

SHANGHAI JIAO TONG UNIVERSITY

**学士学位论文**

BACHELOR’S THESIS



论文题目：

学生姓名:

学生学号:

专 业:

指导教师:

学院(系):

**Contents**

[Chapter One Introduction 3](#_Toc483769809)

[1.1 Previous Work 3](#_Toc483769810)

[1.2 Similar Problems 5](#_Toc483769811)

[1.3 Challenges 6](#_Toc483769812)

[1.3.1 Linear Reasoning 6](#_Toc483769813)

[1.3.2 Word Representation 9](#_Toc483769814)

[1.3.3 Quality of the graph 13](#_Toc483769815)

[Chapter Two Problem Formulation 16](#_Toc483769816)

[2.1 Fetch graph data 17](#_Toc483769817)

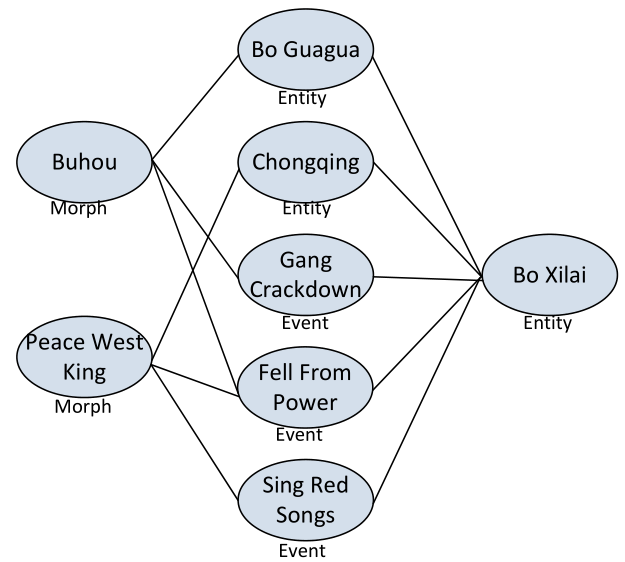
[2.1.1 Extraction from plain text 17](#_Toc483769818)

Chapter One Introduction

The 21st century is an information explosion era. The word ‘Big Data’ is presented to describe the phenomena that numerous amount of data is being generated in the current daily life. Information is exchanged in a rapid speed through Internet. Hotspot no longer springs just one month a time, it springs every day. Moreover, with the wide spreading of smart phone, it has become much more easily for people to fetch the latest information. People currently are used to exploring interesting staff or other information on online social platforms/community, like Sina Weibo, Wechat Group, Zhihu. New terms are created and spread widely since some event that attracts a lot of people happens. These kinds of new terms have a strong relation to the entity they refer, like ‘平西王’ refers to ‘薄熙来’ due to the similarity between these two entities. We call this kind of new term as ‘Morph’, while the one that it refers is called ‘Target Entity’. But due to the aging effect, people who are not familiar with the background of the creation of the morph may feel confused about the origin of the new term. Only when the relevant event is still the hot spot can people easily find the reason of the origination, because numerous We-Media write blogs to analyze and explain the event, which will introduce the meaning of the morph. So when the period of the hot spot passes, it is hard to understand the originating relation between the morph and the target entity. Moreover, due to the blockade of the Internet, some web documents are not accessible to most of people, which aggravate the blindness to understanding the originating relation. Thus it is meaningful to find a way, which can explain such kind of latent relation. This is the purpose of my project.

* 1. Previous Work

Before I presented this project, I’ve read three papers about morph, which inspired me to do this project. Following the temporal order, they are ‘Resolving Entity Morphs in Censored Data’(Huang et al., 2013), ‘Be Appropriate and Funny: Automatic Entity Morph Encoding’ (Zhang et al. 2014), ‘Context-aware Entity Morph Decoding’(Zhang et al. 2015). The first and third paper is to find out morphs from corpus, namely to detect the entity and further detect whether it is a morph or not. The second paper is to generate morphs from some entities. The first paper uses a graph to compute the similarity between a target morph and the base entity, as the Figure X shows. Figure X is the ‘Figure 4: Example of Morph-Related Heterogeneous Information Network’ in (Huang et al., 2013)’s paper.



It calculates the similarity based on the graph, along with other prior knowledge, such as the social features. This graph gives me a lot of inspiration. If machine can learn the strong relation between a morph and its target entity, human beings can also learn from it. But this relation network in Huang’s paper is used to candidate the best morph. It’s a comparison between the numbers. So the absolute number of the score may not be able to reflect the originating relation between the morph and its target entity. The third paper pays more attention on the morph verification and morph candidate ranking. The second paper introduces 7 ways to generate a man-made morph, from phonetic view, spelling view, nickname view, translation and transliteration view, semantic view, historical figure view, and characteristics view. These views show clearly how a morph can relate to an entity.

* 1. Similar Problems

The problem, explaining the originating relation between a morph and its target entity, is very similar to reason the relation between two entities from the Knowledge Graph. This word is presented by Google, whose purpose is to enhance its search engine. This graph is useful in semantic searching. The query like ‘Who is Thomas Jefferson?’ is processed with the help of Knowledge Graph. The nodes within this graph represents the real entities that exist in the real world, the edges within this graph represents the relations between the entities. You may get the result like ‘Born: April 13, 1743’ from this query, inside which ‘Born’ is the relation edge and ‘April 13, 1743’ is the entity node. As to my project, the query may become ‘Why 平西王 refers to 薄熙来?’. It may tell you that they are both related to ‘重庆’, ‘政府’. So it is very similar. But the difference is, to explain the originating relation between the morph and its target entity is a subset of the general Knowledge Graph query, also it needs a more complicated explanation other than just tell you a simple list of relation-entity tuples.

You may hear another word ‘Linked Data’. This term is created by Tim Berners-Lee (the director of W3C), which is a publishing method, aiming to link data to help semantic query. This concept is very similar to Knowledge Graph, but this term becomes a standard to create this kinds of graph. It is based on HTTP, RDF and URI. RDF is used to describe tuples, URI is used to locate resources, HTTP is used as the transport protocol. Following this publishing standard, there are some datasets existed: DBpedia, FOAF, GeoNames, UMBEL, Wikidata. DBpedia is extracted from Wikipedia, has almost 10 thousand tuples in 11 languages. FOAF mainly describe the relation between people. GeoNames has >= 7,500,000 geometric information.

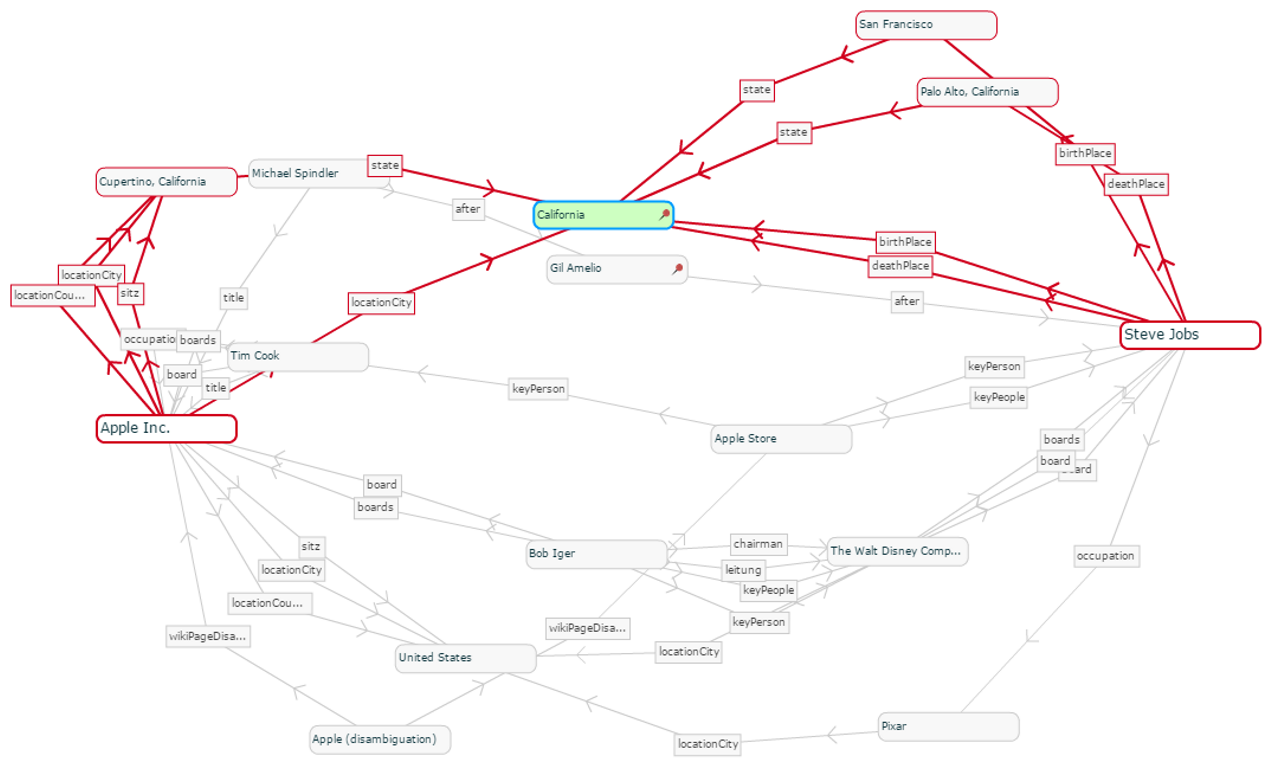
Another good data set is called YAGO. Yet Another Great Ontology. It is extracted from Wikipedia (e.g. categories, redirects, infoboxes), Wordnet (e.g., synsets, hyponymy), and GeoNames. It has need linked to DBpedia ontology and to the SUMO ontology. It provides dumps in Turtle or TSV forms. It also provides query endpoint (Browser or SPARQL).

Ontology is a term which is similar to the namespace in C++. Each field has its own ontology. To separate the knowledge into different fields, we use ontology to describe the aggregation of different knowledge.

Entity Cube is a good relation searching engine build by Microsoft. It is embedded in Bing. When I search ‘薄熙来’ (In Chinese), I can get terms separated into three categories: People, Locations and Organizations. From my perspective, these outs are highly related to ‘Bo Xilai’. But the problem is they are just been put into a big category, which lose the more exact relationship to the target entity. For example, ‘Bo Guagua’ is in the category of People, but it doesn’t show the ‘son’ relation. This is the case when I search in Chinese. The result is better if I search in English. For example, input ‘Bo Xilai’ (In English), ‘Bo Guagua’ is marked the ‘son’ of ‘Bo Xilai’. But the final problem is, it only supports the relations in English corpus. So if I want to leverage this Entity Cube, I have to do a lot of translation work. Which is another hard part of NLP.

* 1. Challenges
     1. Linear Reasoning

Most of the graph reasoning is based on the linear relation chains between two entities, as the Figure Y shows. It illustrates the linear logical path between the entity ‘Steve Jobs’ and the entity ‘Apple Inc.’. They both have one common entity ‘California’, which is the birth/death place of ‘Steve Jobs’ and the location city of ‘Apple Inc.’. People can tell the common part between ‘Steve Jobs’ and ‘Apple Inc.’ easily from this graph. People can say, these two entities are similar due to the common parts they share.



‘Steve Jobs’ has four paths to ‘California’, while ‘Apple Inc.’ has four paths to ‘California’ too. You may wonder, which path pair is the most important one, or, which path can tell the deepest relation between them. From this aspect, we need to compute a score between different path pairs.

One approach is to compute the similarity. This is useful when two entities are similar in prior, like ‘平西王’ and ‘薄熙来’. The sub paths of each relation graph may share some similarity, for example, the relation between ‘平西王’ and ‘重庆’ and the relation between ‘薄熙来’ and ‘重庆’ may be similar. We can even make the constraints looser, such as allowing the end points of two paths are not the same. We can make graph bigger, adding phonetic and spelling relations to the entity.

Besides the similarity between paths, we can also compute the similarity between the sub graph. For graph A = (E1, (R11, E11), (R12, E12)) and graph B = (E2, (R21, E21), (R22, E22)), where E represents the entity node, R represents the relation node. If R11 is similar to R21, R12 is similar to R22, the relation between E11 and E12 is similar to the relation between E21 and E22, then we say these two graphs are similar. This kind of similarity definition can extend the expression ability of the relation graph.

The similarity can help explaining the originating relation between the morph and its target entity, besides the traditional linear logical chains. It can be easily adopted by human beings because we learn new knowledge from comparison. It two items show their similarity, then we can better understand why they are related.

But to define how two relations are similar is very difficult. First, concept has many ways of expression, like the word ‘胜利’, we have a lot of synonyms, like ‘取胜’, ‘打倒’,’摆平’,’获胜’,’成功’,’成就’. They have high similarities, but they are different in characters. Also, they are different in word senses, some are verbs, some are nouns. What’s more, even they are similar, we want to know to which extent are they similar. So we need to score the similarity, not just give a Boolean value to tell whether it is similar or not. We can compute the editing distance, also we can compute the similarity between the context they occur. Second, the noise of the data may interfere the accuracy of the similarity analysis. The data of the relation graph is extracted from web documents like Wikipedia. The content from it are all man-made, so some of them may not follow the format. Also the extraction framework also affects the quality of the information retrieved. Bad format of data may result in the failure of the detection of similarity or the failure of filtering out nonsense similarity. The previous one is called ‘False Negative’, the second one is called ‘False Positive’. For example, instead of ‘胜利’, it is ‘胜利了’, then the additional nonsense word ‘了’ may reduce the similarity between it and ‘打倒’. Another example is, ‘图像大小’. This may refer to the image width of the figure of some entity, but too many of entities have this relation. So at the end this relation has a very high score, but it is very common, whose score should be lowered down.

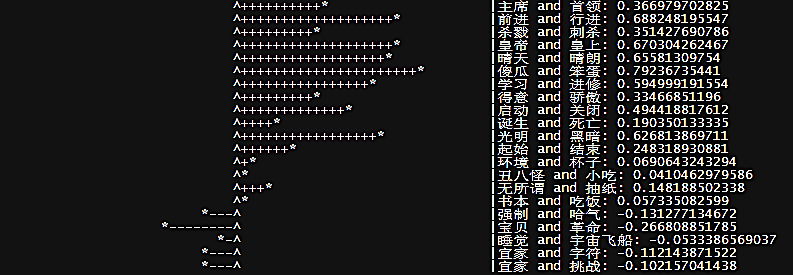
The similarity between entities should also be considered. But most of entities are more abstract than relations. For example, ‘薄熙来’ 🡪 ‘母校’ 🡪 ‘北京大学’, the relation ‘母校’ has a fairly clear meaning, which means the school that one has attended. But ‘薄熙来’ and ‘北京大学’ are abstract, many concepts are related to these two entities. You may say, this project is aimed to explain the similarity between the entities, to compute the similarity between entities should be considered as a sub task. This project aims to use the relation-entity graph to explain the similarity between two entities. The entities inside this relation-entity graph have different levels of abstraction. Some are very obvious, like ‘研究生’. For those not so complex, we may consider explain by its sub relation-entity graph. This means, this a recursive problem. This may result in the endless depth of the problem.

* + 1. Word Representation

Word vector may deal with this problem in a simple but efficient way. Word vector is a vector used to represent a word. The word representation has a closely 30-year history. From 1986 Hinton proposed the concept ‘distributed representation’ to currently Google created a highly-efficient word vector computation algorithm ‘word2vec’, the study to word representation has been continuing for a long time. At the first beginning, people use one-hot representation to represent a word. Within a given corpus, the scale of words is determined. We call the set of words as a dictionary. Inside a dictionary, we give each word an index. The word vector is thus created whose dimension is the size of the dictionary and order is the order of word index. Each word can be represented by a long sparse vector with all zeros but one number one at the position of its index. This kind of representation is useful to turn word into numbers, and use it to do some tasks. For example, to judge the theme of a document. Since each word can be represented as a sparse vector, a sentence can be represented as the sum of these vectors. It is similar to process a document. This is called BOW, bag of words. Due the sparseness of the vector, different document has different distribution of the vector elements. Then we can apply K-means to cluster the documents, thus find different themes. One more step, we can use this model to anticipate the theme of an unknown document. LDA is a more complicated algorithm to deal with this problem.

However, this kind of representation cannot tell the relation between words. Synonyms and antonyms cannot directly be got from this representation. That’s why Hinton proposed ‘distributed representation’. This word currently is called ‘word embedding’, or people can this model as vector space model. This model is famous in information retrieval from corpus whose documents are all in the form of a vector. This vector, which is the sum of word vectors, means the numerical importance of every single word within one document. Thus, two documents’ similarity can be calculated by some similarity measure, such as the cosine method. Word2vec is a fully connected neural network with one hidden layer. It can train a language model for each word as a vector. Similar word will have similar vector, which is computed in cosine distance. Google uses two methods to train word2vec. One is CBOW, continuous bag of words. Another is Skip-Gram Model. In CBOW, each word is surrounded by a context, which consists of some words. CBOW considers this context in the training of each word. Unlike traditional BOW, CBOW scroll a window through each words, instead of computing a single sum of vectors to a sentence or a corpus. This method helps focusing on the context of each word. Skip-Gram Model is similar to CBOW, but actually its strategy is the opposite of CBOW. Skip-gram model makes the target word as the input to the neural network, while the output is repeated to concatenate to be as a context, which should be close the original context of the target word. With the usage of CBOW and Skip-Gram, the neural network trains the words iteratively, and finally make similar words as similar as possible, opposed words as opposed.

Back to this problem. If we want to compute the similarity between entities, we can use word2vec as a solution. Abstract words are hard to be found with other similar words, but word2vec can help do this. Though abstract words are hard to describe its meaning, the context of similar abstract words may share a lot of commons. Thus word2vec will train the vectors for these similar abstract words as similar as possible. For example, within the corpus of Sina Weibo in 2012, the most similar word to ‘总理’ is ‘总书记’, whose cosine similarity is around 0.742168. This model even can detect that ‘平西王’ and ‘薄熙来’ is very close. From this point of view, we can say word2vec is powerful in detecting the similarity between abstract words, but it still cannot explain why they are similar. But it can help us to narrow down the searching area within the relation-entity graph, because the dissimilar entities may indicate that the sub graph of them also cannot reflect the relations between the morph and the target entity.

The usage of word2vec has another problem. If the comparison pair are not single words; say, they are phrases, then how should we compare? The phrase is composed by word. Though most words can find their word embedding, this is not often the same for phrases. To solve this, one straightforward idea is to compare all possible combinations of the words in phrases, then pick the highest score. But this may result ‘False Positive’. For example, ‘termStart’ and ‘termEnd’. Both of them have ‘term’, so it may consider them as a high score pair. But actually, ‘start’ and ‘end’ means totally different meanings. So this score should be given a discount. But this discount should not be too big; because though ‘start’ and ‘end’ are opposite words, their concepts all have relations to the concept of ‘time’. This kind of common part is very latent. From figure X we can see that opposite words ‘光明’ and ‘黑暗’ have a very high similarity even more than the synonym ‘得意’ and ‘骄傲’. Actually the word vector generated by word2vec is not suitable to describe the semantic relation. It can only describe the context similarity. People often mention ‘光明’ and ‘黑暗’ at the same time, or use them separately in the similar scenario. Thus why they share a high similarity. But if two words are totally irrelevant, like ‘宝贝’ and ‘革命’, then they a have strong negative similarity score, as the figure X shows. So we can use word2vec to compute the correlation between two words, not the similarity.

Besides word2vec, another choice is TransE. From this paper ‘Translating Embeddings for Modeling Multi-relational Data’, TransE is a method to learn vector embeddings for both entities and relations. It significantly improves the performance of link prediction between two knowledge graphs. Though the idea of TransE, but is solves the comparison problem of relation. In the model of TransE, the difference between two word embeddings is the vector of a relation. So the comparison between relations can be more abstract and complex, since it doesn’t need to interpret each part of the relation. Also, it is useful to find latent relations, since relations are calculated by the difference of entities, not recoded from information extraction. But to apply TransE, we need a big and accurate graph. In English world, we have big knowledge graph with high quality like ‘Freebase’, but in Chinese language world things are not as simple as in the previous language world. Maybe we can translate the English version of ‘Freebase’ into Chinese, but the word sense disambiguation is not an easy task. So in this project, TransE is not the first choice. I would prefer to use word2vec to be as an experiment. If it works, then it’s find; if it doesn’t, I may try to focus on the information extraction from most Chinese corpus.

* + 1. Quality of the graph

However good the algorithm of similarity between graphs is, the quality of the graph determines the quality of the result. If the nodes inside the graph has a lot of noise, or it is to say that the graph has a low signal-noise-ratio, then the result will be mixed with irrelevant nodes. One kind of the noise is caused by the extraction framework. For example, the linked data dumped from Wikipedia by DBpedia. The Chinese version of this dumped data is mixed of a lot of noise, also it lacks of entity linking, so the connectivity of the graph is very low. For example, one tuple is (‘薄熙来’, ‘儿女’,’薄望知（1977年由李丹宁所生）\n薄瓜瓜（1997年由谷开来所生）’, the third element is very noisy, the better relation tuples should be (‘薄熙来’, ‘儿女’, ’薄望知’), (‘薄望知’,‘母亲’, ‘李丹宁’), (‘薄望知’, ‘生日’, ‘1977’), (‘薄熙来’, ‘儿女’, ’薄瓜瓜’), (‘薄瓜瓜’,‘母亲’, ‘谷开来’), (‘薄瓜瓜’, ‘生日’, ‘1997’). With this extension, the tuples become more clear now, and each third element of the tuple can be represented as an entity, otherwise they are just literals. We cannot compare the similarity between the literals, since they are too long to represent a clear concept. Besides the lack of entity linking, this extraction framework will include many English relations inside the graph. I think it is due to the template mechanism of Wikipedia. The content of Wikipedia follows some format, a structured text format. This template is written in English. Tough you may see Chinese relations in some info-boxes in Wikipedia, actually they are recorded in English in the database. Wikipedia uses a mapping table to map the English relation into Chinese relation. For example, (‘薄熙来’, ‘继任’, ‘张德江’), the relation node of this tuple is ‘继任’, but in Wikipedia database it is recorded as successor[x]. The x is used to distinguish different successor, but the DBpedia extraction framework simply ignores this x. As to this example, I can still use a dictionary to translate it. This is also a sub task of entity linking. But this word, ‘successor’, has no context. It doesn’t appear among a sentence. It just appears as a single word. Many meanings of this word exists, but I don’t know which one should I choose. That’s the problem of the data. It loses some important information. Moreover, relation like ‘termStart’ is more complicated. It is a combination of two words, which works as a marker as the start of a term. The first problem is, how to translate it into Chinese. The second problem is that the concept of it is complex. It works as a verb, meaning when the term starts. This consists of two concepts, one is the noun ‘term’, another is the verb ‘start’. So to compare this compound relation item to other relation item is hard; how can I compare a compound concept to a simple concept?

Even the graph is consisted of a set of extremely clear data, that all meanings of relation is simple and clear, that all non-relation nodes are true entities, the integrity of the graph still affects the quality of the similarity comparison algorithm. For example, the linked data extracted from Info-Box of Wikipedia in Chinese by DBpedia have 8,895,892 lines of tuples, and in total there are 784,840 entities in Chinese Wikipedia. So in average, each entity has 8,895,892/784,840 = 11.3 relation-entity/literal tuples. If we image a relation-entity/literal tuple as attributes of an entity, then 11.3 attributes are too small to describe an entity. Many attributes are very simple, such as the one describing where one was born, who was one’s father, things like these. More meaningful verb-noun pairs are hoped to be added in this graph. We call these verb-noun pairs as events. Extracting events from documents is not an easy work. Even with the help of NLP tools so that we can determine which one is subject, which one is predicate and which one is object, we still cannot extract subject-predicate-object, SPO, from a complex sentence easily. The order of SPO is maybe different from sentence to sentence, also the number of subject, predicate, object is maybe different too. It is hard to make sure which combination of SPO is correct, also the importance of SPO is not easy to be made as a ranking score. We may use dependency analysis a powerful weapon to conquer the challenges of event extraction. With this tool, the relation between many subject, predicate and object items within one sentence is clear. But the running efficiency of this tool is low. With Stanford Core NLP module, applying this algorithm to parse all Chinese Wikipedia documents may take one month to finish. So it is not applicable; it is useful though. Another simpler algorithm is to extract all possible SPO and count the occurrence, then the ones with low occurrence are eliminated. This is workable, but not accurate. It will bring noise to the data. But this can ensure the integrity of the data. So it’s a tradeoff.

Another possible way to extend the graph is to use alias. In Wikipedia articles, editors often use aliases to refer the corresponding entities. For example, ‘薄熙来’ is an entity, but it is also an alias referring to the entity ‘薄熙来事件’. This gives an import extensibility to the graph, because the relation between alias and its source may hard to use a word or phrase to describe, which is the reason why its source doesn’t appear in the Info-Box. But still there some problems with these aliases. First is, how to describe the relation between alias and its source. If it is marked as a universal mark ‘alias’, then it will lose a lot of complex information. For example, in Chinese Wikipedia, ‘北大’ is the alias of ‘北京大学’, but the relation between them is just ‘the same as’. For example, in Chinese Wikipedia, ‘中国’ is the alias of both ‘中华人民共和国’ and ‘中华民国’. The relation between same can be said as ‘the same as’, but should be under different historical background. For another example, in Chinese Wikipedia, ‘苹果’ is the alias of both '苹果 (电影)' and '苹果公司'. In this example, the relation is much more complicated. In daily life, ‘苹果’ refers to a common breed of fruit. But its meaning may differ in other context. So it is hard to use a simple ‘same as’ to be the relation between these kinds of alias and its sources. If we have to use this mark as the relation, then we have to do disambiguation – often we call it as entity linking – to eliminate irrelevant sources.

Chapter Two Problem Formulation

The aim of this project is to explain the originating relation between the morph and its target entity. A morph is something like a metaphor or an alias to a concrete entity. For example, ‘平西王’ is the morph of ‘薄熙来’. People use this to refer to ‘薄熙来’ for some reasons. Maybe it is due to the censorship, or it is just to make fun. The later one is called as sense of humor. The task to detect whether an entity is a morph is called Morph Detection. Take this sentence as an example, ‘庭审的主角无疑是薄熙来,江湖上戏称“平西王”。’The entity ‘平西王’ here is a morph, refers to ‘薄熙来’. But for this sentence, ‘平西王,中国古代封建王朝的一种封号,“王”的地位一般仅次于皇帝。’, the entity ‘平西王’ is not a morph, it just refers to a title. Morph detection is a bit different from entity linking, because it doesn’t need to link actually link the entity, it just need to justify whether is a morph candidate or not. The task to find which entity is the morph that refers to is called Morph Resolution. This step is the further process of Morph Detection. Among all morph candidates, the correlation between them and the target entity is ranked from high to low. If the right morph appears as high as possible, then this model succeeds.

Besides these two tasks, I present a new morph task: explaining the originating relation between the morph and its target entity. Previous tasks can only tell you which are a morph and target entity pair, but they cannot tell you why they are related. So it is meaningful to use some kind of illustration methods to demonstrate how the morph is created for the target entity.

From paper ‘Resolving Entity Morphs in Censored Data’(Huang et al., 2013), Huang used an heterogeneous information network to track the relation between the morph and target entity. At the first beginning, I thought that I can use this network to explain the originating relation between them. Also as I said in the ‘Challenges’ section in chapter one, using linear logical chains inside the knowledge graph has already been studied. So I want to focus more on how to compare the similarity between two graphs. It is well known that people tend to learn new stuff from modeling something similar. From this point of view, it is meaningful to study the similarity between two graphs.

Many sub tasks have to be dealt with for this project. As the ‘Challenges’ section said, the prerequisite for solving this problem is the integrity of the graph, along with its high signal-noise-ratio and high connectivity. Besides it, a good comparison algorithm is also needed. Neither of each part is easy to be solved.

2.1 Fetch graph data

2.1.1 Extraction from plain text

At the very beginning of this project, is to fetch the data. As the ‘Challenges’ section said, it is not an easy problem. One straightforward to do this is to extract the relation tuple from raw text from Wikipedia and Sina Weibo in year 2012. The text in Wikipedia are more formal, and thus with less noise. Also the information in Wikipedia is tighter; it has a lower redundancy than Weibo corpus. But the shortage of it is its advantage; it is so formal that many interesting sentences are not included in it. Also it may lack the ability to include the hot spot in the cyber space.

Considering the timing feature of morph and fruitfully formal concepts that the target entity has, I chose to search the relation tuple in Sina Weibo 2012 for the morph and in Wikipedia for the target entity. For each morph, I plan to search every tweets in Weibo that consist it, and find all terms before and after it and the verbs between them among the same clause to form a (morph, verb, term) or (term, verb, morph) tuple. Here the word ‘term’ refers to a potential entity – a noun or a noun phrase or a verb-object phrase. Then for each term, do this search again, so as to find the tuple whose center is this term. Thus the graph can be expanded. Replicate this iteration, and finally stop before the depth of search reached. For each target entity, I plan to search in Wikipedia and retrieve the tuples similar in what I plan to do in Weibo. Additionally, the text in Wikipedia has a better structure. Many words in Wikipedia text has a hyperlink, some are aliases, some are outer links. This information can be used to expand the graph.

Before I begin to parse the corpus, I have to fetch them. For Weibo corpus, I can download them from a CUHK website. For Wikipedia corpus, at the first beginning, I built a web crawler to fetch the data. Though later I knew that the organization of Wikimedia has published the dumped file of Wikipedia, I found that the dumped one is a bit different from what it shows the wiki page on the website. The biggest difference is from the Info-Box. Also website can be used as a search engine, while the dumped text is hard to be used as a database; this problem has been solved by a SPARQL engine though. Back to the web crawler. I designed a web crawler which can do what a basic web crawler does: fetch one page, extract all links inside it, and finally add this link to the pool. But I have to distinguish the inner links and the outer links. I also designed a thread pool to accelerate the crawling process.

I used several regular expressions to locate the region of content, the links, root link, sub link, tag link:

regionRex = re.compile(r'(?s)<div id="content" class="mw-body" role="main">.\*?</div>(?=\s\*<div id="mw-navigation">)');

linkRex=re.compile( r'<a[^>]\*?href="(?P<href>[^"]\*)"(?:(?!/>)[^>])\*?>\s\*(?!<)(?P<entity\_name>.\*?)</a>');

rootLinkRex = re.compile(r'(?P<rootLink>.\*?://[^/]\*)');

subLinkRex = re.compile(r'^/(?!/).\*');

tagLinkRex = re.compile(r'^#.\*')

Also I designed some cascaded structures to link the wiki pages and entities:

class CascadePage(): def \_\_init\_\_(self, page, cascadeLink): self.page = page; self.cascadeLink = cascadeLink

class CascadeLink(): def \_\_init\_\_(self, link, cascadeEntityName): self.link = link; self.cascadeEntityName = deepcopy(cascadeEntityName);

class CascadeEntityName(): def \_\_init\_\_(self, names): self.names = list(names); def \_\_getitem\_\_(self, index): return self.names[index]; def \_\_len\_\_(self): return len(self.names)

Page and cascaded link are combined together as a cascaded page. Link and the cascaded entity names are combined as a cascaded Link. The list of names is considered as a cascaded entity name. The order of these names means the cascaded relation between them. Say, if A mentions B, C, D, then three CascadeEntityName are created for B, C, D as CascadeEntityName(B) = [CascadeEntityName(A), B], CascadeEntityName(C) = [CascadeEntityName(A), C] and CascadeEntityName (D) = [CascadeEntityName(A), D]. This kind of design can populate the cascaded relation deeper into the search. From this point of view, I used these cascaded structure to build the link extraction framework:

Define a function called as extract\_links with input of a cascaded page. Search the content region according to the regionRex, and build an iterator of a linkRex finder. Iterate all results found by this regular expression, extract the group ‘href’ as the link, parse this link to get the full path of this link, assign this full path link to the link, extract the group ‘entity\_name’ from the found results, cascade this entity\_name to the inputted cascaded page’s cascaded link. Following this way, the newly found link will be cascaded to its parent. This function will expand the graph by one level.

These codes will extract the links according to the regular expression of the link (To locate the links within a certain website, a regular expression is the best choice; It is better than to use a xml parser to parse the whole web documents and then retrieve all hyperlink text.), and record them in a cascaded manner.

The codes have a cache feature. The result of downloaded path is restored in a cascaded folder structure. The following is a sample of the list (74/401) of the names of all links extracted from the wiki page ‘薄熙来’ (each one’s web page is stored in the disk):

三峡大坝，上山下乡运动，世界史，东北，东北地区，中共中央，中共中央书记处，双开，受贿，受贿罪，司法机关，吕福源，吴仪，吴文康，周永康，国务院副总理，尼尔·伍德，尼尔·海伍德，尼爾·海伍德，山东省，山东省济南市中级人民法院，山东省高级人民法院，山西省，市委书记，市政府，市長，布鲁金斯学会，广州，开除党籍、公职，张国光，张德江，张文岳，张晓军，徐明，徐鸣，德维尔，李永金，李铁映，李雪峰，欧元，死亡案，母校，每日电讯报，毛泽东思想，江泽民，汪洋，沈阳市，海伍德死亡案，温家宝，温家宝内阁，滥用职权，滥用职权罪，王奉友，王旭光，王正刚，王立军，王立军事件，王立軍事件，王鸿举，薄一波，薄小莹，薄昌福，薄望知，薄洁莹，薄熙來，薄熙宁，薄熙成，薄熙来事件，薄熙来受贿、贪污、滥用职权一案开庭审理，薄熙来案一审判决书，薄熙来案二审裁定书，薄熙来涉嫌受贿、贪污、滥用职权案提起公诉，薄熙来简历，薄熙来获开除党籍、开除公职处分

From this list we can see a lot of related entities. But these codes haven’t extracted the relations. It is only used to expand the graph. In my design, the expansion of the graph is the first step: fetch the data. The next step is to parse the relation-entity tuples from it. But before I further move on this, I realized that there are already existing some out-of-box data sets that describe the relation-entity network. What’s more, the extraction of relation-entity tuple itself can be a big project. So it makes sense to make this part easier by using something that already existed. But just like what I mentioned in ‘Challenges’ section, the ready-made one may lose the integrity and connectivity of the graph.

Beyond the ‘Entity Cube’, ‘YAGO’ and ‘DBpedia’ that I mentioned before, a group from Fudan University built a CN-DBpedia, whose definition is the chinses version of DBpedia. Table 2.1 is a sample of the searching result of ‘薄熙来’ from CN-DBpedia. The introduction of CN-DBpedia says it is a large scale, universal field applicable, structured encyclopedia. Its data is extracted from Chinese cyclopedia sites, such as Hudong Baike, Baidu Baike and Chinese Wikipedia. It has commercial cooperation with many companies to facilitate different fields of intelligent applications. It has 16,532,759 entities, 213,477,603 of relation-entity tuples. So the average tuples for one entity is 213,477,603/16,532,759 = 12.9. This is even higher than the Chinese data set extracted by DBpedia, which is 11.3. What’s more, the quality of CN-DBpedia is better. Table 2.2 is also a sample of the searching result of ‘薄熙来’ from the Chinese data set of DBpedia. It has 55 tuples, far more than the one from CN-DBpedia. But it has more noise as the same time, like (‘图像大小’, ‘200’), (‘姓名’, ‘薄熙来’). This means, the actual high quality tuples from DBpedia is much less than ones from CN-DBpedia. But the advantage of DBpedia is, from this example, it can extract more information for entities which are under the censorship in China. Since a certain amount of morphs are created due to the censorship, the advantage of DBpedia makes me to use it instead of CN-DBpedia.

Table 2.1 relation-entity tuples of ‘薄熙来’ from CN-DBpedia

|  |  |
| --- | --- |
| Relation | Entity |
| 中文名 | 薄熙来 |
| 主要成就 | 曾任中央政治局委员 |
| 主要成就 | 重庆市委书记 |
| 入党时间 | 1980年10月 |
| 出生地 | 山西定襄 |
| 出生日期 | 1949年07月 |
| 参加工作 | 1968年1月 |
| 国籍 | 中国 |
| 毕业院校 | 中国社会科学院 |
| 民族 | 汉族 |

Table 2.2 relation-entity tuples of ‘薄熙来’ from DBpedia Chinese data

|  |  |
| --- | --- |
| Relation | Entity |
| termStart | 2001 |
| termStart | 2004 |
| termStart | 2007 |
| termStart | 1992 |
| 母校 | 北京市第四中学 |
| 母校 | 中国社会科学院研究生院 |
| 母校 | 北京大学 |
| 图像大小 | 200 |
| office | 遼寧省人民政府省长 |
| office | 大連市人民政府第2任市長 |
| office | 中华人民共和国商务部部长 |
| office | 重庆市委员会书记 |
| office | 中央政治局委员 |
| 专业 | 世界歷史 |
| 专业 | 新闻学 |
| 出生日期 | 1949-07-03 |
| 政黨 | (（1980-2012）) |
| secretary | 曹伯纯于学祥 |
| secretary | 闻世震 |
| 性別 | 男 |
| 父母 | 母亲：胡明 |
| 父母 | 父亲：薄一波 |
| 图像 | VOA-Bo Xilai.jpg |
| 居處 | 秦城监狱(（2012年至今）) |
| 親屬 | 薄熙永(（兄長）) |
| term | 中国共产党第十七届中央委员会 |
| deputy | 王鸿举黄奇帆（市长） |
| 子女 | 薄瓜瓜(（1987年由谷开来所生）) |
| 子女 | 薄望知(（1977年由李丹宇所生）) |
| 出生地點 | （解放区） |
| 籍貫 | 山西省定襄县 |
| premier | 温家宝 |
| termEnd | 2000 |
| termEnd | 2012 |
| termEnd | 2004 |
| termEnd | 2007 |
| 姓名 | 薄熙来 |
| 網站 | http://news.xinhuanet.com/ziliao/2002-02/21/content\_285068.htm |
| generalSecretary | 胡锦涛 |
| 配偶 | 谷开来(（1987年至今）) |
| 配偶 | 李丹宇(（1976年至1984年，离婚）) |
| successor | 张文岳 |
| successor | 陈德铭 |
| successor | 李永金 |
| successor | 张德江 |
| 宗教信仰 | 无 |
| predecessor | 吕福源 |
| predecessor | 张国光 |
| predecessor | 汪洋 |
| predecessor | 魏富海 |
| job | 政治人物 |
| 學歷 | 研究生 |
| 職業 | 政治人物 |
| title | 曾任职务 |
| period | 21 |