IntelligentAutomationAndSoftComputing,2019

Vol.25,no.3,625–635
https://doi.org/10.31209/2019.100000117

A Longest Matching Resource Mapping Algorithm with State Compression Dynamic Programming Optimization

Zhang Min, Teng Haibin, Jiang Ming, Wen Tao, Tang Jingfan

School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou, 310018, China

ABSTRACT

Mapping from sentence phrases to knowledge graph resources is an important step for applications such as search engines, automatic question answering systems based on acknowledge base and knowledge graphs. The existing solution maps a simple phrase to a knowledge graph resource strictly or approximately from the text. However, it is difficult to detect phrases and map the composite semantic resource. This paper proposes a longest matching resource mapping scheme to solve this problem, namely, to find the longest substring in a sentence that can match the knowledge base resource. Based on this scheme, we propose an optimization algorithm based on state compression dynamic programming. Furthermore, we improve the operating efficiency by removing invalid states. Experimental results show that our proposed optimization algorithm considerably improves the efficiency of the benchmark algorithm in terms of running time.

KEY WORDS: Knowledge graphs, NLP application, Phrase detection, question answering systems, Resource mapping, State Compression Dynamic Programming.

1 INTRODUCTION

KNOWLEDGE bases and knowledge graphs have been widely used in search engines and automated question answering systems to improve the accuracy of query results. Traditional search engines can only mechanically retrieve relevant content through keywords, while a search engine or automated question answering system powered by knowledge graphs can understand what the users want and look for answers. To achieve this goal, existing studies (Lopez et al., 2006; Unger et al., 2012; Yahya et al., 2012; Zou et al., 2014) usually begin with the detection of phrases from texts and map them to resources in the knowledge graph.

There are many studies that focus on phrase detection and resource mapping, which use multiple parsers to detect various types of phrases and rely on text corpora to map the relational mappings of the forms (Yahya et al., 2012). Yuanzhe Zhan et al. (2016) used the resource label in the repository to build the resource dictionary and filter the word sequences in the problem though that dictionary. If the *Levenshtein* distance *sim* between resource tags and word sequences is similar, the similarity is higher than a certain threshold θ^3. That is, this word is considered a usable phrase. After selecting candidate words through n-grams, Shizhu He etal. (2014) filter candidate words using constraint conditions such as limiting word spans and then select different schemes to map entities to different types of semantic items.

Existing schemes can strictly match or approximately match phrases and map resources from text, but mapping schemes that are limited to a single word and resource tag will miss the possibility of mapping multiple words to compound words. For example, the *Levenshtein* distance between "*Cyanobacteria*" and "*Cyanobacteria process technology*" will obviously be less than θ^3, and the match will be lost and will map to the wrong item when there are more similar words (Yuanzhe Zhan et al., 2016). Yahayaetal. (2012) choose to map the detected phrases directly to the corresponding types through a dictionary. This scheme also maps "*Cyanobacteria*" to the wrong resources. Although these schemes may infer "*Cyanobacteria process technology*" by structured query construction, this will inevitably increase the depth of semantic processing and the complexity of the knowledge map. We hope to figure out a solution that can solve this problem as simply as possible, during phrase detection and resource mapping.

In this paper, we propose a phrase detection and resource mapping algorithm that can handle complex

CONTACT Zhang Min jiangming@hdu.edu.cn

semantic resources to solve the above problems. In our solution, a single word will be mapped first to the resources in the knowledge graphs that contain it. These resources will be referred to as the relevant set of resources for that word, which in turn is the anchor of the elements within the set. We will find a resource with the highest number of anchors by enumerating a combination of related resource sets for all words. For semantic aggregation considerations, the anchor of the same resource should fall in a small area. After implementing the algorithm, we find that its time overhead is not ideal. Therefore, we propose a dynamic programming optimization algorithm based on state compression. The basic idea of the optimization algorithm is to consider the resource mapping problem of sentences and phrases as a multistage dynamic programming decision process. That is, each word is scanned from left to right in the sentence, and the resource mapping of the scanned word combination is needed as a stage. The best mapping result for each combination will be recursive to the next stage. Then, we will further prune and optimize the algorithm by removing the related resource set, in order to prevent this ineffective state from spreading to the matching of subsequent words. This pruning considerably reduces the running costs. Through the iterative work above, we will be able to find the resource mapping results of all the combinations of the substrings that it cuts down for each word of the sentence. When searching for the best mapping result, we only need to find the states with non-null resource sets and then map words from the set of words and their states, and the elements of its related resource set will be returned as the mapping result.

2 LONGEST MATCHING RESOURCE MAPPING ALGORITHM

2.1 Problem Description

USUALLY, a resource mapping problem can be expressed as follows: for a natural language sentence $S_{NL}=w_1w_2w_3w_4\cdots w_l$, find a substring $s_{il}=w_{i1}w_{i2}...w_{ik}$ such that mapping(s_{il})->*resource'*, where $i_1,i_2,...,i_k \in [1,l]$, *resource'*$\in$*KB*. In general, *KB.*, and s_{il} and resource' will be mapped together as close as possible on a certain spatial scale, and the *resource'* with the highest degree of similarity can be regarded as the result of the mapping of s_{il}. At the time of mapping, we expect the resource that matches $|s_{il}|$ to the largest extent to be the optimal mapping result.

The phrase mapping task, a substring starts with a phrase *s* and tries to identify resources that with high probability correspond to *s*. This step begins with a phrase (one or more words) and attempts to find a set of resources in the underlying KB that correspond to a high probability. For the phrase "Europe",as a example, possible resources in DBpediaare: dbr:Europe(band)[7] (that refers to a Band called Europe), dbr:Europe (that refers to Europe as a continent) and dbr:Europe (dighy)(a particular type of boat).Than, some techniques would be used to determine which of the resources identified during the phrase mapping task are the right ones. In the above example, "Europe" cannot refer to a band or a boat since it does not make sense speak about their population. Therefore, these resources in the Knowledge Base or Knowledge Graph could be link to the nature language (Dennis et al, 2017).

This process is aimed to detect phrase in the question, and map into the resources in KBs (yzzhang et al. 2014). Specifically, the labels of all resources in the employed KBs been exploited to build a resource dictionary. Next, for all the word sequences at dictionary, contained in the question. If the similarity of the Levenshtein distance *sim* between the resource tag and the word sequence is greater than a certain threshold θ^3, the word sequence would be outputedas a detection phrase and select the resource as the corresponding candidate resource. At the same time, set *sim* as the confidence value of the resource and also record the frequency of occurrence of the resource. Note that phrases can be mapped to multiple resources from different KB. Disambiguation is not performed in this step and will be performed in the joint inference step.

Another solution for the phrase mapping task is calculate the distance of word2vec between phrase and resources in the knowledge graph (shizhu et al, 2014). For each phrase detected, can be mapped to the corresponding semantic item in KB (entity, class and relation). For example, software is mapped to dbo:Software, dbo:developer, etc., and California is mapped to dbr:California, dbr:California(wine), etc. In order to mapping phrases to entities, considering that the entities in DBpediaand Wikipedia are consistent, anchor been employed, redirection and disambiguation information from Wikipedia. For mapping phrases to resources in knowledge base such as Wikipedia, considering that classes have lexical variation, especially synonyms, e.g., dbo:Film can be mapped from *film*, *movie* and *show*, we compute the similarity between the phrase and class in the KB with the word2vec tool[8]. The *word2vec* tool computes fixed-length vector representations of words with a recurrent-neural-network based language model (Mikolove et al., 2010). Then calculate the similarity score as specific method and select top-N most similar resources as potential resources. For mapping phrase relations, PATTY (Nakashole et al., 2012) and ReVerb(Fader et al., 2011) a employed. Compute the associations between the ontological relations in DBpediaand the relation patterns in PATTY and ReVerb through instance alignments as in (Berant et al., 2013). Next, if detect a detected phrase is match to some relation pattern, the corresponding ontological relations in DBpedia will be returned as a candidate. This step only generates candidates for every possible

mapping, and the division of the best selection will be performed in the next step.

Consider this sentence as an example: "*We need some technologies to process cyanobacteria*." This sentence under the analyzer should be divided into a string as *We/need/some/technology/to/process/cyanobacteria*. In the ontology library, we have a resource called "*cyanobacterial process technology*". Obviously, we need an algorithm to map the substring of the sentence "technologies *to process cyanobacteria*" to this resource, so we have proposed the Longest Matching Resource Mapping (LMRM) Algorithm.

The idea of LMRM is to find a resource that will be semantically related to as many words as possible in a more compact area of the sentence. That is, "*cyanobacterial process technologies*" contains or is close to "*process*"," *cyanobacterial*" and "*technology*" in the original sentence, and they are considered to be related. Hence, it is believed that "*technology to process cyanobacteria*" is related to "*cyanobacterial process technologies*". The LMRM solution is to find all the resource sets that include all words in the original sentence and find intersections among these sets. We find that "*cyanobacteria process technologies*" appears differently. The largest number in the collection is that where each word of the substring "*processing/cyanobacteria/technology*" falls within "*technology process cyanobacteria*" and has the longest length. Therefore, this sentence, considered to be a part of the "*technology to process cyanobacteria*", will map to this resource.

2.2 Baseline Longest Matching Resource Mapping Algorithm

The Baseline Longest Matching Resource Mapping Algorithm is an implementation of the LMRM, which is aimed at finding long-range corpus resources that are as accurate as possible from a single sentence.

Considering the tightness of semantics, the words that make up a resource will converge in a local area as close as possible. In this local area, the resources pointed to by as many words as possible are obviously more representative of the meaning of the local representation. That is, for any string strin the specified character set, it is divided into a sequence of words w_i by the corresponding analyzer, and the first-order synonyms set of the extended by word vector approximation is c_i. A resource in semantic networks containing ci is considered as a related resource of w_i. All related resources about w_i are contained in the related resource set rrs_i. Enumerating all substrings in the sentence, the elements in the intersection of the related resource sets of words in each substring are obviously related to all the words in the substring. Therefore, when the related resource set intersects with the longest nonempty substring, the resources in the set are the mapping results.

The Baseline Longest Match Resource Mapping Algorithm is as follows:

1) Use the corresponding analyzer to break the sentence into separate words to form a word stream

2) For each word as w_i in the stream of words, find the synonym for w_i and constitute as a semantic set of it denoted by c_i.

3) For the i-th word in the sentence, find all resources containing any of c_i, denoted by *Related Resource[i]*

4) According to the given length n, enumerate each partial sequence of length n in the entire word stream from left to right

5) For a subsequence of $w_iw_{i+1}...w_{i+n-1}$, the enumeration extracts each combination of at least two words, w_iw_{i+1}, $w_iw_{i+1}w_{i+2},...,w_i...w_k...\ w_{n-1},...,w_{n-2}w_{n-1}$,

6) For a composition of$w_i...w_j...w_k$, i<j<k, its potential mapped set of resource

resourceSet = relatedResource[i] ∩ *rrelatedResource[j]* ∩ *relatedResource[k]*

7) For all nonempty resource sets, select a related resource set that was hit as much as possible. That means maximizing $|w_i...w_j...w_k|$ in all compositions, and the nonempty resource set pointed to by that composition is the mapped result of the local area.

Remark:

Step 1) refers to the analyzer, i.e., the word segmentation tool. The word segmentation tools for Romance and Germanic families, such as English, are usually separated by spaces and punctuation. In the Chinese context, special segmentation tools are used.

Step 2) finds synonyms of words needed to establish a synonym and word vector model in advance. In LMRM, the words that have the closest cosine value in the word vector space and the synonym lists that are sorted out by hand-made dictionaries such as Chinese dictionaries are sought. Using inverted indexes and nonrelational databases can improve the efficiency.

Step 3), similar to step 2), creates an inverted index for the resources in the knowledge graphs, which will effectively improve the efficiency of partial retrieval from compound words to the complete resource.

2.3 Complexity Analysis of the Baseline Longest Matching Resource Mapping Algorithm

The BLMRM algorithm enumerates, from left to right, all the participating word states in each of the length N local areas. For a sentence of length L, the longest matching solution of the current region is solved for (L-N+1) local areas. In the process of finding the longest matching solution in each local area, we enumerate all possible mapping combinations. In each of these combinations, each word has two states, "participating mapping" and "not participating in mapping". There are 2^N cases in the local domain of length N that need to be mapped. In summary, the complexity of a sentence should be $O((L-N)*2^N+L*C)$.

In view of the problem of repeated matching in calculations, we propose the following optimization algorithm.

3 LONGEST MATCHING RESOURCE MAPPING ALGORITHM WITH STATE COMPRESSION DYNAMIC PROGRAMMING OPTIMIZATION

STATE compression dynamic programming is a typical dynamic programming. It is usually used in small-scale solutions of NP problems. Although it has exponential-level complexity, the speed is faster than searching, which is an idea worth learning. In the optimal algorithm, we use state compression dynamic programming ideas to optimize the phrase detection and resource mapping described in Chapter 2.

3.1 State compression of word sequences

When performing phrase detection and resource mapping algorithms, we solve all the combinations by enumeration. This causes problems, such as repeated calculations and taking up a lot of space, when the words window is moved. This problem can be avoided by reusing the intermediate results of the already calculated words. To reuse the intermediate results, the solution needs to store certain state data (a data value representing a word state in a substring). Each state datum is usually represented by a binary system. This requires each unit of the state data to have only two states. A bit 0 or 1 of the state string indicates whether the corresponding word participates in the resource mapping calculation. Therefore, the mapping state of the entire string must be a binary number.

For a string $w_1w_2w_3w_4w_5$..., an analyzer generates an isolated word string $w_1|w_2|w_3|w_4|$.... For a 0-1 status string equal to the length of the string, $status_i=1$ indicates that w_i is valid in this word combination, and the set of resources to which it maps should also participate in the mapping operation in this calculation.

As shown in Figure 1, for a substring consisting of $w_1w_2w_3w_4w_5$, the status string 10011 indicates that we want to find some resources that can be mapped by w_1,w_4,w_5 at the same time. Therefore, we will find the resources containing w_1, w_4, and w_5 separately and construct the related resource set rrs_1,rrs_4,rrs_5 for each word. Then, we calculate$rrs_1 \cap rsc_4 \cap rrs_5$ and get the r_3 as the mapping result under this word combination. Obviously, r_3 has three words w_1, w_4, and w_5 as anchor points in the string. The status string 01010 represents the attempt to find the resources to which w_2 and w_4 map to. After obtaining the respective set of related resources and performing the intersect operation, {r_3, r_4} is obtained. This is the optimal result of the mapping in that state.

To quickly locate the relationship between words and related resources, complex semantic resources will be collated and indexed in an inverted index. A string representing a composite semantic resource will be split by the segmentation tool of the corresponding language into the smallest semantic unit, i.e., the word. Each word segmented by a composite semantic resource is linked to the resource itself. Because a word usually has multiple links, this is often stored in the form of a key-value pair in which the word is a key name with a different linked list.

As shown in Figure 2, the resources r_1 and r_2 in the semantic network composed of $w_1w_2w_3w_4$ and $w_3w_2w_6w_7$, respectively, are split into eight pairs of words and resources, i.e., records. These records are indexed separately in the inverted index library based on the word. In this indexing mode, when we need to query the related resources of a word, we only need to find the value with the word as the key name to get all the words that contain it.

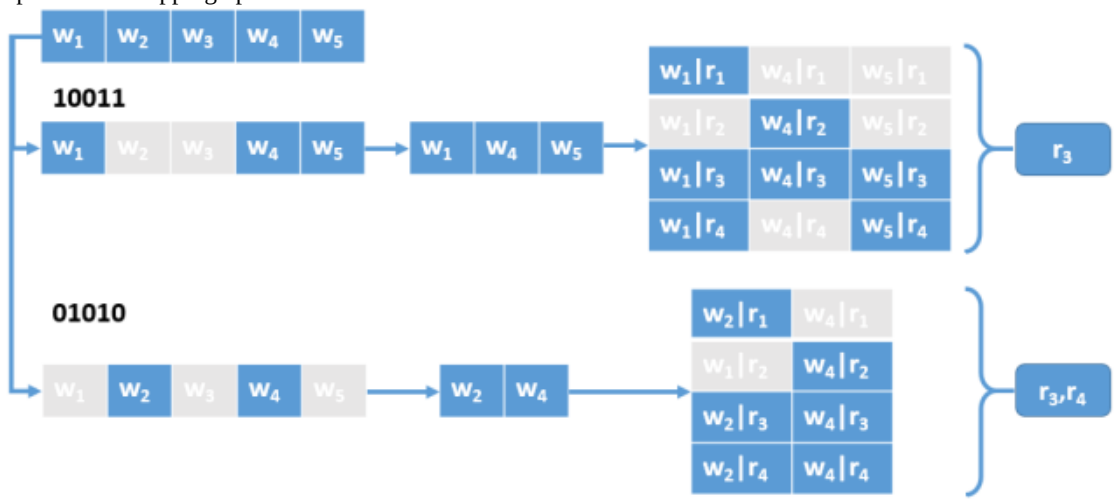

Figure 1. **Relationship between the state string, word and resource set.**

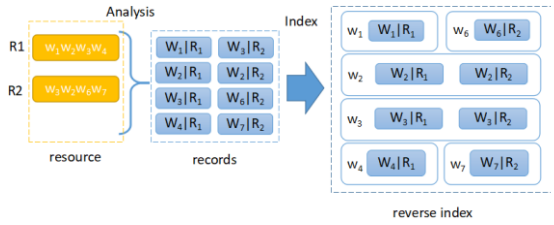

Figure 2. Mapping relationship between a word and resource set.

3.2 State Compression Dynamic Programming

The most important part of the dynamic planning of the multistage decision-making process is to divide the stage and find the state transfer equation. To solve the problem of mapping relations between phrases and resources, we need to find the longest word sequence with a mapped result. Therefore, we can use each new word as a new stage in the process of solving the previous calculation results. Another important factor is the state transfer equation. The state transfer equation is the basis for retrieving state data. To add new words as a new stage, we have constructed the following state transfer equations:

$$dp_{i,status+0} = dp_{i-1,status}, sta \in LS_{i-1} (1)$$

$$dp_{i,status+1} = map(w_i) \cap dp_{i-1,status}, status \in LS_{i-1} (2)$$

In the recursion function (1)(2),$dp_{i,status}$ represents the set of related resources when recursively to w_i and the word combination state is status. In *Function(1)*, w_i does not participate in the operation. Therefore, adding 0 to the end of the state string indicates that the current word is invalid, and the mapping result also inherits directly from the state corresponding to the previous round. In *Function* (2), add 1 to the end of the state string to denote that w_i is related to the mapping result. Therefore, that result of the relevant resource set of w_i intersected with the related resource set under the predecessor state is equal to the resource mapped by w_i and the word effective in state.

Among them, LS_{i-1} is a set of word combination states capable of adding new words recursively up to $word_{i-1}$, that is, a set of available states for performing resource mapping operations with w_i. In the LMRM algorithm, we specify a threshold N as the size of the detection window. Under this constraint, we want to filter out the state where the former in bits are 0. Otherwise, it indicates a word outside the window is involved in the resource mapping operation, and that violates our rules. Therefore, the transfer rules from LS_{i-1} to LS_i are as follows:

for status in LS_{i-1}:
    if status[0] == 0:
        push status+0 and status+1 to LS_i

In the table of new state generation rules, a phenomenon can be observed: after consecutive operations, every element in LS_i has a leading 0 from the beginning to the (i-n+1)th bit, which satisfies the earliest constraint condition and ensures that only all participating mappings are performed when the words of the operation are all within the window of length N that contains w_i. During the detection of a long sentence, the length of the state string increases, and a sentence of length L produces 2^Lstates without proper pruning. An obvious pruning scheme is to stop the extension of the state in which LS_{i-1} cannot perform the mapping operation with w_i, which is in accordance with the processing of the state in which the (i-n+1)th bit is 1 in the LS_{i-1}. This pruning scheme plays a decisive role in the optimization algorithm of our state compression dynamic programming, which reduces the overall computational complexity from $O(2^L)$ of the entire sentence length to $O(L*2^N)$. As resource mapping requires the cohesiveness of words, we look for a relatively short resource as a short substring among very long sentences. This means that the length of N is usually much smaller than L, which means that the optimized space complexity will fall into the correct range.

Taking Figure 3 as an example, the previous operation result set $dp_{0,0}$ derives two result sets $dp_{1,00}$ and $dp_{1,01}$ after querying the common related resource set with word c_i, where $dp_{1,00}$ is the complete set containing all the resources of the knowledge graph, but $dp_{1,01}$ is an empty set representing no common related resources. Obviously, the latter does not have any common related resource set even with other words in the source string, so removing the state string *01* from LS_1 to prevent the invalidation state from further degradation does not affect the accuracy of the calculation result.

The second optimization program for pruning is to remove the empty set state, i.e., to remove status from LS_i when it is found that $dp_{wi,status}$ is already an empty set in one of the LS_i states. Because there is an empty set of results in LS_i, this state will not participate in the generation of the final mapping result, nor will it produce a usable mapping set with subsequent words.

Table 1. Rules of new state generation

state in LS_{i-1}	state at i-n+1	Operate	status in LS_i
00…01XXX	1	Pop	––
00…00XXX	0	mapping calculate with w_i	00…00XXX1
		not mapping calculate with wi	00...00XXX0

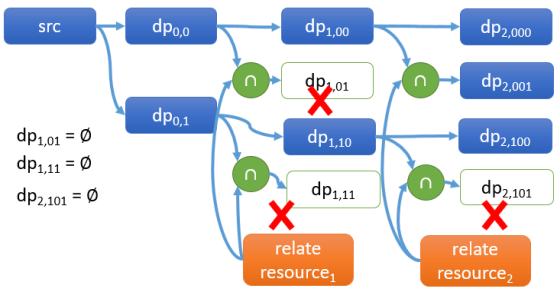

Figure 3. Remove empty resource set state

Another optimization that can be done is removing leading 0s. From the generation rule of the new state of the constraint condition, it can be found that there are many leading 0s of the status string, and the length of the state string starting from 0 is increased from LS_{i-1} to LS_i. Therefore, after we have discarded the state that cannot be mapped with w_i, we can also reduce the length of the state string by further removing the common leading 0s in the state set, reducing the space we need to save the state and reducing the time spent on the retrieval of the state string. When looking for a legal state string, the state of the (i-n)th bit is also judged as the state of the first bit, which leads to a clearer logical representation.

Table 2. Remove Leading 0 in the state

state in LS_{i-1}	first bit	operate	new status	remove leading 0	reality status in LS_i
1XXXX	1	pop	---	---	---
0XXXX	0	map calculation with w_i	0XXXX1	XXXX1	00…00XXXX1
	0	map calculation without w_i	0XXXX0	XXXX0	00…00XXXX0

Table 3. Algorithm for DPLMRM

	Algorithm: Dynamic Programming with State Compression LMRM
1	dp_0 = {0:null, 1:related_resource[w_0]} # dp as a list consist by key-value
2	for i in range(1, l):
3	for i in range(1, l): #scan all word
4	for (status, r_set) in dp_{i-1} items:
5	if status == 0:
6	dp_i = {0:null, 1:related_resource[w_i]}
7	else:
8	if status &first_one != 0:
9	Continue
10	else:
11	tmp_set = $dp_{i-1,status}$ ∩ related_resource[w_i]
12	Iftmp_set != ∅:
13	push ((status<<1)|1, tmp_set) to dp_i
14	push (status<<1, $dp_{i-1,status}$) to dp_i

The longest matching resource mapping algorithm for dynamic programming optimization based on state compression is as follows:

3.3 Longest matching resource mapping results

The dynamic programming after the state compression saves the result of the mapping operation of words under a certain state string. It represents the collection of resources pointed to by these words. The longest matching resource mapping algorithm requires that the set of resources to be mapped to is the one with the most words in this local space. Among the state strings, *status[i]= 1* means that w_i participates in the mapping operation, so the number of 1s in a state string represents the number of words involved in the operation. Thus, the result of the longest matching resource mapping from the intermediate results of dynamic programming is:

for status in LSi:

select max count one in status and $dp_{i,status}$ *!= Øas id*

Return$dp_{i,id}$

In this way, we filter out a local longest matching resource mapping result with w_i as the right boundary.

3.4 Complexity Analysis of the State Compression Dynamic Program LMRM Algorithm

Without considering any optimization such as empty-set pruning, a sentence of length l will be mapped $\sum_i^l 2^i = 2^{i+1}$ times; therefore, obviously its complexity is $O(2^l)$. Through the transfer mechanism of the state set, it can be found that the first n words will cause $\sum_1^n 2^i = 2^{n+1}$ mapping operations, and the n+1th to the i-th words will be generated each time from the last 2^{n-1} legal states. There will be 2^n new states and new states. The number of set operations for the entire sentence is $2n+1+(l-n)*2^n$. In other words, we can reduce the complexity from $O(2^l)$ to $O(l*2^n)$ by removing the invalid preamble state.

Although the state compression dynamic programming LMRM algorithm without null-set state pruning optimization is similar to the simple LMRM algorithm in terms of complexity, the former is much more optimized than the latter in terms of coefficients. The simple LMRM algorithm repeatedly calculates from w_{i-n+1} to w_{i-1} mapping results when the window moves. The state compression dynamic programming LMRM algorithm retains the above calculation result through the state string and can be directly used for the mapping operation with w_i.

In the empty-set states pruning scheme, the states without mapping results will be discarded and will no longer participate in the mapping calculation of the following words. As the actual performance of this optimization is closely related to the sentences and corpora involved in the operation, we cannot obtain a mathematical expression through accurate derivation.

However, in practice, there are considerable differences between the single word mapping resource sets in the neighboring domain. This makes it very common for the result to be an empty set and to be discarded. Because the number of valid nonempty mapping states will be very small, this optimization works well in practical use.

4 EXPERIMENT

TO verify the related calculations of time complexity in Chapter 4, an experiment comparing the running time of BLMRM and DPLMRM was conducted. The experimental platform is a MacbookPro 13-inch with retina, the CPU is Intel i5-6300, the memory is 8G, and the operating system is Mac OS High Sierra. Both algorithms are implemented using Java, JDK version 1.8.0_66. To speed up the retrieval of knowledge graph resources, we use the reverse indexing service provided by Lucene, whose version is Apache Lucene 6.6.1.

The experimental dataset was based on the acquisition of 51,520 articles on water conservation and 10,801 articles on hydrological policy. These materials constitute our hydrology corpus. We randomly selected 10 documents as experimental materials. After data cleaning, ensuring that the literature is composed of plain text, we divide the content into separate clauses by breaking ideographic symbols, such as commas, periods, and semicolons, and then we look for resources that each local area may map to in clauses through the BLMRM and DPLMRM.

The data we get from the government website will be cleaned. In the pre-processing, data such as policy messages in different formats are converted into plain text files. Then select a dedicated tokenizer to divide these articles into word streams. These word flows flow into the BLMRM and DPLMRM resource mappers, respectively, to record their time and space overhead.

Considering that the length of the articles' content and the length of the sentences differ considerably, the two LMRM algorithms are more dependent on the length of the sentences, that is, the number of words in the sentences. It is obvious that it is more reasonable for the cumulative number of detected words rather than the number of documents or the number of sentences to serve as a benchmark for time comparison. Simultaneously, the size of the local area N is also an important indicator. We compare the time performance of time consumption of N=3, N=5 and N=8. The comparison results are shown in Fig. 4. The abscissa shows the number of accumulated words of the test content, and the horizontal axis shows the time consumed by the algorithm.

In Figure 4, we can see that the line representing the BLMRM algorithm has some fluctuation when N=3, while there is a simple linear increase when N=5 and N=8. Furthermore, DPLMRM shows a simple linear growth with N=3, N=5, and N=8, which means that the BLMRM can be considerably influenced by the content when the window is small. This can be blamed on the fact that the words in some sentences have several relevant resources, leading to high time consumption when calculating the intersection set. As the window becomes larger, the number of calculated intersections within the window increases, and the fluctuations of a few words with several related resources are covered.

Another obvious fact is that, despite the approximately linear growth of the two, regardless of the size of the window, RLMRL costs much more to run than DPLMRM. As N increases, the proportion of the two will also increase accordingly, consistent with the analysis of BLMRM and DPLMRM in section 3.5 that when a local domain is moved to a new word, a larger local window means more repeated calculations, while the number of calculations for DPLMRM decreases further.

From Fig. 4d and Fig. 4e, it can be found that with the growth of N, the running time overhead of the BLMRM becomes drastically larger, while the time growth of the DPLMRM is in a very limited range. Furthermore, in the experiments where N = 5 and N = 8, there was a time when the situation was closer. This is consistent with our previous expectation that BLMRM has a large number of repetitive operations, while DPLMRM can reduce the number of operations by generating states through the multiplexing and iteration of optimal substructures.

These figures show that in the BLMRM algorithm, the nature language string to be processed is split into a stream of words by corresponding analyzer. A sliding window traverses the word stream. In the words in the sliding window, each potential combination of words is enumerated and the corresponding set of related resources is obtained. Among all the states that have a collection of non-empty related resources, the most useful words, that is, the set of related resources to which the longest substring is mapped, is the mapping result. In DPLMRM, the combination of words is represented by a status string represented by the 01 binary string. The size of the sliding window is controlled by constraining the derived condition of the state string. The meaningless matching calculation is pruned by filtering the status string that no longer matches the new word. With a variety of optimization

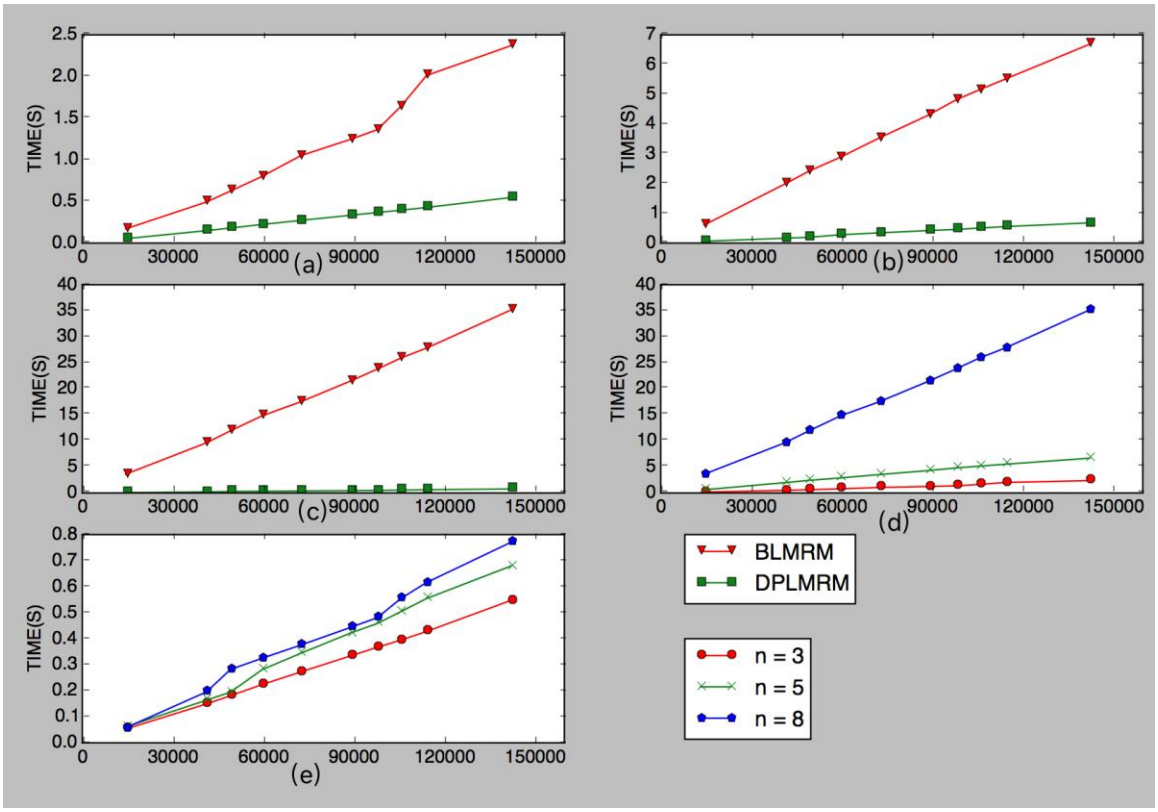

Figure 4. **Performance for BLMRM and DPLMRM**

measures, DPLMRM has significant performance optimizations over BLMRM in terms of time and space overhead.

5 CONCLUSIONS

TO map phrases in sentences to the composite semantic resources, a longest matching resource mapping scheme is proposed in this paper. This solution solves the problem of mapping the composite semantic resources in the resource mapping phase by finding the longest substring in the sentence that can match the knowledge graphs resource, without additional means such as structured query. This work effectively reduces the depth of semantic processing and reduces the complexity of knowledge graph construction.

We propose an optimization algorithm based on state compression dynamic programming to increase the processing speed. Simultaneously, with state compression, we can reduce the running overhead by removing invalid state pruning schemes. Experimental results show that the proposed optimization algorithm considerably improves the efficiency of the baseline algorithm in terms of time consumption.

6 REFERENCES

Amaral C, Figueira H, Martins A, et al. Priberam's Question Answering System for Portuguese[M]// Accessing Multilingual Information Repositories. Springer Berlin Heidelberg, 2005:364-371.

Blooma M J, Chua Y K, Goh H L, et al. Towards a Hierarchical Framework for Predicting the Best Answer in a Question Answering System[M]// Asian Digital Libraries. Looking Back 10 Years and Forging New Frontiers. Springer Berlin Heidelberg, 2007:497-498.

Breck E, Burger J D, Ferro L, et al. How to Evaluate your Question Answering System Every Day and Still Get Real Work Done[C]// 2000:1495--1500.

Buchholz S, Daelemans W. Complex answers: a case study using a WWW question answering system[J]. Natural Language Engineering, 2002, 7(4):301-323.

Cao, Y. G., Liu, F., Simpson, P., Antieau, L., Bennett, A., & Cimino, J. J., et al. (2011). Askhermes: an online question answering system for complex clinical questions. Journal of Biomedical Informatics, 44(2), 277-288.

Cimiano, P., Lopez, V., Unger, C., Cabrio, E., Ngomo, A. C. N., & Walter, S. (2013, September). Multilingual question answering over linked data (qald-3): Lab overview. In International Conference of the Cross-Language Evaluation Forum for European Languages (pp. 321-332). Springer, Berlin, Heidelberg.

Collaborative QoS Prediction for Mobile Service with Data Filtering and SlopeOne Model. Mobile Information Systems 2017: 7356213:1-7356213:14 (2017)

Collaborative Service Selection via Ensemble Learning in Mixed Mobile Network Environments. Entropy 19(7): 358 (2017)

Dima, C. (2013, September). Intui2: A Prototype System for Question Answering over Linked Data. In CLEF (Working Notes).

Güler F M, Birturk A. Natural intelligence: commonsense question answering with conceptual graphs[C]// International Conference on Conceptual Structures: From Information To Intelligence. Springer-Verlag, 2010:97-107.

Hao Li, Yong Ma, Kun Liang, et al. Rapid matching algorithm for hyperspectral image based on norm sifting[J]. Chinese Optics Letters, 2012(1):67-70.

He, S., Liu, K., Ji, G., & Zhao, J. (2015, October). Learning to represent knowledge graphs with gaussian embedding. In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (pp. 623-632). ACM.

He, S., Liu, K., Zhang, Y., Xu, L., & Zhao, J. (2014). Question answering over linked data using first-order logic. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 1092-1103).

Heck L, Huang H. Deep learning of knowledge graph embeddings for semantic parsing of Twitter dialogs[C]// Signal and Information Processing. IEEE, 2015:597-601.

Hu H, Jiang P, Ren F, et al. A New Question Answering System for Chinese Restricted Domain[J]. IEICE - Transactions on Information and Systems, 2006, E89-D(6):1848-1859.

Jia K, Pang X, Li Z, et al. Query expansion based on semantics and statistics in Chinese question answering system[J]. Wuhan University Journal of Natural Sciences, 2008, 13(4):505-508.

Jo, Y., Lagoze, C., & Giles, C. L. (2007, August). Detecting research topics via the correlation between graphs and texts. In Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 370-379). ACM.

Katz B, Lin J, Felshin S. Gathering Knowledge for a Question Answering System from Heterogeneous Information Sources[C]// Human Language Technology. 2001.

Li, H., Wang, Y., de Melo, G., Tu, C., & Chen, B. (2017, April). Multimodal question answering over structured data with ambiguous entities. In Proceedings of the 26th International Conference on World Wide Web Companion (pp. 79-88). International World Wide Web Conferences Steering Committee.

Liu P, Li L, Li Z. Research on the Question Answering System in Chinese Based on Knowledge Represent of Conceptual Graphs[M]// Applied Informatics and Communication. Springer Berlin Heidelberg, 2011:205-214.

Liu P, Li L, Li Z. Research on the Question Answering System in Chinese Based on Knowledge Represent of Conceptual Graphs[C]// The, International Conference on Computational Intelligence and Industrial Application. 2010:132-133-134-135-136.

Lukovnikov D, Fischer A, Lehmann J. Neural Network-based Question Answering over Knowledge Graphs on Word and Character Level[C]// International Conference on World Wide Web. International World Wide Web Conferences Steering Committee, 2017:1211-1220.

Molaei, M. R., Anvari, M. H., & Haqiri, T. (2007). On relative semi-dynamical systems. Intelligent Automation & Soft Computing, 13(4), 405-413.

Ning Q, Wang Q, Zou Y, et al. Intelligent Question Answering System Based on Data Mining[J]. Journal of Zhengzhou University, 2007.

Park, S., Kwon, S., Kim, B., Han, S., Shim, H., & Lee, G. G. (2015). Question Answering system using multiple information source and open type answer merge. In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations (pp. 111-115).

Pechsiri C, Piriyakul R. Developing a Why–How Question Answering system on community web boards with a causality graph including procedural knowledge[J]. Information Processing in Agriculture, 2016, 3(1):36-53.

Pujara J, Miao H, Getoor L, et al. Ontology-aware partitioning for knowledge graph identification[J]. 2013, 71(7):19-24.

Purwarianti A, Tsuchiya M, Nakagawa S. A machine learning approach for Indonesian question answering system[C]// Iasted International Conference on Artificial Intelligence and Applications. 2007:573-578.

QoS Prediction for Web Service Recommendation with Network Location-Aware Neighbor Selection. International Journal of Software Engineering and Knowledge Engineering 26(4): 611-632 (2016)

Sadeghi, F., Kumar Divvala, S. K., & Farhadi, A. (2015). Viske: Visual knowledge extraction and question answering by visual verification of relation phrases. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1456-1464).

Selvi, P., & Bnerjee, A. K. (2010). Automatic short - answer grading system (asags). Computer Science.

Sherkat, E., & Farhoodi, M. (2014). A hybrid approach for question classification in Persian automatic question answering systems. International Econference on Computer and Knowledge Engineering (pp.279-284). IEEE.

Suchanek, F., & Weikum, G. (2013, June). Knowledge harvesting in the big-data era. In Proceedings of the 2013 ACM SIGMOD International

Conference on Management of Data (pp. 933-938). ACM.
Sun, J., Shaban, K., Podder, S., & Karry, F. (2003). Fuzzy semantic measurement for synonymy and its application in an automatic question-answering system. International Conference on Natural Language Processing and Knowledge Engineering, 2003. Proceedings (pp.263-268). IEEE.
Sven Hartrumpf. Extending Knowledge and Deepening Linguistic Processing for the Question Answering System InSicht[J]. 2005.
Taieb M A H, Aouicha M B, Hamadou A B. Computing semantic relatedness using Wikipedia features[J]. Knowledge-Based Systems, 2013, 50(50):260-278.
Vang K J. Ethics of Google's Knowledge Graph: some considerations[J]. Journal of Information Communication & Ethics in Society, 2014, 11(4):245-260(16).
Verner, I., & Ahlgren, D. (2007). Robot projects and competitions as education design experiments. Intelligent Automation & Soft Computing,13(1), 57-68.
Wang, Z., Zhang, J., Feng, J., & Chen, Z. (2014). Knowledge graph and text jointly embedding. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1591-1601).
Xia, Q., Chang, B., & Sui, Z. (2017). A Progressive Learning Approach to Chinese SRL Using Heterogeneous Data. arXiv preprint arXiv:1702.06740.
Xu, Kun, Reddy, Siva, Feng, Yansong, et al. Question Answering on Freebase via Relation Extraction and Textual Evidence[J]. 2016:2326-2336.
Yahya, M., Berberich, K., Elbassuoni, S., Ramanath, M., Tresp, V., & Weikum, G. (2012). Natural language questions for the web of data. Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (pp.379-390). Association for Computational Linguistics.
Yahya, M., Berberich, K., Elbassuoni, S., Ramanath, M., Tresp, V., & Weikum, G. (2012, April). Deep answers for naturally asked questions on the web of data. In Proceedings of the 21st international conference on World Wide Web (pp. 445-449). ACM.
Yamamoto Y, Tanaka K. Towards Web Search by Sentence Queries: Asking the Web for Query Substitutions[C]// Database Systems for Advanced Applications -, International Conference, DASFAA 2011, Hong Kong, China, April 22-25, 2011, Proceedings. DBLP, 2011:83-92.
Yin Y Y, Li Y, Deng S G, et al. Verifying Consistency of Web Services Behavior[C]// Asia-Pacific Services Computing Conference, 2008. APSCC '08. IEEE. IEEE, 2009:1308-1314.
Yin Y Y, Wu Q, Wu B. Towards High-Availability for Services Oriented Application[C]// International Symposium on Computer, Consumer and Control. IEEE, 2012:60-63.
Yuyu Yin, Song Aihua, Gao Min, Yueshen Xu, Wang Shuoping:
Yuyu Yin, Wenting Xu, Yueshen Xu, He Li, Lifeng Yu:
Yuyu Yin, Yueshen Xu, Wenting Xu, Min Gao, Lifeng Yu, Yujie Pei:
Zhang, Y., He, S., Liu, K., & Zhao, J. (2016, February). A Joint Model for Question Answering over Multiple Knowledge Bases. In AAAI (pp. 3094-3100).
Zhuang, H., & Wongsoontorn, S. (2008). Design and tuning of fuzzy control surfaces with bezier functions. Intelligent Automation & Soft Computing, 14(1), 13-28.

7 NOTES ON CONTRIBUTORS

Min Zhang received the Ph.D. degree in computer science from Zhejiang University in 2012, and currently works at College of Computer Science at Hangzhou Dianzi University. His research areas include image processing, machine vision and machine learning.

HaibinTeng is studying for a master's degree in computer science and technology, Hangzhou Dianzi University, HangZhou, China. He is currently doing research on nature language processing based on deep learning, and focus about apply NLP in computer game.

Ming Jiang received the Ph.D. degree in computer science from Zhejiang University in 2004, and currently is a professor of College of Computer Science at Hangzhou Dianzi University. His research areas include data mining, image processing and artificial intelligence.

Tao Wen is studying for a master's degree in Software Engineering, Hangzhou Dianzi University, HangZhou, China. He is currently doing research on natural language processing based on deep learning.

Tang Jingfan, received the Ph.D. degree in computer science from Zhejiang University in 2005, and currently works at College of Computer Science at Hangzhou Dianzi University. His research areas include Software Engineering and artificial intelligence.