



**THE UNIVERSITY
OF QUEENSLAND**
A U S T R A L I A

**The quality of knowledge graph is
improved by Named Entity
Recognition and Link Prediction**

by

Hang Pan

School of Information Technology and Electrical Engineering,
University of Queensland.

Submitted for the degree of
Master of computer science(pass).

June & 2022

Hang Pan
Hang.pan@uqconnect.edu.au

June 8, 2022

Prof Michael Bruenig
Acting Head of School
School of Information Technology and Electrical Engineering
The University of Queensland
St Lucia QLD 4072

Dear Professor Michael Bruenig,

In accordance with the requirements of the Degree of Master of Computer Science in the School of Information Technology and Electrical Engineering, I submit the following thesis entitled

“The quality of knowledge graph is improved by Named Entity Recognition and Link Prediction”

The thesis was performed under the supervision of Professor Xue Li. I declare that the work submitted in the thesis is my own, except as acknowledged in the text and footnotes, and that it has not previously been submitted for a degree at the University of Queensland or any other institution.

Yours sincerely

Hang Pan

Hang Pan

To . . .

Acknowledgments

I would like to thank my esteemed supervisor – Prof. Xue Li and Ms.Chunzi Lyu for their invaluable supervision, support and tutelage during the course of my Master degree. This field is a brand new knowledge for me. Professor Li Xue always answers our doubts and academic questions. Due to the epidemic, I was unable to attend the University of Queensland, Australia, and I chose to study online, overcoming all kinds of difficulties. Teaching assistants and professors communicated with us online every week, which gave me more understanding and in-depth research on knowledge graphs and deep learning, and finally guided me to complete my graduation thesis.

Abstract

In recent years, knowledge graphs have become increasingly popular, and research into knowledge graph technology is an essential component of artificial intelligence. Build and improve base with semantic processing and open connectivity capabilities, as well as intelligent information services like search, question and answer, and tailored suggestion. There are several types of knowledge graphs available today, including knowledge graphs with open connectivity capabilities: YAGO[1], WN18RR[2] and so on. Novels, movies, and music knowledge graphs are all examples of knowledge graphs, and these knowledge networks are frequently linked to downstream applications. We serve individuals with high-quality, consistent services when we work together. However, developers have long struggled with how to increase the quality of knowledge graphs. To assure and improve the quality of the knowledge graph, this paper will build a movie knowledge graph and provide ways to improve entity recognition from unstructured data and also link prediction. However, knowledge graphs frequently suffer from the problem of missing linkages, which restricts their use in related downstream activities. The knowledge graph completion job was created to address this issue. The goal of knowledge graph completion is to derive new facts from current facts in the knowledge graph in order to make it more complete. The last section highlights the entity extraction and connection prediction model, which may help enhance the quality of movie knowledge graphs while developing them. Then, as an application, we build the movie recommendation system.

Key word: knowledge graph, NLP, entities extraction, link prediction ,deep learning, recommendation system

Contents

Table of Contents

Acknowledgments	6
Abstract	7
Key word: knowledge graph, NLP, entities extraction, link prediction ,deep learning, recommendation system	7
Contents	8
List of Figures	10
List of Tables	11
Chapter 1	1
Introduction	1
Chapter 2	6
Relative work	6
2.1 NER	6
2.2 Graph embedding for Link prediction	10
Chapter 3	13
Background	13
3.1 Entity extraction	13
3.2 Word Representations	15
3.3 Models for NER introduction	16
Table 1: Confusion matrix	25
Chapter 4	30
Methodology	30
4.1 Data sets:	31
4.2 Data pre-processing:	32
4.3 Network architectures	35
4.4 Graph embedding to link prediction	37

Chapter 5	40
Results and discussion	40
Chapter 6	43
Application	43
6.1 Movies Recommendation System	43
Chapter 7	45
Conclusions	45
7.1 Summary and conclusions	45
7.2 Possible future work	45
Bibliography	13

List of Figures

4.1	Experimental two-way active crossover (op-amp version).....	5
4.2	Modeling a discrete-time LTI system using z -transforms.....	5

List of Tables

5.1	<i>Fraction of air volume involved in heat exchange for second mode (right column) vs. filling factor (left column). The plain-text headings represent f, m, μ_2 and f_2.</i>	6
-----	---	---

Chapter 1

Introduction

The following is the official Wikipedia definition of a knowledge graph: A knowledge graph is a knowledge base that Google utilises to improve the search engine's functioning[3]. In essence, a knowledge graph is a semantic network that shows links between elements and may be used to formally describe real-world objects and their interactions. The term "knowledge graph" is currently used to describe a wide range of large-scale information systems. Triples, for example, are a generic representation of knowledge graphs: $G = (E, R, S)$, like $E = \{e_1, e_2, e_3, \dots, e_{|E|}\}$ is a knowledge base collection of entities that contains $|E|$ various entities; $R = \{r_1, r_2, r_3, \dots, r_{|R|}\}$ is a collection of relationships in the knowledge base that contains $|R|$ distinct relationships. A collection of triples in the knowledge base is represented as $S \in E \times R \times E$.

Entity 1, Relation, Entity 2, idea, attribute, attribute value, and so on are the most fundamental kinds of triples. The knowledge graph's most fundamental part is the entity, and various entities have different relationships. The term "concept" refers to a group, category, object kind, or type of entity, such as people, geography, and so on. The term "attribute" is used to describe the traits, features, characteristics, characteristics, and parameters that an item may have, such as nationality, birthday, and so on. The value of an object's given attribute is referred to as attribute value. The knowledge graph may also be separated into two types based on coverage: general knowledge graph and industry knowledge graph[4]. The general knowledge graph stresses the integration of additional items and focuses on their breadth. It is difficult to standardize its entities, attributes, and relations among entities with the support ability of ontology library to axioms, rules, and constraints when compared to the industrial knowledge graph, and it is impacted

by the breadth of ideas. In intelligent search and other domains, a general knowledge graph should be employed. The industrial knowledge graph is frequently built using data from a given industry and has industry-specific importance. Entity properties and data patterns are frequently abundant in the industry knowledge graph, and diverse business scenarios and users must be addressed[4].

In recent years, knowledge graphs have grown in size and complexity due to their widespread use in knowledge discovery. Numerous knowledge graphs have been constructed utilizing automated building methods and crowdsourcing. The graph may have a substantial number of syntactic and semantic mistakes, which negatively affect its quality. A knowledge network of low quality results in applications of low quality. Consequently, measuring the quality of a knowledge graph is required for the development of high-quality applications[5].

Data management depends on both the amount and quality of data. As the era of big data approaches, more people are focusing on data, processing and analyzing huge amounts of heterogeneous data at high speeds. For example, they promote building a knowledge base from heterogeneous data using a query optimization algorithm, but they don't take data quality seriously. Real-world data has inconsistencies, errors, incompleteness, obsolescence, duplication, etc. If data quality cannot be adequately evaluated and faults detected cannot be rectified and fixed, accurate answers cannot be achieved and the data cannot be successfully used. Big data requires data quality study[6].

To evaluate data quality, a set of quality dimensions and measurement methodologies must be determined. However, there is no uniform dimension standard for knowledge graph quality, and various downstream activities and data sets have distinct quality needs. Data quality evaluation is "fit for use" if it's appropriate to specified activities; technically, it's "free of defects," meaning the

computer can't identify faults. Distinct works have different quality aspects, such as user interaction and representation, but data has five: accuracy, consistency, integrity, timeliness, and repeatability. Often, various quality dimensions necessitate trade-offs in practical application[6].

Recurrent neural network (RNN) and recurrent neural Network (RNN) Recurrent neural network is a kind of neural network used for processing sequence data, which can learn nonlinear features of sequence with high efficiency[7]. It can handle larger sequences and most recurrent neural networks can handle variable length sequences. Recursive neural networks are a form of computational graph that is an extension of recurrent neural networks. They are built as deep trees rather than chains of recurrent neural networks. When each parent node is connected to only one child node, recursive neural networks are identical to fully connected recurrent neural networks. In neural networks where the input includes data structures, such as natural language processing and computer vision, the potential use of recursive neural networks for learning inferences has been effectively used. Bidirectional RNN (bidirectional cyclic neural network), LSTM, and other traditional models are examples. Language modeling and text production, machine translation, speech recognition, and so on are examples of typical uses. An autoencoder is a neural network that may be taught to replicate input to output. H is a hidden layer in the autoencoder that may create encoded representations of input. The network is made up of two parts: an encoder (represented by the function $h = f(x)$) and a decoder (represented by the function $r = g(h)$) that generates reconstruction. Typically, we will not send the encoder created as input to the output precisely equivalent, but rather, we will place certain limitations on the encoder, so that it can only approximate to copy, and can only copy similar to the training data input. The constraint drives the model to consider the input data which sections need to be copied first, so it can learn the relevant aspects of data quickly.

A graph is a common data format that may be found in a wide range of real-world

situations. Effective graph analytics gives customers a better knowledge of what's underlying the data, which may help with node categorization, node suggestion, connection prediction, and other applications. Most graph analytics approaches, on the other hand, have a significant computational and storage cost. Graph embedding is a simple yet effective method of tackling the graph analytics challenge. It translates graph data into a low-dimensional space that preserves as much graph information about the structure and graph attributes as possible[8].

Data was represented in knowledge bases (KBs) as a directed graph with indicated edges (relationships) between nodes (entities). Natural redundancy between recorded relationships frequently allows missing knowledge base items to be filled in. For example, not all entities record the CountryOfBirth relationship, but it may be simply inferred if the CityOfBirth relationship is known. The purpose of link prediction is to automatically recognize this trend. The combination of the two facts that "isBornIn" (John, Athens) and "isLocatedIn" (Athens, Greece) does not always suggest that "HasNationality," despite the fact that John was born in Athens and that Athens is located in Greece (John,Greece)[9]. As a result, other facts affecting these connections or entities must be handled probabilistically. In this work, I will utilize a graph embedding model to forecast relationships, which will result in the completeness of a knowledge map.

Neo4j is a NOSQL database that runs on the JVM. Its model is straightforward and expressive, matching closely to your whiteboard domain model as the leading graph database[10]. Neo4j is hundreds of times quicker than relational databases for densely linked data, making it excellent for managing complex data across a wide range of fields, from financial to social, telecommunications to GIS.

The main movies datasets source from Wikipedia. Wikipedia is a compendium of the world's knowledge[11].It also provides a data crawl API, which makes it easy for experimenters to get the data and types they want. And the wikipedia data is

reliable and interpretative.

In general, in this paper, we need to improve the accuracy of entity recognition and link prediction. We will use NLP, deep learning and some statistical methods to improve the score of entity extraction model and apply graph embedding to do link prediction, and use Neo4j to visualize the knowledge graph and do a movies recommendation system , so as to improve the data quality of the knowledge graph.

Chapter 2

Relative work

2.1 NER

The fundamental goal of named entity recognition (NER) is to recognize entity terms in a phrase and their associated entity types[12]. It is, for example, the NER task in several cases in the following example. The sorts of entities that need to be recognized vary depending on the circumstance.

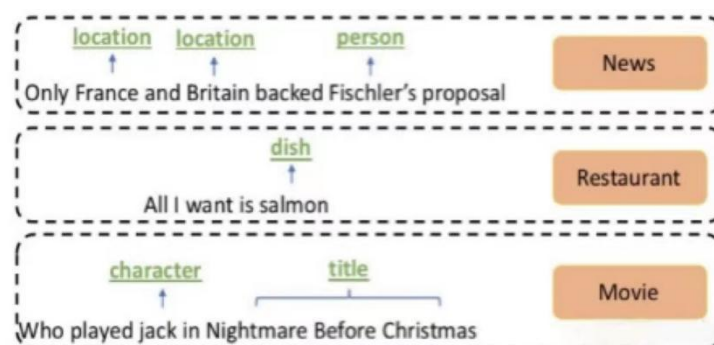


Figure 1.Example of NER processing

The basic solution of the NER task is regarded as a sequence classification task. Generally, annotation prediction methods such as BIO and BIOES are used. Here, the BIO annotation method is mainly introduced. The BIO labeling method is to label each word in the sentence with a label. This label consists of two parts: one part is the location of the entity to which the word belongs, where B indicates that the word is the first word of the entity, and I indicates the word. It is the middle word of the entity, and O means not an entity; the other part is the entity type corresponding to the word. For example, in the above News type NER task, it is necessary to predict whether the word belongs to location or person. Therefore, finally each word is annotated in the form of BIO + Entity type, which is a text sequence classification task[13].

Xiang Xiaowen et al. published a Chinese named entity recognition system in 2005 that combined statistics and rules. The entire recognition process is separated

into two phases: first, the hidden Markov model is utilized for part-of-speech tagging, and then matching rules with priority level are employed to change and convert the first step's findings[14]. The system also makes a rudimentary attempt at context-dependent named entity recognition at the same time. The accuracy rate, recall rate, and F value of the system in the evaluation of named entity recognition of 863 organizations were 81.93 %, 78.20 %, and 80.02 %, respectively [15].

For the TREC 2009 English product named Entity (EPNE) identification problem, Zhang Chao-sheng et al. developed an English product named entity recognition technique based on the Conditional Random Field model (CRF) in 2010[16]. In conditional random fields, the technique uses word as the granularity of segmentation, makes extensive use of context and English product name specific indicator information as classification features, and models with a manually built brand word list. The procedure produces good outcomes, according to the findings of the experiments[17].

In 2002, Carreras X et al. address two issues in the CoNLL2002 competition, Named Entity Extraction (NEE) and Named Entity Classification (NEC), using machine learning-based models. Specifically, the Binary AdaBoost classifier. On the NER problem, the simplest BIO approach gets the best results. Regarding overall system performance, Spanish outperforms Dutch. This is presumably due to the unpredictability of Dutch statistics[18].

This research proposes a classifier-combination experimental paradigm for named entity identification in which four distinct classifiers (robust linear classifier, maximum entropy, transformation-based learning, and hidden Markov model) are merged under various scenarios. On the identical English corpus job, it is 17 to 21 percent superior to earlier studies. And presented The Robust Risk Minimization Classifier (RRM) approach, which is based on deep learning[19].

For example, Ronan Collobert et al. suggested a multilayer neural network design to solve NLP tasks such as part-of-speech tagging, chunking, named entity identification, and semantic role tagging, and it improved processing speed and accuracy. Collobert et al use a CNN with a CRF layer on top of a succession of word embeddings. This may be regarded of as our first model sans character-level embeddings and a CNN in place of the bidirectional LSTM[20]. The framework of multilayer neural network architecture as follow:

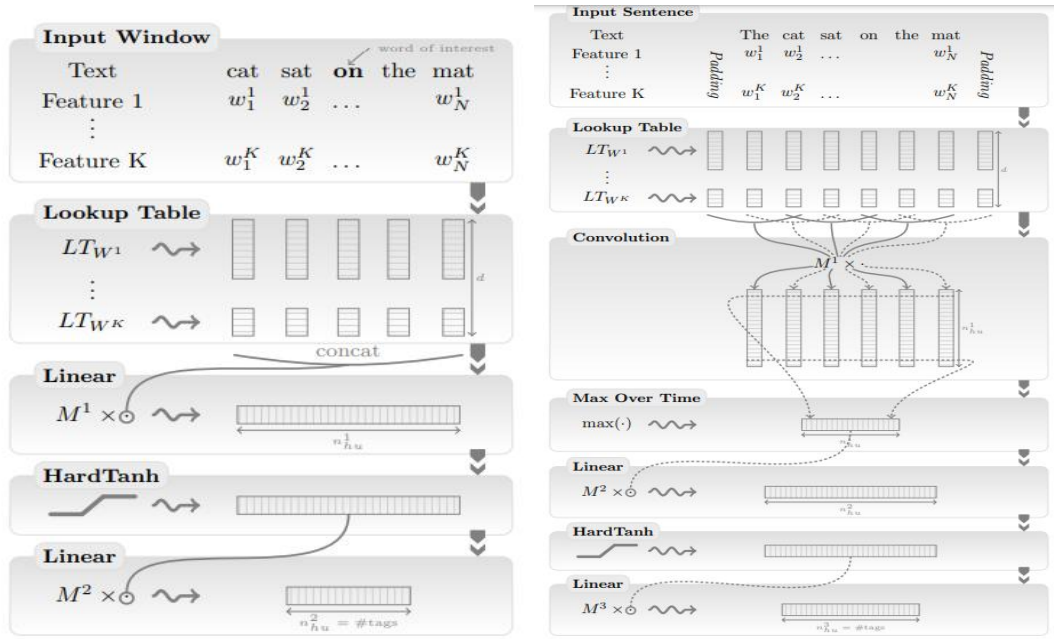


Figure2: Window approach network and Sentence approach network.

The Window approach network and the Sentence approach network are used in this architecture. Every word's characteristics are extracted throughout the first layer. The second layer retrieves characteristics from a window of words or the entire phrase as a sequence with local and global structure (i.e., it is not treated like a bag of words). The layers that follow are conventional NN layers. The architecture learns various layers of feature extraction that process the inputs based on the input text. Back-propagation is used to automatically train the features calculated by the deep layers of the network to be relevant to the job[20].

Zhiheng Huang et al. proposed a number of sequence tagging models based on Long Short-Term Memory (LSTM) in 2015. Bidirectional LSTM (BI-LSTM),

LSTM with a Conditional Random Field (CRF) layer, and Bi-LSTM-CRF are some of the techniques used. These methods are based on the LSTM algorithm. They also demonstrated that these models were effective[21].

Ekbal A et al. discuss the construction of a Bengali Named Entity Recognition (NER) system employing statistical Conditional Random Fields in 2008. (CRFs). The algorithm uses diverse contextual information from the words, as well as a range of attributes, to predict various named entity (NE) classes. The method was built using a piece of a partly NE tagged Bengali news corpus derived from an online archive of a popular Bengali daily. A NE tagset of seventeen tags has been manually annotated on the training set, which contains 150K words. The proposed CRF-based NER system performs well in the 10-fold cross validation test, with average Recall, Precision, and F-score values of 93.8 %, 87.8%, and 90.7 %, respectively[22].

Pretrained word embeddings are used to establish our lookup table, same as Collobert et al. (2011) did. Pretrained word embeddings outperform randomly initialized ones by a substantial margin. Skip-n-gram (Ling et al., 2015a)[24], a version of word2vec (Mikolov et al., 2013a) that accounts for word order, is used to pretrain embeddings[25]. During training, these embeddings are fine-tuned. In this experiment, we also apply Word embedding, which improves the model's training and prediction effects.

We present a range of neural network-based models for sequence tagging in this research. LSTM networks, bidirectional LSTM networks (BI-LSTM), and bidirectional LSTM networks with a CRF layer are among the models available (BILSTM-CRF). The following is a list of our contributions. 1) We assess the performance of the aforementioned models on NLP tagging data sets in a systematic way; 2) This is the first time a bidirectional LSTM+CRF (dubbed BI-LSTM-CRF) model has been applied to NLP benchmark sequence tagging data

sets. Thanks to a bidirectional LSTM component, this model can exploit both past and future input properties. In addition, owing to a CRF layer, this model may employ sentence level tag information. On POS, chunking, and NER data sets, our model can achieve state-of-the-art (or close to) accuracy; 3) We show that the BI-LSTM-CRF model is resilient and has less reliance on word embedding than prior observations (Collobert et al., 2011). It is capable of producing accurate tagging results without the use of word embedding.

2.2 Graph embedding for Link prediction

Three phases are commonly involved in the creation of a knowledge graph embedding model: 1) Define the representation of entities and connections; 2) Define the scoring function to assess the logic of the triple combination; and 3) Train and learn the embedded representation of things and relationships [26]. The higher the value of the scoring function, the more logical the triad is, and hence the more probable it is to be accurate. The optimization aim while training the embedded representation of learning entities and relations is to get the score of existing triples in the knowledge graph to be as high as feasible compared to non-existent triples. Knowledge graph embedding models may be loosely separated into distance-based model, bi-linear model, and neural network model depending on the defining form of scoring function[27].

Bishan Yang et al. suggested utilizing the neural-embedding technique to learn representations of items and relations in knowledge graphs in 2015[28]. They demonstrate that most existing models, such as NTN (Socher et al., 2013)[29] and TransE (Bordes et al., 2013b)[30], can be generalized using a unified learning framework in which entities are low-dimensional vectors learned from a neural network and relations are bilinear and/or linear mapping functions. They showed that using a simple bilinear formula, they were able to attain new state-of-the-art outcomes for the challenge (improved accuracy by nearly 20 percent on Freebase).

They also proposed a new method for mining logical rules based on learnt relational embedding[28].

Johannes Welbl introduces complex space into knowledge graph embedding. Conjugate vectors in complex space allow the traditional dot product to be used in asymmetric relations. DistMult does a good job of predicting symmetric relationships, but not modeling asymmetric ones. The TransE model does not perform well for both symmetric and asymmetric relationships. The ComlEX model was largely inspired by the DistMult model with a few improvements, but one that was subtle[31].

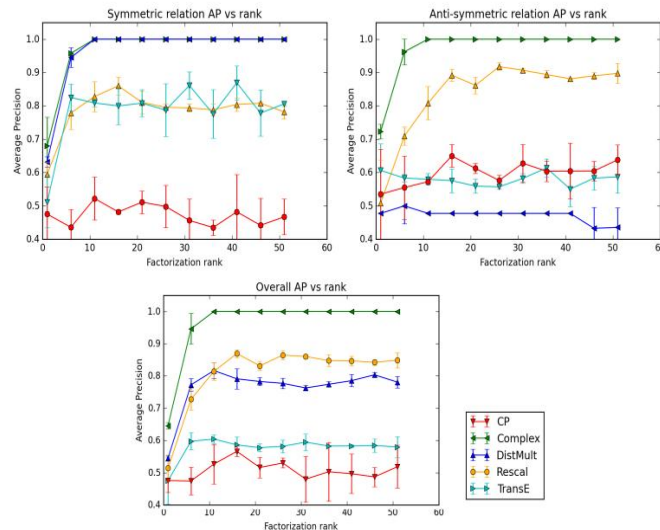


Figure 3. Result of different graph embedding models

Complex and DistMult embedding are both capable of capturing the semantic content of symmetric connections and making accurate predictions. Complex captures and predicts the semantics of asymmetric connections substantially better than other models, as seen in the upper image. The key to automatically understanding the structure of a big knowledge base is link prediction. They presented potential factorization as a solution to this problem in prior works. However, a significant variety of binary relations, including symmetric and antisymmetric relations, may be handled using a mixture of complex embeddings. This approach based on complex embedding is simpler than neural tensor networks and holographic embedding since it simply requires the Hermitian dot

product, the complex equivalent component of the ordinary dot product between real vectors. Because it remains linear in space and time, this strategy grows to enormous data sets while consistently beating other approaches in conventional link prediction benchmarks.

Chapter 3

Background

In general, the relationship extraction problem is changed into a classification problem, in which all the entity pairings in a phrase are first listed, and then the classifier is used to find the connections we truly need[32]. The more complicated the tagging system, the more accurate it will be in most circumstances, but the training time will also grow. As a result, it's critical for people to choose an acceptable labeling strategy based on the situation[32].

3.1 Entity extraction

The automated identification of named entities from the initial corpus is referred to as early entity extraction, also known as named entity learning or named entity recognition. Because an entity is the most fundamental piece in the knowledge graph, its extraction integrity, correctness, recall, and other characteristics will have a direct impact on the knowledge base's quality. As a result, entity extraction is the most fundamental and crucial phase in the knowledge extraction process. Determine the font string limits of text that has a certain meaning and sort it into predetermined groups. Traditional identification tasks identify events, institution names, places, and so on, but with the steady development of programs to detect particular specified categories, recognition tasks have become more sophisticated[33].

Entity extraction methods are classified into four categories: encyclopedia site or vertical site extraction, rule- and dictionary-based approaches, statistical machine learning-based methods, and open-domain-oriented extraction methods. The rule-based technique often requires the creation of a template for the target object, which is then matched against the original corpus, as in: The technique based on

statistical machine learning primarily employs machine learning to train the original corpus, and then uses the trained model to identify entities; open domain-focused extraction will be geared to enormous Web corpus[33].

1) Extract from an encyclopedia or vertical site, which is a very straightforward extraction method. Extract entity names from the titles and links of encyclopedic sites. It has the advantage of obtaining the most common entity name on the open Internet, but it has the disadvantage of having restricted coverage for low frequency. Unlike generic web sites, vertical class sites may gather domain-specific entities during entity extraction. For example, you can get various entity lists from Dou Ban channels (music, books, movies, etc.). This technique relies heavily on crawl technology to achieve and obtain[34].

2)Entity extraction approach based on dictionaries and rules Early entity extraction in the limit text field, under conditions of restricted semantic unit type, mostly uses a technique based on rules and dictionaries, such as extracting text of person names, place names, organization names, and certain temporal entities, for example, using established criteria. Rau L F. built an entity extraction system capable of extracting firm names for the first time, relying mostly on a heuristic algorithm and a rule template. However, the rule-template-based technique, also known as the compromise method, requires a large number of specialists to design rules or templates, which is relatively accurate but only covers a limited number of fields and is difficult to adapt to new data requirements[34].

3) Statistical machine learning-based entity extraction approach The researchers then attempt to extract named things using a supervised learning approach in machine learning. For example, Liu X, Zhang S, and colleagues used the KNN algorithm with a conditional random field model to recognize entities in Twitter text data[35]. The performance of the basic supervised learning algorithm is restricted not only by the training set, but also by the algorithm's accuracy and

recall rate. After recognizing the supervised learning method's limitation, the researchers attempted to merge the supervised learning algorithm with the rules, with some success. Lin Y F et al., for example, employed the maximum entropy technique to perform an entity extraction experiment on the GENIA data set of Medline abstracts, and the experiment's accuracy and recall rate were both above 70% [36].

4) An open field object extraction technique This paper proposes an iterative way extended entity corpus solution, aimed at how to automatically discover from a small number of entity instances have to distinguish the force model, and then extended to massive amounts of text entities to do classification techniques problems. Its basic idea is that a characteristic model is established by a small number of entity instances, and then the model is applied to new data sets to obtain new named entities. In 2010, Jain A et al introduced an unsupervised learning-based open domain clustering technique, the core concept of which is to search logs for named entities based on semantic properties of known entities, and then conduct clustering [37].

3.2 Word Representations

The author, like Collobert et al. (2011), uses pretrained word embeddings to create our lookup table[23]. Pretrained word embeddings outperform randomly initialized ones by a substantial margin. Skip-n-gram (Ling et al., 2015a)[38], a version of word2vec (Mikolov et al., 2013a) that accounts for word order[39], is used to pretrain embeddings. During training, these embeddings are fine-tuned. When it comes to NLP tasks in general, word embedding can help the model perform better[23].

3.3 Models for NER introduction

In this section, we describe the models used in this paper: BI-LSTM, CRF, HMM, LSTM-CRF and BI-LSTM-CRF models.

LSTM:

Recurrent neural networks (RNN) have been employed to produce promising results on a variety of tasks including language model (Mikolov et al., 2010; Mikolov et al., 2011)[40] and speech recognition (Graves et al., 2005)[41]. A RNN maintains a memory based on history information, which enables the model to predict the current output conditioned on long distance features.

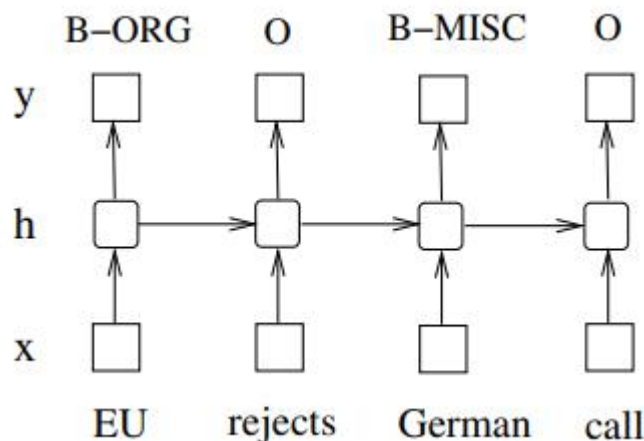


Figure 4: A simple RNN model.

The RNN structure (Elman, 1990) contains an input layer x , a hidden layer h , and an output layer y (see Figure 4)[42]. In the context of named entity tagging, x denotes input characteristics and y denotes tags. Figure 1 shows a named entity recognition system in which each word is assigned to one of four entity types: person (PER), location (LOC), organization (ORG), or miscellaneous (MISC). B-ORG, O, B-MISC, O tags show the starting and intermediate locations of entities in the phrase EU rejects German call.

At time t , an input layer represents features. One-hot encoding for word features, dense vector features, or sparse features are all possibilities. The dimensionality of

an input layer is the same as the dimensionality of the feature size. A probability distribution across labels at time t is represented by an output layer. It's the same dimensions as a label's size. A RNN, unlike a feedforward network, adds a link between the prior hidden state and the present hidden state (and thus the recurrent layer weight parameters). This recurrent layer is used to keep track of past events. The hidden and output layers' values are calculated as follows:

$$\begin{aligned}\mathbf{h}(t) &= f(\mathbf{U}\mathbf{x}(t) + \mathbf{W}\mathbf{h}(t-1)), \\ \mathbf{y}(t) &= g(\mathbf{V}\mathbf{h}(t)),\end{aligned}$$

where U , W , and V are the training-time connection weights, and $f(z)$ and $g(z)$ are the sigmoid and softmax activation functions, respectively.

$$\begin{aligned}f(z) &= \frac{1}{1 + e^{-z}}, \\ g(z_m) &= \frac{e^{z_m}}{\sum_k e^{z_k}}.\end{aligned}$$

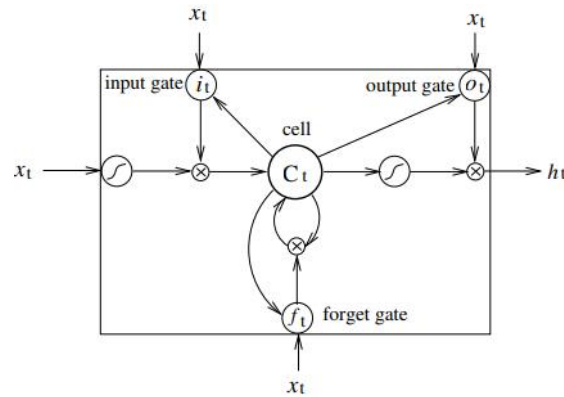


Figure 5. Example of LSTM cell

Hochreiter and Schmidhuber utilize Long Short-Term Memory to sequence tagging in this study. Long Short-Term Memory networks are similar to RNNs, but instead of hidden layer updates, purpose-built memory cells are used[43]. As a result, they may be more adept at detecting and exploiting long-range data relationships. A single LSTM memory cell is shown in Figure 5[45]. The following is how the LSTM memory cell is implemented:

$$\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \\
c_t &= f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \\
h_t &= o_t \tanh(c_t)
\end{aligned}$$

Bi-LSTM:

It is possible that the forecast will need to be decided jointly by numerous prior and subsequent inputs, which will result in a more accurate prediction. As a result, a bidirectional recurrent neural network is presented, with the following network structure. The Forward and Backward layers are both connected to the output layer, which has 6 common weights w_1 - w_6 [45].

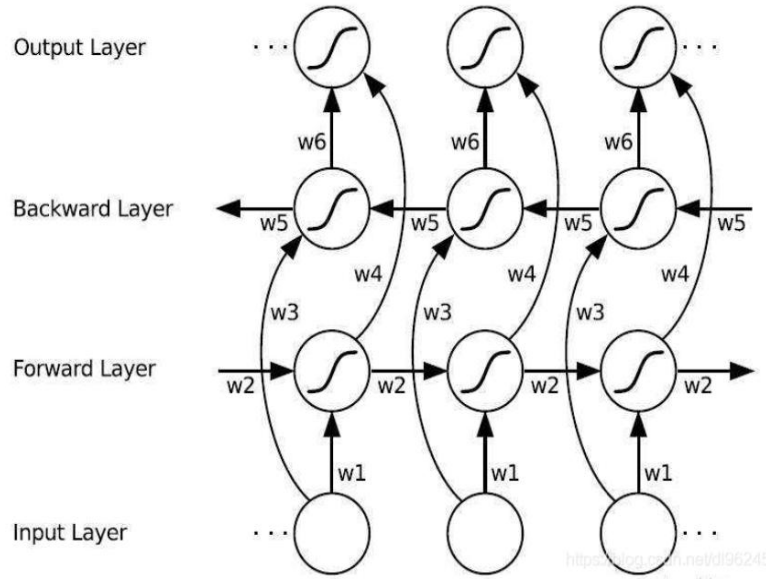


Figure 6. Structure of BI-LSTM model

In the Forward layer, the forward calculation is performed from time 1 to time t , and the output of the forward hidden layer at each time is obtained and saved. In the Backward layer, it is calculated in reverse from time t to time 1, and the output of the backward hidden layer at each time is obtained and saved. Finally, the final output is obtained by combining the output results of the Forward layer and the Backward layer at the corresponding moment at each moment. The mathematical expression is as follows[45]:

$$h_t = f(w_1x_t + w_2h_{t-1})$$

$$h'_t = f(w_3x_t + w_5h'_{t+1})$$

$$o_t = g(w_4h_t + w_6h'_t)$$

Unlike HMM and CRF, LSTM depends on the neural network's extremely nonlinear fitting capabilities. The samples are exposed to complicated nonlinear transformations in high-dimensional space during training in order to learn the function from sample to label and subsequently apply it. For the given sample, the function guesses the label for each token. A schematic model of sequence labeling using bidirectional LSTM (bidirectional captures sequence dependencies better) is shown below:

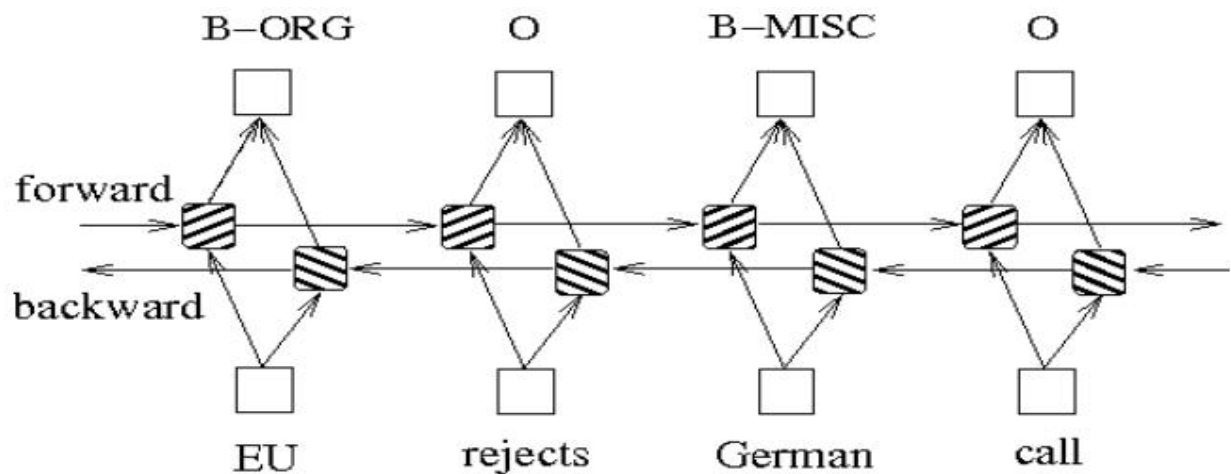


Figure 7. There is an example of Bi-LSTM to do tagging tasks

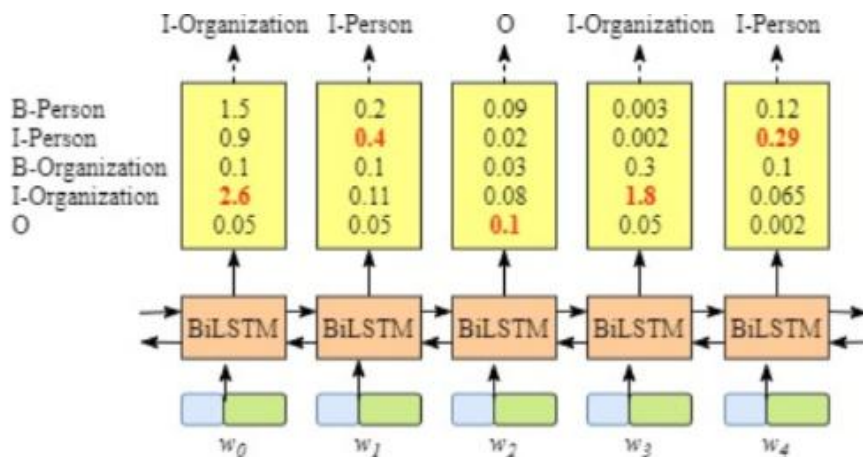


Figure 8. The bilstm model without CRF layer output

The output from the BiLSTM layer represents the score for each category for the word.

w_0 , "B-person" has the highest score (1.5), so we can select "B-Person" as the prediction result;

w_1 , is "I-Person";

w_2 , is "O";

w_3 , is "B-Organization" ;

w_4 , is "O";

Although we get the correct result for the sentence in this example, this is not always the case. Obviously, this classification is not accurate. "I-organization i-person" and "B-organization i-person" are both incorrect[46].

HMM:

In essence, named entity recognition may be thought of as a sequence labeling problem. We can observe the series of words (observation sequence) while utilizing HMM to solve the sequence labeling issue of named entity recognition. is the label (sequence of states) corresponding to each word[47].

The initial state distribution is the probability of each annotation being initialized, the state transition probability matrix is the probability of moving from one annotation to the next, and the observation probability matrix is the probability of generating a specific word under a specific annotation.

The HMM model's training procedure is analogous to the hidden Markov model's learning issue. In fact, the maximum likelihood technique is used to estimate the three parts of the model, namely the starting state distribution, the state transition probability matrix, and the observation indicated above, based on the training data. After the model has been trained, use it to decode, that is, discover the label

matching to each word in a phrase for a given sentence. We utilize the Viterbi algorithm to solve this decoding problem.

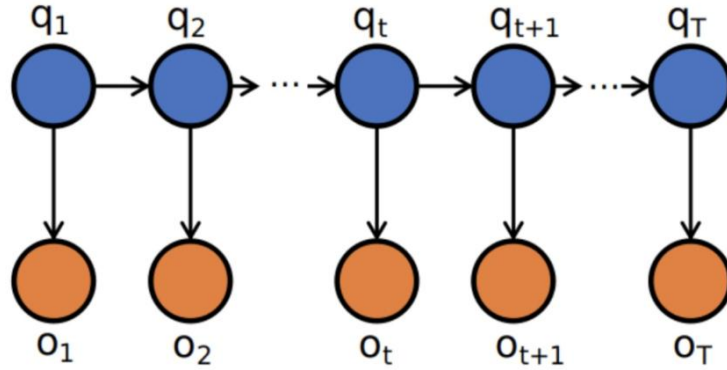


Figure 9.Example of HMM models

state sequence: $Q = q_1, q_2, \dots, q_T$

state value set: $S = \{s_1, s_2, \dots, s_N\}$

observation sequence: $O = o_1, o_2, \dots, o_T$

set of observations: $V = \{v_1, v_2, \dots, v_M\}$

In the idea graph model, HMM belongs to the directed graph model. A random variable is represented by a node, while a dependence is represented by a directed edge in the diagram above. For instance, O_1 is reliant on q_1 . It's worth noting that the observation sequence is influenced by the concealed state! [48] In the HMM model, there are three main components.

$$\lambda = \{A, B, \Pi\}$$

A is the state transition matrix: the probability of transition between each hidden state. B is the emission matrix called the observation probability matrix: the probability of observing different predicted values in a certain hidden state. π is the initialization probability: the probability that each hidden state occurs in the initial state[48].

CRF:

The HMM model is based on two assumptions: the output observations are absolutely independent, and the current state is only connected to the prior state during the state transition process. That is to say, in the case of named entity

recognition, HMM considers each word in the observed phrase to be independent of the others, and the current annotation is solely connected to the prior annotation. However, named entity identification frequently necessitates the addition of other elements such as part of speech, word context, and so on, and the annotation of the present moment should be linked to the annotations of the preceding and subsequent moments. Because these two assumptions exist, it is clear that the HMM model fails to solve the problem of named entity recognition.

In 2001, Lafferty et al. employed Conditional Random Fields (CRFs) to calculate the conditional probability of values on defined output nodes given values on other designated input nodes[49]. Given an observation series $O = \langle o_1, o_2, \dots, o_T \rangle$, the conditional probability of a state sequence $S = \langle s_1, s_2, \dots, s_T \rangle$ is determined as:

$$P_{\Lambda}(s|o) = \frac{1}{Z_0} \exp\left(\sum_{t=1}^T \sum_k \lambda_k \times f_k(s_{t-1}, s_t, o, t)\right),$$

Where $f_k(s_{t-1}, s_t, o, t)$ is a feature function with a weight λ_k that must be learnt during training. The feature functions can have values ranging from $-\infty, \dots, +\infty$, however they are usually binary. We must compute the normalization factor to make all conditional probabilities equal one.

$$Z_0 = \sum_s \exp\left(\sum_{t=1}^T \sum_k \lambda_k \times f_k(s_{t-1}, s_t, o, t)\right),$$

Like HMMs, may be efficiently produced by dynamic programming. The penalized log-likelihood of the state sequences given the observation sequences is the goal function to be maximized when training a CRF:

$$L_{\Lambda} = \sum_{i=1}^N \log(P_{\Lambda}(s^{(i)}|o^{(i)})) - \sum_k \frac{\lambda_k^2}{2\sigma^2},$$

where $\{\langle o^{(i)}, s^{(i)} \rangle\}$ is the labeled training data. The second sum corresponds to a zero-mean, σ^2 variance Gaussian prior over parameters, which facilitates

optimization by making the likelihood surface strictly convex. Here, we set parameters λ to maximize the penalized log-likelihood using Limited-memory BFGS [50], a quasi-Newton method that is significantly more efficient, and which results in only minor changes in accuracy due to changes in λ . When applying CRFs to the NER problem, an observation sequence is a token of a sentence or document of text and the state sequence is its corresponding label sequence. While CRFs generally can use real-valued functions, in our experiments maximum of the features are binary valued. A feature function $f_k(s_{t-1}, s_t, o, t)$ has a value of 0 for most cases and is only set to be 1, when s_{t-1} , s_t are certain states and the observation has certain properties[51].

By introducing a self-defined characteristic function, the conditional random field can not only express the dependence between observations, but also express the complex dependence between the current observation and multiple states before and after, which can effectively overcome the problems faced by the HMM model.

The Viterbi algorithm:

The Viterbi algorithm (VA) is a recursive best solution for predicting the state sequence of a discrete-time finite-state Markov process in memoryless noise.

Many issues in fields like digital communications may be expressed in this way.

This article provides an overview of the method, as well as how it is implemented and examined. The applications that have been submitted so far have been examined. The algorithm is expected to be used in a growing number of sectors.[52].

BI-LSTM+CRF:

The benefit of LSTM is that it can learn bidirectionally the relationships between observation sequences (input words). LSTM may extract characteristics of observation sequences according to the aim (such as recognizing entities) during

the training phase, but the drawback is that it cannot be taught. The relationship between state sequences (output annotations), you should know that there is a certain relationship between annotations in the named entity recognition task, such as type B annotations (representing the beginning of an entity) will not be followed by a B Class labeling, so when LSTM solves sequence labeling tasks like NER, it has the disadvantage of not being able to learn the labeling.

CRF, on the other hand, has the advantage of being able to model the hidden state and learn the properties of the state sequence, but it also has the disadvantage of requiring human extraction of the sequence features. To reap the benefits of both, it is common practise to add a layer of CRF after the LSTM.

A bidirectional LSTM network and a CRF network are combined to generate a Bi-LSTM-CRF network, which is similar to an LSTM-CRF network. A BiLSTM-CRF model can incorporate future input characteristics in addition to the past input features and sentence level tag information used in an LSTM-CRF model. Experiments will show that the added characteristics can improve tagging accuracy.

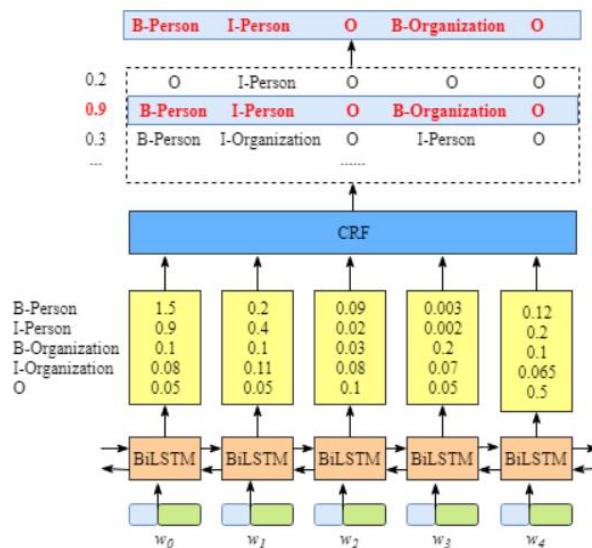


Figure 10: Layers in a Bi-LSTM-CRF model

The picture above illustrates that the outputs of BiLSTM layer are the scores of

each label. For example, for w_0 , the outputs of BiLSTM node are 1.5 (B-Person), 0.9 (I-Person), 0.1 (B-Organization), 0.08 (I-Organization) and 0.05 (O). These scores will be the inputs of the CRF layer. Then, all the scores predicted by the BiLSTM blocks are fed into the CRF layer. In the CRF layer, the label sequence which has the highest prediction score would be selected as the best answer.

Evaluation matrix

An Evaluation Matrix is a tool that compares alternative design concepts by utilising a table, or matrix. On the side, there's a list of criteria, and on top, there's a list of possible design options. Engineers can then give numerical values to each solution based on how well it "scores" on each criterion.

True Classification	Predicted classification			
	+	+	-	Total
	+	TP (True Positives)	FN (False Negatives)	TP + FN (Actual Positive)
	-	FP (False Positives)	TN (True Negatives)	FP + TN (Actual Negative)
	Total	TP + FP (Predicted Positive)	FN + TN (Predicted Negative)	TP + FP + FN + TN

Table 1: Confusion matrix

Accuracy:

The accuracy of a testing data is defined as ratio of the proportion of successfully classified samples by the classifier to the total number of samples. In other words, total accuracy on test datasets is 0-1 loss when the loss function is 0-1 loss. The accuracy of a model may be used to judge its overall performance; in general, the better the classifiers, the greater the accuracy. The accuracy formula is as follows:

$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN}$$

Precision:

Precision is a statistic for determining how many of the anticipated positive samples are truly positive. Predicting a positive class may be done in two ways: one is to predict a positive class (TP), and the other is to anticipate a negative class (FP), which is

$$Precision = \frac{TP}{TP + FP}$$

Recall:

The recall of a good classifier should be one (high). Recall becomes 1 only when the numerator and denominator are identical, i.e. $TP = TP + FN$. This also means that FN is equal to zero. As FN increases, the denominator becomes larger than the numerator, lowering the recall value (which we don't want). The following is the definition of recall, also known as sensitivity or true positive rate:

$$Recall = \frac{TP}{TP + FN}$$

F1-score:

The F1 Score becomes 1 only when both accuracy and recall are 1. The F1 score can only improve when both accuracy and recall are high. Because it represents the harmonic mean of precision and recall, the F1 score is a superior statistic than accuracy. The F1-score is a statistic that takes accuracy and recall into account, and it's calculated as follows:

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Graph embedding to link prediction:

This is the example of graph embedding :

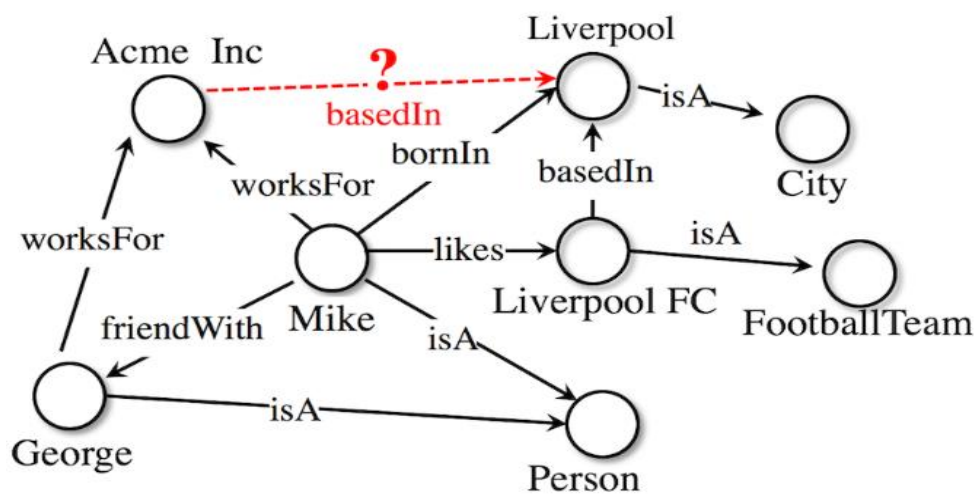


Figure 11.Example of graph embedding(1)

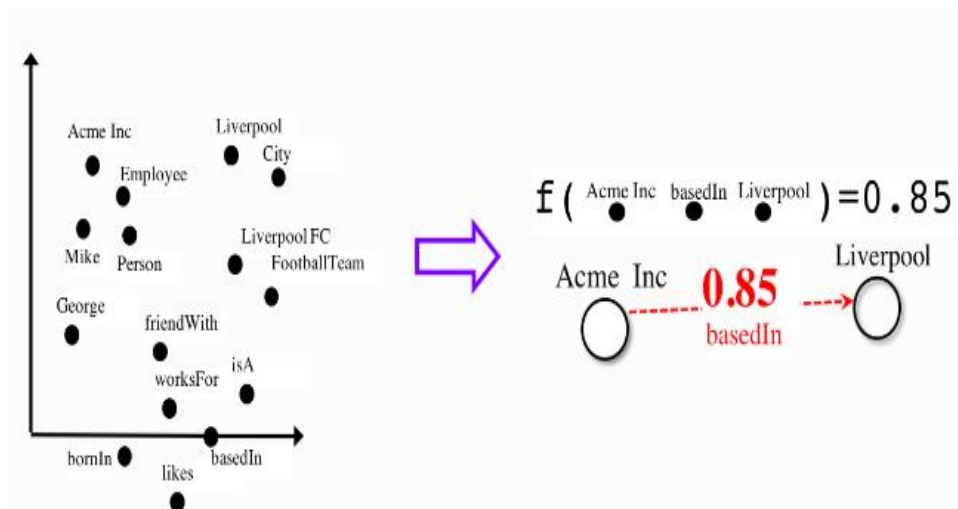


Figure 12.Example of graph embedding(2)

AmpliGraph's machine learning models generate knowledge graph embeddings, which are metric space vector representations of ideas. It then use embeddings in

conjunction with model-specific scoring functions to predict previously undiscovered and novel links[53].

We will explore the usage of complex embeddings for low-rank matrix factorization in this research, and we will demonstrate this by looking at a simple link prediction job with only one relation type. Grasp complex space factorization leads to a better theoretical understanding of the matrices that may be approximated by dot products of embeddings. These are the so-called normal matrices, which have the same unitary basis for the left and right embeddings.

Fristly , we need to introduce DisMult model(Figure 13) and its algorithm(Figure 14)[54]:

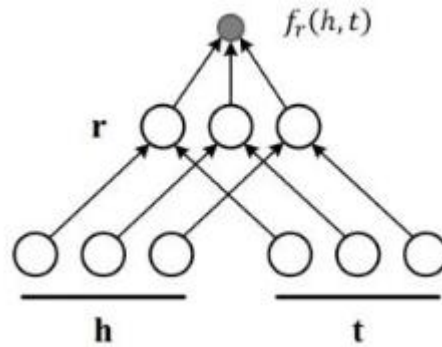


Figure 13.DisMult model

Algorithm 1 EMBEDRULE

- 1: **Input:** $KB = \{(e_1, r, e_2)\}$, relation set R
 - 2: **Output:** Candidate rules Q
 - 3: **for each** r in R **do**
 - 4: Select the set of start relations $S = \{s : \mathcal{X}_s \cap \mathcal{X}_r \neq \emptyset\}$
 - 5: Select the set of end relations $T = \{t : \mathcal{Y}_t \cap \mathcal{Y}_r \neq \emptyset\}$
 - 6: Find all possible relation sequences
 - 7: Select the K -NN sequences $P' \subseteq P$ for r based on $dist(\mathbf{M}_r, \mathbf{M}_{p_1} \circ \dots \circ \mathbf{M}_{p_n})$
 - 8: Form candidate rules using P' where r is the head relation and $p \in P'$ is the body in a rule
 - 9: Add the candidate rules into Q
 - 10: **end for**
-

Figure 14. DisMult of algorithm

The Complex Embedding model extends the DisMult model to Complex space, so as to better model the anti-symmetric and reversible relations. Its scoring

function is:

$$f_r(\mathbf{h}, \mathbf{t}) = \Re(\mathbf{h}^\top \mathbf{g} - (\mathbf{r})\bar{\mathbf{t}})$$

And $\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{C}^k$ represents the conjugate of the vector \mathbf{t} , $\Re(\cdot)$ stands for taking the real part.

ComplEx scoring function is based on the trilinear Hermitian dot product in \mathbb{C} , \mathbf{C} is represent vectors with entries[55]:

$$f_{ComplEx} = \Re(\langle \mathbf{r}_p, \mathbf{e}_s, \overline{\mathbf{e}_o} \rangle)$$

where \mathbf{e}_s is the embedding of the subject, \mathbf{r}_p the embedding of the predicate and $\overline{\mathbf{e}_o}$ the embedding of the object, which means this RDF has reflexivity relationship[56].

Evaluation for Graph embedding

For evaluation purposes, the ranks of all triplet test sets with all possible head and tail substitutions were measured. The evaluation indicators that are used are MRR (Mean Reciprocal Rank) and Hits@ m , and both of these evaluation indicators have two values (original value and sample value, the latter being the indicator produced by deleting data from the sample).

MRR_score: An algorithm is used to determine a vector's reciprocal rank mean.

Hits_@_n_score: How many elements of a ranking ranks vector make it to the top n spots is calculated by the function

Chapter 4

Methodology

This section presents the methods we use to train our models, the results we obtained on various tasks and the impact of our networks' configuration on model performance.

The software PyCharm is used to run python, and Neo4j to build knowledge graph. Another reason for using PyCharm is the user-friendly interface, We use Pandas, Numpy and Jupyter to pre-process and cleaning the data.

To evaluate the performance of our proposed model, we conduct a wide range of experiments. In this section, we describe the datasets used for training and evaluation, discuss hyper-parameters, baselines and the evaluation measures.

We employ the PyTorch programme for the NER assignment, which is a Python-based scientific computing software that makes advantage of graphics processing units' capabilities. It's also one of the most popular deep learning research platforms, with a focus on flexibility and speed[57].

For graph embedding for link prediction, we use ampligraph package, which is a suite of neural machine learning models for relational Learning, a branch of machine learning that deals with supervised learning on knowledge graphs[53].

4.1 Data sets:

We use the datasets about different movies' information.

The datasets download from Kaggle which is a public datasets area, and the datasets has 1000 rows and 13 columns.

Poster_Link - Link of the poster that imdb using

Series_Title - Name of the movie

Released Year - Year at which that movie released

Certificate - Certificate earned by that movie

Runtime - Total runtime of the movie

Genre - Genre of the movie

IMDB_Rating - Rating of the movie at IMDB site

Overview - mini story/ summary

Meta_score - Score earned by the movie

Director - Name of the Director

Star1,Star2,Star3,Star4 - Name of the Stars

Noofvotes - Total number of votes

Gross - Money earned by that movie

Poster_Link	Series_Title	Released_Year	Certificate	Runtime	Genre	IMDB_Rating	Overview	Meta_score	Director	Star1	Star2	Star3	Star4	No_of_seasons
https://m.media-amazon.com/images/M/MV5BMTkxMjY0NDUyOV5BMl5Banl... 	The Shawshank Redemption	1994	A	142 min	Drama	9.3	Two imprisoned men bond over a number of years...	80.0	Frank Darabont	Tim Robbins	Morgan Freeman	Bob Gunton	William Sadler	23
https://m.media-amazon.com/images/M/MV5BMjMyNTUxMDQwNF5BMl5Banl... 	The Godfather	1972	A	175 min	Crime, Drama	9.2	An organized crime dynasty's aging patriarch t...	100.0	Francis Ford Coppola	Marlon Brando	Al Pacino	James Caan	Diane Keaton	16
https://m.media-amazon.com/images/M/MV5BMTkxMjY0NDUyOV5BMl5Banl... 	The Dark Knight	2008	UA	152 min	Action, Crime, Drama	9.0	When the menace known as the Joker wreaks hav...	84.0	Christopher Nolan	Christian Bale	Heath Ledger	Aaron Eckhart	Michael Caine	23

Picture 1. example of datasets

4.2 Data pre-processing:

Since the existing data is not enough, for the direction of my research, I need to go to wikipedia to crawl the text type (unstructured) data we need, and these data are more credible and have explanatory.

Here is some brief information about each movie, including the movie's title, release year, genre, director, source, and more. But this is just unstructured data, we need to identify entity sets and relationships from unstructured data.

The Shawshank Redemption is a 1994 American drama film written and directed by Frank Darabont, based on the 1982 Stephen King novella Rita Hayworth and Shawshank Redemption. It tells the story of banker Andy Dufresne, who is sentenced to life in Shawshank State Penitentiary for the murders of his wife and her lover, despite his claims of innocence. Over the following two decades, he befriends a fellow prisoner, contraband smuggler Ellis "Red" Redding, and becomes instrumental in a money-laundering operation led by the prison warden Samuel Norton. William Sadler, Clancy Brown, Gil Bellows, and James Whitmore appear in supporting roles.

The Godfather is a 1972 American crime film directed by Francis Ford Coppola, who co-wrote the screenplay with Mario Puzo, based on Puzo's best-selling 1969 novel of the same name. The film stars Marlon Brando, Al Pacino, James Caan, Richard Castellano, Robert Duvall, Sterling Hayden, John Marley, Richard Conte, and Diane Keaton. It is the first installment in The Godfather trilogy. The story, spanning from 1945 to 1955, chronicles the Corleone family under patriarch Vito Corleone (Brando), focusing on the transformation of his youngest son, Michael Corleone (Pacino), from reluctant family outsider to ruthless mafia boss.

The Dark Knight is a 2008 superhero film directed, co-produced, and co-written by Christopher Nolan. Based on the DC Comics character Batman, the film is the second installment of Nolan's The Dark Knight Trilogy and a sequel to 2005's Batman Begins, starring Christian Bale and supported by Michael Caine, Heath Ledger, Gary Oldman, Aaron Eckhart, Maggie Gyllenhaal, and Morgan Freeman. In the film, Bruce Wayne / Batman (Bale), Police Lieutenant James Gordon (Oldman) and District Attorney Harvey Dent (Eckhart) form an alliance to dismantle organized crime in Gotham City, but are menaced by an anarchistic mastermind known as the Joker (Ledger), who seeks to undermine Batman's influence and throw the city into chaos.

Figure. example of text data

In order to improve efficiency, we choose to use nltk and spacy tools to complete the text annotation task together. The task of named entity recognition is to assign a named entity label to each word in a sentence. A single named entity can span multiple tokens in a sentence. Sentences are usually represented in IOB format (Inside, Outside, Beginning), where each token is marked with a B-label if the token is the beginning of a named entity, and I if it is inside the named entity but not the first token -label is inside a named entity, otherwise O. In order to train the model better, we need to remove all punctuation marks in the text, and then only need to keep the last punctuation mark of each sentence "." .

And for this step, we need to identify entities with some text from each movie in spaCy. This is what happens to the description of each movie.

```

from spacy import displacy

displacy.render(doc, style='ent', jupyter=True)

```

The Shawshank Redemption **WORK_OF_ART** is a 1994 **DATE** American **NORP** drama film written and directed by Frank Darabont **PERSON**, based on the 1982 **DATE** Stephen King **PERSON** novella Rita Hayworth and Shawshank Redemption **WORK_OF_ART**.

reference:[1]<https://towardsdatascience.com/named-entity-recognition-with-nltk-and-spacy-8c4a7d88e7da#targetText=Named%20entity%20recognition%20%28NER%29is,monetary%20values%2C%20percentages%2C%20etc.>

In sequence annotations, we need to label each element (token) of a sequence. Generally speaking, a sequence refers to a sentence, and an element (token) refers to a word or a word in a sentence. For example, information extraction can be regarded as a sequence labeling problem, such as extracting the time and place of the meeting.

Label types are generally defined as follows:

PERSON: People, including fictional

NORP: Nationalities or religious or political

FAC: Buildings, airports, highway, bridges, etc.

ORG: Companies, agencies, institutions, etc.

GPE: Countries, cities, states

LOC: Non-GPE location, mountain ranges, bodies of water.

EVENT: Named hurricanes, battles, wars, sports events, etc.

WORK_OF_ART: Titles of books, songs, movies, etc.

DATE: Dates or periods

Define	The Full Name	Note
B	Begin	The beginning of the physical fragment
I	Intermediate	The middle of the entity fragment
O	Other/Outside	Characters that do not belong to

		any entity (including punctuation, etc.)
--	--	---

Table 1 of BIO define

This project tried to use a variety of different models (including HMM, CRF, BI-LSTM, bi-LSTM +CRF) to solve the problem of named entity recognition. The format of data is as follows. Each line of the data consists of a word and its corresponding annotation, Figure 7.

```

The B-WORK_OF_ART
Shawshank I-WORK_OF_ART
Redemption I-WORK_OF_ART
is O
a O
1994 B-DATE
American B-NORP
drama O
film O
written O
and O
directed O
by O
Frank B-PERSON
Darabont I-PERSON
based O
on O
the O
1982 B-DATE

```

Picture 2. Sample of Tagging Datasets

We need to divide the data set into three parts: training set, development set, test set: 60% , 20%, 20%.

We test HMM, BI-LSTM, CRF, and BI-LSTM-CRF models in PyTorch on named entity tagging. Table 1 shows the size of sentences, tokens, and labels for training, validation and test sets respectively. POS assigns each word with a unique tag that indicates its syntactic role. In chunking, each word is tagged with its phrase type. For example, tag B-NP indicates a word starting a noun phrase. In NER task, each word is tagged with other or one of four entity types: Person, Location, Organization, or Miscellaneous. We use the BIO annotation standard for chunking and NER tasks.

4.3 Network architectures

HMM model :

It is actually according to the training data and the three elements of Maximum_likelihood_estimation estimation model[60], namely, the initial state distribution, state transition probability matrix and observation probability matrix mentioned above. An example is taken to help understand that when estimating the initial state distribution, if the number of times a mark is marked as the first word of a sentence in the data set is K and the total number of sentences is N, then the probability of the mark as the first word of a sentence can be estimated as K/N approximately.

The training of HMM is to estimate the model parameters according to the training corpus. Because we have the observation sequence and its corresponding state sequence, we can use the maximum likelihood estimation method to estimate the parameters of the hidden Markov model.

Model Parameters:

word_lists: list, where each element consists of a list of words, such as ['is','an','American','movie']

tag_lists: list, where each element is a list of corresponding labels, such as ['O','O','B-NORP','O']

word2id: Mapping words to I D

tag2id: Dictionary, mapping labels to I D

After the model training is completed, the trained model should be used to decode, that is, to find the corresponding annotation of each word in the sentence for the sentences that have not been seen in the given model. For this decoding problem, we use the Viterbi algorithm. The Viterbi algorithm is used to obtain the state sequence for the given observation sequence. Here is the sequence composed of pairs of words, and the corresponding annotation is obtained. In fact, Viterbi

algorithm uses dynamic programming to solve hidden Markov model prediction problems, namely, dynamic programming to find the maximum probabilistic path (optimal path).

For the model of CRF

we need to use sklearn_crfsuite package, We need to extract the features of a single word, extract sequence features, train model decoding, predict the annotation of a given sentence, and then carry out prediction and model evaluation. The best parameters for CRF model implement by PyTorch:

algorithm='lbfgs'

c1=0.1

c2=0.1

max_iterations=100

For the model of BI-LSTM

In addition to the above two probabilistic graph model-based methods, LSTM is also often used to solve sequence labeling problems. Different from HMM and CRF, LSTM relies on the super nonlinear fitting ability of neural network. During training, samples are learned from the function from samples to annotations through complex nonlinear transformation in high-dimensional space, and then the function is used to predict the annotations of each token for the specified samples. Compared with CRF model, the biggest advantage of LSTM is that it is simple. It does not need to do complex feature engineering and can be directly trained. Meanwhile, compared with HMM, LSTM has higher accuracy.

For the model of BI-LSTM+CRF

we need to set and find the best parameters to evaluation:

Mould Parameters:

Batch_size=128

Learn rate=0.001

Epoches=100

Emb_size=128

Hidden_size=128

Algorithm 1 Bidirectional LSTM CRF model training procedure

```
1: for each epoch do
2:   for each batch do
3:     1) bidirectional LSTM-CRF model forward pass:
4:       forward pass for forward state LSTM
5:       forward pass for backward state LSTM
6:     2) CRF layer forward and backward pass
7:     3) bidirectional LSTM-CRF model backward pass:
8:       backward pass for forward state LSTM
9:       backward pass for backward state LSTM
10:    4) update parameters
11:   end for
12: end for
```

Picture 3. algorithm of BiLSTM+CRF

When we complete the task of entity extraction, the next step is to extract the relationship, and the last step is to establish the knowledge map and visualization.

4.4 Graph embedding to link prediction

For this movie knowledge graph, the relationship between entities is relatively simple and uniform. We just need to define some relationships in advance and let the model match, and then we can get a large number of triples. And the knowledge graph can be constructed. After we complete the NER task, we use regular expressions to match predefined relationships.

And then, we use

```
Avatar,IS_PRODUCED,Ingenious Film Partners
Avatar,IS_PRODUCED,Twentieth Century Fox Film Corporation
Avatar,IS_PRODUCED,Dune Entertainment
Avatar,IS_PRODUCED,Lightstorm Entertainment
Spectre,IS_PRODUCED,Columbia Pictures
Spectre,IS_PRODUCED,Danjaq
Spectre,IS_PRODUCED,B24
Tangled,IS_PRODUCED,Walt Disney Pictures
Tangled,IS_PRODUCED,Walt Disney Animation Studios
Titanic,IS_PRODUCED,Paramount Pictures
Titanic,IS_PRODUCED,Twentieth Century Fox Film Corporation
```

Picture 4. Example of RDF triples

```
Gattaca,HAS_GENRE,Romance
Batman,HAS_GENRE,Fantasy
Batman,HAS_GENRE,Action
21,HAS_GENRE,Drama
21,HAS_GENRE,Crime
Trainwreck,HAS_GENRE,Comedy
Species,HAS_GENRE,Science Fiction
Species,HAS_GENRE,Horror
Species,HAS_GENRE,Action
```

Picture 5.Example of RDF triples

For we need to apply `ampligraph.latent_features.Complex()` to make a graph embedding model.

First, we divided the data set into 80% test set and 20% training set. Secondly, training a model and set the parameters:

```
batches_count=100,
seed=0,
epochs=200,
k=150,
eta=5,
optimizer='adam',
optimizer_params={'lr':1e-3},
loss='multiclass_nll',
regularizer='LP',
regularizer_params={'p':3, 'lambda':1e-5},
verbose=True
```

Explain of the parameters:

k: the dimensionality of the embedding space

eta: the number of negative, or false triples that must be generated at training runtime for each positive, or true triple

batches_count: the number of batches in which the training set is split during the training loop. If you are having into low memory issues than settings this to a higher number may help.

epochs: the number of epochs to train the model for.

optimizer: the Adam optimizer, with a learning rate of $1e-3$ set via the optimizer_params kwarg.

loss: pairwise loss, with a margin of 0.5 set via the loss_params kwarg.

regularizer: Lp regularization with $p=2$, i.e. l2 regularization. $\lambda = 1e-5$, set via the regularizer_params kwarg[61].

And the last step is evaluating the model, and the result put in next chapter.

Chapter 5

Results and discussion

Evaluation HMM module:

正在训练评估HMM模型...

	precision	recall	f1-score	support
B-PERSON	0.8661	0.7391	0.7976	1602
B-NORP	0.8875	0.8130	0.8486	262
B-PRODUCT	0.5000	0.0769	0.1333	13
I-ORG	0.7114	0.4753	0.5699	223
I-MONEY	0.2500	0.2000	0.2222	5
B-CARDINAL	0.7000	0.6774	0.6885	93
B-TIME	0.2500	0.3333	0.2857	6
I-PERSON	0.8974	0.7333	0.8071	1646
B-LANGUAGE	0.5882	0.6250	0.6061	16
I-DATE	0.7833	0.5434	0.6416	173
I-GPE	0.8083	0.7578	0.7823	128
B-QUANTITY	0.2000	0.3333	0.2500	6
I-PRODUCT	0.0000	0.0000	0.0000	7
B-WORK_OF_ART	0.6537	0.6504	0.6521	389
B-GPE	0.7405	0.6158	0.6724	190
B-FAC	0.5714	0.2857	0.3810	14
I-NORP	0.2609	0.4000	0.3158	15
I-EVENT	0.6981	0.4005	0.5692	77
B-DATE	0.8353	0.8771	0.8557	480
B-EVENT	0.6667	0.5556	0.6061	36
I-LOC	0.4318	0.2794	0.3393	68
O	0.9048	0.9722	0.9373	13379
B-LOC	0.5714	0.3571	0.4396	56
B-ORG	0.6140	0.4730	0.5344	148
I-QUANTITY	0.5000	0.6000	0.5455	15
I-CARDINAL	0.6923	0.3214	0.4390	28
B-MONEY	0.0000	0.0000	0.0000	4
I-FAC	0.1250	0.0500	0.0714	20
I-WORK_OF_ART	0.7138	0.5423	0.6163	828
B-ORDINAL	0.6606	0.9231	0.7701	78
I-TIME	0.1818	0.2857	0.2222	7
avg/total	0.8693	0.8743	0.8682	20012

Picture 6. Result of HMM

Evaluation CRF module:

正在训练评估CRF模型...

	precision	recall	f1-score	support
B-PERSON	0.8387	0.7984	0.8180	1602
B-NORP	0.9333	0.7481	0.8305	262
B-PRODUCT	0.0000	0.0000	0.0000	13
I-ORG	0.8376	0.4395	0.5765	223
I-MONEY	0.5000	0.4000	0.4444	5
B-CARDINAL	0.8594	0.5914	0.7006	93
B-TIME	1.0000	0.1667	0.2857	6
I-PERSON	0.8496	0.8512	0.8504	1646
B-LANGUAGE	0.9091	0.6250	0.7407	16
I-DATE	0.7101	0.5665	0.6302	173
I-GPE	0.8485	0.6562	0.7401	128
B-QUANTITY	0.0000	0.0000	0.0000	6
I-PRODUCT	0.0000	0.0000	0.0000	7
B-WORK_OF_ART	0.8365	0.6838	0.7525	389
B-GPE	0.8056	0.4579	0.5839	190
B-FAC	1.0000	0.0714	0.1333	14
I-NORP	0.8750	0.4667	0.6087	15
I-EVENT	0.8367	0.5325	0.6508	77
B-DATE	0.9227	0.7208	0.8094	480
B-EVENT	0.8571	0.5000	0.6316	36
I-LOC	0.9231	0.1765	0.2963	68
O	0.9074	0.9712	0.9382	13379
B-LOC	0.8462	0.1964	0.3188	56
B-ORG	0.9630	0.3514	0.5149	148
I-QUANTITY	0.0000	0.0000	0.0000	15
I-CARDINAL	0.5625	0.3214	0.4091	28
B-MONEY	0.5000	0.2500	0.3333	4
I-FAC	0.0000	0.0000	0.0000	20
I-WORK_OF_ART	0.7560	0.7597	0.7578	828
B-ORDINAL	0.9524	0.7692	0.8511	78
I-TIME	1.0000	0.1429	0.2500	7
avg/total	0.8829	0.8874	0.8791	20012

Picture 7. Result of CRF

Evaluation BI-LSTM module:

```

评估bilstm模型中...
precision    recall    f1-score   support

B-PERSON    0.8590    0.7603    0.8066    1602
B-NORP      0.8354    0.7748    0.8040    262
B-PRODUCT   0.0000    0.0000    0.0000     13
I-ORG       0.8208    0.3901    0.5289    223
I-MONEY     0.3333    0.0000    0.4286     5
B-CARDINAL  0.8286    0.6237    0.7117     93
B-TIME      0.0000    0.0000    0.0000     6
I-PERSON    0.8605    0.8208    0.8402    1646
B-LANGUAGE  0.7143    0.6250    0.6667     16
I-DATE      0.6780    0.4624    0.5498    173
I-GPE       0.7500    0.6797    0.7131    128
B-QUANTITY  0.0000    0.0000    0.0000     6
I-PRODUCT   0.0000    0.0000    0.0000     7
B-WORK_OF_ART 0.5993    0.4653    0.5239    389
B-GPE       0.5158    0.6000    0.5547    190
B-FAC       0.5000    0.0714    0.1250     14
I-NORP      1.0000    0.3333    0.5000     15
I-EVENT     0.7805    0.4156    0.5424     77
B-DATE      0.8551    0.7500    0.7991    480
B-EVENT     0.7500    0.3333    0.4615     36
I-LOC       0.5789    0.1610    0.2529     68
O           0.9011    0.9844    0.9409    13379
B-LOC       0.3913    0.1607    0.2278     56
B-ORG       0.7000    0.3311    0.4495    148
I-QUANTITY  0.8000    0.2667    0.4000     15
I-CARDINAL  0.7000    0.2500    0.3684     28
B-MONEY     0.0000    0.0000    0.0000     4
I-FAC       0.0000    0.0000    0.0000     20
I-WORK_OF_ART 0.7561    0.4831    0.5895    828
B-ORDINAL  0.9355    0.7436    0.8286     78
I-TIME      0.6667    0.2857    0.4000     7
avg/total  0.8647    0.8751    0.8638    20012

```

Picture 8. Result of BI-LSTM

Evaluation BI-LSTM+CRF module:

```

评估bilstm_crf模型中...
precision    recall    f1-score   support

B-PERSON    0.8346    0.6835    0.7515    1602
B-NORP      0.8304    0.7099    0.7654    262
B-PRODUCT   0.0000    0.0000    0.0000     13
I-ORG       0.1146    0.5202    0.1879    223
I-MONEY     0.0000    0.0000    0.0000     5
B-CARDINAL  0.8214    0.4946    0.6174     93
B-TIME      0.0000    0.0000    0.0000     6
I-PERSON    0.8321    0.7078    0.7649    1646
B-LANGUAGE  0.8889    0.5000    0.6400     16
I-DATE      0.6506    0.3121    0.4219    173
I-GPE       0.8646    0.6484    0.7411    128
B-QUANTITY  0.0000    0.0000    0.0000     6
I-PRODUCT   0.0000    0.0000    0.0000     7
B-WORK_OF_ART 0.6834    0.5938    0.6355    389
B-GPE       0.6194    0.4368    0.5123    190
B-FAC       0.0000    0.0000    0.0000     14
I-NORP      1.0000    0.2000    0.3333     15
I-EVENT     0.5667    0.4416    0.4964     77
B-DATE      0.7285    0.7604    0.7441    480
B-EVENT     0.9091    0.2778    0.4255     36
I-LOC       1.0000    0.0147    0.0290     68
O           0.9219    0.9605    0.9408    13379
B-LOC       1.0000    0.0179    0.0351     56
B-ORG       0.3571    0.2345    0.2846    148
I-QUANTITY  0.0000    0.0000    0.0000     15
I-CARDINAL  0.5000    0.0714    0.1250     28
B-MONEY     0.0000    0.0000    0.0000     4
I-FAC       0.0000    0.0000    0.0000     20
I-WORK_OF_ART 0.7474    0.6075    0.6702    828
B-ORDINAL  0.8929    0.6410    0.7463     78
I-TIME      0.0000    0.0000    0.0000     7
avg/total  0.8646    0.8456    0.8470    20012

```

Picture 9. Result of BI-LSTM+CRF

We could get the result as follow:

	Precision	Recall	F1-score
HMM	86.93%	87.43%	86.82%
CRF	88.29%	88.74%	87.91%
Bi-LSTM	86.74%	87.51%	86.38%
Bi-LSTM +CRF	89.54%	89.91%	89.13%

Table 2. Result of model

The result for graph embedding:

```
from magraph.evaluation import mr_score, arr_score, hits_at_n_score

arr = arr_score(ranks)
print("MRR: %.2f" % (arr))

hits_10 = hits_at_n_score(ranks, n=10)
print("Hits@10: %.2f" % (hits_10))
hits_3 = hits_at_n_score(ranks, n=3)
print("Hits@3: %.2f" % (hits_3))
hits_1 = hits_at_n_score(ranks, n=1)
print("Hits@1: %.2f" % (hits_1))
```

MRR: 0.12
Hits@10: 0.25
Hits@3: 0.14
Hits@1: 0.05

Picture 10. Result of BI-LSTM+CRF

Discussion

By comparing the results of different models, we can find bilstm + CRF model obtained the best score, thanks to its neural network structure, and then we go to do the experiment of graph embedding model, but found that score is not high, the most direct reason is the data set is too small and the data preprocessing steps need further improvement. “Neural Architectures for Named Entity Recognition”[58] as our main reference material, we verified that neural networks can indeed make NER tasks more efficient and accurate.

Chapter 6

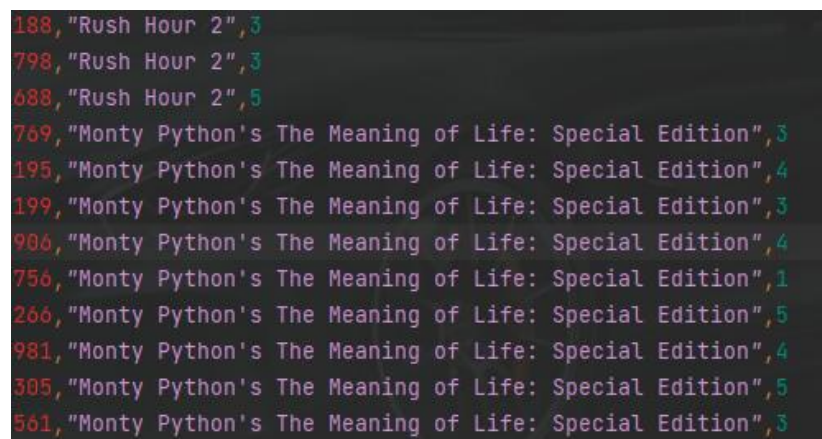
Application

6.1 Movies Recommendation System

The initial construction of the film knowledge graph is completed, and the similarity relationship between users is added to complete the calculation of similarity between users based on cosine similarity, so that the information of the film knowledge graph is more complete and the relevant knowledge information foundation of users is provided for the next recommendation system. In terms of user data and ratings, we used Netflix's datasets and download from Kaggle ,like figure , to build our system. The movie relationship specifically constructed in the Neo4j graph database is shown in the following figure, for users datasets :

HAS_GENRE	Movie_HAS_GENRE—>GENRE
HAS_KEYWORD	Movie_HAS_KEYWORD—>KEY WORD
HAS_PRODUCUTOR	Movie_HAS_PRODUCUTOR—>PRODUCUTOR
SIMILARITY	User SIMILARITY—>User

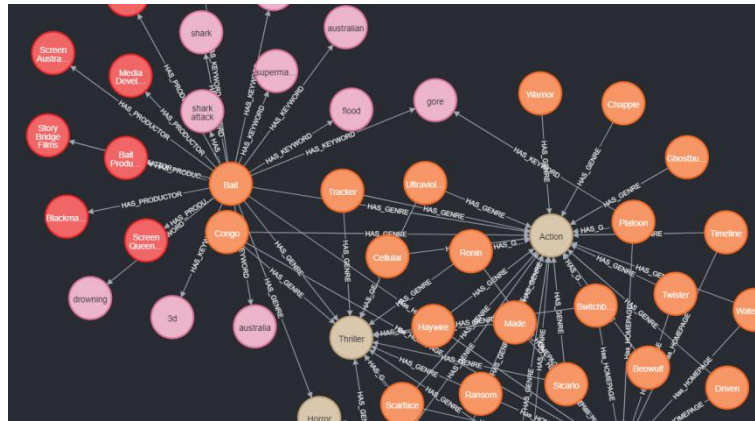
Picture11. Example of relationship in knowledge graph



188	"Rush Hour 2"	3
798	"Rush Hour 2"	3
688	"Rush Hour 2"	5
769	"Monty Python's The Meaning of Life: Special Edition"	3
195	"Monty Python's The Meaning of Life: Special Edition"	4
199	"Monty Python's The Meaning of Life: Special Edition"	3
906	"Monty Python's The Meaning of Life: Special Edition"	4
756	"Monty Python's The Meaning of Life: Special Edition"	1
266	"Monty Python's The Meaning of Life: Special Edition"	5
981	"Monty Python's The Meaning of Life: Special Edition"	4
305	"Monty Python's The Meaning of Life: Special Edition"	5
561	"Monty Python's The Meaning of Life: Special Edition"	3

Picture 12. Example of users' information. Column1 is ID which represents users. Columns2 is Movie'title entity. Columns3 is grade for movies.

This is a picture of Neo4j, when we visualized the movie knowledge graph:



Picture 13 . Visualization of movies knowledge graph

We use Python+Flask to build a friendly front-end page to do a movie recommendation system demo, which can help users get better experience and effect. For the recommendation algorithm, we mainly use cosine similarity. The following is the realization of cosine similarity in Neo4j and the picture of the system:

Movies Recommendation System

首页 登录 注册 搜索 电影数据库

Your rating are the following:

Index	Title	Grade	Genre
1	Control Room	5	None
2	The Hurricane	5	None
3	Ray	5	Drama
4	The Life Aquatic with Steve Zissou	5	None
5	Garden State	5	None
6	This Is Spinal Tap	5	None
7	Ray	5	Music
8	BASketball	5	None
9	The Professional	5	None
10	So I Married an Axe Murderer	5	None
11	Dogma	5	Adventure
12	Dogma	5	Comedy
13	Dogma	5	Fantasy
14	Lock	5	None
15	Y Tu Mama Tambien	5	None

For id:735 We would recommend these movies:

Index	Title	Average grade	Num recommenders	Genres	Homepage
1	Man on Fire	5.0	2	[]	None
2	Don't Say a Word	4.499814951862145	2	[]	None
3	The Village	4.0	2	[]	None
4	Something's Gotta Give	4.0	2	[]	None
5	Ghost	3.999643643940807	2	["Romance", "Fantasy", "Thriller", "Mystery", "Drama"]	None

Picture 14. Result of Movies Recommendation system

We use cosine similarity to id: 735 to recommend a series of movies. The first table displayed on the page is what movies ID: 735 has seen, and the second one is the movie recommended this time.

Chapter 7

Conclusions

7.1 Summary and conclusions

We have presented a multilayer neural network architecture that can handle a number of NLP tasks with both speed and accuracy. The design of this system was determined by our desire to avoid task-specific engineering as much as possible. Instead we rely on large unlabeled data sets and let the training algorithm discover internal representations that prove useful for all the tasks of interest. Using this strong basis, we have engineered a fast and efficient “all purpose” NLP tagger that we hope will prove useful to the community.

7.2 Possible future work

In the process of researching this topic, I found a relatively new model BERT, this model does not require a lot of data for training, because it is too expensive to choose the method of human labeling to produce training data. The paper "A Neural Probabilistic Language Model" says that each article is inherently a training corpus. Don't need manual labeling? Answer, no need. we would try our best to do this, because this model has good performance to do NLP task.

The latest records for 11 NLP tasks that BERT has refreshed so far include: pushing the GLUE benchmark to 80.4% (7.6% absolute improvement), MultiNLI accuracy reaching 86.7% (5.6% absolute improvement), pushing the SQuAD v1.1 question answering The test F1 score record was refreshed to 93.2 points (an absolute increase of 1.5 scores), surpassing human performance by 2.0 points[59].

Bibliography

- [1]<https://yago-knowledge.org/>
- [2]<https://paperswithcode.com/dataset/wn18rr>
- [3]https://en.wikipedia.org/wiki/Knowledge_Graph
- [4]Xu Zenglin, Sheng Yongpan, He Lirong, et al. Journal of University of Electronic Science and Technology of China, 2016, 45(4): 589-606.
- [5]Chen H, Cao G, Chen J, et al. A practical framework for evaluating the quality of knowledge graph[C]//China conference on knowledge graph and semantic computing. Springer, Singapore, 2019: 111-122.
- [6]<https://zhuanlan.zhihu.com/p/371765354>
- [7]https://en.wikipedia.org/wiki/Recurrent_neural_network
- [8]H. Cai, V. W. Zheng and K. C. -C. Chang, "A Comprehensive Survey of Graph Embedding: Problems, Techniques, and Applications," in IEEE Transactions on Knowledge and Data Engineering, vol. 30, no. 9, pp. 1616-1637, 1 Sept. 2018, doi: 10.1109/TKDE.2018.2807452.
- [9]Trouillon T, Welbl J, Riedel S, et al. Complex embeddings for simple link prediction[C]//International conference on machine learning. PMLR, 2016: 2071-2080.
- [10]Jim Webber. 2012. A programmatic introduction to Neo4j. In Proceedings of the 3rd annual conference on Systems, programming, and applications: software for humanity (SPLASH '12). Association for Computing Machinery, New York, NY, USA, 217–218. <https://doi.org/10.1145/2384716.2384777>
- [11]<https://en.wikipedia.org/wiki/Wikipedia>
- [12]Mansouri A, Affendey L S, Mamat A. Named entity recognition approaches[J]. International Journal of Computer Science and Network Security, 2008, 8(2): 339-344.
- [13]<https://www.zhihu.com/question/354081237/answer/2382243772>
- [14]Eddy S R. What is a hidden Markov model?[J]. Nature biotechnology, 2004, 22(10): 1315-1316.
- [15]Xiang Xiaowen, Shi Xiaodong, Zeng Hualin. Chinese Named Entity Recognition System based on Statistics and Rules. Computer Applications, 2005, 25(10): 2404-2406.
- [16]Hongbin W, Hongkui G, Qiang S, et al. Thai Language Names, Place Names and Organization Names Entity Recognition[J]. Journal of System Simulation, 2019, 31(5): 1010.
- [17]Zhang Chao-sheng, Guo Jian-yi, Xian Yan-tuan, et al. Computer Engineering and Science, 2010, 32(6): 115-117. (in Chinese)

- [18]Carreras X, Márquez L, Padró L. Named Entity Extraction using AdaBoost, proceeding of the 6th Conference on Natural language learning[C]//Association for Computational Linguistics. 2002.
- [19]Florian R, Ittycheriah A, Jing H, et al. Named entity recognition through classifier combination[C]//Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003. 2003: 168-171.
- [20]Collobert R, Weston J, Bottou L, et al. Natural language processing (almost) from scratch[J]. Journal of machine learning research, 2011, 12(ARTICLE): 2493– 2537.
- [21]Huang Z, Xu W, Yu K. Bidirectional LSTM-CRF models for sequence tagging[J]. arXiv preprint arXiv:1508.01991, 2015.
- [22]Ekbal A, Bandyopadhyay S. Bengali named entity recognition using support vector machine[C]//Proceedings of the IJCNLP-08 workshop on named entity recognition for south and south east Asian languages. 2008.
- [23]Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. The Journal of Machine Learning Research, 12:2493–2537.
- [24]Ling W, Tsvetkov Y, Amir S, et al. Not all contexts are created equal: Better word representations with variable attention[C]//Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. 2015: 1367-1372.
- [25]Mikolov T, Chen K, Corrado G, et al. Efficient estimation of word representations in vector space[J]. arXiv preprint arXiv:1301.3781, 2013.
- [26]Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge Graph Embedding: A Survey of Approaches and Applications. In TKDE.
- [27]https://blog.csdn.net/qq_27590277/article/details/109006902?utm_medium=distribute.pc_relevant.none-task-blog-2~default~baidujs_baidulandingword~default-1-109006902-blog-79081541.pc_relevant_default&spm=1001.2101.3001.4242.2&utm_relevant_index=4
- [28]Yang B, Yih W, He X, et al. Embedding entities and relations for learning and inference in knowledge bases[J]. arXiv preprint arXiv:1412.6575, 2014.
- [29]Paccanaro, Alberto and Hinton, Geoffrey E. Learning distributed representations of concepts using linear relational embedding. IEEE Transactions on Knowledge and Data Engineering, 13(2): 232–244, 2001.
- [30]Bordes, Antoine, Usunier, Nicolas, Garcia-Duran, Alberto, Weston, Jason, and Yakhnenko, Oksana. Translating embeddings for modeling multi-relational data. In NIPS, 2013b.
- [31]Trouillon T, Welbl J, Riedel S, et al. Complex embeddings for simple link

- prediction[C]//International conference on machine learning. PMLR, 2016: 2071-2080.
- [32]Zhang Q, Chen M, Liu L. A review on entity relation extraction[C]//2017 Second International Conference on Mechanical, Control and Computer Engineering (ICMCCE). IEEE, 2017: 178-183.
- [33]<https://zhuanlan.zhihu.com/p/165886601>
- [34]Rau L F. Extracting company names from text[C]//Proceedings the Seventh IEEE Conference on Artificial Intelligence Application. IEEE Computer Society, 1991: 29, 30, 31, 32-29, 30, 31, 32.
- [35]Liu X, Zhang S, Wei F, et al. Recognizing named entities in tweets[C]//Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies. 2011: 359-367.
- [36]Lin Y F, Tsai T H, Chou W C, et al. A maximum entropy approach to biomedical named entity recognition[C]//Proceedings of the 4th International Conference on Data Mining in Bioinformatics. 2004: 56-61.
- [37]Jain A, Pennacchiotti M. Open entity extraction from web search query logs[C]//Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010). 2010: 510-518.
- [38]Ling W, Tsvetkov Y, Amir S, et al. Not all contexts are created equal: Better word representations with variable attention[C]//Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. 2015: 1367-1372.
- [39]Mikolov T, Le Q V, Sutskever I. Exploiting similarities among languages for machine translation[J]. arXiv preprint arXiv:1309.4168, 2013.
- [40]Mikolov T, Kombrink S, Burget L, et al. Extensions of recurrent neural network language model[C]//2011 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE, 2011: 5528-5531.
- [41]Graves A, Schmidhuber J. Framewise phoneme classification with bidirectional LSTM and other neural network architectures[J]. Neural networks, 2005, 18(5-6): 602-610.
- [42]Elman J L. Distributed representations, simple recurrent networks, and grammatical structure[J]. Machine learning, 1991, 7(2): 195-225.
- [43]Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.
- [44]Graves A, Schmidhuber J. Framewise phoneme classification with bidirectional LSTM and other neural network architectures[J]. Neural networks, 2005, 18(5-6): 602-610.
- [45]Lample G, Ballesteros M, Subramanian S, et al. Neural architectures for named entity

recognition[J]. arXiv preprint arXiv:1603.01360, 2016.

[46]<https://blog.csdn.net/u013250861/article/details/122753115>

[47]Eddy S R. Hidden markov models[J]. Current opinion in structural biology, 1996, 6(3): 361-365.

[48]<https://wenku.baidu.com/view/7a4c4639c6da50e2524de518964bcf84b9d52d10.html>

[49]Lafferty J, McCallum A, Pereira F C N. Conditional random fields: Probabilistic models for segmenting and labeling sequence data[J]. 2001.

[50]Sha F, Pereira F. Shallow parsing with conditional random fields[C]//Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics. 2003: 213-220.

[51]Ekbal A, Haque R, Bandyopadhyay S. Named entity recognition in Bengali: A conditional random field approach[C]//Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II. 2008.

[52]Forney G D. The viterbi algorithm[J]. Proceedings of the IEEE, 1973, 61(3): 268-278

[53]<https://docs.ampligraph.org/en/1.4.0/>

[54]Lei Y, Yang K, Wang B, et al. Response of inland lake dynamics over the Tibetan Plateau to climate change[J]. Climatic Change, 2014, 125(2): 281-290.

[55]Trouillon T, Welbl J, Riedel S, et al. Complex embeddings for simple link prediction[C]//International conference on machine learning. PMLR, 2016: 2071-2080.

[56]<https://docs.ampligraph.org/en/1.0.3/background.html>

[57]<https://geek-docs.com/pytorch/pytorch-tutorial/what-is-pytorch.html>

[58]Lample G, Ballesteros M, Subramanian S, et al. Neural architectures for named entity recognition[J]. arXiv preprint arXiv:1603.01360, 2016.

[59]Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.

[60]https://en.wikipedia.org/wiki/Maximum_likelihood_estimation

[61]<https://github.com/Accenture/AmpliGraph/blob/master/docs/tutorials/AmpliGraphBasicsTutorial.ipynb>