*How many rivers does Mississippi have* ?
Ground Truth	`answer(A,count(B,(river(B),const(C,stateid(Mississippi)),loc(B,C)),A))`
Infer	`answer(A,count(B,(river(B),const(C,riverid(Mississippi)),loc(B,C)),A))`

Figure 3.3: POI Type Recognition Without Spatial Comprehension Model

Therefore, we propose another deep model for the purpose of spatial semantic understanding; despite the fact that we use the same parse training data, an external model targeted on understanding the context is able to infer the correct spatial semantics (Challenge II) precisely. With the spatial semantics retrieved from the spatial comprehension model, we adopt symbol insertion strategy [14] to inject external information and help the seq2seq to infer an unseen sample correctly (Challenge I).

Overview To address the aforementioned challenges, we present our *SpatialNLI* overview shown in Figure 4.4. The workflow of our SpatialNLI involves the following steps:

1. Identify ambiguous spatial semantics in the NL query.

2. Build a spatial comprehension model that is able to understand a spatial-related question semantically.

3. Injecting spatial semantics retrieved from the spatial comprehension model into the question ($q \longrightarrow q'$).

4. "Translating" the question into a structured query (Lambda expression in our example) ($q' \longrightarrow l'$).

5. Replace the symbols injected to their original text ($l' \longrightarrow l$).

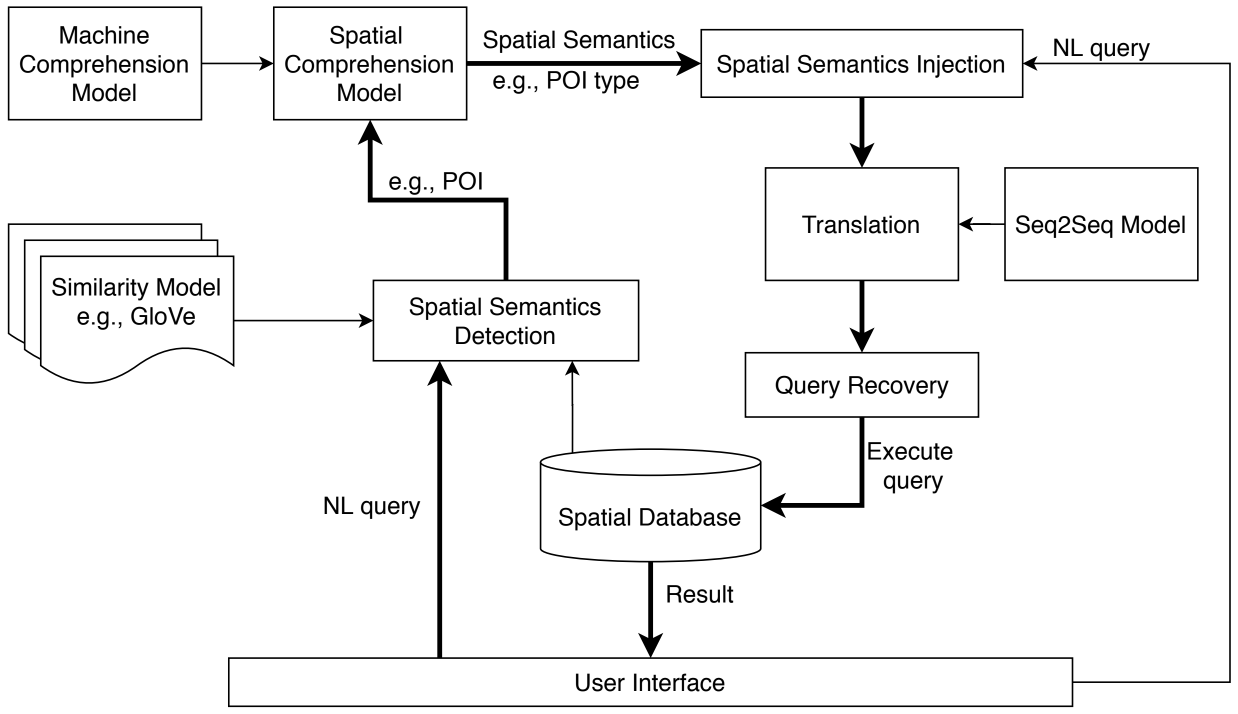


Figure 3.4: SpatialNLI Workflow.

3.1.3 Interface Design

Since most of the keywords or data elements in spatial queries (e.g., lambda expression) are spatial-related, we propose a strategy to inject latent spatial semantics into the natural language question to help the seq2seq model to capture the semantic meaning of the question. For example, for the question "*How many rivers does Mississippi have* ?", its correspondence lambda expression is "`count(B,(river(B), const(C,stateid(Mississippi)), loc(B,C))`", which has five keywords "*count*", "*river*", "*const*", "*stateid*", "*loc*", and three of them are spatial-related. We will illustrate the workflow of our SpatialNLI with this running example.

Our SpatialNLI model is composed of the following steps (corresponding to Algorithm 1)

1. **Spatial Semantics Detection**. Having access to GeoSpatial databases, we detect potential keywords or data elements using 1) string match, 2) edit distance, and 3) cosine distance in semantic embedding space (e.g., Glove). In the aforementioned question, "*Mississippi*" can be detected by comparing against the data in the databases using

string match, "*river*" can be detected by edit distance since "*rivers*" is in the table. We will detail semantic distance measurement later in Section 3.1.3.

2. **Spatial Comprehension Model**. For ambiguous spatial phrases, we propose a spatial comprehension model to resolve the ambiguity. As we mentioned in Section 3.1.2, "*Mississippi*" is an ambiguous POI, and it will be identified as a river type using our spatial comprehension model.

3. **Spatial Semantics Injection**. With identified keyword "*river*" and data element "*Mississippi*" (river name), we inject such information into the question by inserting predefined symbols "*How many* $\langle k0 \rangle$ *rivers* $\langle eok \rangle$ *does* $\underline{\langle k1 \rangle\ stateid\ \langle eok \rangle}$ $\langle v0 \rangle$ *Mississippi* $\langle eov \rangle$ *have* ?"

4. **Seq2seq Translation**. We then feed the modified question to a seq2seq translation model. In the previous example, the predicted output sequence is "`answer(A, count(B, (`$\langle k0 \rangle$`(B), const(C,`$\langle k1 \rangle$`(`$\langle v0 \rangle$`)), loc(B,C)), A))`".

5. **Query Recovery**. The generated sequence of the seq2seq model is then recovered to an executable query. Following the previous example, we have "`answer(A, count(B,(river(B), const(C, stateid(Mississippi)), loc(B,C)), A))`".

Algorithm 1 SpatialNLI

```
1: function SpatialNLI($q$, $D$, $E$)
2:     $P$, $V$ = SpatialMapper($D$, $q$, $E$);
3:     $q'$, $s2p$ =SpatialInjection($q$, $V$, $P$);
4:     $l'$ = Seq2seq($q'$);
5:     $l$ = Recover($l'$, $s2p$);
6:     Return $l$;
7: end function
```

Spatial Semantics Detection

Even though our major contribution is spatial comprehension, we formally define our strategy to detect keywords and data elements mentioned in the question (denotated as $SpatialMapper$) to keep our work self-contained.

$$P, V = \textsc{SpatialMapper}(D, q, E)$$

The inputs are the GeoSpatial database D, a natural language question q, and an embedding function E (e.g., Glove). E will change a word to a high-dimensional vector, which represents its location in the embedding space. We collect the table names, column names, and column values from D, thus D refers to a collection of entities in our spatial mapper. The table names (e.g., river) and column names (e.g., river length) in D are potential keywords of executable queries (e.g., Lambda expression), and column values (e.g., Mississippi) are potentially data elements that might be mentioned in executable queries. The detail of the algorithm is presented in Algorithm 2, in the aforementioned example, $P = [\langle rivers, river\rangle, \langle Mississippi, Mississippi\rangle]$, since "*Mississippi*" is detected by exact string match, and "*rivers*" (in q) and "*river*" (in D) has a small edit distance. We define semantic distance measurement as

$$\text{semantic\_distance}(a, b) = 1 - \frac{E(a) \cdot E(b)}{||E(a)||_2 ||E(b)||_2}$$

The semantic distance is also the spatial distance in the embedding space. If any operand is a phrase which comprises multiple tokens, for example, A is a list of tokens, we define $E(A) = \text{avg}_{a \in A}\big(E(a)\big)$.

Taking question "*Where is the lowest spot in Iowa*?" as an example, its corresponding logic form is `answer(A,lowest(A,(place(A), loc(A,B),const(B,stateid(Iowa)))),` "*spot*" in the NL question is matched to keyword "*place*" since semantic_distance($place$, $spot$) < 0.368, which is relatively small.

Algorithm 2 Spatial Semantics Mapper

```
1: function SpatialMapper($D, q, E$)
2:     $P = \emptyset$;                                    ▷ Spatial semantics matching pairs.
3:     $V = \emptyset$;                          ▷ Spatial values with its semantic meaning.
4:     for $k$ in $K..1$ do                          ▷ Iterating from $K$-gram to 1-gram
5:         for all $p_q \in k$-gram of $q$ do
6:             for all $c \in D$ do
7:                 if $p_q == c$ or semantic_distance($p_q, c$) $< \tau_{sem}$ or edit_distance($p_q, c$) $< \tau_{ed}$
   then ▷ $\tau_{ed}$ is the threshold for edit distance. $\tau_{sem}$ is the threshold for semantic distance.
8:                     $P$.add($\langle p_q, c\rangle$);
9:                     if $c$ is a column value then
10:                        for all table $tb$ that has $c$ do
11:                            $p_q.types$.add($tb$)
12:                        end for
13:                        $V$.add($p_q$)
14:                    end if
15:                end if
16:            end for
17:        end for
18:    end for
19:    Return $P$, $V$;
20: end function
```

We care about the spatial phrases that have semantic ambiguities (e.g., *Mississippi*). An intuitive solution is to use pre-collected human knowledge. However, to devise an automatic and intelligent approach, we propose using Geospatial database; for example, we discover that "*Mississippi*" is an ambiguous value by simply searching for this phrase in the database, and it appears in two tables River and State. In Algorithm 2 Line 9-13, if a phrase appears in multiple tables, we collect all the ambiguous information in V. For example, in V, "*Mississippi*"*.types* = [River, State], "*New York*"*.types* = [City, State] and "*Alabama*"*.types* = [State]. In other words, for a spatial phrase that is a value, we collect the tables it belongs to and stored in V. For most of the Geospatial databases, the table names are able to represent the meaning of the value. In the question "*How many rivers does Mississippi have?*", V = ["*Mississippi*"] and "*Mississippi*"*.types* = [River, State].

It is worth noticing, we use minimum human knowledge to cover phrase mapping that is not covered by Glove. For example, in a question where "*population per km*^{2}" refers to "*population density*", such mapping is not easy to be covered by Glove or a deep model, thus human knowledge is necessary. However, such cases are rare in practice, and we only require minimum human knowledge.

Spatial Comprehension

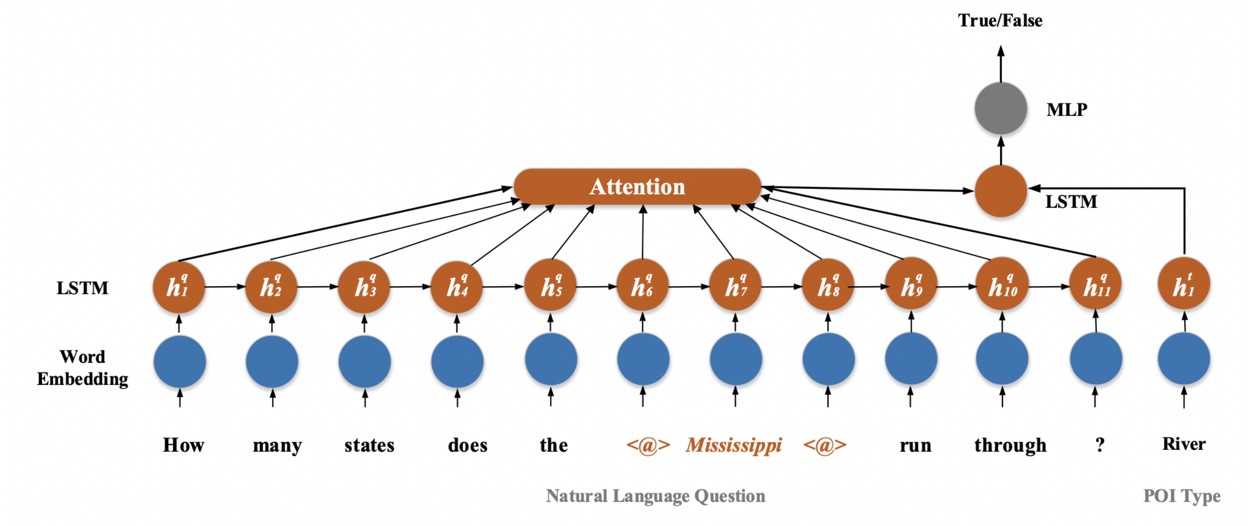


Figure 3.5: Spatial Comprehension Model.

Question	POI Type	Label
How many states does ⟨@⟩ *Mississippi* ⟨@⟩ *traverse?*	State	False
How many states does ⟨@⟩ *Mississippi* ⟨@⟩ *traverse?*	City	False
How many states does ⟨@⟩ *Mississippi* ⟨@⟩ *traverse?*	River	True

Figure 3.6: Using Spatial Comprehension for POI Type Selection. $T = \{State, City, River\}$.

A critical challenge in understanding the spatial question is the meaning of an ambiguous phrase, such as a point of interest (POI). For example in Figure 4.2, where "*Mississippi*" could be a river or a state, we have to differentiate its meaning by the context and understanding the semantic meaning of the question. In the question "How many people live in *Mississippi* ?", we would interpret "*Mississippi*" as a state name, and in the question "How many states does the *Mississippi* run through ?", people would understand "*Mississippi*" is referring to a river.

With large training corps, a deep model might be able to capture that information; however, existing spatial question answer data sets are inadequate and sparse due to the difficulty in collecting ground truth. To address this challenge, we propose a principle method to enable semantic understanding of spatial questions using sparse training data, which relieves the burden of collecting large training sets. By our definition, *Spatial Comprehension* is spatial semantic understanding using machine comprehension. Our strategy is to exploit pre-trained Glove embedding to understand spatial keywords in the question first, then use a seq2seq machine comprehension model to learn the semantic meaning of the question (context) without the burden of extracting the spatial relations.

Model Structure Our spatial comprehension model is designed to understand the "meaning" (e.g., type of POI) of an ambiguous spatial phrase mentioned in the question based on its context.

Inspired by the machine comprehension model using an attention flow [23], we propose our spatial comprehension model composed of two stacked LSTM layers on each input with another shared attentive LSTM layer. The design of the bi-directional attentive workflow [23] is to answer a question given a premise – i.e., locate the sentences in the premise that is most relevant to the answer of the question. The task of the machine comprehension is different from ours; however, they do share the same strategy: understanding one of them (question and premise) semantically based on the context of the other. So we use LSTM to pre-process our NL question and the possible meaning of the ambiguous spatial phrase separately, and conduct an attentive workflow over the hidden states of the question (shown in Figure 3.5). Also inspired by [14], we enclose the ambiguous spatial phrase in special symbols to indicate it has more influence than the other tokens in the question.

We denote a question as $q = [q_1, ..., q_n]$ and the meaning of the ambiguous spatial phrase (mentioned in the question), such as a POI type, as $t = [t_1, ..., t_m]$, both of which are fed to a word embedding layer ϕ (initialized with GloVe [59]). On top of that, we use an LSTM layer to capture the hidden states of each time step i.

We build the same structure for both q and t. We denote the hidden states of the top stacked LSTM layers as

$$H^q = \text{LSTM}(\phi(q)) = (h_1^q, \cdots, h_n^q)H^t = \text{LSTM}(\phi(t)) = (h_1^t, \cdots, h_m^t)$$

Inspired by natural language understanding proposed in [15, 23], we build an extra LSTM layer on H^q with attention over H^t as the follows

$$\begin{aligned} d_i &= \text{LSTM}\left([h_i^t, \beta_{i-1}], d_{i-1}\right) \\ e_{ij} &= v^T \text{Tanh}(W_0 H^t + W_1 h_j^q + W_2 d_i) \\ \alpha_{ij} &= e_{ij} / \sum_{j'} e_{ij'} \\ \beta_i &= \sum_{j=1}^{n} \alpha_{ij} h_j^q \end{aligned}$$

$d_0 = \mathbf{0}$. W_0, W_1, W_2, v, and U are model parameters. Here i is the time step while enumerating t, and j enumerates each token in q. The final output d_m is fed to a multi-layer perceptron (MLP) and then resized to a binary prediction. If t involves a sequence of tokens, we use bi-directional attentive flow as in [23] and compute bi-directional output $d_i = [\overrightarrow{d_i}, \overleftarrow{d_i}]$.

With the attentive flow on type t while reading the question, our spatial comprehension model is able to make the prediction based on the memory of the context. However, with the observation that, given a question "*How many rivers in Mississippi*?" and a POI type "*river*", the machine comprehension model is highly likely to produce a positive prediction (false prediction), since "*river*" is mentioned in the question by "*rivers*", but the model would fail to capture our intention to categorize the type of "*Mississippi*". In order to feed our intention into the model, we insert special symbols (e.g., ⟨@⟩) to enclose the POI mentioned in the question.

The corresponding model structure is shown in Figure 3.5. For the question shown in Figure 3.5, to address the ambiguity of "*Mississippi*", we feed three records shown in

Figure 3.6. Our spatial comprehension function is defined as follows:

$$\textsc{SpatialComprehension}(q, p, t)$$

where q is the question, p is the ambiguous phrase, and t is the meaning of the phrase. If SpatialComprehension returns true, the semantic meaning of p is identified as t.

Spatial Semantics Injection

Now we are able to understand the context and recognize the meaning of each ambiguous spatial phrase correctly through the spatial comprehension model. The question is how to inject the external information to the main seq2seq model. We propose an injection strategy shown in Figure 3.7. The general idea is to insert symbols into the question at the locations before and after every spatial phrase to emphasize its semantics, then feed the inserted question into a seq2seq model.

The first step is to search for components in the question that need a spatial semantics injection–i.e., ambiguous spatial semantics. For example, in the question "*What is the population of San Antonio*?", we do not feed extra information for every word, instead, we only focus on spatial information or tokens that could be shown in the corresponding logic form (e.g., keywords). In other words, we only care about the tokens in the NL question that contribute to its logic form. The tokens such as question word "what" and stop words "is" "the" "of" do not contain the question's information, thus are not annotated.

For the ambiguity of spatial phrases, we believe it is necessary to feed the meaning of the phrase in the question. For example, we feed the **Type of POI** in the question to address the POI ambiguity. We present our *Information Injection Format With Type Feeding* in Figure 3.7. We will validate in the experiment section that our type feeding improves the accuracy dramatically.

We propose a general purpose automatic injection algorithm shown in Algorithm 3. For the input natural language question q, after we recognize the meaning (e.g., type) t_i of each phrase p_i (e.g., POI) correctly through spatial comprehension model, for each pair of ¡p_i, t_i¿ $\in < P,T >$, we will insert the type t_i before the p_i in the input question q. Also, for $p_i \in P$, we store the symbol *sym* of each phrase p_i in *s2p*, which will be used later in the query recovery. For example, as Figure 3.7 shows, the symbol for the phrase "*san antonio*" is $\langle v0 \rangle$. Then the ¡$\langle v0 \rangle$, *san antonio*¿ is stored in *s2p* which will later be used for the recovery of $\langle v0 \rangle$ in the output logic form query.

Algorithm 3 Spatial Semantics Injection

```
1: function SpatialInjection($q$, $V$, $P$)
2:     $s2p = \emptyset$;                                            ▷ Symbol phrase mapping.
3:     $index_v = 0$;                                               ▷ Value Symbol index.
4:     $index_k = 0$;                                               ▷ Keyword symbol index.
5:     $q' = q$;
6:     for all $\langle p_q, c \rangle \in P$ do                              ▷ Iterate each matched pairs.
7:         if $c$ is a keyword then
8:             $sym$ = '$k$'+$index_k$;
9:         else if $c$ is a value then
10:            $sym$ = '$v$'+$index_v$;
11:            Search for $c.types$ from $V$;
12:            if —$c.types$—¿1 then                          ▷ $c$ is an ambiguous spatial phrase
13:                $T = p.types$
14:                for all $t \in T$ do
15:                    if SpatialComprehension($q, p, t$) is $True$ then
16:                        $c.type = t$;
17:                    end if
18:                end for
19:            else if —$c.types$—==1 then                 ▷ $c$ is not an ambiguous spatial phrase
20:                $c.type = c.types[0]$;
21:            end if
22:            Insert $c.type$ to $q'$ (using symbol '$k$'+$index_k$)
23:        end if
24:        $index_k = index_k + 1$;
25:        $index_v = index_v + 1$;
26:        $s2p$.add($\langle sym, c \rangle$);
27:        Insert $sym$ to $q'$;
28:    end for
29:    Return $q'$, $s2p$;
30: end function
```

Figure 3.7 presents our detailed injection format. For a phrase or token in the question that is identified as a keyword (e.g., population), that phrase or token will be enclosed with $\langle ki \rangle$ and $\langle eok \rangle$. For values such as "*San Antonio*" that appear in the question, we enclose them with $\langle vi \rangle$ and $\langle eov \rangle$ where $\langle eok \rangle$ represents "end of keyword" and $\langle eov \rangle$ represents "end of value". Note that we use the spatial databases and the grammar of executable queries to identify keywords and values without referring to the ground truth. Here i indicates it is the i-th spatial semantics that is injected. For a value $\langle v \rangle$, if ambiguity exists, we predict its spatial meaning using spatial comprehension model and feed the spatial semantics into the question using the symbol $\langle k \rangle$.

Question q	*What is the population of San Antonio ?*
Keyword Detection	*What is the population of San Antonio ?*
Symbol q' Injection	*what is the* $\langle k0 \rangle$ *population* $\langle eok \rangle$ *of* $\langle k1 \rangle$ *cityid* $\langle eok \rangle$ $\langle v0 \rangle$ *San Antonio* $\langle eov \rangle$?
Seq2seq Model	
Output l'	`answer(A,`$\langle k0 \rangle$`(B,A),const(B,`$\langle k1 \rangle$`(`$\langle v0 \rangle$`)))`
Recover l	`answer(A,population(B,A),const(B,cityid(San Antonio)))`

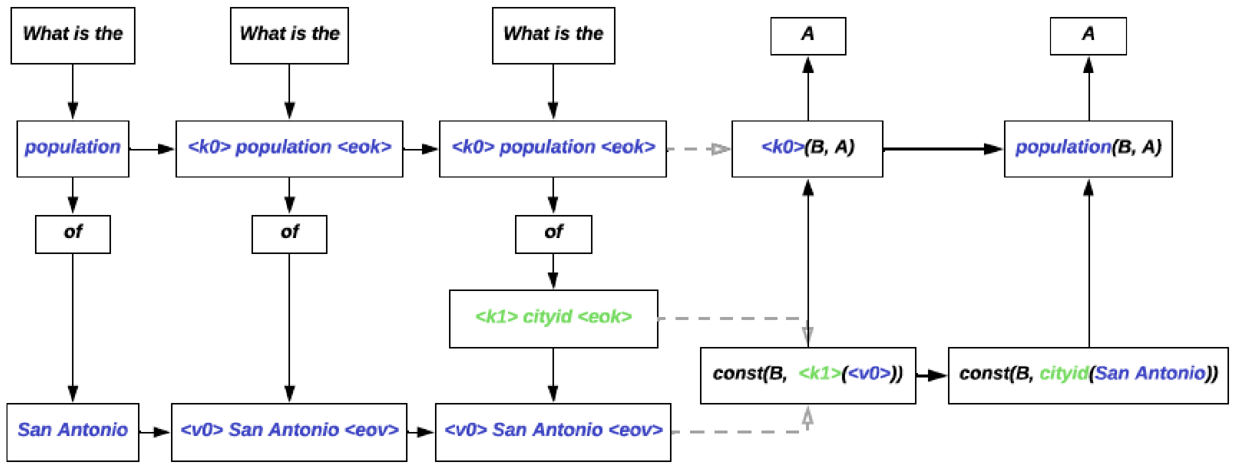


Figure 3.7: An example of Information Injection Format With Type Feeding

As shown in Figure 3.7, since the output of the seq2seq model involves symbols that are inserted into the question which need to be transformed to its original literal form, we propose a query recovery model. The detailed algorithm will be presented in Section 3.1.3.

Following the aforementioned example in spatial comprehension q = "*How many states does the Mississippi run though*?", if we do not address the ambiguity problem and rely on the seq2seq model to infer the spatial meaning of "*Mississippi*", q' will be "*How many* $\langle k0 \rangle$ *states* $\langle eok \rangle$ *does the* $\langle v0 \rangle$ *Mississippi* $\langle eov \rangle$ *run through*?". After translated by the seq2seq model, the recovered query is likely to be `answer(A, count(B, (state(B), const(C, stateid (mississippi)), traverse(C,B)), A))` since "*states*" is mentioned in the question. Our model is fundamentally built upon a seq2seq translation model, where the context is transformed into a weighted sum of all the tokens, in which "*states*" will be embedded as part of the context, and the model is easy to be confused and outputs "*Mississippi*" as a state name. We will mention in Section 3.1.3 later that, since the output vocabulary size is much smaller than the input vocabulary size and most of the tokens in output appear in the input as well, we adopt Copying Mechanism [49], where the output token has a higher chance to be copied from the input sequence. Copying Mechanism makes the ambiguity harder to address, such as in the aforementioned question, "*river*" does not appear in the input question, the model has a higher probability of copying "*state*" from the input sequence. Thus we need to insert '*riverid*' before the word "*Mississippi*" in the input sentence to help the model make the translation. The final input will be "*How many* $\langle k0 \rangle$ *states* $\langle eok \rangle$ *does the* $\langle k1 \rangle$ *riverid* $\langle eok \rangle$ $\langle v0 \rangle$ *Mississippi* $\langle eov \rangle$ *run through*?".

Translation Model

Since seq2seq models have been widely adopted in translation tasks, and our NLI task is simpler than a translation task due to small vocabulary size. We believe a seq2seq model is able to capture the logic and the spatial structure of the question as long as it is able to understand the entities that are mentioned in the question. So we adopt a seq2seq model as in Figure 3.8 with copying mechanism following [14].

$$l' = \textsc{seq2seq}(q')$$

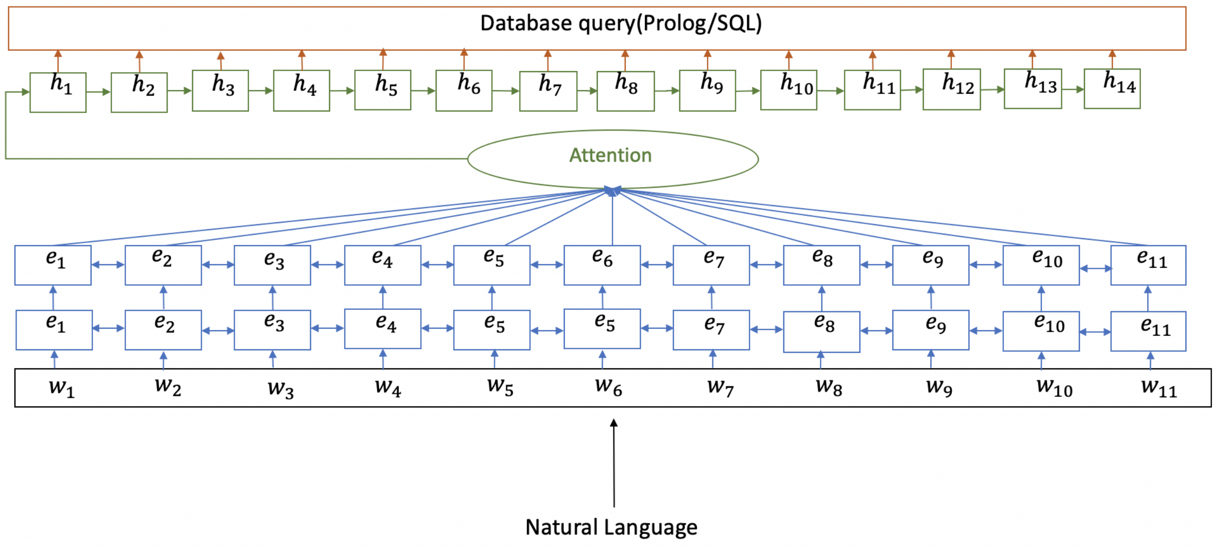

Database query(Prolog/SQL)
h_1 h_2 h_3 h_4 h_5 h_6 h_7 h_8 h_9 h_{10} h_{11} h_{12} h_{13} h_{14}
Attention
e_1 e_2 e_3 e_4 e_5 e_6 e_7 e_8 e_9 e_{10} e_{11}
e_1 e_2 e_3 e_4 e_5 e_5 e_7 e_8 e_9 e_{10} e_{11}
w_1 w_2 w_3 w_4 w_5 w_6 w_7 w_8 w_9 w_{10} w_{11}
Natural Language

Figure 3.8: Seq2seq Model.

Query Recovery Model

Algorithm 4 Symbol Recovery

```
1: function RECOVER($l'$, $s2p$)
2:     $l = l'$;
3:     for all $\langle sym, c\rangle \in s2p$ do
4:         $l$.replace($sym$,$c$);
5:     end for
6:     Return $l$;
7: end function
```

We detail our strategy of query recovery through Algorithm 4, whose inputs are l', the output of translation model, and $s2p$, the symbol-phrase pairs detected by spatial semantics injection model (Algorithm 3). Just as Figure 3.7 shows, the output l' of the seq2seq translation model is a sequence of the symbol, for example, $\langle k0\rangle$, $\langle k1\rangle$ and $\langle v0\rangle$ here. Then we need to recover the output logic form query. For each pair $\langle sym, c\rangle \in s2p$, every symbol $sym\ l'$ needs to be replaced by the original phrase c. After replacing all symbols in l', we finally get the output logic form. In Figure 3.7, for output l', we replace the symbols $\langle k0\rangle$, $\langle k1\rangle$ and $\langle v0\rangle$ by their corresponding phrase "*population*","*cityid*" and "*San Antonio*" based on the pairs in $s2p$. After recovery, we finally get the right output logic form.

Data Augmentation By Shuffling

Just as mentioned before, one of the Challenges right now is the lack of training set. The sparsity of training data causes two problems: 1). The semantic structures of questions are sparse; 2). The data entities mentioned in the questions are inadequate. Problem 2 can be simply addressed by replacing data entities. However, addressing problem 1 is non-trivial. So we propose to shuffle the prepositional phrases to augment the semantic structures of the training set.

We propose our unique augmentation strategy as follows: If a question has a prepositional phrase (PP as a POS tag), and the question can be decomposed as $q = q^{\text{prefix}}|q^{\text{PP}}$ or $q = q^{\text{PP}}|q^{\text{suffix}}$, where q^{prefix} are the words placed before the prepositional phrase and q^{suffix} are the words placed after the prepositional phrase, we will shuffle the position of the prepositional phrase. Note that we only consider the questions that start with a prepositional phrase or end with a prepositional phrase.

Consider the example in Figure 3.9, for the query *"Which states does the Mississippi river run through"*, the format of the question is $q = q^{\text{prefix}}|q^{\text{PP}}$, and q^{prefix}="*Which states does the Mississippi river run*" and q^{PP} ="*through*". By exchanging q^{prefix} and q^{PP}, we can get a new sentence "*Through which states does the Mississippi river run*" and the meaning of the new query remains the same. Also, for the other question "*In what state is Mount Mckinley* ", $q = q^{\text{PP}}|q^{\text{suffix}}$, q^{PP}="*In what state*", and q^{suffix}="*is Mount Mckinley*". we can shuffle the position of the prepositional phrase to get a new sentence.

q	$\underbrace{\textit{Which states does the Mississippi river run}}_{q^{\text{prefix}}} \underbrace{\textit{through}}_{q^{\text{PP}}}$?
Augment	$\underbrace{\textit{Through}}_{q^{\text{PP}}} \underbrace{\textit{which states does the Mississippi river run}}_{q^{\text{prefix}}}$?
q	$\underbrace{\textit{In what state}}_{q^{\text{PP}}} \underbrace{\textit{is Mount Mckinley}}_{q^{\text{suffix}}}$?
Augment	$\underbrace{\textit{Mount Mckinley is}}_{q^{\text{suffix}}} \underbrace{\textit{in what state}}_{q^{\text{PP}}}$?

Figure 3.9: Two Examples Of Data Augmentation

3.1.4 Experimental Validation

Experimental Settings

Configuration All our experiments are conducted on a machine equipped with 2 Intel CPU E5-2670 v3 running at 2.3GHz with 256GB of RAM and 2 NVIDIA Tesla K80 GPUs.

Dataset To evaluate the effectiveness of our system, we performed an experimental evaluation on dataset Geoquery and Restaurants.

- *Geoquery* [39] is a collection of 880 natural language questions and corresponding executable database query pairs about U.S. geography. The answers in this dataset are defined in λ-calculus logical form. We follow the standard training-test split to that of [30], of which the dataset was divided into 600 training examples and 280 test examples respectively. As [49], [38] and [60], we determine its *Acc* based on the denotation match.

- *Restaurant* (Rest) [40, 11] is a dataset with 251 question-answer pairs about restaurants, their food types, and locations. The questions are all human natural language and the answers are in λ-calculus logical form.

Data Augmentation

We not only propose our new data augmentation strategy by shuffling in Section 3.1.3, but also adopt a data augmentation strategy that is based on the recombination [49] of a sentence itself.

For example, as in Figure 3.10, for the query "*What is the highest point in Florida?*", we can simply identify that the word "*Florida*" is the name of a state based on the spatial database. Given this example, we change it to new questions, in which the word "*Florida*" is replaced by the name of other states in the database. Here, the word "*Florida*" is replaced by "*Rhode Island*". For the second example in Figure 3.10, for the query "*What is the highest point in Florida?*" and the query "*What state has the smallest population density?*", we can infer that the entire expression of the second sentence could map to the word "*Florida*" in the first query since this query is asking about one state. Then we can generate one new question by replacing the word "*Florida*" with the second sentence. For the third example, the two queries, "*What state has the largest population?*" and "*What state has no rivers?*", are both asking about one state, so we combine them together to generate a new query.

Type 1	Original	*What is the highest point in Florida* ?
	Augment	*What is the highest point in Rhode Island* ?
Type 2	Original	*What is the highest point in Florida* ?
		what state has the smallest population density ?
	Augment	*What is the highest point in state that has the smallest population density* ?
Type 3	Original	*what state has the largest population* ?
		what state has no rivers ?
	Augment	*what state has the largest population and has no rivers* ?

Figure 3.10: Three Examples Of Data Augmentation

Spatial Comprehension Model

We preprocess the dataset for spatial comprehension model so that each record contains (1) A question with each POI phrase enclosed with symbols (e.g., ⟨@⟩) indicating the attentive position; (2) A POI type (e.g., River, State, and City.).

Dataset		Train	Test
Geoquery	Acc_{rcd}	97.4%	91.9%
	Acc_{qu}	98.3%	98.1%
Rest(aurant)	Acc_{rcd}	100.0%	100.0%
	Acc_{qu}	100.0%	100.0%

Table 3.1: Spatial Comprehension Model Evaluation.

For a question "*How many states does the Mississippi run through?*" with one ambiguous POI "Mississippi", we have the three records as shown in Figure 3.6. To balance the positive and negative samples in the training set, we replicate positive samples. For samples in Figure 3.6, we replicate positive samples by 2 times.

We run experiments with 200 hidden units and 300-dimensional pre-trained Glove embedding. We minimize the cross entropy using Adam Optimization Algorithm. We evaluated the performance of our spatial comprehension model in Table 3.1. Acc_{rcd} represents the percentage of correctly predicted records. Acc_{qu} represents the percentage of correctly predicted questions where all POIs are recognized correctly. For the example in Figure 3.6, the total number of samples for Acc_{rcd} is 3, and the total number of samples for Acc_{qu} is 1. Even the training objective function is to optimize Acc_{rcd}. In fact Acc_{qu} is what we are trying to optimize, and we prove that Acc_{rcd} and Acc_{qu} are optimized simultaneously.

We evaluate on Geoquery and Rest datasets, respectively (shown in Table 3.1). Test Acc_{qu} is 98.1 percent for Geoquery, and 100.0 percent for Restaurant data, respectively. All the Acc_{qu} is not less than Acc_{rcd}. In other words, our spatial comprehension model is able to recognize the spatial semantics with high confidence.

Evaluation

For the encoder and the decoder of our seq2seq model, we use one layer of Gated Recurrent Unit (GRU) with a hidden size of 800 and $800 * 2$, respectively. The input and output of both encoder and decoder share the same embedding layer, which is initialized with 300-dimensional pre-trained Glove embedding. Special symbols inserted (e.g., k_1 and v_1) are treated as special tokens; they are represented by the concatenation of an embedding of the symbol type (e.g., k and v) and an index, where the embedding of the symbol type and the index are randomly initialized with 150-dimension (the concatenation has a dimension of 300). The other unknown token is initialized with a 300-dimension random vector. For training, we use gradient clipping with a threshold 5.0, and for inference, we use beam search with width 5.

	q	*How many states does the Mississippi run through ?*
+SC	q'	$\langle k0\rangle$ *How many* $\langle eok\rangle$ $\langle k1\rangle$ *states* $\langle eok\rangle$ *does the* $\langle k2\rangle$ *riverid* $\langle eok\rangle$ $\langle v0\rangle$ *Mississippi* $\langle eov\rangle$ *run* $\langle k3\rangle$ *through* $\langle eok\rangle$?
	Infer	`answer(A,`$\langle k0\rangle$`(B,(`$\langle k1\rangle$`(B),const(C,`$\langle k2\rangle$`(`$\langle v0\rangle$`)),`$\langle k3\rangle$`(C,B)),A))`
	Recover	`answer(A,count(B,(state(B),const(C,riverid(Mississippi)),traverse(C,B)),A))`
-SC	q'	$\langle k0\rangle$ *How many* $\langle eok\rangle$ $\langle k1\rangle$ *states* $\langle eok\rangle$ *does the* $\langle k2\rangle$ *stateid* $\langle eok\rangle$ $\langle v0\rangle$ *mississippi* $\langle eov\rangle$ *run* $\langle k3\rangle$ *through* $\langle eok\rangle$?
	Infer	`answer(A,`$\langle k0\rangle$`(B,(`$\langle k1\rangle$`(B),const(C,`$\langle k2\rangle$`(`$\langle v0\rangle$`)),`$\langle k3\rangle$`(C,B)),A))`
	Recover	`answer(A,count(B,(state(B),const (C,stateid(Mississippi)),traverse(C,B)),A))`

Figure 3.11: Spatial Comprehension Case Study.

*+SC means using Spatial Comprehension, -SC means without.

Table 3.2 presents our experiment results for (a) Geoquery dataset and (b) Rest(aurant) dataset. For Geoquery, compared with the previous models, our method outperforms the state-of-the-art. The conventional methods are overdependent on predefined templates and manually designed features, which have lower accuracy on the test set. For neural network-based methods such as ASN [47] and TRANX [22], they convert word space to action space, which inevitably introduces transformation error. SQL2TREE [48] and JL16 [49] use seq2seq model as well, but fail to address spatial semantics ambiguity. For Rest, the state-of-the-art achieves 100 percent accuracy, which states the Rest dataset is an easier task than Geoquery. Our model exhibits excellent downward compatibility by achieving 100 percent accuracy on Rest dataset.

To validate the performance of our system, several ablation experiments were conducted by the removal of (1) Copy Mechanism, (2) Spatial Comprehension Model, (3) Data Augmentation, (4) Type Feeding, and (5) Information Injection, respectively. By the removal of the spatial comprehension model, we random guess the meaning (type) of ambiguous POI and inject it to the question. For removing information injection, we feed the original content to the model without inserting any symbols. By the removal of type feeding, we conduct symbol injection but omit to inject the extra spatial information (e.g., $\langle k1 \rangle$ cityid $\langle eok \rangle$ in Figure 3.12).

First, we measure the contribution of the spatial comprehension mechanism to the overall performance of the model. We train and evaluate two models: one with the spatial comprehension model and one without. Training is done with data augmentation and information injection. In Table 3.2, for Geoquery and Restaurant, with the removal of spatial comprehension model, the denotation match accuracy drops 4 percent for Geo and drops 3.9 percent for Rest. Since only 19.3 percent of the test set for Geoquery and 4 percent of the test set for Restaurant has POI ambiguity problem, it is obvious that our machine comprehension model is able to resolve the majority of them.

As shown in Figure 3.11, by comparing against the spatial database, "*Mississippi*" appears in two tables: River table and State table. Without spatial comprehension, if we are using random guess, "*river*" has only 50 percent chance to be correctly categorized. The '+comprehension' in the figure means we use the spatial comprehension model and '-comprehension' is for the result without a right understanding of "*Mississippi*". Without the spatial comprehension model, it is possible for the system to recognize the "*Mississippi*" as a state name. As the figure shows, once "*Mississippi*" is recognized as a state, it will insert "*stateid*" in the input question and finally get a wrong result after recovery. One interesting thing is that the infer for '+comprehension' and '-comprehension' are the same, both correct. This is because, for seq2seq model, it just outputs the result with $\langle ki \rangle$, not the specific word. Here the "*riverid*" and "*stateid*" are both replaced by $\langle k2 \rangle$. Thus we get the same infer result from seq2seq model. However, after recovery, the result without spatial comprehension model is wrong.

Table 3.2 shows that by removing type feeding, the accuracies drop 5.4 percent on Geoquery and 29.4 percent on Restaurant. The symbol injection significantly improves the accuracy of the Restaurant dataset since for most samples in the Restaurant dataset, one token in the input question always corresponds to multiple tokens/symbols in the output sequence, which relationships are hard for the seq2seq model to capture.

As shown in Table 3.2, the information injection component improves test accuracy by 7.5 percent on Geoquery and 39.2 percent on Restaurant. When we stop injecting information into the natural language question, the seq2seq is not able to capture all the necessary information to infer correctly and suffers from a large accuracy decrease. A case study of our symbol injection strategy is shown in Figure 3.12, where a seq2seq model generates outputs token by token and a large number of entities involve a sequence of tokens. Without symbol injection, the seq2seq model has to infer "*San Antonio*" token by token using two steps. On the other hand, with symbol injection, the seq2seq model generates $v0$ as a representation of "*San Antonio*", which only requires one step. Our symbol injection format is able to replace a name entity composed of a sequence of tokens to a single symbol, which prevents wrong name entity caused by a long sequential generation.

-SI	q	*What is the population of San Antonio* ?
	Infer	`answer(A, population(B,A), const(B,cityid(San Jose)))`
+SI	q	*What is the population of San Antonio* ?
	q'	*what is the* $\langle k0 \rangle$ *population* $\langle eok \rangle$ *of* $\langle k1 \rangle$ *cityid* $\langle eok \rangle$ $\langle v0 \rangle$ *San Antonio* $\langle eov \rangle$?
	Infer	`answer(A,` $\langle k0 \rangle$`(B,A), const(B,`$\langle k1 \rangle$`(`$\langle v0 \rangle$`)))`
	Recover	`answer(A, population(B,A), const(B,cityid(San Antonio)))`

Figure 3.12: A Case Study using Symbol Inject (Geoquery).

* +/- SI means with/without Symbol Injection.

We also jointly train both datasets in a shared model compared with separate training, shown in Table 3.5. Jointly training achieves an accuracy of 90.7 percent. Our experiment results show that a shared model performs better than two separate models.

3.1.5 Conclusion

In this work, we propose an NLIDB applied for the spatial domain to convert natural language queries to structured queries executable by database. The main contribution of our work is to recognize the meaning of the ambiguous spatial phrases based on contextual interpretation and capture the semantic structure of the question by the seq2seq model with injecting spatial information. Our extensive experimental analysis demonstrates the advantage of our approach over state-of-the-art methods.

3.2 Spatial Interface With Flexible Back End

3.2.1 Challenges Of Spatial NLI With Flexible Back End

Besides the challenges in the Section 3.1.2, adding a flexible back-end to spatial NLI could be more challenging.

We argue that if an NLI model only supports a single query language (e.g., SQL), its commercial value would be limited since different database engines are better at different tasks. For example, Prolog [1] works well for logic-intensive tasks where a lot of reasoning is involved; NoSQL works better for unstructured or semi-structured data; and SQL is widely adopted in industry. It is important to incorporate flexible back-end when designing a natural language interface to databases.

To address this issue, a prefix symbol is introduced to support the flexible back-end. The system overview will be the same as Section 3.1.2. But we prefix a query type token to indicate the target query language so that the 4th step, "Translating", in the Section 3.1.2 is able to support more types of database query such as Prolog and SQL.

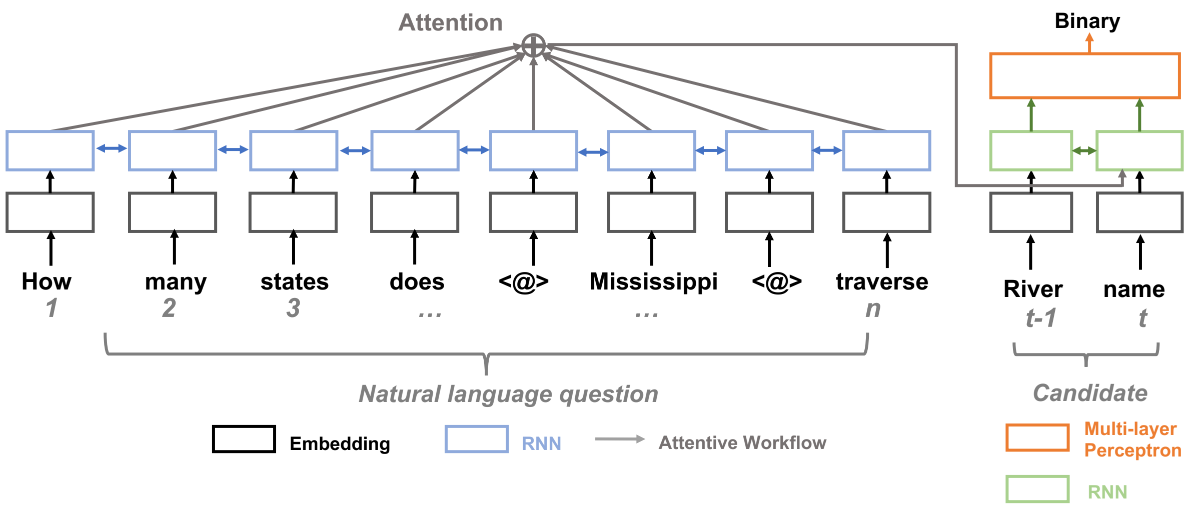


Figure 3.13: Spatial Comprehension Model.

Question	Table	Label
What are the populations of states through which the Mississippi traverses?	State	True
What are the populations of states through which the Mississippi traverses?	City	False
What are the populations of states through which the Mississippi traverses?	River	True
What are the populations of states through which the Mississippi traverses?	Border	False
What are the populations of states through which the Mississippi traverses?	Mountain	False

Figure 3.14: Using Spatial Comprehension for schema linking.

$T' =$ {State, River}.

3.2.2 Spatial Comprehension With Table Selection

The corresponding model structure is shown in Figure 3.13. We build the same structure for both q and t. We denote the hidden states of the top stacked RNN layers as

$$H^q = \text{RNN}(\phi(q)) = (h_1^q, \cdots, h_n^q)$$

$$H^t = \text{RNN}(\phi(t)) = (h_1^t, \cdots, h_m^t)$$

[1]https://www.swi-prolog.org/pldoc/man?section=db

Inspired by natural language understanding proposed in [15, 23], we build an extra RNN layer on H^q with attention over H^t as follows:

$$\begin{aligned}
d_i &= \text{RNN}\left([h_i^t, \beta_{i-1}], d_{i-1}\right) \\
e_{ij} &= v^T \text{Tanh}(W_0 H^t + W_1 h_j^q + W_2 d_i) \\
\alpha_{ij} &= e_{ij} / \sum_{j'} e_{ij'} \\
\beta_i &= \sum_{j=1}^{n} \alpha_{ij} h_j^q
\end{aligned}$$

$d_0 = \mathbf{0}$. W_0, W_1, W_2, v, and U are model parameters. Here i is the time step while enumerating t, and j enumerates each token in q. The final output d_m is fed to a multi-layer perceptron (MLP) and then resized to a binary prediction. If t involves a sequence of tokens, we use bi-directional attentive flow as in [23] and compute bi-directional output $d_i = [\overrightarrow{d_i}, \overleftarrow{d_i}]$.

When a database is too large to be scanned for each question and the NLI has to identity related schema first (i.e., schema linking), we adopt spatial comprehension model for schema linking. Specifically, T is the collection of tables, SpatialComprehension(q, T) will return $T' \subset T$ where T' are related tables. Unlike ambiguity challenge, we do not need to annotate anything in q and feed it as the original format (shown in Figure 3.14). After schema linking, our method will redefine D as T' tables.

3.2.3 NLI With Flexible Back End

Inspired by Google multilingual translation model [63], we propose a multi-language natural language interface (i.e., flexible back end) [15]. The idea is simple but effective: we prefix a query type token to indicate the target query language. For example, considering a natural language query "*What are the populations of states through which the Mississippi traverses*?". As shown in Figure 3.15,

- If we prefix $\langle Prolog \rangle$ to the model input, the model will generate

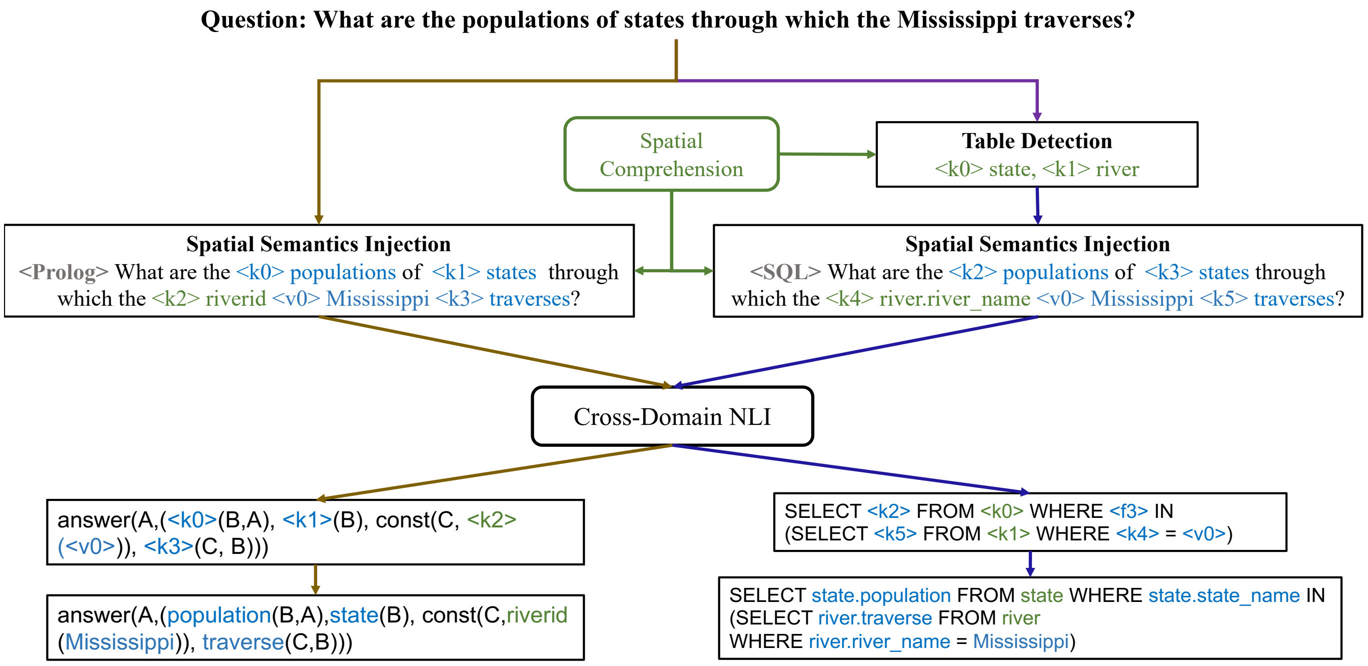

Question: What are the populations of states through which the Mississippi traverses?
Spatial Comprehension
Table Detection
<k0> state, <k1> river
Spatial Semantics Injection
<Prolog> What are the <k0> populations of <k1> states through which the <k2> riverid <v0> Mississippi <k3> traverses?
Spatial Semantics Injection
<SQL> What are the <k2> populations of <k3> states through which the <k4> river.river_name <v0> Mississippi <k5> traverses?
Cross-Domain NLI
answer(A,(<k0>(B,A), <k1>(B), const(C, <k2>(<v0>)), <k3>(C, B)))
answer(A,(population(B,A),state(B), const(C,riverid(Mississippi)), traverse(C,B)))
SELECT <k2> FROM <k0> WHERE <f3> IN (SELECT <k5> FROM <k1> WHERE <k4> = <v0>)
SELECT state.population FROM state WHERE state.state_name IN (SELECT river.traverse FROM river WHERE river.river_name = Mississippi)

Figure 3.15: Flexible Back-End Overview

Prolog (left) and SQL (right).

answer(A,(population(B,A),state(B),const(C,riverid(Mississippi)),traverse(C,B)))

- If we prefix $\langle SQL \rangle$ to the model input, the model will generate

  SELECT state.population FROM state where state.state_name IN (SELECT river.traverse FROM river WHERE river.river_name = Mississippi)

3.2.4 Experimental Validation

Experimental Settings

Dataset

To evaluate the effectiveness of our system, we performed an experimental evaluation on dataset Geoquery in both SQL and Prolog formats.

Geoquery [39] is a collection of 880 natural language questions and corresponding executable database query pairs about U.S. geography. The original answers in this dataset are defined in λ-calculus logical form (Prolog), later the answers are converted to SQL[2].

The dataset was divided to 600 training examples and 280 test examples [30]. For SQL format, the accuracy is calculated by exact query match. For Prolog, we determine its accuracy based on the denotation match following [49, 38, 60].

SQL Evaluation

We compare our method with the following baselines:

- Transformer [25]. Transformer is the proposed for contextual understanding. We test our data on the pre-trained Transformer [3].
- RAT-SQL [62]. RAT-SQL is the state-of-the-art for Spider[4] dataset. As the sketch of SQL in Spider dataset is very similar to ours. We compare with RAT-SQL [5].
- Spatial Semantics Injection. As our spatial semantics injection strategy is orthogonal to most of the designs, we inject our information when applicable.

  (1)Bert Embedding [6] + Seq2Seq. We use pre-trained bert embedding with sequence-to-sequence model.

  (2)Transformer. We adopt the previous setting, and the only difference is the input data has our injected information.

Table 3.4 presents our evaluation on SQL query language. Similar to our schema-aware strategy, RAT-SQL encodes the database relations and learns the alignment between schema columns and their mentions in a natural language query. In this section, we compare

[2]https://github.com/jkkummerfeld/text2sql-data
[3]https://github.com/Kyubyong/Transformer
[4]https://yale-lily.github.io/spide
[5]https://github.com/microsoft/rat-sql
[6]https://huggingface.co/Transformers/

different schema-aware strategies where RAT-SQL uses direct schema encoding and schema linking, while our design uses an external model to focus on schema linking (especially spatial ambiguity), and feed the schema alignment to the translation model using predefined symbols. We argue that even though direct schema linking is a more compact design, but it fails to address challenges posed by spatial domain (e.g., spatial ambiguity). Our design outperforms all the state-of-the-arts and is specifically designed for spatial domain.

We also compare our work with Transformer [25], and the accuracy is 55.2 percent. We hypothesize that the contribution of pre-trained embeddings (BERT) is limited since the major challenge of an NLIDB is schema linkage and alignment rather than contextual understanding. To further test our hypothesis, we run our dataset with information injection on Transformer, and the accuracy was largely improved (62.3%). However, Transformer does not take the consideration that the output space is much smaller than the input space and most output tokens are directly copied from the input.

Information Injection Performance Our information injection strategy works well on both Transformer and seq2seq models, which proves that our design is orthogonal to the model structure and can be adopted by any NLIDB model.

SQL and Prolog Performance Gap We also notice that Prolog accuracy is much higher than SQL. Existing high-accuracy achievements [16, 61] on translating natural language questions to SQL are often limited to simple SQL sketch, shown as the following:

```
SELECT $AGG $SELECT_COL
WHERE $COND_COL $OP $COND_VAL
(AND $COND_COL $OP $COND_VAL)*
```

Datasets with complex SQL sketches (e.g., JOIN, nested SQL) suffer from relatively low accuracy. For example, the deep learning state-of-the-art for Spider dataset is only

65.6 percent accuracy. Existing deep models can hardly handle complex SQL with nested queries, GROUP BY, and ORDER BY, etc. A most challenging bottleneck of such translation is schema linking, i.e., aligning entity referenced in the natural language question to schema [62].

For logic-intensive queries, it would be very hard for NLI itself to handle heavy schema linking and alignment for SQL. Our Geoquery dataset is a Spider-style dataset which requires complex table join. For such queries, Prolog is designed to handle logic-intensive queries with a lot of reasoning, and it is reasonable to observe that Prolog has a higher accuracy than SQL.

Jointly Learning for Different Query Languages

As we mentioned earlier, a NLI model should be able to support different query languages, otherwise the practical value would be diminished. In this section, we test our model on both SQL and Prolog.

As shown in Figure 3.15, we prefix an indicator (i.e., ⟨SQL⟩ or ⟨Prolog⟩) to each natural language question for training and inference. We up sampling less numbered dataset so that each query language has equal number of training samples. Even though Prolog dataset experiences slight performance decrease, we believe it is due to the greater challenges of modeling SQL queries where the schema linkage and alignments require more complex logical inferences.

3.2.5 Conclusion

In this work, we propose an NLIDB applied for the spatial domain to convert natural language queries to multiple types of queries executable by databases. The main contribution of our work is to support a flexible back-end translation of spatial natural language including

Prolog and SQL which are the most commonly used queries in the database. Our extensive experimental analysis demonstrates the advantage of our approach over state-of-the-art methods.

3.3 Transfer Learning of Spatial Data Query

3.3.1 Transfer Learning and Spatial Query

When a model is developed for one activity, it is then used in another study to help in the enhancement of learning for the second target task, this is referred to as "transfer learning." Specifically, in this part, we propose a transfer learning technique for addressing the problem of spatial natural language translation to database queries. We train a base model from natural language to the Prolog database query dataset, and then we apply it to another kind of database query dataset, the SPARQL query dataset, to see how it performs. The models are validated against the Geoquery dataset and are shown to outperform known methodologies in terms of transferability, as shown by the experiment result.

3.3.2 Experiment Validation

In order to verify the transferability of our model, we trained the SpatialNLI network on the Prolog dataset and then used the same model on the Sparql dataset to ensure that it was transferable. The model's capacity to translate from normal language to Sparql is shown in the table 3.6. Upon further fine tuning, the model demonstrates the capacity to translate from natural language to Sparql, with the outcome demonstrating that it outperforms the Transformer model in this regard.

The reason we chose Prolog and SPARQL for the transfer learning experiment rather than SQL is that when working with a SQL dataset, there is an assumption that the table name must first be detected before the translation can be continued in the NLI, which is not the case when working with Prolog and SPARQL. Seeing as how both Prolog and Sparql could be put straight into the neural network without any modification, we could utilize the model trained by Prolog to finish the transfer learning job for SPARQL by using the model learned by Prolog.

Dataset	Geoquery	
Conventional	ZC05 [30]	79.3%
	ZC07 [32]	86.1%
	UBL [33]	87.9%
	DCS+L [38]	87.9%
	FUBL [34]	88.6%
	ZH15 [37]	88.9%
	KCAZ13 [35]	89.0%
Deep Model	ASN [47]	87.1%
	SQL2TREE [48]	87.1%
	TRANX [22]	88.2%
	JL16 [49]	89.3%
Ours	SpatialNLI	**90.4**%
	– Copy Mechanism	88.9 %
	– Spatial Comprehension	86.4 %
	– Type Feeding	85.0 %
	– Data Augmentation	83.2 %
	– Information Injection	82.9 %

(a)

Dataset	Restaurant	
Conventional	PEK03 [11]	97.0%
	TM00 [40]	99.6%
Deep Model	FKZ18 [46]	100.0%
Ours	SpatialNLI	**100.0**%
	– Spatial Comprehension	96.1 %
	– Copy Mechanism	94.1 %
	– Data Augmentation	92.2 %
	– Type Feeding	70.6 %
	– Information Injection	60.8 %

(b)

Table 3.2: Experiment Results

"–" means the removal of each component. The accuracy is measured as denotation match [49] on test set.

Training	Geoquery	Restaurant
Separately	90.4%	100%
Jointly	**90.7**%	**100**%

Table 3.3: Evaluation Of Jointly Training On Denotation Match.

Data	Test	
Previous	Transformer [25]	55.2 %
	RAT-SQL [62]	56.2%
	Information Injection	
	+ Bert Embedding + Seq2Seq	60.5%
	+ Transformer	62.3%
Ours	SpatialNLI	**66.3**%
	– Copy Mechanism	63.1 %
	– Spatial Comprehension	63.8 %
	– Data Augmentation	62.7 %
	– Information Injection	40.8 %

Table 3.4: SQL Evaluation.

"–" means the removal of each component. For RAT-SQL methods, we use schema information for fair comparisons.

Training	Prolog	SQL
Separately	**90.4**%	66.3%
Jointly	86.4%	**67.7**%

Table 3.5: Evaluation Of Jointly Training Of Prolog And SQL.

Data	SpatialNLI	Transformer
	60.36%	51.79%

Table 3.6: Evaluation Of Transfer Learning from Prolog to Sparql.

Chapter 4

Deep Real Estate Forecasting

4.1 Deep Real Estate Dynamics Encoding

4.1.1 Introduction

As a result of the global economy's rapid evolution, the economies of many countries face a diverse variety of difficulties and opportunities. It is still hard to forecast the future of the global economy, despite the fact that economic development has received considerable attention in recent years. Economic development cannot be predicted with high confidence using static data, dynamic data, or other types of data due to the massive number of elements involved, some of which are unknown at the moment, and the great number of variables that must be considered. We seek to visualize and forecast economic growth at a fine-granular level in this study by using an example from the real estate. We anticipate that our findings will aid individuals in making data-driven decisions. The objective of this research is to identify and anticipate real estate investment opportunities that give a higher rate of return and a greater possibility that the hotspot we recommend will rise significantly in the near future.

For years, real estate prediction was a manual process. However, as data accumulates daily, it is impractical for prospective real estate buyers to manually assess the pertinent information online. Individuals are seeking the assistance of automated analysis supported by deep learning to accurately evaluate the worth of real estate property. Due of the problems inherent in integrating static and non-static data from possibly inaccurate sources, economic prediction research (e.g., real estate) is rudimentary.

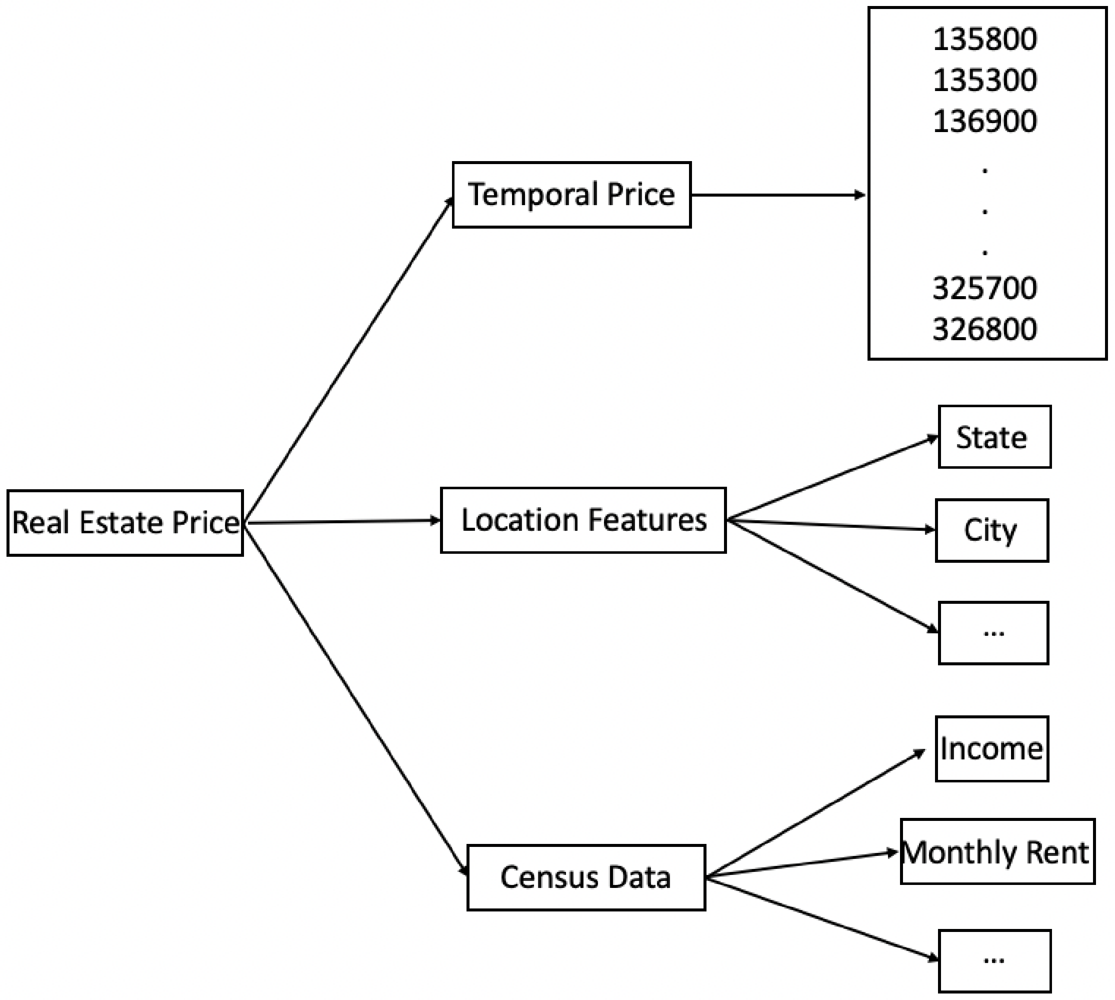


Figure 4.1: Related features.

Temporal price history, census data, and location features are all important factors in predicting real estate prices.

In previous works, collecting static data online for the purpose of predicting real estate values has been explored. Static characteristics, on the other hand, are insufficient to accurately predict the value of real estate property. Prospective buyers frequently rely on price history to determine the worth of real estate. As illustrated in Figure 4.1, the difficulties arise from recording temporal dynamics and merging them with static information in order to completely represent a listing's real estate potential.

To solve this issue, we present a practical approach for encoding temporal price listings using the Transformer-based model and combining it with other static real estate characteristics. In brief, we utilize Zillow data on neighborhood listings and supplement it with data from the United States public census records collected by us.

To demonstrate our idea, we structure our problem as a multi-class categorization indicating which regions see bigger average value increments. We concentrate on region-by-region aggregated value increments to ensure the dataset's privacy and anonymity. Additionally, we normalize data to remove the effect of US dollar inflation. Please keep in mind that we are forcasting real estate hotspots that may experience value growth, which does not necessarily indicate whether a location is desirable or not.

Our contributions are described as follows:

- A large-scale real estate-related dataset is created by merging numerical real estate data from Zillow with a publicly available dataset from the Census Bureau.
- A properly developed Transformer-based forecasting model is proposed that is capable of capturing changes in real estate values and making investment predictions.
- We incorporate both static and temporal information in order to overcome prediction challenges.
- The experiment's results indicate that our proposed model is highly accurate and outperforms all baseline models.

All of the specifics of our new dataset are provided in Section 4.1.2. Our model architecture and specific design are described in depth in Section 4.1.3. In Section 4.1.4, we conduct an experimental evaluation of our proposed approach and dataset. Section 4.1.5 is the section that discusses the conclusions .

Datasets	# records	# features
Manjula et al. [87]	21000	7
Limsombunchai [84]	200	40
Lee et al. [83]	116	62
Park et al. [90]	5359	76
Nur et al. [64]	9	16
Goodman et al. [76]	28561	46
Sampathkumar et al. [92]	n/a	13
Ours	7437	294

Table 4.1: Comparisons of real estate datasets.

We compare the number of records and features in datasets used in other literature.

4.1.2 Proposed Dataset

Existing Datasets

There are several public datasets that provide information on housing, and these databases span a wide range of countries and geographic areas. Table 4.2 examines the geographical locations (State, Metro, and so on) of several real estate datasets in the United States. Spatial locations are critical in our data gathering since the data integration is based on administrative spatial units such as counties and regions, which are the primary data collecting places. One of the publicly accessible real estate-related datasets is a real estate value dataset from Zillow [75], which is one of the datasets that is currently available. With housing data spanning from April 1996 to September 2019, it covers 7,436 neighborhoods in 567 cities and 304 counties in 225 metropolitan areas and 50 states throughout the United States. Zillow Research provides statistics on a weekly and monthly basis. However, one disadvantage of utilizing the Zillow dataset is that it only contains data that is directly connected to the worth of a property, such as rental prices, listing prices, and sale prices, and does not include relevant economic and demographic data, which may be useful.

Dataset	State/Province Level	Metro Level	County Level
Zillow Dataset	+	+	+
Census Bureau's Dataset[1]	+	+	+
NBSC Quarterly Housing Dataset[2]	+	+	−
FHFA's House Price Index Dataset[3]	+	+	+
Magicbricks' Dataset[4]	+	+	−
NJOP House Price Dataset	+	+	−
NYC Property Sales Dataset[5]	−	−	−
	Neighborhood Level	Building Unit Level	
Zillow Dataset	+	−	
Census Bureau's Dataset	−	−	
NBSC Quarterly Housing Dataset	−	−	
FHFA's House Price Index Dataset	−	−	
Magicbricks' Dataset	−	−	
NJOP House Price Dataset	−	−	
NYC Property Sales Dataset	+	+	

[1] https://www.census.gov/

[2] http://www.stats.gov.cn/english/

[3] https://www.fhfa.gov/

[4] https://www.magicbricks.com/

[5] https://www1.nyc.gov/site/finance/taxes/property-rolling-sales-data.page

Table 4.2: Existing Public Dataset Comparisons.

Existing Public Dataset Comparisons. We compare the scale of data in these public datasets. The scale of data is categorized to state/province level, metro level, county level, neighborhood level, and building unit level.

In addition, academics often use the housing statistics provided by the Census Bureau. This dataset, which is based on survey data, contains information on housing in the United States from 1991 to 2019. It contains a variety of housing-related data, such as size, age, house prices, and rentals; it also contains economic and demographic data on a county, zip code, or tract level. As stated in Table 4.4, we have gathered static data related to real estate price change.

The quarterly home price dataset from the National Bureau of Statistics of China (NBSC) might be a valid data source for studying the Chinese real estate market, according to [95]. This dataset contains historical data on commercial housing sales prices at the nation, province, and city levels from 1999 to 2020, as well as data from the 2010 Chinese population census, which includes data from all households in the country. The House Price Index Dataset from the Federal Housing Finance Agency (FHFA) contains house price indexes that assess changes in single-family home prices based on data from 50 states and more than 400 cities throughout the United States of America. It contains statistics on housing indices spanning the period from the mid-1970s until 2020. Monthly Purchase-Only Indexes, Quarterly Purchase-Only Indexes, etc. are all accessible in their public datasets. Magicbricks.com contains a comprehensive collection of Indian homes, including residential and commercial housing statistics for key Indian cities from 2007 to 2020 [87]. Alfiyatin et al. [64] model and evaluate housing prices in Indonesia using data from Nilai Jual Objek Pajak (NJOP). This dataset offers time series data on housing in key cities across Indonesia from 2014 to 2017. It contains information about the property's identification number, location, coordinates, year, building area, and land area, as well as the NJOP land price (IDR/$m2$). The NYC Property Sales dataset contains information on every building or unit (apartment, condo, etc.) sold in New York City within a 12-month period. It comprises information on the location, address, type of housing, selling price, and date of sale of all building units sold.

In Table 4.1, We compare the amount of records and characteristics in other studies' datasets to our own. While our dataset may not have the most records, it does contain 294 characteristics, which is 218 more than the [90] that comes in second place.

Dataset Construction

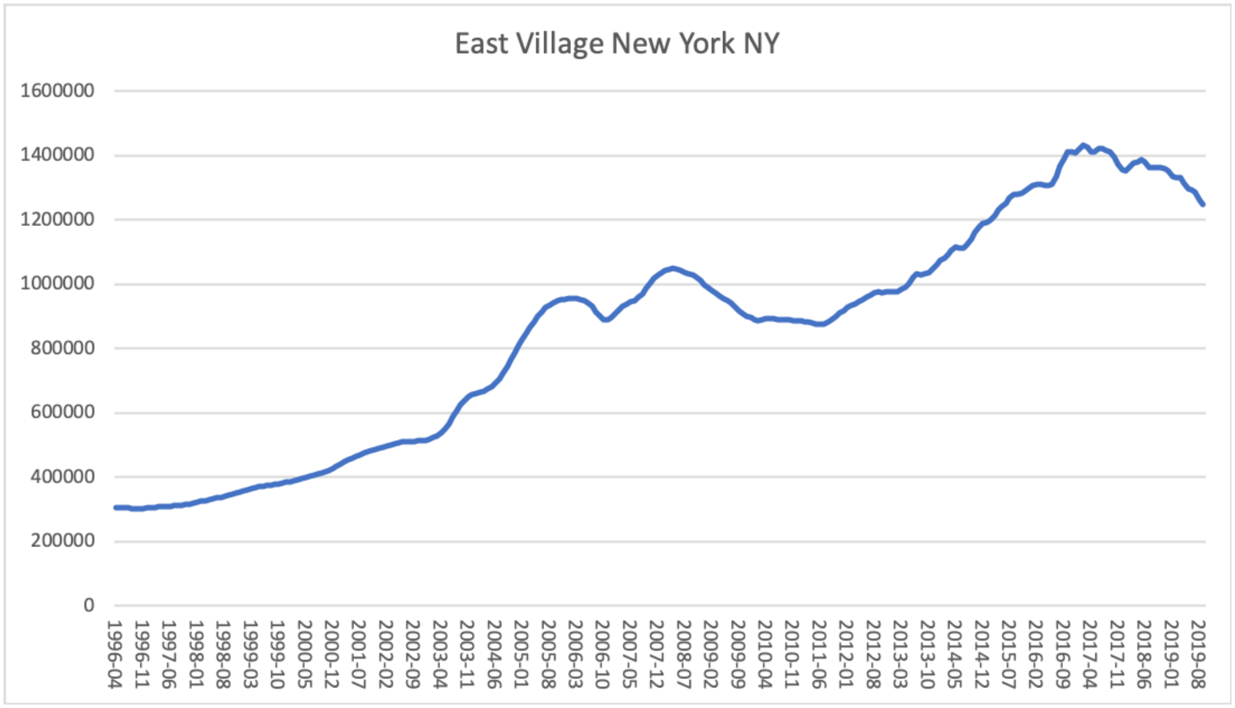


Figure 4.2: An example of a real estate price trend.

We propose a new dataset to capture the dynamics of the real estate market. We gathered geographically correlated data from a variety of sources, and the data is linked together based on their physical placements (State, County). The gathered data are based on geographical correlation, and our suggested dataset comprises a multi-level spatial hierarchy (State, Metro, County, City, Neighborhood). There are three components of our data collection:

- Census data. The specifics are shown in the following Table 4.4. Each record indicates the region's aggregated average. We utilize factors that are directly connected to the price of housing: Median Value, Total Price Asked, Wage or Salary Income, Total Household Income, transportation (time) to work.

- Pricing History. The pricing history format is described in Table 4.3, which describes temporal price history in county level. Real estates in the same county but different regions share the same census record.

- Location Features. The dataset includes neighborhood location information. The inner relationships between neighborhoods are examined. For instance, we determined if two neighborhoods are situated inside the same city/county/metropolitan area/state.

RegionID	274772
RegionName	Northeast Dallas
City	Dallas
State	TX
CountyName	Dallas County
SizeRank	1
Time Stamps	Average Price
April, 1996	135800
May, 1996	135300
June, 1996	136900
...	...
September, 2019	325700

Table 4.3: Temporal data format.

It has data on the region name, city, state, county name, size rank, time stamps, and average price.

The Figure 4.2 depicts the average trend in East Village (New York, NY) real estate prices from 1996 to 2019. The real estate market is heavily associated with economic conditions and exhibits year-to-year fluctuations. This observation motivates us to encrypt time-series price histories in order to capture the full range of historical real estate dynamics.

ID	Data Source
1	Median Gross Rent (monthly)
2	Total Household Income (yearly)
3	Total population
4	Transportation to Work
5	Wage or Salary Income
6	Health Insurance Coverage
7	Employment Status
8	Bedrooms
9	Total Contract Rent
10	Total Rent Asked
11	Total Gross Rent
12	Aggregate Gross Rent
13	Median Value
14	Total Price Asked
15	Selected Monthly Owner Costs
16	Housing Cost (monthly)

Table 4.4: Census Data Examples.

Data Collection

As depicted in Figure 4.3, we gather the dataset in five steps:

1. **Price History Collection**: We gather region-wise aggregated price history. Each record has a monthly price history spanning the years 1996 to 2019.

2. **Census Data Collection**: The county-wise census data is obtained from the public data warehouse. Because the county is the most fine-grained granularity accessible from the public data repository, information is gathered for each county.

3. **Location Data Collection**: The location data for each real estate, including state, city, metropolitan area, county, neighborhood information, is collected.

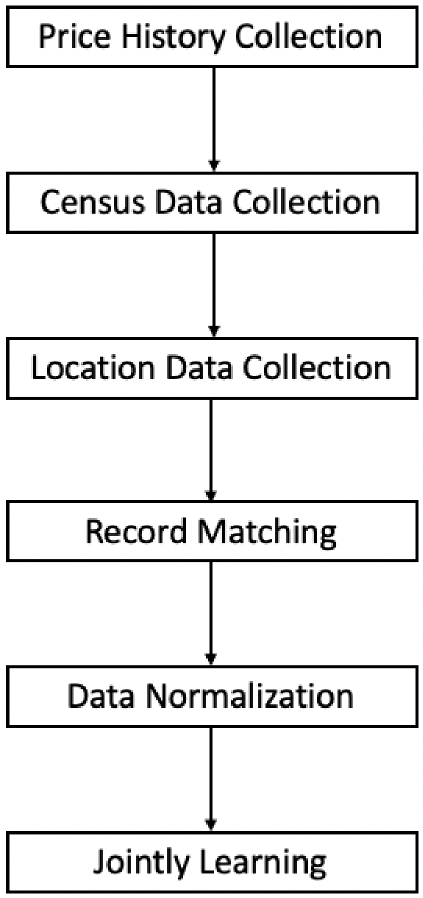
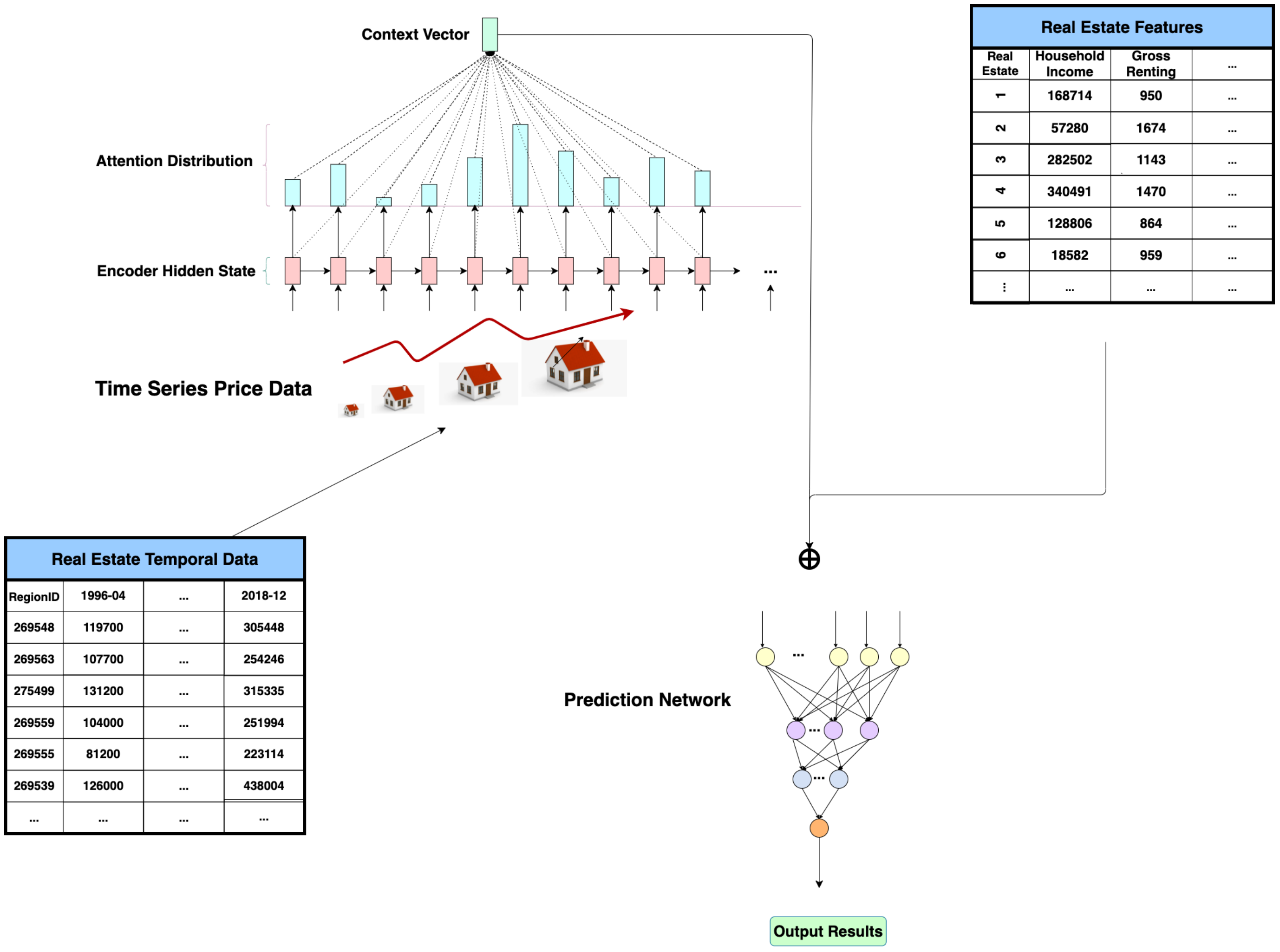


Figure 4.3: Data collection process.

4. **Record Matching**: Because a county is divided into areas, numerous pricing history records are associated with the same census data. We utilize the same census data from each county for all areas within that county.

5. **Data Normalization**: It is necessary to take inflation into account, and the price data is normalized.

6. **Jointly Learning**: In our deep learning model, we learn both temporal and census data concurrently.



Real Estate Features

Real Estate	Household Income	Gross Renting	...
1	168714	950	...
2	57280	1674	...
3	282502	1143	...
4	340491	1470	...
5	128806	864	...
6	18582	959	...
...	...	...	...

Real Estate Temporal Data

RegionID	1996-04	...	2018-12
269548	119700	...	305448
269563	107700	...	254246
275499	131200	...	315335
269559	104000	...	251994
269555	81200	...	223114
269539	126000	...	438004
...	...	...	...

Figure 4.4: Design flow of the Transformer-based prediction model.

We first employ a Transformer layer to encode the information of the time-series real estate price data, then concatenate it with the local features. By combining the time series data with other features, a four-layer MLP model is used for the final prediction.

4.1.3 Model Design

This section explains our model architecture and the advantages of our approach. Our data is classified into two categories:

- Temporal price data x_t. Modeling real estate dynamics necessitates consideration of prior price trends. The temporal price history is defined as x_h with N timestamps,

$$x_t = \{x_t^1, x_t^2, ..., x_t^N\}$$

$x_t^i \ _{i \in \{1..N\}}$ represents the average price for the i-th timestamp.

- Census data x_c. As shown in Table 4.4, by using census data, we can gain insight into the elements that influence the real-estate market.

We normalized the temporal data we collected in the data preprocessing step by applying the US inflation rate from 1996 to 2018 to the temporal data. We then used the normalized monthly temporal data for 2017 and 2018 to generate the median value for both years and then further generated the hotspot index as the prediction model's target value.

Figure 4.4 illustrates our proposed architecture. We begin by encoding time series price data with a Transformer encoder and then combining it with real estate-related census information. To forecast the future hotspot index, the combined latent encoding is fed into a multi-layer perceptron.

Model Architecture

I. Preliminaries

The structure of our Transformer-based prediction model is shown in Figure 4.5. It first consists of an **Encoder** and then **Multi-layer Perceptron** (MLP) layers. The major component in the Transformer is the **Multi-head self-attention**.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

where the K is the key matrix and V is the value matrix. Q is the query matrix that will map against a set of keys.

II. Transformer-based Encoding and Prediction

Predicting the fluctuations in real estate values is accomplished through the use of a precisely constructed forecasting model based on Transformer. We were able to fulfill the prediction challenge by using not only time-series data on real estate prices, but also data on the features of the properties. Local factors such as the property's location, gross rental revenue, household income, transportation, and health insurance, etc. are all considered while determining the instant real estate price in a given area. These features are all considered to have a great impact on the real estate price. We use this local information to help prediction in our model.

Price history, as can be seen in Figure 4.2, is an important indicator for predicting future prices. Such temporal data, on the other hand, is insufficient to capture the complete picture and must be supplemented with census information. Examples include two regions A and B, which have both experienced an increase in price in the past, while A has a comparatively high Median Gross Rent (as shown in the Table 4.4) and B has a relatively low Median Gross Rent. It is most likely that A will exhibit a stable price change whereas B may experience a value decrease in this situation.

We make the prediction using a Transformer-based model. In recent years, it has been demonstrated that a Transformer model [25] works effectively in time series data. In our approach, we first use a Transformer layer to encode the information contained in the time-series real estate price data. This Transformer layer could assist in capturing the change rule of data as it changes over time. Then, after embedding the time series data into the model, we concatenate it with the local characteristics (e.g., local gross renting, household income, etc.). Precision prediction will be aided by these region characteristics. A four-layer MLP model is used to make the final prediction after merging the time series data with additional features.

III. Jointly Learning Model

We propose concatenating temporal and non-temporal data in the same latent space and combining the latent vectors for the future prediction job. Multi-layer perceptrons are used to encode the data census features x_c. Formally, we define the l-th layer network as

$$h_c^{(l)} = ReLu(W_c^{(l)} h_c^{(l-1)} + b_c^{(l)})$$

where

$$h_c^{(0)} = x_c$$

Assuming we use L^c layers in total, and we use the final layer to summarize census information, which is defined as

$$\mathbf{h_c} = h_c^{(L^c)}$$

and $\mathbf{h_c} \in R^{d_c}$.

Temporal price history is encoded by using a Transformer encoder [25], and $\mathbf{h_t} \in R^{d_t}$.

Since x_t and x_c are encoded as $\mathbf{h_t}$ and $\mathbf{h_c}$, the merged hidden state is defined as $\mathbf{h_m}$

$$\mathbf{h_m} = [\mathbf{h_t}, \mathbf{h_c}]$$

To further process the merged data, we employ another multi-layer perceptron for the prediction task. Similarly, we define the l-th layer in the network as (assuming L^m layers in total)

$$h_m^{(l)} = ReLu(W_m^{(l)} h_m^{(l-1)} + b_m^{(l)})$$

where $h_m^{(l-1)}$ is the input of the (l-1)-th layer in the i-th position. $W_m^{(l)}$ and $b_m^{(l)}$ are model parameters.
We use the output from the last layer for prediction

$$\bar{y} = sigmoid(h_m^{(L^m)})$$

Loss is measured with the Mean Squared Error (MSE)

$$loss = MSEloss(\bar{y}, y)$$

$$MSEloss = \frac{1}{n} \sum_{i=1}^{n} (y - \bar{y})^2$$

where n is the number of items; y is the actual value and $\bar{y}$ is the predicted value.

4.1.4 Experimental Validation

The train-to-test split ratio in our dataset is 7:3.

Ground Truth

The ground truth for model training and testing is set by using neighborhood pricing history. The true label in the training data is defined as 2018 price minus 2017 price. The price history before 2017 is used as training data.

Data Normalization

Inflation has a long-run effect on house prices as well as the pricing of non-housing products and services [66]. As a result, inflation is considered, as property values often increase in lockstep with inflation over time. The adjusted value, which takes inflation into account, is supposed to assist disclose the underlying value of residential real estate.

In order to capture the pricing trends across a 23-year period, we normalize the listing prices for each year based on the fluctuation of the U.S. dollar against the euro and other major currencies. The real house listing price will be determined using historical data on the inflation rate[1] in the United States . The temporal price history is defined as x_h with N timestamps,

$$x_t = \{x_t^1, x_t^2, ..., x_t^N\}$$

The U.S. inflation rate historic data is defined as I_h with N timestamps,

$$I_t = \{I_t^1, I_t^2, ..., I_t^N\}$$

$$x_{actual} = \{x_t^1 * I_h^1, x_t^2 * I_h^1, ..., x_t^N * I_h^N\}$$

[1]https://www.macrotrends.net/countries/USA/united-states/inflation-rate-cpi

Evaluation Metrics

In order to quantify the performance of our model, we use the metrics of Accuracy, Recall, Precision, F1-score, and Roc for further evaluation. Let t_p be the number of true positives, f_p be the number of false positives, f_n be the number of false negatives, t_n be the number of true negatives.

Accuracy measures how many observations, both positive and negative, were correctly classified. It is simply a ratio of correctly predicted observation to the total observations.

$$acc = \frac{t_p + t_n}{t_p + f_p + t_n + f_n}$$

Precision, also called positive predictive value, is the number of true positives divided by the total number of elements labelled as belonging to the positive class.

$$precision = \frac{t_p}{t_p + f_p}$$

Recall, also known as sensitivity, is defined as the ratio of true positives and total number of elements that actually belong to the positive class. It measures how many of the true positives were found.

$$recall = \frac{t_p}{t_p + f_n}$$

F1-score is the harmonic mean of the precision and recall. It balances both the concerns of Precision and Recall in one number.

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

ROC is the receiver operating characteristic curve [98], which is a graphical plot that illustrates the performance of a binary classifier system. It is created by plotting the fraction of

L	Precision	Recall	F1-score
2	0.88	0.98	0.93
3	0.92	0.94	0.93
4	0.94	0.88	0.91
5	0.92	0.86	0.89

(a) class 0

L	Precision	Recall	F1-score
2	0.98	0.90	0.94
3	0.80	0.89	0.84
4	0.73	0.81	0.77
5	0.66	0.77	0.71

(b) class 1

L	Precision	Recall	F1-score
2	n/a	n/a	n/a
3	0.97	0.88	0.93
4	0.80	0.69	0.74
5	0.58	0.72	0.64

(c) class 2

L	Precision	Recall	F1-score
2	n/a	n/a	n/a
3	n/a	n/a	n/a
4	0.87	0.95	0.91
5	0.78	0.54	0.64

(d) class 3

L	Precision	Recall	F1-score
2	n/a	n/a	n/a
3	n/a	n/a	n/a
4	n/a	n/a	n/a
5	0.88	0.88	0.88

(e) class 4

L	Accuracy	ROC
2	93.5%	0.941
3	90.1%	n/a
4	83%	n/a
5	74.6%	n/a

(f) Results with all classes

Table 4.5: Hyper-parameter L evaluation.

The accuracy decreases as L increases. It indicates that when having more classes, it would be more difficult to make accurate predictions.

TPR (True Positive Rate) vs. the fraction of FPR (False Positive Rate).

$$TPR = \frac{t_p}{t_p + f_n}$$

$$FPR = \frac{f_p}{f_p + t_n}$$

Hotspot Prediction

Study Area

Figure 4.6 presents the location of the study area in this study. It consists of 7,436 neighborhoods, 567 cities, 304 counties, 225 metros, and 50 states across the U.S.

Predictions of hotspots and their increase in values over time are the primary focus of our problem settings, as previously noted. Price comparisons are not what we're interested in. For instance, If location A has an average value of $100,000$ and location B has an average value of $10,000$, A will rise by $1,000(1\%)$ and B will rise by $500(5\%)$. Because of the greater increment percentile, we conclude B is a hotspot rather than A. Figure 4.7 displays the U.S. hotspots for the year 2018.

The purpose of hotspot prediction is to identify areas that will see a significant increase in the value of the houses in the near future. Early detection and prediction of hotspot locations is extremely significant since it can assist property buyers and investors in identifying properties with high potential value. Figure 4.7 illustrates the hotspots' position. For the purpose of establishing the ground truth for hotspot prediction, we further define the increment of 2018 minus 2017 of greater than 7.5 percent as high (2), greater than 4.5 percent but less than 7.5 percent as medium (1), and less than 4.5 percent as low (0).

We denote the increase rate of a region as

$$z = \frac{P_2 - P_1}{P_1}$$

where P_2 is 2018 average price, P_1 is 2017 average price. We formalize the hotspot prediction problem as a multi-category classification with L **classes**. We partition the increase rate range to L intervals

$$(-\infty, \mu_1),\ [\mu_1, \mu_2),\ ...,\ [\mu_{L-1}, \infty)$$

Each label is defined as

$$\begin{aligned} z \geq \mu_{L-1} &\quad label = L-1 \\ &\cdots \\ \mu_1 \leq z < \mu_2 &\quad label = 1 \\ z < \mu_1 &\quad label = 0 \end{aligned}$$

Note that the intervals are partitioned with equal distribution. Taking L = 3 as an example

$$\begin{aligned} z \geq \mu_2 &\quad label = 2 \\ \mu_1 \leq z < \mu_2 &\quad label = 1 \\ z < \mu_1 &\quad label = 0 \end{aligned}$$

$\mu_2 = 7.5\%$, $\mu_1 = 4.5\%$ in our experimental setting.

In this thesis, the term hotpot refers to a location with a greater investment potential and a greater likelihood of future value increasement. Multi-class classification is used in the hotspot prediction challenge, and the higher the score (label), the greater the increment.

Hyper-parameter Evaluation

In this section, we use a different number of classes L to conduct evaluation tasks. We test $L \in \{2, 3, 4, 5\}$. The number of hotspot locations will be reduced when more classes are used, and the prediction results will reveal a more detailed classification for neighborhoods with varying house value increment rates. As shown in Table 4.5, as L increases, the prediction result decreases. This suggests that forecasting accurately becomes more difficult as the number of classes increases. As an example, when $L = 2$, we have the highest accuracy with our proposed model, 93.5 percent, and when $L = 5$, we have the lowest accuracy, 74.6 percent. Additionally, when $0 < L \leq 3$, the proposed model has achieved high precision, recall, and F1-score in all classes. However, when $L > 3$, the proposed model has a relatively poor performance in some of the classes. For example, when $L = 4$, the proposed model has 0.73 in precision when class is 2 and has 0.69 in recall when class is 3. When $L = 4$, the proposed model has 0.66 in precision when predicting class 1, 0.58 in precision when predicting class 2, and has 0.54 in recall when predicting class 3.

Baselines

We use the following baseline methods:

- Logistic Regression (LR) [89]
- Long Short Term Memory (LSTM) [7]
- Random Forest (RF) [80]
- Decision Tree (DT) [85]
- Support Vector Machine (SVM) [68]
- Multilayer Perceptron (MLP) [65]

The experimental results are shown in Tables 4.6, 4.7, 4.8 and 4.9.

Performance Analysis

The comparisons in Tables 4.6 and 4.7 indicate that the deep models outperform traditional machine learning models when $L = 2$ or $L = 3$. LR outperforms all other classic machine learning methods. It predicts hotspots with a 90 percent accuracy when $L = 2$, and with a 78.3 percent accuracy when $L = 3$. The proposed Transformer-based model has achieved 93.5 percent accuracy when $L = 2$ and 90.1 percent accuracy when $L = 3$, outperforming all of our baseline models. When $L = 2$, the model that has the second-best performance is LR, which is 3.5 percent worse than the result of the proposed model. When

Model/Precision	**class0**	**class1**
RF	0.79	0.84
SVM	0.77	0.78
DT	0.73	0.77
LR	0.91	0.90
MLP	0.82	0.94
LSTM	0.79	0.96
Ours	0.88	0.98

(a) Presicion

Model/Recall	**class0**	**class1**
RF	0.84	0.79
SVM	0.76	0.78
DT	0.77	0.73
LR	0.90	0.91
MLP	0.93	0.84
LSTM	0.96	0.79
Ours	0.98	0.90

(b) Recall

Model/F1-score	**class0**	**class1**
RF	0.81	0.81
SVM	0.77	0.78
DT	0.75	0.75
LR	0.91	0.90
MLP	0.87	0.89
LSTM	0.86	0.87
Ours	0.93	0.94

(c) F1-score

Model	**Accuracy**	**Roc**
RF	0.81	0.813
SVM	0.772	0.772
DT	0.765	0.754
LR	0.90	0.904
MLP	0.878	0.879
LSTM	0.866	0.874
Ours	0.935	0.941

(d) Accuracy and Roc

Table 4.6: Model performance comparisons (L=2). The proposed Transformer-based model has achieved 93.5% accuracy.

Model/Precision	**class0**	**class1**	**class2**
RF	0.68	0.54	0.76
SVM	0.67	0.48	0.66
DT	0.59	0.47	0.66
LR	0.82	0.64	0.81
MLP	0.72	0.66	0.94
LSTM	0.76	0.74	0.92
Ours	0.92	0.80	0.97

(a) Presicion

Model/Recall	**class0**	**class1**	**class2**
RF	0.73	0.51	0.75
SVM	0.59	0.49	0.72
DT	0.58	0.48	0.66
LR	0.80	0.63	0.82
MLP	0.96	0.58	0.80
LSTM	0.88	0.65	0.90
Ours	0.94	0.89	0.88

(b) Recall

Model/F1-score	**class0**	**class1**	**class2**
RF	0.70	0.52	0.76
SVM	0.62	0.49	0.69
DT	0.59	0.48	0.66
LR	0.81	0.64	0.81
MLP	0.82	0.62	0.87
LSTM	0.82	0.69	0.91
Ours	0.93	0.84	0.93

(c) F1-score

Model	**Accuracy**
RF	0.668
SVM	0.60
DT	0.579
LR	0.76
MLP	0.783
LSTM	0.821
Ours	0.901

(d) Accuracy

Table 4.7: Model performances comparisons (L=3). The proposed Transformer-based model has achieved 90.1% accuracy.

Model/Precision	class0	class1	class2	class3
RF	0.71	0.49	0.50	0.69
SVM	0.59	0.38	0.43	0.58
DT	0.69	0.43	0.41	0.61
LR	0.66	0.44	0.46	0.69
MLP	0.72	0.42	0.37	0.74
LSTM	0.61	0.50	0.59	0.74
Ours	0.94	0.73	0.80	0.87

(a) Presicion

Model/Recall	class0	class1	class2	class3
RF	0.75	0.51	0.48	0.66
SVM	0.54	0.40	0.41	0.63
DT	0.69	0.42	0.43	0.59
LR	0.77	0.35	0.52	0.64
MLP	0.88	0.69	0.05	0.82
LSTM	0.84	0.48	0.42	0.75
Ours	0.88	0.81	0.69	0.95

(b) Recall

Model/F1-score	class0	class1	class2	class3
RF	0.73	0.50	0.49	0.68
SVM	0.56	0.39	0.42	0.60
DT	0.69	0.42	0.42	0.60
LR	0.71	0.39	0.49	0.66
MLP	0.79	0.52	0.09	0.78
LSTM	0.71	0.49	0.49	0.74
Ours	0.91	0.77	0.74	0.91

(c) F1-score

Model	Accuracy
RF	0.597
SVM	0.49
DT	0.532
LR	0.57
MLP	0.595
LSTM	0.614
Ours	0.83

(d) Accuracy

Table 4.8: Model performances comparisons (L=4). Our proposed model outperforms all the baseline models on the accuracy, precision, recall, and f1-score. It has 83% accuracy.

$L = 3$ or $L = 4$, the second-best model is LSTM, which has an accuracy of 82.1 percent, 8 percent less than that of our proposed Transformer based model.

Additionally, when $L = 3$, all of our baseline models have relatively higher precision and recall for predicting high (2) and low (0) areas, but a lower precision and recall in predicting medium (1) areas. The baseline models such as RF, SVM, LR, MLP, have a low precision and recall in predicting medium (1) increase area; LSTM has higher precision, but low recall and F1-score in predicting medium (1) increase area.

As illustrated in Figures 4.9 and 4.10, our proposed Transformer-based model produces balanced prediction outcomes for all classes; for example, when $L = 3$, our model has a precision of 0.80 and a recall of 0.86 for predicting areas with the medium increase rate. Figure 4.8 shows the ROC curve of the model performance when $L = 2$. An AUC value of 0.5 indicates no discrimination, 0.7 to 0.8 indicate acceptable discrimination, 0.8 to 0.9 is considered excellent discrimination, and greater than 0.9 is considered outstanding [86]. When $L = 4$, In Table 4.8, our proposed model outperforms all the baseline models on accuracy, precision, recall, and F1-score. It has an accuracy of 83 percent, which is 21.6 percent higher than the accuracy of the second-best model, LSTM.

Ablation Analysis

This section evaluates the effectiveness of eliminating census data. These tests aim to assess the influence of temporal and census data. The baseline models (RF, SVM, DT, LR) and deep learning models (MLP, LSTM, Transformer) are run on both the temporal data only dataset and the concatenated dataset (temporal and census data). The results are shown in Table 4.9. An intriguing finding is that when only temporal data is used, baseline models such as RF, DT, and MLP have shown an improvement in accuracy. For example, RF achieves a 73.4 percent accuracy, which is 6.6 percent higher than executing on a concatenated dataset containing temporal and census data. The prediction accuracy of models such as DT, SVM, LR, and MLP is lower than that of concatenated datasets. One

Model/Precision	class0	class1	class2
RF	0.77	0.59	0.81
SVM	0.66	0.48	0.65
DT	0.73	0.45	0.78
LR	0.75	0.58	0.75
MLP	0.62	0.51	0.97
LSTM	0.84	0.75	0.92
Ours	0.92	0.76	0.93

(a) Presicion

Model/Recall	class0	class1	class2
RF	0.82	0.53	0.84
SVM	0.57	0.49	0.72
DT	0.73	0.46	0.76
LR	0.81	0.49	0.80
MLP	0.99	0.39	0.72
LSTM	0.93	0.73	0.87
Ours	0.83	0.84	0.92

(b) Recall

Model/F1-score	class0	class1	class2
RF	0.79	0.56	0.82
SVM	0.61	0.48	0.68
DT	0.73	0.46	0.77
LR	0.78	0.53	0.78
MLP	0.76	0.44	0.82
LSTM	0.88	0.74	0.89
Ours	0.87	0.80	0.92

(c) F1-score

Model	Accuracy
RF	0.734
SVM	0.598
DT	0.66
LR	0.71
MLP	0.70
LSTM	0.844
Ours	0.868

(d) Accuracy

Table 4.9: Model performances comparisons temporal data only (L=3).

Our proposed model, DT, SVM, LR, and MLP have lower prediction accuracy than using the concatenated dataset, while RF, DT, and MLP have improved their accuracy.

example is that, LR has a 71 percent accuracy with only temporal data, which is 5 percent less than using the concatenated dataset.

4.1.5 Conclusion

Modeling the real estate market only on the basis of temporal price is exceedingly challenging. A new dataset that incorporates both price history and census data is proposed in this work to better represent real estate dynamics. Also, to encode temporal data, we propose employing a sequential encoder and combining it with static census features. It has been demonstrated experimentally that our technique outperforms the baseline models for price prediction tasks, and the Transformer model structure has been shown to be the most appropriate model structure for temporal data encoding in our problem setting.

4.2 Deep Real Estate Encoding with Location Features

4.2.1 Prediction Model with Location Features

Real estate prices can be affected by multiple features, not limited to temporal features. For example, spatial location play an important role in measuring the local real estate value. In this section, based on the model of the Figure 4.4, the relationship between each real estate locations and the corresponding location features are added and encoded into the model to help the prediction. Here we model the real estate locations with a location graph and do the encoding work based on the the graph embedding techniques [99, 100, 101, 102, 103].

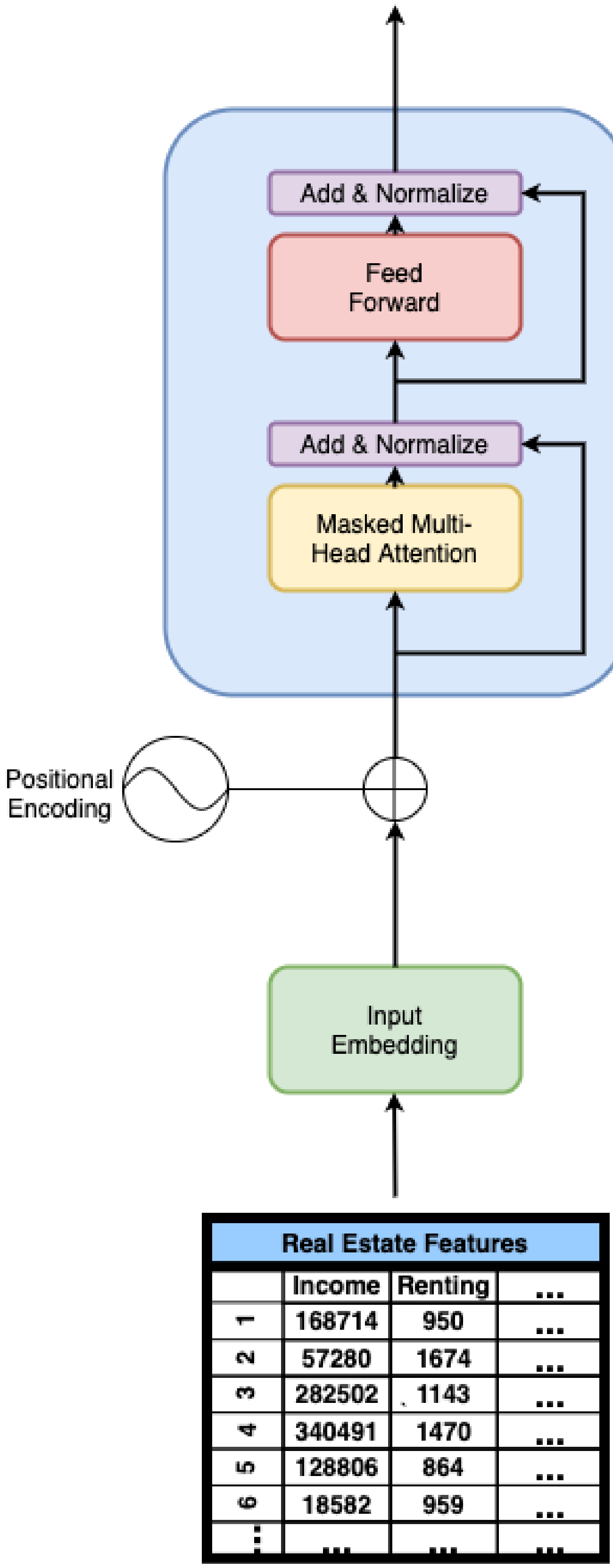
As the Figure 4.11 shows, same as the design in Figure 4.4, the Transformer is used to encode the time-series real estate input for further prediction. To better help the forecast, instead of concatenating the features directly with the Transformer output, here, a Graph Convolutional Network (GCN) [102] is carefully designed and added. The GCN is used to encode the real estate location information and its corresponding location features. We build a graph based on the location information and use the GCN to encode it. Each distinct real estate location is considered as a node in the graph, and the edges in the graph are built if two real estates are in the same location. If two real estates are in the same state, the graph edge between these two nodes is set to one. If two real estates are in the same metro, the edge is set to two. Three is for the same city and four is for the same county. The edge will be 0 if there is no relationship between these two real estates locations. Also, there are location features helping the GCN to encode the location information. After encoding, the output of the GCN is then merged into the latent output vector of the Transformer for further prediction. In the last step, an MLP network is used to complete the prediction of the price of the input.

4.2.2 Experimental Validation

Same as the section 4.1.4, Accuracy, Recall, Precision, F1-score, and Roc are used as the criteria of the performance. We evaluate the model using the number of classes L = 2, compared with the results before. Also, We use the following as the baseline methods: Logistic Regression (LR), Long Short Term Memory (LSTM), Random Forest (RF), Decision Tree (DT), Support Vector Machine (SVM), Multilayer Perceptron (MLP) and our previous Transformer-based model.

As shown in Table 4.10, with adding the location features encoded by GCN, our deep outperforms the traditional machine learning models and our previous Transformer-based model. The previous proposed Transformer-based model has achieved 93.5 percent accuracy when $L = 2$ and it outperforms all of our baseline models. Our new model with the location features beding encoded outperforms the previous Transformer-based model. The model has the performance of 95.4 percent when $L = 2$, which is 1.9 percent higher than the previous result.

Another interesting finding is that, compared with the previous Transformer-based model, our new model could help to balance the prediction result. As the Table 4.10 illustrates, the Precision for class 0 was 0.88 before, which was much lower than the Precision of class 1, which was 0.98. But for our new model with adding the location features helping the prediction, The Precision of class 0 increased to 0.93. The difference between the two classes decreases in the new model. Same in the Recall, The difference between two classes changed from 0.8 to 0.3, which shows the prediction result becomes more balanced, with using the GCN to encode the location relationship and features of the real estate.



	Income	Renting	...
1	168714	950	...
2	57280	1674	...
3	282502	1143	...
4	340491	1470	...
5	128806	864	...
6	18582	959	...
...	...	...	...

Figure 4.5: Transformer Architecture.

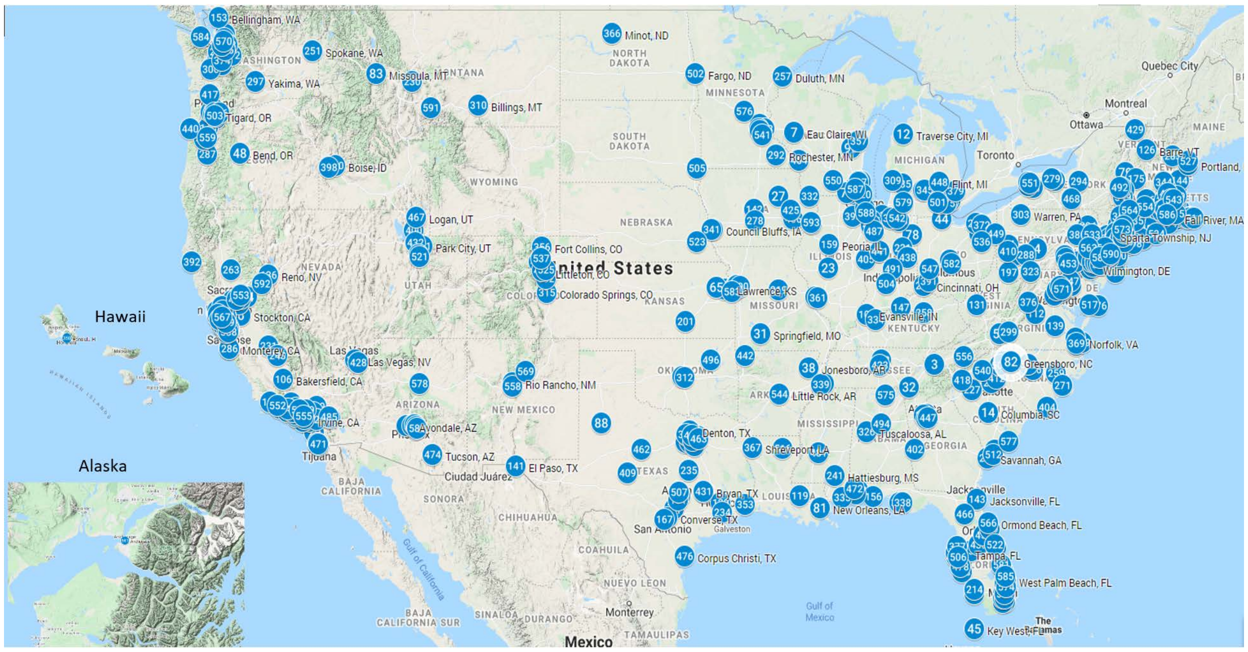


Figure 4.6: Study Area. It consists of 7,436 neighborhoods, 567 cities, 304 counties, 225 metros, and 50 states across the U.S.

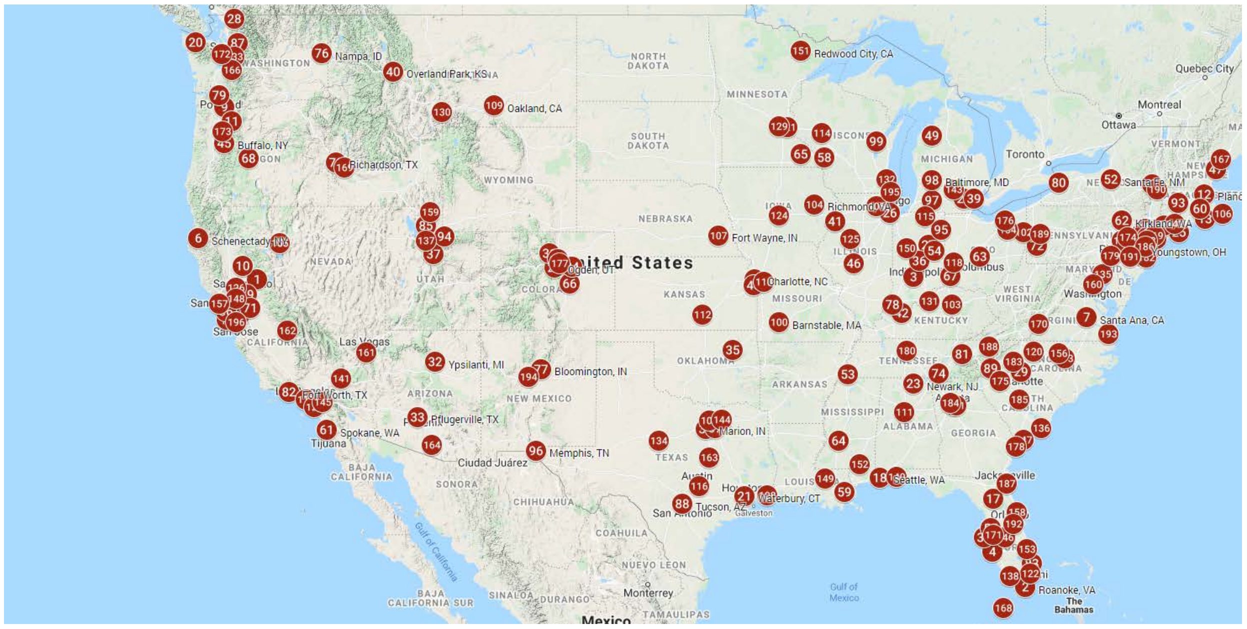


Figure 4.7: Hotspot Areas. It shows the ground truth of the hotspot region in the U.S in 2018.

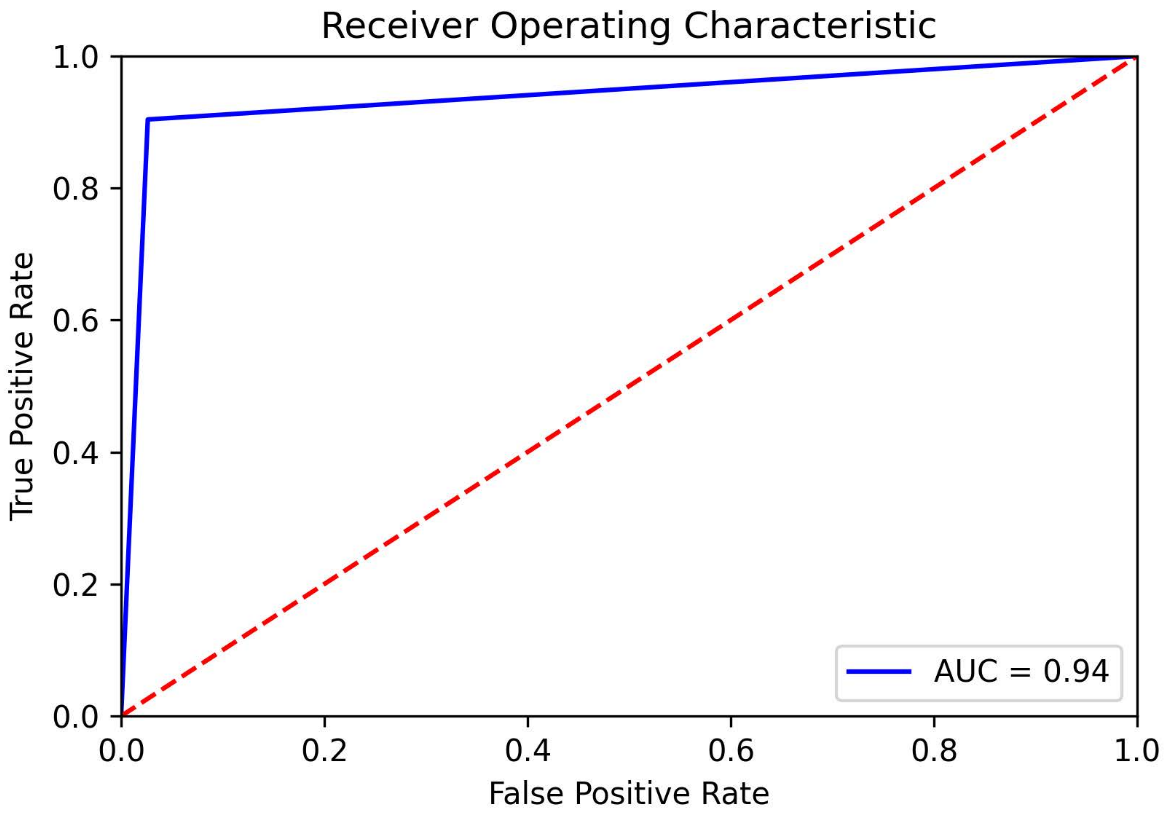


Figure 4.8: ROC curve (L=2). An AUC score of 0.5 suggests no discrimination, and 0.7 to 0.8 is considered acceptable, 0.8 to 0.9 is considered excellent, and more than 0.9 is considered outstanding [86].

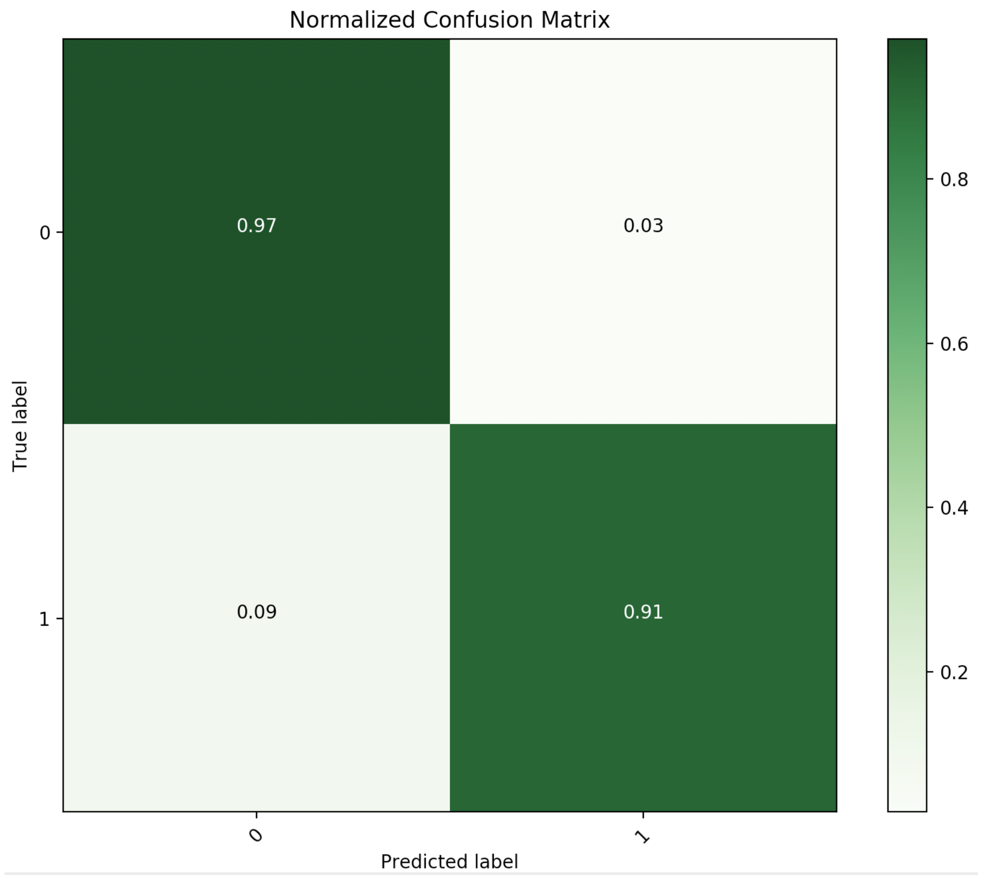


Figure 4.9: Normalized Confusion Matrix (L=2). It shows that our model has a high true positive and true negative rates over both classes, and a low false positive and false negative rates for both classes.

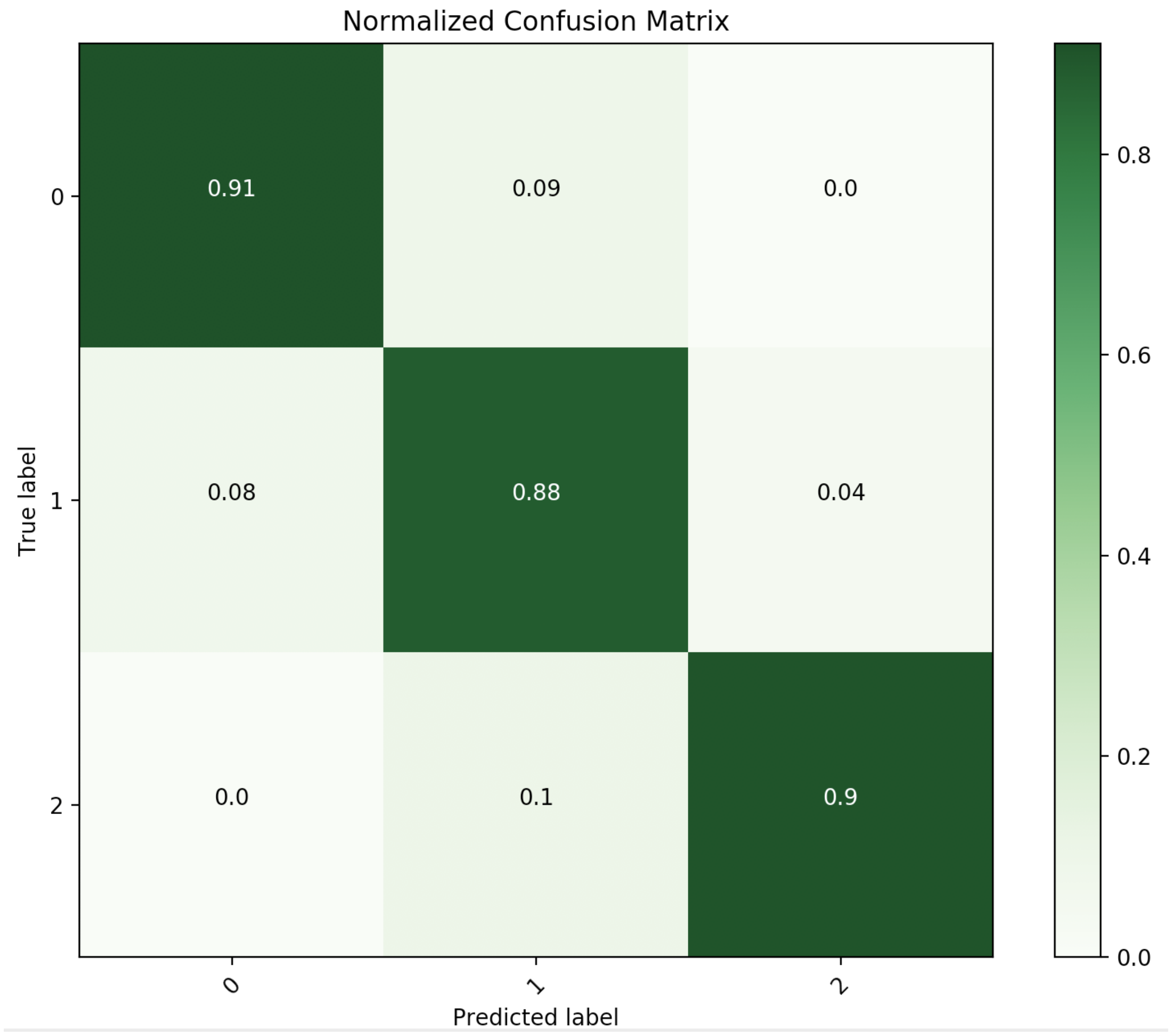
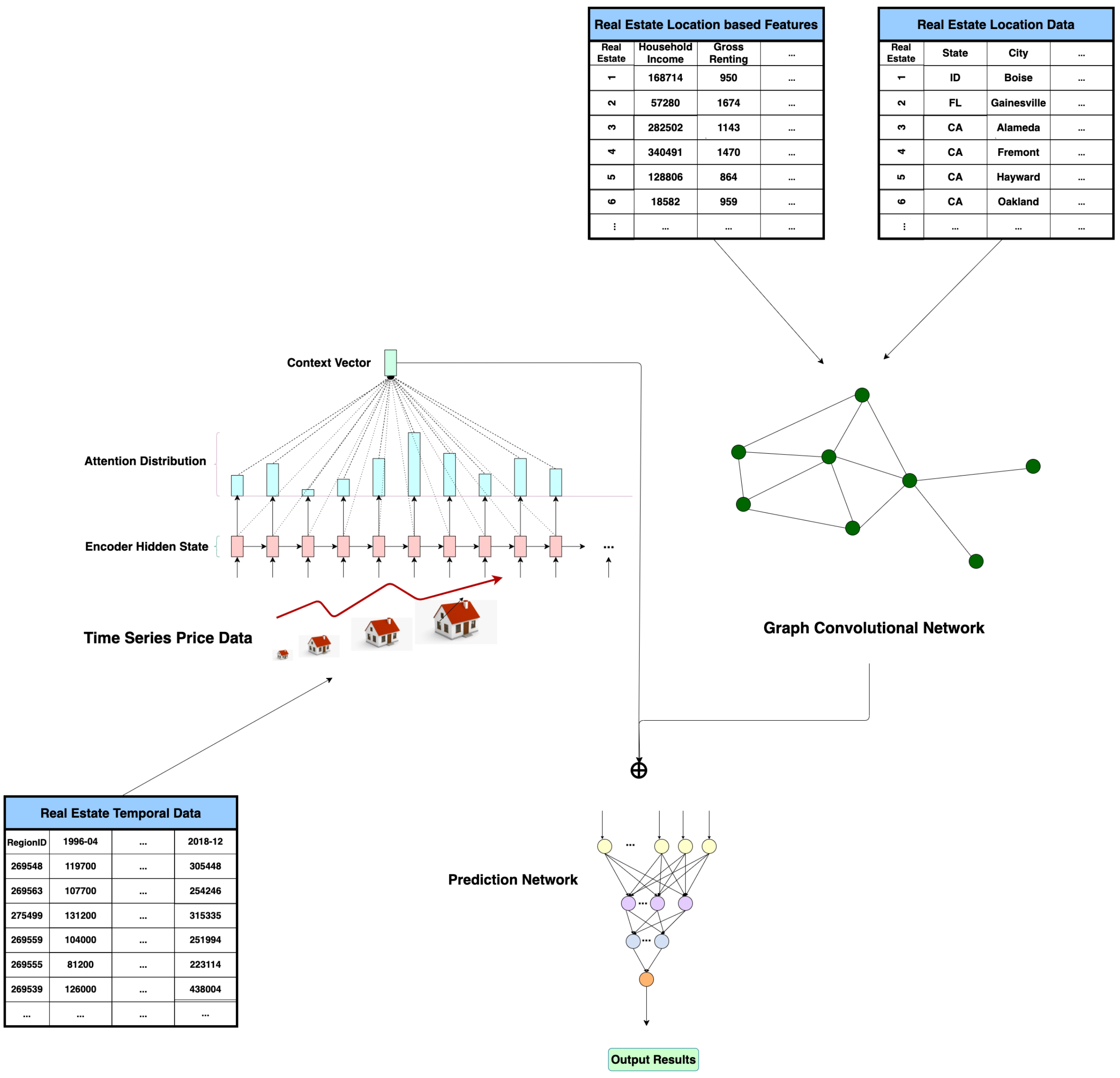


Figure 4.10: Normalized Confusion Matrix (L=3). It shows that our model has a high positive rate over all three classes and a low negative rate for all three classes.



Real Estate Location based Features			
Real Estate	Household Income	Gross Renting	...
1	168714	950	...
2	57280	1674	...
3	282502	1143	...
4	340491	1470	...
5	128806	864	...
6	18582	959	...
...	...	...	...

Real Estate Location Data			
Real Estate	State	City	...
1	ID	Boise	...
2	FL	Gainesville	...
3	CA	Alameda	...
4	CA	Fremont	...
5	CA	Hayward	...
6	CA	Oakland	...
...	...	...	...

Real Estate Temporal Data			
RegionID	1996-04	...	2018-12
269548	119700	...	305448
269563	107700	...	254246
275499	131200	...	315335
269559	104000	...	251994
269555	81200	...	223114
269539	126000	...	438004
...	...	...	...

Figure 4.11: Design flow of the Transformer-GCN-based prediction model.

A Transformer is employed to encode the information of the time-series real estate price data.
A Graph Convolutional Network(GCN) is employed to encode the location information and location features.
Then concatenate the results from GCN and Transformer. By combining the time series data with other features, a MLP model is used for the final prediction.

Model/Precision	**class0**	**class1**
RF	0.79	0.84
SVM	0.77	0.78
DT	0.73	0.77
LR	0.91	0.90
MLP	0.82	0.94
LSTM	0.79	0.96
Ours	0.88	0.98
Ours(+GCN)	0.93	0.97

(a) Presicion

Model/Recall	**class0**	**class1**
RF	0.84	0.79
SVM	0.76	0.78
DT	0.77	0.73
LR	0.90	0.91
MLP	0.93	0.84
LSTM	0.96	0.79
Ours	0.98	0.90
Ours(+GCN)	0.97	0.94

(b) Recall

Model/F1-score	**class0**	**class1**
RF	0.81	0.81
SVM	0.77	0.78
DT	0.75	0.75
LR	0.91	0.90
MLP	0.87	0.89
LSTM	0.86	0.87
Ours	0.93	0.94
Ours(+GCN)	0.95	0.96

(c) F1-score

Model	**Accuracy**	**Roc**
RF	0.81	0.813
SVM	0.772	0.772
DT	0.765	0.754
LR	0.90	0.904
MLP	0.878	0.879
LSTM	0.866	0.874
Ours	0.935	0.941
Ours(+GCN)	0.954	0.955

(d) Accuracy and Roc

Table 4.10: Model performance. The proposed Transformer-GCN-based model has achieved 95.4% accuracy.

Chapter 5
Conclusion and Future Work

5.1 Conclusion

In this thesis, we focus on the deep time series model and its application. We proposed models for the translation of the natural language sequences, and the prediction of the real estate data. The experiment results show our proposed model could achieve high accuracy and outperform all baseline models.

In the first part, we propose an NLIDB applied for the spatial domain to convert natural language queries to structured queries executable by databases. The main contribution of our work is to not only recognize the meaning of the ambiguous spatial phrases based on contextual interpretation but also support a flexible back-end translation to database queries under the deep learning model. Also, a transfer learning approach is proposed to address the problem of translation from spatial language to database queries. Our extensive experimental analysis demonstrates the advantage of our approach over state-of-the-art methods.

In the second part, we focus on the deep learning methods for processing the real estate-related dataset. We propose a large-scale real estate-related dataset for the value prediction task consists of numerical real estate price history data from public dataset, with both static and dynamic features. Also, a carefully designed Transformer-based forecasting model is proposed to encode the temporal data. We experimentally validate that our strategy performs well for price prediction tasks and the Transformer is proven to be the most suitable model structure for temporal data encoding in our problem setting.

5.2 Future Work

- Continue to develop NLP models and transfer learning approaches in order to improve the accuracy of natural language to database query translation, e specially for the translation of natural language to complicated database queries involving joined tables.
- Continue to develop the deep time series model and related machine learning techniques, and expand its application to a broader range of issues.

Bibliography

[1] Rosenblatt, Frank. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Cornell Aeronautical Lab Inc Buffalo NY, 1961.

[2] Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[3] Androutsopoulos, Ion, Graeme D. Ritchie, and Peter Thanisch. "Natural language interfaces to databases-an introduction." arXiv preprint cmp-lg/9503016 (1995).

[4] Brad, Florin, Radu Iacob, Ionel Hosu, and Traian Rebedea. "Dataset for a neural natural language interface for databases (NNLIDB)." arXiv preprint arXiv:1707.03172 (2017).

[5] Utama, Prasetya, Nathaniel Weir, Fuat Basik, Carsten Binnig, Ugur Çetintemel, Benjamin Hättasch, Amir Ilkhechi, Shekar Ramaswamy, and Arif Usta. "An end-to-end neural natural language interface for databases." arXiv preprint arXiv:1804.00401 (2018).

[6] Li, Fei, and Hosagrahar V. Jagadish. "NaLIR: an interactive natural language interface for querying relational databases." In Proceedings of the 2014 ACM SIGMOD international conference on Management of data, pp. 709-712. 2014.

[7] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9, no. 8 (1997): 1735-1780.

[8] Gers, Felix A., Jürgen Schmidhuber, and Fred Cummins. "Learning to forget: Continual prediction with LSTM." Neural computation 12, no. 10 (2000): 2451-2471.

[9] Cho, Kyunghyun, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. "On the properties of neural machine translation: Encoder-decoder approaches." arXiv preprint arXiv:1409.1259 (2014).

[10] Li, Yunyao, Huahai Yang, and H. V. Jagadish. "Nalix: an interactive natural language interface for querying xml." In Proceedings of the 2005 ACM SIGMOD international conference on Management of data, pp. 900-902. 2005.

[11] Popescu, Ana-Maria, Oren Etzioni, and Henry Kautz. "Towards a theory of natural language interfaces to databases." In Proceedings of the 8th international conference on Intelligent user interfaces, pp. 149-157. 2003.

[12] Saha, Diptikalyan, Avrilia Floratou, Karthik Sankaranarayanan, Umar Farooq Minhas, Ashish R. Mittal, and Fatma Özcan. "ATHENA: an ontology-driven system for natural language querying over relational data stores." Proceedings of the VLDB Endowment 9, no. 12 (2016): 1209-1220.

[13] Iyer, Srinivasan, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. "Learning a neural semantic parser from user feedback." arXiv preprint arXiv:1704.08760 (2017).

[14] Wang, Wenlu, Yingtao Tian, Hongyu Xiong, Haixun Wang, and Wei-Shinn Ku. "A transfer-learnable natural language interface for databases." arXiv preprint arXiv:1809.02649 (2018).

[15] Wang, Wenlu. "A cross-domain natural language interface to databases using adversarial text method." Database 1 (2019): q2.

[16] Wang, Wenlu, Yingtao Tian, Haixun Wang, and Wei-Shinn Ku. "A Natural Language Interface for Database: Achieving Transfer-learnability Using Adversarial Method for Question Understanding." In 2020 IEEE 36th International Conference on Data Engineering (ICDE), pp. 97-108. IEEE, 2020.

[17] Wang, Wenlu, Ji Zhang, M-T. Sun, and W-S. Ku. "Efficient parallel spatial skyline evaluation using mapreduce." In Proceedings of the 20th international conference on extending database technology. 2017.

[18] Wang, Wenlu, Ji Zhang, Min-Te Sun, and Wei-Shinn Ku. "A scalable spatial skyline evaluation system utilizing parallel independent region groups." The VLDB Journal 28, no. 1 (2019): 73-98.

[19] Wang, Wenlu, and Wei-Shinn Ku. "Dynamic indoor navigation with bayesian filters." SIGSPATIAL Special 8, no. 3 (2017): 9-10.

[20] Wang, Wenlu, and Wei-Shinn Ku. "Recommendation-based smart indoor navigation." In Proceedings of the Second International Conference on Internet-of-Things Design and Implementation, pp. 311-312. 2017.

[21] Zlatev, Jordan. "Spatial semantics." The Oxford handbook of cognitive linguistics (2007): 318-350.

[22] Yin, Pengcheng, and Graham Neubig. "TRANX: A transition-based neural abstract syntax parser for semantic parsing and code generation." arXiv preprint arXiv:1810.02720 (2018).

[23] Seo, Minjoon, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. "Bidirectional attention flow for machine comprehension." arXiv preprint arXiv:1611.01603 (2016).

[24] Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. "Language models are unsupervised multitask learners." OpenAI blog 1, no. 8 (2019): 9.

[25] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need." arXiv preprint arXiv:1706.03762 (2017).

[26] Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

[27] Radford, Alec, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. "Improving language understanding by generative pre-training (2018)." (2018).

[28] Wang, Shuohang, and Jing Jiang. "Machine comprehension using match-lstm and answer pointer." arXiv preprint arXiv:1608.07905 (2016).

[29] Brown, Tom B., Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan et al. "Language models are few-shot learners." arXiv preprint arXiv:2005.14165 (2020).

[30] Zettlemoyer, Luke S., and Michael Collins. "Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars." arXiv preprint arXiv:1207.1420 (2012).

[31] Peters, Matthew E., Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. "Deep contextualized word representations." arXiv preprint arXiv:1802.05365 (2018).

[32] Zettlemoyer, Luke, and Michael Collins. "Online learning of relaxed CCG grammars for parsing to logical form." In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), pp. 678-687. 2007.

[33] Kwiatkowksi, Tom, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. "Inducing probabilistic CCG grammars from logical form with higher-order unification." In Proceedings of the 2010 conference on empirical methods in natural language processing, pp. 1223-1233. 2010.

[34] Kwiatkowski, Tom, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. "Lexical generalization in CCG grammar induction for semantic parsing." In Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pp. 1512-1523. 2011.

[35] Kwiatkowski, Tom, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. "Scaling semantic parsers with on-the-fly ontology matching." In Proceedings of the 2013 conference on empirical methods in natural language processing, pp. 1545-1556. 2013.

[36] Wang, Adrienne, Tom Kwiatkowski, and Luke Zettlemoyer. "Morpho-syntactic lexical generalization for CCG semantic parsing." In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1284-1295. 2014.

[37] Zhao, Kai, and Liang Huang. "Type-driven incremental semantic parsing with polymorphism." arXiv preprint arXiv:1411.5379 (2014).

[38] Liang, Percy, Michael I. Jordan, and Dan Klein. "Learning dependency-based compositional semantics." Computational Linguistics 39, no. 2 (2013): 389-446.

[39] Zelle, John M., and Raymond J. Mooney. "Learning to parse database queries using inductive logic programming." In Proceedings of the national conference on artificial intelligence, pp. 1050-1055. 1996.

[40] Tang, Lappoon R., and Raymond Mooney. "Automated construction of database interfaces: Intergrating statistical and relational learning for semantic parsing." In 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, pp. 133-141. 2000.

[41] Tang, Lappoon R., and Raymond J. Mooney. "Using multiple clause constructors in inductive logic programming for semantic parsing." In European Conference on Machine Learning, pp. 466-477. Springer, Berlin, Heidelberg, 2001.

[42] Ge, Ruifang, and Raymond Mooney. "A statistical semantic parser that integrates syntax and semantics." In Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005), pp. 9-16. 2005.

[43] Kalchbrenner, Nal, and Phil Blunsom. "Recurrent continuous translation models." In Proceedings of the 2013 conference on empirical methods in natural language processing, pp. 1700-1709. 2013.

[44] Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." arXiv preprint arXiv:1409.3215 (2014).

[45] Cho, Kyunghyun, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." arXiv preprint arXiv:1406.1078 (2014).

[46] Finegan-Dollak, Catherine, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. "Improving text-to-sql evaluation methodology." arXiv preprint arXiv:1806.09029 (2018).

[47] Rabinovich, Maxim, Mitchell Stern, and Dan Klein. "Abstract syntax networks for code generation and semantic parsing." arXiv preprint arXiv:1704.07535 (2017).

[48] Dong, Li, and Mirella Lapata. "Language to logical form with neural attention." arXiv preprint arXiv:1601.01280 (2016).

[49] Jia, Robin, and Percy Liang. "Data recombination for neural semantic parsing." arXiv preprint arXiv:1606.03622 (2016).

[50] Fan, Xing, Emilio Monti, Lambert Mathias, and Markus Dreyer. "Transfer learning for neural semantic parsing." arXiv preprint arXiv:1706.04326 (2017).

[51] Susanto, Raymond Hendy, and Wei Lu. "Neural architectures for multilingual semantic parsing." In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pp. 38-44. 2017.

[52] Herzig, Jonathan, and Jonathan Berant. "Neural semantic parsing over multiple knowledge-bases." arXiv preprint arXiv:1702.01569 (2017).

[53] Zhong, Victor, Caiming Xiong, and Richard Socher. "Seq2sql: Generating structured queries from natural language using reinforcement learning." arXiv preprint arXiv:1709.00103 (2017).

[54] Shen, Qijun, Xueying Zhang, and Wenming Jiang. "Annotation of spatial relations in natural language." In 2009 International Conference on Environmental Science and Information Application Technology, vol. 3, pp. 418-421. IEEE, 2009.

[55] Bateman, John A., Joana Hois, Robert Ross, and Thora Tenbrink. "A linguistic ontology of space for natural language processing." Artificial Intelligence 174, no. 14 (2010): 1027-1071.

[56] Kordjamshidi, Parisa, Paolo Frasconi, Martijn Van Otterlo, Marie-Francine Moens, and Luc De Raedt. "Relational learning for spatial relation extraction from natural language." In International Conference on Inductive Logic Programming, pp. 204-220. Springer, Berlin, Heidelberg, 2011.

[57] Khan, Arbaz, Maria Vasardani, and Stephan Winter. "Extracting Spatial Information From Place Descriptions." (2013).

[58] Ramalho, Tiago, Tomáš Kočiský, Frederic Besse, S. M. Eslami, Gábor Melis, Fabio Viola, Phil Blunsom, and Karl Moritz Hermann. "Encoding spatial relations from natural language." arXiv preprint arXiv:1807.01670 (2018).

[59] Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. "Glove: Global vectors for word representation." In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp. 1532-1543. 2014.

[60] Wang, Yushi, Jonathan Berant, and Percy Liang. "Building a semantic parser overnight." In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 1332-1342. 2015.

[61] He, Pengcheng, Yi Mao, Kaushik Chakrabarti, and Weizhu Chen. "X-SQL: reinforce schema representation with context." arXiv preprint arXiv:1908.08113 (2019).

[62] Wang, Bailin, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. "Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers." arXiv preprint arXiv:1911.04942 (2019).

[63] Johnson, Melvin, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat et al. "Google's multilingual neural machine translation system: Enabling zero-shot translation." Transactions of the Association for Computational Linguistics 5 (2017): 339-351.

[64] Alfiyatin, Adyan Nur, Ruth Ema Febrita, Hilman Taufiq, and Wayan Firdaus Mahmudy. "Modeling house price prediction using regression analysis and particle swarm optimization." International Journal of Advanced Computer Science and Applications 8, no. 10 (2017): 323-326.

[65] Almeida, Luis B. "C1. 2 Multilayer perceptrons." Handbook of Neural Computation C 1 (1997).

[66] Anari, Ali, and James Kolari. "House prices and inflation." Real Estate Economics 30, no. 1 (2002): 67-84.

[67] Borovykh, Anastasia, Sander Bohte, and Cornelis W. Oosterlee. "Conditional time series forecasting with convolutional neural networks." arXiv preprint arXiv:1703.04691 (2017).

[68] Boser, Bernhard E., Isabelle M. Guyon, and Vladimir N. Vapnik. "A training algorithm for optimal margin classifiers." In Proceedings of the fifth annual workshop on Computational learning theory, pp. 144-152. 1992.

[69] Cao, Lijuan, and Francis EH Tay. "Financial forecasting using support vector machines." Neural Computing Applications 10, no. 2 (2001): 184-192.

[70] Chung, Junyoung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. "Empirical evaluation of gated recurrent neural networks on sequence modeling." arXiv preprint arXiv:1412.3555 (2014).

[71] Crawford, Gordon W., and Michael C. Fratantoni. "Assessing the forecasting performance of regime-switching, ARIMA and GARCH models of house prices." Real Estate Economics 31, no. 2 (2003): 223-243.

[72] De Gooijer, Jan G., and Rob J. Hyndman. "25 years of time series forecasting." International journal of forecasting 22, no. 3 (2006): 443-473.

[73] Fu, Yanjie, Yong Ge, Yu Zheng, Zijun Yao, Yanchi Liu, Hui Xiong, and Jing Yuan. "Sparse real estate ranking with online user reviews and offline moving behaviors." In 2014 IEEE International Conference on Data Mining, pp. 120-129. IEEE, 2014.

[74] Fu, Yanjie, Guannan Liu, Spiros Papadimitriou, Hui Xiong, Yong Ge, Hengshu Zhu, and Chen Zhu. "Real estate ranking via mixed land-use latent models." In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 299-308. 2015.

[75] Fu, Yanjie, Hui Xiong, Yong Ge, Zijun Yao, Yu Zheng, and Zhi-Hua Zhou. ”Exploiting geographic dependencies for real estate appraisal: A mutual perspective of ranking and clustering.” In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 1047-1056. 2014.

[76] Goodman, Allen C., and Thomas G. Thibodeau. ”Housing market segmentation and hedonic prediction accuracy.” Journal of Housing Economics 12, no. 3 (2003): 181-201.

[77] Guirguis, Hany S., Christos I. Giannikos, and Randy I. Anderson. ”The US housing market: Asset pricing forecasts using time varying coefficients.” The Journal of real estate finance and economics 30, no. 1 (2005): 33-53.

[78] Hall, Stephen, Zacharias Psaradakis, and Martin Sola. ”Switching error-correction models of house prices in the United Kingdom.” Economic Modelling 14, no. 4 (1997): 517-527.

[79] Hamilton, James Douglas. Time series analysis. Princeton university press, 2020.

[80] Ho, Tin Kam. ”Random decision forests.” In Proceedings of 3rd international conference on document analysis and recognition, vol. 1, pp. 278-282. IEEE, 1995.

[81] Kayacan, Erdal, Baris Ulutas, and Okyay Kaynak. ”Grey system theory-based models in time series prediction.” Expert systems with applications 37, no. 2 (2010): 1784-1789.

[82] KK Lai et al.2005. Crude oil price forecasting with TEI@ I methodology.Journalof Systems Science Complexity18, 2 (2005), 145–166.

[83] Lee, Kanghyeok, Hanbeen Kim, and Do Hyoung Shin. ”Forecasting short-term housing transaction volumes using time-series and internet search queries.” KSCE Journal of Civil Engineering 23, no. 6 (2019): 2409-2416.

[84] Limsombunchai, Visit. ”House price prediction: hedonic price model vs. artificial neural network.” In New Zealand agricultural and resource economics society conference, pp. 25-26. 2004.

[85] Loh, Wei-Yin. ”Classification and regression trees.” Wiley interdisciplinary reviews: data mining and knowledge discovery 1, no. 1 (2011): 14-23.

[86] Mandrekar, Jayawant N. ”Receiver operating characteristic curve in diagnostic test assessment.” Journal of Thoracic Oncology 5, no. 9 (2010): 1315-1316.

[87] Manjula, Raja, Shubham Jain, Sharad Srivastava, and Pranav Rajiv Kher. ”Real estate value prediction using multivariate regression models.” In IOP Conference Series: Materials Science and Engineering, vol. 263, no. 4, p. 042098. IOP Publishing, 2017.

[88] McNally, Sean, Jason Roche, and Simon Caton. ”Predicting the price of bitcoin using machine learning.” In 2018 26th euromicro international conference on parallel, distributed and network-based processing (PDP), pp. 339-343. IEEE, 2018.

[89] Nelder, John Ashworth, and Robert WM Wedderburn. ”Generalized linear models.” Journal of the Royal Statistical Society: Series A (General) 135, no. 3 (1972): 370-384.

[90] Park, Byeonghwa, and Jae Kwon Bae. ”Using machine learning algorithms for housing price prediction: The case of Fairfax County, Virginia housing data.” Expert systems with applications 42, no. 6 (2015): 2928-2934.

[91] Potepan, Michael J. ”Explaining intermetropolitan variation in housing prices, rents and land prices.” Real Estate Economics 24, no. 2 (1996): 219-245.

[92] Sampathkumar, V., M. Helen Santhi, and J. Vanjinathan. ”Forecasting the land price using statistical and neural network software.” Procedia Computer Science 57 (2015): 112-121.

[93] Tay, Francis EH, and Lijuan Cao. ”Application of support vector machines in financial time series forecasting.” omega 29, no. 4 (2001): 309-317.

[94] WANG, Jing, and Peng TIAN. ”Real Estate Price Indices Forecast by Using Wavelet Neural Network.” Computer Simulation 2 (2005).

[95] Yan, Yan, Xu Wei, Bu Hui, Song Yang, Wen ZHANG, Y. U. A. N. Hong, and Shou-yang WANG. ”Method for housing price forecasting based on TEI@ I methodology.” Systems Engineering-Theory Practice 27, no. 7 (2007): 1-9.

[96] Yang, Nan, and L. C. Xing. ”Application of Grey-Markov model on the prediction of housing price index.” In Statistics Information Forum, vol. 25, no. 9, pp. 65-67. 2006.

[97] Zhang, G. Peter. ”Time series forecasting using a hybrid ARIMA and neural network model.” Neurocomputing 50 (2003): 159-175.

[98] Zweig, Mark H., and Gregory Campbell. ”Receiver-operating characteristic (ROC) plots: a fundamental evaluation tool in clinical medicine.” Clinical chemistry 39, no. 4 (1993): 561-577.

[99] Perozzi, Bryan, Rami Al-Rfou, and Steven Skiena. ”Deepwalk: Online learning of social representations.” In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 701-710. 2014.

[100] Grover, Aditya, and Jure Leskovec. ”node2vec: Scalable feature learning for networks.” In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 855-864. 2016.

[101] Wang, Daixin, Peng Cui, and Wenwu Zhu. ”Structural deep network embedding.” In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 1225-1234. 2016.

[102] Kipf, Thomas N., and Max Welling. ”Semi-supervised classification with graph convolutional networks.” arXiv preprint arXiv:1609.02907 (2016).

[103] Hamilton, William L., Rex Ying, and Jure Leskovec. "Inductive representation learning on large graphs." In Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 1025-1035. 2017.