THU QUANTA at TAC 2009 KBP and RTE Track

Fangtao Li, Zhicheng Zheng, Fan Bu, Yang Tang, Xiaoyan Zhu, Minlie Huang
State Key Laboratory on Intelligent Technology and Systems,
Tsinghua National Laboratory of Intelligent Technology and Systems (LITS),
Department of Computer Science and Technology,
Tsinghua University, Beijing, 100084, China
Fangtao06@gmail.com; zxy-dcs@tsinghua.edu.cn;

Abstract

This paper describes the systems of THU QUANTA in Text Analysis Conference (TAC) 2009. We participated in the Knowledge Base Population (KBP) track, and the Recognizing Textual Entailment (RTE) track. For the KBP track, we investigate two ranking strategies for Entity Linking task. We employ a Listwise "Learning to Rank" model and Augmenting Naïve Bayes model to rank the candidate. We try to use learned patterns to solve the Slot Filling task. For the RTE track, we propose an interesting method, SEGraph (Semantic Elements based Graph). This method divides the Hypothesis and Text into two types of semantic elements: Entity Semantic Element and Relation Semantic Element. The SEGraph is then constructed, with Entity Elements as nodes, and Relation Elements as edges for both Text and Hypothesis. Finally we recognize the textual entailment based on the SEGraph of Text and SEGraph of Hypothesis. The evaluation results show that our proposed two frameworks are very effective for KBP and RTE tasks, respectively.

1 Introduction

In this year's Text Analysis Conference, we participated in two tracks: the Knowledge Base Population (KBP) track and the Recognizing Textual Entailment (RTE) track. This paper reports on our developed systems for the two tracks.

This is the first year for KBP track. It is designed to discover information about named entity and to incorporate this information in a knowledge source. It contains two subtasks: Entity Linking task that links names to entities in the Knowledge Base, and Slot Filling task that extracts related attributes about entities from text. Most of previous studies focus on using similarity or classification based strategies to solve Entity Linking task. In this paper, we investigate two types of "Learning to Rank" strategies. We employ a Listwise learning to rank model and Augmenting Naïve Bayes model to rank the entity candidates. For Slot Filling task, we try to use learned patterns to extract the attributes. We also employ Knowledge Base information and Wikipedia to improve the performance. From the evaluation results, we can see that our proposed methods are effective for these two tasks.

In this year's RTE task, we propose an interesting method, called SEGraph (Semantic Elements based Graph), for recognizing textual entailment. This method divides the Hypothesis and Text into two types of semantic elements: Entity Semantic Element and Relation Semantic Element, where Entity Semantic Element describes the entity referred in the text, and Relation Semantic Element describes the relations between the Entity Semantic Elements. We then construct SEGraph, with Entity Elements as nodes, and Relation Elements as edges for both Text and Hypothesis. We recognize the textual entailment based on the SEGraph of Text and SEGraph of Hypothesis. Due to different degree of variations for entity and relation, we employ different strategies to detect the Entity Element Entailment and Relation Element Entailment. The Entity Element Entailment is recognized with knowledge based method, and Relation Element Entailment is determined by supervised classification method. In this year's RTE track, the best result we achieved is 67.0% in accuracy and 70.1% in average precision.

2 KBP Track

The aim of KBP Track is to automatically increase the existed Knowledge Base, such as Wikipedia. To achieve the purpose, there are 2 related tasks in the Track: Entity Linking, where names must be aligned to entities in the KB, and

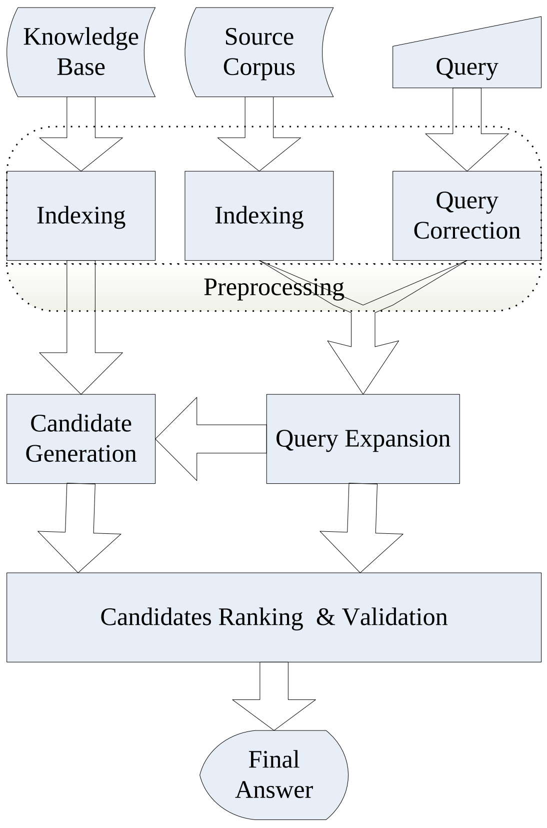


Fig. 1, Framework for Entity Linking task

Slot Filling, which involves mining information about entities from text.

2.1 KBP Entity Linking

In the task, we need to process a list of queries. Each query contains an entity's name string and a reference to disambiguating text from news document set. We should return the corresponding Entity ID which represents an Entity in the Knowledge Base if the Entity is in the Knowledge Base or return NIL if the Entity isn't in the Knowledge Base. We dealt with the task as the framework shown in Figure 1.

2.1.1 Preprocessing

In the first modules, we have to preprocess the corpus data and the query data.

We get two corpus from the KBP Track: The Knowledge Base Corpus (formed by some selected articles from Wikipedia), and The Source Corpus (formed by news articles, used in disambiguation).

To deal with the task, we index the 2 corpus. For the KB Corpus, we build indexes on 2 fields: the title of the article and the text of the article. For the Source Corpus, we simply build index on the text of the news articles.

Although most of the queries' name strings are well formed, there are still some spelling errors. So we try to correct the spelling errors. We used the query correction function supplied by search engines like Google, AltaVista etc. We input the query's name string in the search engine, and then the search engine will return a corrected spelling of the name string if the original one was wrong.

2.1.2 Query expansion

It's obviously that is not sufficient if we only use the query's name string as our query to search in Knowledge Base. Some query strings like "MND" (short for Ministry of National Defense or others), it's quite hard to retrieve the correct entity from Knowledge Base without any expansion. Therefore, we use 3 ways to expand the query.

(a) Expand the query from the source text given in the query. This is used for expanding those query strings which are abbreviations. If the query contains only one word which starts and ends with both capital letters, we would look for its expansion form in the source text. We define the expansion form of the original name string is: there are continuous k words; all words start with capital letters or are stop words; if we concatenate the k words' first letter (with or without the stop word), it will form the original word. Sometimes we may find the expansion form of the name string in the source text. If the expansion form was found, then we use this to replace the original name string. In order to express easily and clearly, we use Q to represent a query need to be dealt with, Q.nameString represents the name string of Q, Q.sourceText represents the source text of Q, and Q.querySet represents the queries used to retrieve entity in Knowledge Base. Q.querySet at least contains one query: Q.nameString.

(b) Only a name string is not sufficient even if we process the step (a). Sometimes we can't get the expansion form of an abbreviated name string in the source text because of the abbreviated form is well known, or sometimes the query is a nick name such as "Gus Dur" (nick name for Abdurrahman Wahid). Wikipedia use the redirect link to avoid same entity occurring many times due to different queries. So we collected the redirect links of English Wikipedia (All the articles before May, 2009), and use these to form additional query. For a given Q.nameString, if there is a redirect link that redirects Q.nameString to target article of Wikipedia, we add the target article's title string to the query set of Q.

(c) The articles in Wikipedia contain a lot of cross-over links. The anchor text of the links supply us some more information. For a given Q.nameString, if there is an anchor text of cross-over link equals with Q.nameString, then we add the target article's title string to the query set of Q. To filter out the noises, we select those anchor text and cross-over link pairs occur at least twice and only add the top 5.

The mainly function of the module is to generate the Q.querySet.

2.1.3 Candidates Generation

In the module, we have got an expanded query set. To generate candidates from the Knowledge Base, we search the Knowledge Base with these queries.

For each query string in the query set Q.querySet, we treat all these words in the string as "or"-relationship, and then retrieve top N (N set to be 20 in practice) in Knowledge Base Corpus on the field of title. Now we get a candidate set called CSet.

We also use the Q.nameString to retrieve entities in the Knowledge Base on the field of text, we treat the words in the string as "and"-relationship, and restrict them to occur together with a gap no more than k (k set to be 2 in practice) words. We also add the top N entities to the CSet, and define Q.textRetrievalSet as the all entities retrieved here.

The mainly function of the module is to generate the CSet.

2.1.4 Ranking Candidates & Final Answer Selection

All of our three submits are the same in the first two modules, but they differ in this module. We tried two different methods to rank the candidates and validate the final answer.

We use the sample corpus (version 2) given by TAC committee and the training set supplied by Paul McNamee.

2.1.4.1 List wise Learning to rank & 2-class SVM

After generating CSet, we need to output the final answer. An intuitive method is to train a classifier to classify each candidate into 2-class (whether is the target entity of the query). If more than one candidate is classified as target entity, then pick the one with highest probability as final answer. But here it would meet a problem that is how to balance the 2-class training samples' proportion in the training set. For each query Q, we may get a CSet contains a lot of candidates, most of which are negative samples and at most one will be positive sample. So we change the strategy: we first rank the candidates in the CSet, and then use a 2-class classifier to judge whether the top 1 entity is target entity of the query Q.

2.1.4.1.1 List wise Learning to rank —— ListNet
Obviously, we can extract a lot of features from the candidates in CSet, but how we can combine all the features to rank the candidates. We adopt ListNet, an algorism of learning to rank posted by Zhe Cao etc in [1].

There are totally 416 (119 + 297) queries in the training set, and 285(75+210) of them have the target entity in the Knowledge Base. We use these 285 queries as training set to train a rank model to rank the CSet. We adopt the ListNet with k = 1(same as stated in [1], and it fit our problem quite well) and also use the linear Neural Network model in the ListNet.

The features used for ranking are list in Table 1.

Table 1. Features used in our ranking mode

Name of Feature	**Value Type**	**Feature Type**	**Definition for Feature**
StrSimSurface	Double	Surface	$\mathrm{Max}_{s \in \mathrm{C.titleExpand}} \left(\mathrm{Similarity}(s, \mathrm{Q.nameString})\right)$
ExactEqualSurface	{0,1}	Surface	Whether there is a string s in C.titleExpand, and s equals with Q.nameString.
ContainsQuery	{0,1}	Surface	Whether there is a string s in C.titleExpand and s contains Q.nameString.
SubstringInQuery	{0,1}	Surface	Whether there is a string s in C.titleExpand and s was a sub-string of Q.nameString.
EqualWordNumSurface	Int	Surface	For each string s in C.titleExpand, calculate how many words are the same between s and Q.nameString, and the maximum value would be the value of this feature.
MissWordNumSurface	Int	Surface	For each string s in C.titleExpand, calculate how many words are the different between s and Q.nameString, and the minimum value would be the value of this feature.
TFSimContext	Double	Context	Calculate the TF-IDF similarity between the C.article and Q.sourceText (In practice, we just treat all words' IDF value as 1).
TFSimRankContext	Double	Context	$\frac{1}{\mathrm{Rank\ of\ C.TFsimRankContext\ in\ CSet}}$
AllWordsInSource	{0,1}	Context	Whether all words in C.title exist in Q.sourceText
QueryInArticle	{0,1}	Context	Whether Q.textRetrievalSet contains C
NENumMatch	Int	Context	Calculate the number of same name entity between C.nameEntitySet and Q.nameEntitySet
NENumMissed	Int	Context	Calculate the number of the missing name entities in Q.nameEntitySet compared to C.nameEntitySet.
CountryInTextMatchPer	double	Context	$\frac{\sum_{c \in (\mathrm{Q.countrySet} \cap \mathrm{C.countrySet})} \mathrm{num}\,(c\ \mathrm{exist\ in\ Q.sourceText})}{\sum_{c \in Q.countrySet} num\,(c\ exist\ in\ Q.sourceText)}$
TypeMatch	{0,1}	Other	Whether the C.type equals to Q.type, if C.type is UNK, then the value is also 1, same as the situation of equal.

Here, C represents a candidate in CSet. C.title represents the title of corresponding Wikipedia article of C. C.titleExpand represents the union set of the redirect set of C and the anchor text set of C. C.article represents the Wikipedia article of C. C.nameEntitySet represents the set of all name entity in C.article labeled by Stanford NER. C.type would be one in set {PER, ORG, GPE, UNK}, and is labeled already in the Knowledge Base. Q.type should be labeled by us. To label it, we simply use Stanford NER to label Q.nameString and Q.nameString in Q.sourceText, and then combine the result with linear weight sum method.

All the features list in the table can be divided into 3 groups: Surface, Context and Other.

The features in Surface group are used to measure the similarity between query string and candidate entity's name string.

The features in Context group are used to measure the context relativity between query and the candidate entity. TF-IDF is a raw way to measure this. The name entity co-occurrence also can reflect the relativity. By analysis of the text, we find that country is a quit significant entity typ, so we add it to the feature set.

The KBP Track already defines 3 types of queries, so we design TypeMatch feature to make use of the information.

We trained 3 ranking models for different Q.type, in order to reflect different importance of features in different type queries.

We normalized the features into [0, 1] before training the ranking models to avoid noise caused by large Integer value or small double value.

2.1.4.1.2、Using 2-class SVM to validate the final answer

To train a 2-class SVM to validate the final answer, we select training samples to construct a training set as following:

a) For all the 416 queries, use the ranking models trained as previous section to select top 1 entity from their corresponding CSet. This would be training samples in the training set.

b) Due to whether the entity is the target entity of the query, label the sample as positive (for yes) or negative.

Same as the ranking step, we trained 3 SVM classifiers due to Q.type for the same purpose. We adopt features almost like features used in ranking step, but do not normalize the feature values.

2.1.4.2 Bayes Method

We also tried Bayes method to rank candidate and select the final answer. All surface features together with TFSimContext, TFSimRankContext, QueryInArticle and TypeMatch are used as attributes in Bayes model. All features having continuous value are discretized. Assuming thatA_1,A_2, … ,A_n are n attributes, a candidate d is represented by a vector $(a_1, a_2, \dots, a_n)$ where a_i is the value of A_i. Let C = {"linking", "unlinking"} represent the class variable and c represent the value that C takes. Our problem is defined as follows:

$$g(d) = \arg\max_c p(c|a_1, a_2, \dots, a_n). \quad (1)$$

According to Bayes' theorem,

$$p(c|a_1, a_2, \dots, a_n) \sim p(c)p(a_1, a_2, \dots, a_n|c) \quad (2)$$

Thus,

$$g(d) = \arg\max_c p(c)p(a_1, a_2, \dots, a_n|c) \quad (3)$$

There are a variety of methods to imate $p(a_1, a_2, \dots, a_n|c)$. For example, in naïve Bayes, all attributes are assumed independent given the class; that is,

$$p(a_1, a_2, \dots, a_n|c) = \prod_{i=1}^{n} p(a_i|c) \quad (4)$$

Naïve Bayes is easy to estimate, but the assumption of independency is too strong for most of the real-world applications. To address this problem, paper [2] proposed a Hidden Naïve Bayes (HNB) model which to some extents captures the dependency among attributes. Formally, the conditional distribution can be represented as follows.

$$P(A_1, A_2, \dots, A_n|C) = \prod_{i=1}^{n} P(A_i|pa_i, C) \quad (5)$$

Where pa_i denotes the parents of A_i from attribute nodes.

If every pa_i is empty, then (5) becomes Naïve Bayes. For HNB, the distribution $P(A_i|pa_i, C)$ is approximated using the weighted one-dependence estimators as follows.

$$P(A_i|pa_i, C) \approx \sum_{j=1, j\neq i}^{n} W_{ij} \times P(A_i|A_j, C) \quad (6)$$

Where W_{ij} is set as follows.

$$W_{ij} = \frac{I_p(A_i; A_j|C)}{\sum_{j=1, j\neq i}^{n} I_p(A_i, A_j|C)} \quad (7)$$

$I_p(X; Y|Z)$ is conditional mutual information defined as

$$I_p(X; Y|Z) = \sum_{x,y,z} p(x, y, z) \log \frac{p(x,y|z)}{p(x|z)p(y|z)} \quad (8)$$

Given a candidates set D, we sort all candidates by their posterior probability given by HNB. If g(top1)="linking", then the top1 candidate is returned, otherwise NULL is returned.

2. 1.5 Results

Table 2. Entity Linking Task Evaluation Results

Submits	P	P(Non-NIL)	P (NIL)
QUANTA1 (ListNet + SVM)	0.8033	0.7725	0.8264
QUANTA2 (ListNet + SVM) (no country feature)	0.8012	0.7707	0.8241
QUANTA3	0.7871	0.6478	0.8919

Although we didn't get the highest precision for all queries, QUANTA1 got the highest precision for Non-NIL queries. This shows that the learning to rank method can achieve quite good performance for ranking candidates. It also shows that the training set for 2-Class classifier has different distributions from the real test queries (In training set, there are more positive values than negative values; the portion is about 2: 1. But the portion is about 1: 1.5 in real test queries). So QUANTA1 performs not well at NIL value queries. When the training set for QUANTA3 contains too many negative samples, it achieves the highest precision of NIL queries, but failed to recognize the correct answers.

2.2 KBP Slot Filling Task

2.2.1 Preprocess

In preprocessing phase, we collect training sentences for each slot. For all the slots which have value, we use the slot value together with the corresponding entity name as a query. For each sentence in each document returned, if it contains both entity name and slot value, the sentence is collected into training set. Finally, the entity name is replaced by "<target>" and the slot value is replaced by "<value>" in each training sentence.

2.2.2 Pattern Generation

In this phase, we first generate candidate patterns from training sentences and then filter the ones of low quality.

Candidate Patterns are generated as follows. For each training sentence, we only retain words between "<target>" and "<value>". Then, a Stanford POS tagger is used to tag the sentence and all nouns, verbs, adjectives, adverbs and numerals are replaced by their POS tags.

In order to remove the patterns of low quality, we filter all the candidate patterns which do not contain verb, preposition, colon, parenthesis, quote mark and "'s". Then, the top n (n depends on slots) most frequent patterns are selected as patterns for each slots.

2.2.3 Answer Extraction & Remove Redundant Answers

This module is divided into two phases: extract candidate answers by patterns from documents and validate each candidate by knowledge base or redundancy-based method.

In Candidate Extraction phase, we first use the expanded entity name as query and collect the top 700 most relevant documents returned by local search engine. Then each document is segmented into sentences and tagged. For each sentence, a regular-expression matcher built from corresponding patterns is used to extract candidates. As mentioned in 2.2.2, each pattern is a sequence of words and POS tags which starts (ends) with "<target>" or "<value>". A word can match with the same word with any POS tags and a POS tag can match any words with the same POS tag. The "<target>" mark is replaced by the corresponding entity name. For those slot expecting noun phrase, we replace "<value>" by proper noun sequence of any length. For those concerned with time or number "<value>" is replaced by corresponding regular-expression. Due to sparseness, patterns are not used to extract websites. We collect website candidate as long as it appears in sentence containing entity name.

In Answer Validation phase, we employ different strategies for different slots. For single-valued slots, we select the most frequent candidate as the final answer. For alternate_name slot, we use wiki redirection page to find more candidate answer to candidate set. For slots concerned with person or organization, a Stanford NER is employed to tag candidates. If all words in a candidate are tagged with right name entity type, then it is retained. For slots expecting location names such as org:headquaters, all candidates are filtered by city list and country list. Only the candidates appeared in one of the lists are retained. For the rest, we only remove lexically duplicate candidates.

2.2.4 Additional Answers From Knowledge Base and Wikipedia

If the queries have target entities in Wikipedia, we also try to extract answers from Wikipedia.

First, we search the query's corresponding entity in the Knowledge Base or latest Wikipedia; if we can get the right entity from Knowledge Base or Wikipedia, then we just extract answers from these articles and find support document in news corpus (We deal with all the PER and GPE queries).

Second, we use some pattern to extract answers from Wikipedia articles. The articles are better formed than news text, so the pattern is clear and the answer extraction is much easier.

Third, we find supporting document for these answers, based on co-occurrence and words occurring between the answers and query's entity name string. If we can't find the supporting document for the answer, we just drop the answer.

Finally, same as we do in Answer Validation phase, we remove the duplicate answers.

2.2.5、Results

Table 3. Slot Filling Task Evaluation Results

	F-Score	**Recall**	**Precision**
All	0.748		
Single Slots	0.784(Accuracy)		
List Slots	0.712		
Single Slots(Non-NIL)		0.436	0.279
Single Slots(NIL)		0.847	0.943
List Slots(Non-NIL)	0.251		
List Slots(NIL)		0.873	0.878

Since no submissions are better than the All NIL Baseline at SF-Score values as shown in Table 3, we consider that as following reasons:

a) There is little information about those entities mentioned in news corpus but not extracted as entities' facts. So it's hard to retrieve more useful information.

b) The evaluation methods assign equally to both NIL value slots and Non-NIL value slots, so if the system tried to get high SF-Score, it need to submit Non-NIL value only if it had high confidence about the value. But most submitted teams didn't pay attention to this.

Table 4. Question Taxonomy from UIUC

(Note: the categories with underline are selected as factoid question taxonomies to identify entity)

Coarse	Fine
ABBR	abbreviation, expression
DESC	definition, description, manner, reason
ENTY	*animal*, *body*, *color*, *creation*, *currency*, *disease/medicine*, *event*, *food*, *instrument*, *language*, letter, other, *plant*, *product*, *religion*, *sport*, *substance*, symbol, *technique*, term, *vehicle*, word
HUM	description, *group*, *individual*, title
LOC	*city*, *country*, *mountain*, *other*, *state*
NUM	*code*, *count*, *date*, *distance*, *money*, *order*, *other*, *percent*, *period*, *speed*, *temperature*, *size*, *weight*

Although we didn't achieve high SF-Score, we got a relatively good F-Score at List Slots (Non-Nil). The pattern based system is useful for retrieving list value type information.

3 RTE Track

Given two text fragments 'Text' and 'Hypothesis', Textual Entailment Recognition is the task of determining whether the meaning of the Hypothesis is entailed (can be inferred) from the Text. This year we focus on investigating the textual representation in the RTE task. We believe that the text can be divided into two types of semantic units: One is what objects the text referred; the other is the description to the object: such as the characteristic of object and relation with other objects. In this year's paper, we use Semantic Element to represent this semantic unit. Two types of semantic elements are defined: Entity Semantic Element refers to the object, and Relation Semantic Element describes the relations among these objects. In the following sections, we will introduce how to construct SEGraph based on Entity Element (short for Entity Semantic Element) and Relation Element (short for Relation Semantic Element), and we also discuss how to recognize textual entailment based on SEGraph.

3.1 SEGraph Construction

The SEGraph is constructed with Entity Element as node and Relation Element as edge. We first show you how to identify the Entity Elements from text and then describe how to identify Relation Elements and construct SEGraph based on the dependency parser tree.

3.1.1 Entity Element Identification

We consider Entity Element as the object described in the text. In this section, we will show you how to identify Entity Element. The most intuitive way is to use Named Entity Recognition tools, such as Stanford NER. However, the Stanford NER methods only identify three or four types of entities. It is not enough in the SEGraph construction.

We propose to employ factoid question taxonomy to identify Entity Elements. The factoid questions seek simple entities as answers, like "What is Hawaii's state flower?" and "What is the length of the coastline of the state of Alaska?" We define Entity Semantic Element as the entity, which can be used as an answer for factoid question. We use question taxonomy to recognize these entities. Question taxonomy defines the expected semantic categories of answers to the questions. The following table shows general question classification taxonomy developed by UIUC [4]. It contains 6 coarse types and 50 fine types. We select about thirty factoid categories, denoted with underline as shown in the following table. The answers to these questions can be recognized as Entity Elements.

We then propose four methods to detect if an entity belongs to these taxonomies:

1) WordNet based Identification

For some categories, we can find a corresponding node in WordNet, which can represent the semantic meaning of this category. For example, "animal" in WordNet is the corresponding node for category "animal", "monetary unit" is the corresponding node for category "money". For a new word, if its hypernyms contain the category corresponding node, this word belongs to the corresponding category, and can be determined as an Entity Element. This WordNet based method is suitable for most categories of ENTITY and LOCATION.

2) Wikipedia based Identification

For some entities, especially for new concept, it may be not contained in WordNet, such as entity “MV Princess of the Stars”. Wikipedia is a good expansion for WordNet. Similar as WordNet based method, we also find some corresponding node for factoid category in Wikipedia. We find ships, car associated category nodes in Wikipedia for “vehicle” category.

3) Pattern based Identification

For number and date categories, we construct patterns for entity identification. We use regular expressions to extract the number and date Entity Element.

4) NER based Identification

We also use Named Entity Recognition Tools to identify Entity Element. We employ Stanford NER tools to extract named entities. It can recognize four categories: Location, Organization, Person and Misc.

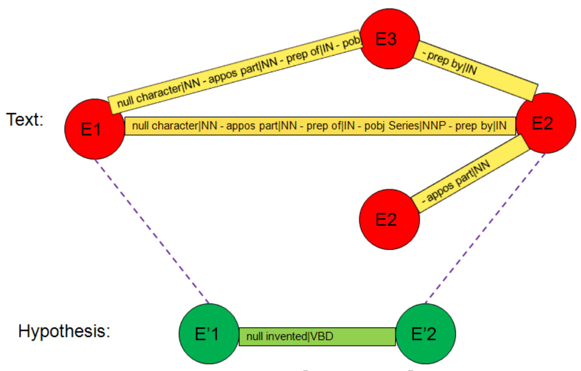
Besides above four methods, we also assume that if the noun word appears in both Text and Hypothesis, we also recognize this noun word as Entity Element. We also use the Entity Element integration strategies to reduce the number of entities, such as if several successive Entity Elements appear in both Text and Hypothesis, we recognize their combination as one Entity Element.

3.1.2 Relation Element Identification

After identifying Entity Elements, we employ dependency tree to identify Relation Element. We extract the shortest path between two Entity Elements in the dependency tree as Relation Element.

3.1.3 SEGraph Construction

After Recognizing both Entity Element and Relation Element, we set Entity Element as node and Relation Element as edge to construct SEGraph. For example, given the T-H pair 998 in RTE 4 test set, the Text is “Preem Palver is a fictional character, part of the Foundation Series by the sci-fi writer Isaac Asimov.” And Hypothesis is “Isaac Asimov invented Preem Palver.” Its corresponding SEGraph is shown in Figure 2.


Text:
E1
E3
E2
E2
null character|NN - appos part|NN - prep of|IN - pobj
- prep by|IN
null character|NN - appos part|NN - prep of|IN - pobj Series|NNP - prep by|IN
- appos part|NN
Hypothesis:
E'1
E'2
null invented|VBD
Fig. 2 SEGraph Example

We need to note that here we just consider Entity Element and Relation Element. However, there is still some Hypothesis, which may contain only one Entity Element, or contain Entity Element and Relation Element, but encounter characteristic mismatch problem. We analyze the “No Entailment” T-H pairs for RTE4 test set. We find that about 60% “No Entailment” comes from relation mismatch, and 20% for entity mismatch, 10% for characteristic mismatch, 10% for other mismatch. In this year, we only focus on handling the major problem: entity mismatch and relation mismatch, and for simplicity, we just consider the major special relation, which contains verb in its path. For other types, we use our baseline and post-check processing to deal with, which will be discussed in the framework section.

3.2 Textual Entailment Recognition with SEGraph

After constructing SEGraph for Text and Hypothesis, we will recognize textual entailment with these two SEGraphs. By analysis of the text, we find that the degree of variation is different for Entity Element and Relation Element. The entities have few variations, and these variations can be easily recognized by lexicons. For example, if we want to describe “computer”, limited number of expressions can be used, such as “computing device”, “computing machine”, and these expressions can be found with WordNet. However, the relations may be described in many ways. If I want to express “I like you”, various expressions can be used. Therefore, we employ two different strategies to detect the Entity Element Entailment, and Relation Element Entailment.

3.2.1 Entity Entailment Recognition

We use knowledge based methods to detect entity entailment. The knowledge resource contains WordNet, Wikipedia and other knowledge base. For all Entity Element in Hypothesis, several strategies are used to detect entailment:

1) Determine if all the original entity words appear in Text

2) The edit distance is used to determine entity presence in Text. If the edit distance is lower than a threshold, we say the entities match each other.

3) Wordnet relations, such as synonym, hypernym, hyponym, are used to expand the entity to determine entity entailment.

4) Wikipedia is also employed to detect the entity entailment. Wikipedia redirection set contains all the redirection information. Since the synonymous entities are redirected into one entity. This set can be used as a synonymous entity set.

Besides the above four strategies, we also use the entity category information. When we identify entities based on factoid taxonomy, each entity is assigned a category. We also compare the category match, especially for date and other number.

If there is one Entity Element mismatch in Hypothesis, we take this T-H pair as "Entity Not Entailment".

3.2.2 Relation Entailment Recognition

For Relation Entailment detection, most of previous studies use unsupervised methods, such as DIRT and TEASE. While here we try to investigate supervised methods to detect relation entailment. We first show you the constructed training set, and then describe the employed features and machine learning methods.

We use RTE 3 training set, RTE 3 test set and RTE 4 test set as our training set. There are totally 2600 pairs, where RTE 3 contains 1600 pairs with 800 for training and test set, and RTE 4 contains 1000 pairs. For false entailment T-H pairs, we manually label the false relation entailment: we first determine whether this entailment is caused by relation mismatch; if so, we also denote which two entities cause the false entailment. For example, for the following false entailment pair, we label the relation between entity "Annan" and entity "IECI" as false relation entailment.

<t>Annan praised the courage of the people and congratulated the Independent Election Commission of Iraq (IECI) and UN officials for successfully run-ning the poll.</t>
<h>Annan is a member of the IECI.</h>

For true entailment T-H pairs, we define all the relation entailment is true. That is to say, every relation between two entities in Hypothesis can be entailed by the corresponding relation in Text. We use the extracted true entailment relations and manually labeled false entailment relations to construct training set. Since the number of true entailment relation is much larger than the number of false entailment relation, we use all the false relation entailment from RTE3 and RTE4 as false relation entailment set, and just use the true relation entailment from RTE4 as true entailment set. This can solve the imbalance problem. For simplicity, we only consider the relation with verb in its path.

We then design our framework for relation entailment classification. We employ a lot of features, not only use T-H relation intra similarity, but also investigate the impact of the cross similarity among T-H relation pairs. For T-H intra-similarity features, they are divided into two types: sentence-level features and path-level features. Sentence-level features denote the similarity feature from the two sentences, which contain T, H Relation Elements. The path-level features denote the shortest path similarity from dependency tree. Both are shown in Table 5.

Table 5. Features used in relation entailment detection

	Feature type	**Feature Description**
Sentence-Level Features		
LLM-similarity	double	Lexical similarity [3]
Entity-similarity	double	Named Entity similarity [3]
Path-Level Features		
Path-LLM-similarity	double	Lexical similarity in the path
Path-Relation-similarity	double	The path similarity
Verb-synonym	{0,1}	Contain synonym for verb?
Verb-antonym	{0,1}	Containing antonym for verb?
V-N-derivation	{0,1}	Containing Noun-Verb derivation relationship using WordNet?

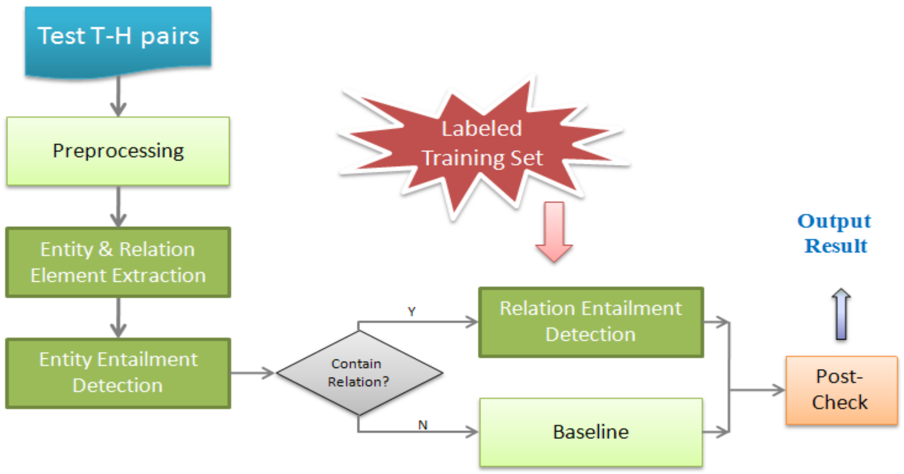


Fig. 3 SEGraph based RTE Framework

We use Weka tool [5] as our classifier. We also test the cross-pair similarity features, but it didn't achieve a better result. This year only focuses on intra-pair similarity. We plan to design more elaborate features for relation entailment as our future works.

If there is one false Entity Element entailment, or one false Relation Element entailment, we will consider this T-H pair as false entailment.

3.3 SEGraph based RTE Framework

Based on our proposed SEGraph method, we design our RTE framework, as shown in the Figure 3. We first preprocess all test T-H pairs. It contains dependency parsing, Named Entity Recognition, co-reference Resolution, and WordNet Sense Disambiguation. Then the following three models with dark green color are related with our SEGraph framework: the "Entity & Relation Element Extraction" module identifies two types of our defined Semantic Elements with the methods proposed in Section 3.1; the "Entity Entailment Detection" module detect entity entailment with the knowledge based methods proposed in Section 3.2.1; the "Relation Entailment Detection" module detects relation entailment with the methods proposed in Section 3.2.2. For simplicity, in this year, we only consider the relation with verb in its path. For the T-H pair without this path, we use our baseline method to process it. The baseline mainly use Weka classifier with lexical similarity features and Entity similarity features to classify T-H pair as entailment or no entailment, which has been described in our last year's notepaper[3]. Finally, we also use post-check processing methods to improve the performance. The post-check processing use some patterns to detect mismatch. In this year task, we use Negation rules, Subjunctive rules, Quantifier Modification rules, which can be found in our last year's notepaper[3]. We also collect some verb, which contains negative meaning, like "reject", "refuse", and use these verb to construct mismatch rules.

3.4 Submissions and Evaluations

We submit two results. The first submission QUANTA1 employs the whole procedure described in the previous section. QUANTA2 removes the "Relation Entailment Detection" module. We just submit results for two-way task. The evaluations are shown on Table 6.
From the table, we can see that our proposed methods are very effective. But we also find it is a very challenging task to recognize relation entailment. In the future work, we will focus on improving the relation entailment detection.

4 Conclusions

In this paper, we describe our systems for the KBP and RTE tracks. For the KBP track, we propose to use a Listwise learning to rank model and Augmenting Na ïve Bayes model for Entity Linking task, and use learned patterns to solve Slot Filling task. For RTE track, we propose to use SEGraph method. The idea of SEGraph is that we should use different strategies to recognize entity entailment and relation entailment. From evaluation results, we can see that both KBP and RTE systems achieve competitive results. We find that "Learning to Rank" strategy is really effective for Entity Linking task, and relation entailment recognition is really a challenging problem for RTE task.

Table 6. RTE Task Evaluation Results

	QUANTA1	QUANTA2	Best	Median	Worst
Accuracy	0.67	0.6633	0.7350	0.6117	0.5000
Avg. precision	0.7011	0.6755			

Acknowledgments

The work was supported by NSFC under grant No.60803075, the National Basic Research Program (973 project in China) under grant No.2007CB311003. The work was also supported by an IRCI project from the International Development Research Center, Canada.

References

[1] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: From pairwise approach to listwise approach. In ICML, 2007.

[2] H. Zhang, L. Jiang, and J. Su. Augmenting naive bayes for ranking. In ICML ’05: Proceedings of the 22nd international conference on Machine learning, pages 1020–1027, New York, NY, USA,2005.ACM Press.

[3] Fangtao Li, Zhicheng Zheng, Yang Tang, Fan Bu, Rong Ge, Xian Zhang, Xiaoyan Zhu and Minlie Huang. THU QUANTA at TAC 2008 QA and RTE track. Text Analysis Conference (TAC 2008), Gaithersburg, Maryland USA, November 2008

[4] Li, X. and Roth, D. Learning question classifiers. In Proceedings of the 19th international Conference on Computational Linguistics pages 1~7, 2002.

[5] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1