# A Multi-strategy Learning Approach to Competitor Identification

Tong Ruan[1], Yeli Lin[1], Haofen Wang[1], and Jeff Z. Pan[2]

[1] East China University of Science and Technology, Shanghai, 200237, China
{ruantong,whfcarter}@ecust.edu.cn, lin_yeli@163.com
[2] The University of Aberdeen, Scotland
jeff.z.pan@abdn.ac.uk

**Abstract.** Competitor identification tries to find competitors of some entity in a given field, which is the key to the success of market intelligence. Manually collecting competitors is labor-intensive and time consuming. So automatic approaches are proposed for this purpose. However, these approaches suffer from the following two main challenges. Competitor information might not only be contained in semi-structured sources like lists or tables, but also be mentioned in free texts. The diversity of its sources make competitor identification quite difficult. Also, these competitors might not always occur in form of their full names. The occurrences of name variants further increase the diversity, and make the task more challenging. In this paper, we propose a novel unsupervised approach to identify competitors from prospectuses based on a multi-strategy learning algorithm. More precisely, we first extract competitors from lists using some predefined heuristic rules. By leveraging redundancies among competitor information in lists, tables, and texts, these competitors are fed as seeds to distantly supervise the learning process to find table columns and text patterns containing competitors. The whole process is iteratively performed. In each iteration, the newly discovered competitors of high confidence from various sources are treated as new seeds for bootstrapping. The experimental results show the effectiveness of our approach without human intentions and external knowledge bases. Moreover, the approach significantly outperforms traditional named entity recognition approaches.

**Keywords:** Competitor Mining, Unsupervised Learning, Distant Supervision, Wrapper Induction

## 1 Introduction

Competitor mining tries to identify competitors of certain companies in a given domain. It is quite important to the success of market intelligence. The information of competitors is not only useful for individual companies, but also vital for market analyzers and investors. In the new economic environment of China, stockbroking companies and stock exchanges begin to have rights to recommend or approve companies to IPO (Initial Public Offering) in a brand new board. These agencies are very active in looking for companies with great potentials. Given that a competitor of a well-known company in a specific field is a good candidate, the above mentioned company finding problem can be treated as a competitor identification task.

Due to the large number of companies on markets and the emergence of new companies, it is labor-intensive and time consuming to manually collect competitors. Therefore, automatic approaches have been proposed for this purpose [?,?]. However, these approaches suffer from the following two main challenges. Competitor information might not only be contained in semi-structured sources like lists or tables, but also be mentioned in free texts. We call it the *source diversity* challenge. Traditional information extraction methods only focus on a particular type of data sources. For example, Ciravegna et al. [?], Milne et al. [?], and Limaye et al. [?] studied how to extract knowledge from lists, tables, or Semantic Web. On the other hand, Web-based IE methods and systems like Snowball [?], OpenIE/TextRunner [?], and KnowItAll [?] mainly extract data from texts. While systems such as LODIE [?] extract information from both free texts and structured data, how to fully utilize different kinds of data sources is not fully investigated especially for competitor identification. In addition, competitors might not always be mentioned in form of their full names. The occurrences of name variants further increase the diversity, and make the task more challenging. We call it the *expression diversity* challenge.

In this paper, we focus on competitor identification in prospectuses. Each prospectus contains competitors of a particular company occurring in tables, lists or free texts with different name descriptions. In order to tackle both challenges, we propose a novel unsupervised approach based on a multi-strategy learning algorithm. More precisely, we first extract competitors from lists using some predefined heuristic rules. By leveraging redundancies among competitor information in lists, tables, and texts, these competitors are fed as seeds to distantly supervise the learning process to find table columns and text patterns containing competitors. The whole process is iteratively performed. We carried out experiments on prospectuses of Chinese listed companies obtained from Shanghai Stock Exchange. The experimental results show the effectiveness of our approach without human interventions and external knowledge bases. Moreover, the approach significantly outperforms traditional named entity recognition approaches.

The rest of the paper is organized as follows. Section 2 lists several aspects of related work. Section ?? analyzes the structure as well as the competitor occurrences of prospectus. It then gives a overview of multi-strategy learning for the competitor identification task. Section ?? presents our approach in details. Section ?? shows experiment results of our work. Finally, Section ?? concludes the paper and points out the future direction of our work.

## 2 Related Work

While our work is the first to identify competitors with combined strategies on prospectuses, there exist several aspects of related work.

### 2.1 Competitor and Competitive Mining

Lappas et al. [?] defined a formal definition of competitiveness between products sold on B2C Web sites. They developed an algorithm called CMiner to find top-$k$ competitive items for a given product. They run their algorithm on different datasets ranging from Amazon.com, Booking.com to TripAdvisor.com. The results show that the algorithm is effective and can be applied to different domains. Cominer [?] extracted competitors of an object (a company, a sports team etc) from Web. Given the name of an object, it queried the search engine with predefined linguistic patterns to gather its competitor name and rank these competitors accordingly. Cominer also mined competitive domain and competitive evidence. Since competitors are expressed in different ways on the Web, the linguistic patterns cannot cover all situations, and thus Cominer can only mine competitive relationship between well-known companies whose information is very redundant on the Web.

### 2.2 Information Extraction

Competitor identification is an application of information extraction which combines relation extraction with named entity recognition. Information extraction has been studied intensively over the past few years. *Wrapper induction* is a sort of information extraction, which extracts knowledge from semi-structured data. Multi-view learner [?] and Vertex! [?] use supervised learning algorithms to learn data extraction rules from manually labeled training examples. Other systems like SKES [?] and LODIE [?] use unsupervised methods. Moreover, Dalvi et al. [?] presented a generic framework to learn wrappers across Web sites. Another kind of information extraction is to extract structured information from texts, which is called *text mining*. Snowball [?] and TextRunner [?] are two typical examples. The input to Snowball is a corpus of text documents and a small set of seeds. Extracted patterns can be learned by summarizing the occurrence patterns of seeds in the corpus. TextRunner learns all relations in a corpus without any predefined rules or hand-tagged seeds, thus the method is called "Open Information Extraction".

One trend of information extraction is to utilize various data published on the Web including Web pages, Linked Open Data as well as lists and tables on dynamic Web sites. Gentile et al. [?] proposed a methodology called *multi-strategy learning* which combines text mining with wrapper induction to extract knowledge from tables, lists and Web pages. While the method seems promising, there are no clear evaluation results in their paper. On the other hand, *distant supervision* is an effective mean to leverage redundancies among different sources, which has been used in [?,?]. Mintz et al. [?] leveraged entity pairs of a certain

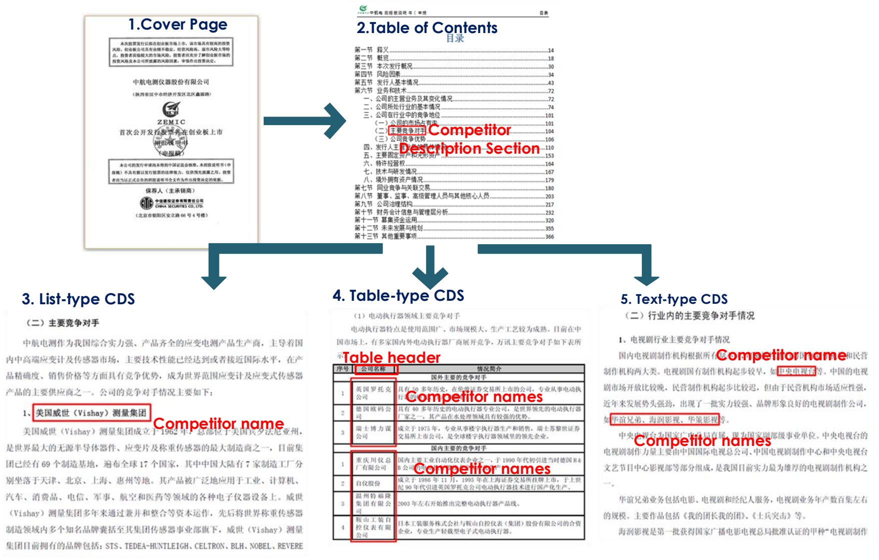

**Fig. 1.** Observations of Competitor Information in Prospectuses

relation from Freebase as seeds and collected sentences containing these pairs as *weakly labeled data* from a text corpus. Then they extracted syntactic and semantic features of context words around entity pairs to train a multi-class logistic classifier for relation extraction. As a result, 10,000 instances of 102 relations were extracted at a precision of 67.6%. Roth et al. [?] used distant supervision in pattern learning and built a system called RelationFactory [?]. RelationFactory RelationFactory achieved top ranked F1-Score at 37.3% in TAC KBP 2013 English Slot Filling evaluation. To the best of our knowledge, distant supervision has not been used for competitor identification.

## 3 Approach Overview

### 3.1 Problem Analysis

After we analyze more than 800 prospectuses about companies in different fields from the Chinese stock market, we have the following observations.

- **Observation 1**. Almost every prospectus has a specific section to describe the competitors of a company. The section is called *Competitor Description Section (CDS)*. Each CDS contains one or more paragraphs, and nearly 80% titles of CDS contain the word “竞争对手 (Competitors)”. The right upper part (i.e. “Table of Contents”) of Figure **??** shows a CDS called “主要竞争对手 (Major Competitors)”.

**Table 1.** Distribution of Different CDS Types in One Prospectus

| | List | Text | Table | List+Text | List+Table | Table+Text | All |
|---|---|---|---|---|---|---|---|
| Amount | 246 | 147 | 142 | 148 | 44 | 94 | 18 |
| Ratio | 29.32% | 17.52% | 16.92% | 17.64% | 5.24% | 11.20% | 2.15% |

– **Observation 2**. Competitors mentioned in a CDS might appear in different forms (e.g., a list, a table or free text). Moreover, a CDS may contain competitor information of more than one form. As shown in the lower parts of figure **??**, competitor names are listed as titles of subsections in a list-type CDS, are contents of the same column of a table in a certain table-type CDS, and are mentioned in a text-type CDS respectively.
– **Observation 3**. Competitor information is redundant in prospectuseses. Redundancies not only exist in different forms of the same prospectus, but also can be found different prospectuses of the same domain. The former is called *intra redundancy*, and the later is called *inter redundancy.*

Table **??** shows the distribution of different CDS types in one prospectus. From the table, we can see a large proportion of prospectuses contain at least two forms of CDS, which indicates big intra redundancies. We then check the inter redundancies between different prospectuses. 99 appear in list-type CDSs of at least two prospectuses, 85 come from table-type CDSs of different prospectuses, and another 85 are from text-type CDSs. Furthermore, 209 competitors occur in a list-type CDS of one prospectus but in a table-type CDS or a text-type CDS of another prospectus with the same names. 203 is the number of inter redundancies for the situation when one is from a table-type CDS and we find matches from another type of CDS in a different prospectus. Similarly, 195 is the answer for the third situation. Observation 2 and 3 are the basis of our multi-strategy learning algorithm. Redundant competitors from CDSs of one type can be used to annotate their occurrences in CDSs of other two types.

## 3.2 Overall Architecture of Our Approach

The objective of our approach is to find a way to extract competitor information in a language-independent way without the use of any named entity recognition (NER) tools or any prior knowledge about company information. According to observations introduced in Section **??**, competitors are mentioned in different kinds of CDSs (list-type, table-type, and text-type). We also find rich intra- and inter-redundancies between different types. In such circumstances, competitors are first extracted from structured sources using specific wrappers. As far, the main concern is whether we can use a limited set of heuristic rules or some automatic mechanism to get these wrappers which can cover most cases. Then the extracted competitors are further used as seeds to help competitor identification from free texts, which can be modeled as a distant supervision process. The overall architecture of our approach is shown in Figure **??**. We have two main steps namely *Competitor Description Section Detection* and *Multi-Strategy Learning.*

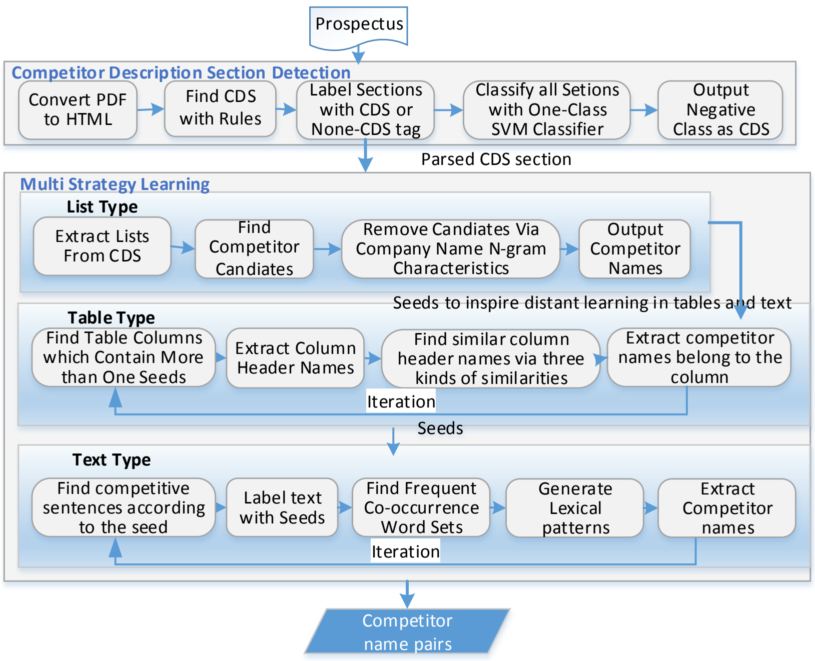
Prospectus
Competitor Description Section Detection
Convert PDF to HTML
Find CDS with Rules
Label Sections with CDS or None-CDS tag
Classify all Setions with One-Class SVM Classifier
Output Negative Class as CDS
Parsed CDS section
Multi Strategy Learning
List Type
Extract Lists From CDS
Find Competitor Candiates
Remove Candiates Via Company Name N-gram Characteristics
Output Competitor Names
Seeds to inspire distant learning in tables and text
Table Type
Find Table Columns which Contain More than One Seeds
Extract Column Header Names
Find similar column header names via three kinds of similarities
Extract competitor names belong to the column
Iteration
Seeds
Text Type
Find competitive sentences according to the seed
Label text with Seeds
Find Frequent Co-occurrence Word Sets
Generate Lexical patterns
Extract Competitor names
Iteration
Competitor name pairs

**Fig. 2.** Overall Architecture of Our Approach

– **Competitor Description Section Detection**. As mentioned in Observation 1, there exists a specific section describing competitors and competitive information in each prospectus called CDS. Compared with the whole prospectus, a CDS is more focused and is thus more appropriate for competitor identification. So the first step of our work is to find the CDS for each prospectus. Since titles of a large number of CDSs contain the word "竞争对手 (Competitors)", heuristic rules can be used to find these CDSs easily. In order to find CDSs for the remaining prospectuses, we use a classification-based method. The details will be described in Section **??**.
– **Multi-Strategy Learning**. According to Observation 2, competitor names may occur in lists, tables or free texts in CDSs. Since list-type CDSs are the easiest to deal with, we first identify competitive lists and extract competitor names from these lists, as shown in Figure **??**. Then these competitor names are served as seeds to extract more competitors from table-type CDSs and text-type CDSs. Furthermore, the extracted competitors of high confidence from table-type CDSs and text-type CDSs can also be fed as seeds to each other. For table-type CDSs, we detect competitive columns in tables based on the input seeds, and then header names of detected columns are used to find similar column headers. In the iterative process, cell contents within the detected columns are extracted as competitor names. The iteration will not terminate until the newly discovered column contents do not conform to a n-gram model of company names. For text-type CDSs, competitive sentences containing seeds are collected. Competitive lexical patterns are then

learned from contexts of these seeds in sentences, and competitor names are extracted by applying the above patterns to new sentences in other text-type CDSs. The process is also iterative. More details of multi-strategy learning are introduced in Section **??**.

## 4 Approach Details

### 4.1 Competitor Description Section Detection

If the title of a section has the word "竞争对手(Competitors)", the section is a CDS. Since each prospectus has a well-organized table of contents, the CDS section is easy to locate. However, still 20 percent prospectuses cannot be covered. Since the heuristic rule can find CDSs with high precision, we use CDSs found by the rule as positive examples while other sections in these prospectuses are treated as the background corpus. Then the CDS detection task is modeled as a one-class classification problem, and we use one-class SVM (Support Vector Machine) as the classification model. Each section is represented as a word feature selection. When applying the model to each section of a prospectus, CDSs will be detected. For feature selection, *Information Gain* is used to select words which can distinguish CDSs with the background corpus for performance improvements. Since prospectuses are in PDF formats, certain preprocessing steps are required. Firstly, prospectuses are converted into HTML formats, then HTML tags are removed and texts are extracted. At last, texts are segmented into words using natural language processing tools, and stop words are removed. The overall accuracy of CDS detection is higher than 95%.

### 4.2 Multi-Strategy Learning

**Seeds Extraction from List-type CDSs** Lists containing competitor names in list-type CDSs have the following characteristics. These lists are parallel structures in CDSs and each item in such a structure starts with sequence numbers. There are two types of such structure. One is subsections in a CDS, as shown in Figure **??**. The other is in a text paragraph whose precedent words are "竞争对手是 (competitors are)" or "竞争对手有 (competitors include)". Since some lists might contain false positives, the extracted texts from these lists have no relationship with competitor names. In order to filter out unrelated strings, we use three rules to check whether a string is an organization name. If any of the rules does not hold, the corresponding string does not refer to an organization. The rules are learned by calculating statistics collected from the organization list provided by Shanghai Bureau of Public Security [**?**].

– *Lengths of organization names.* Organization names have a limited number of characters. The length distributions of organization names, including full names and abbreviations, are shown in Figure **??** and Figure **??** respectively. In most cases, the lengths of full organization names are between 4 and 24. If the length of a string is out of the range, we can filter it out.

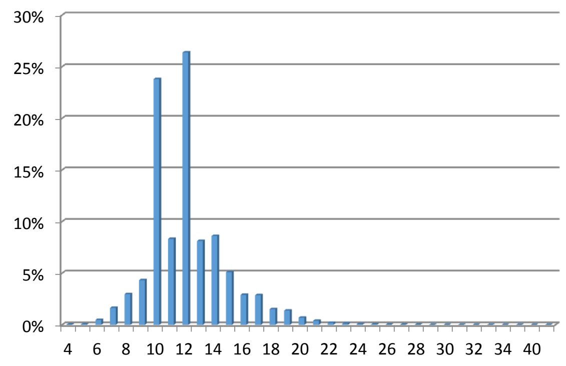

**Fig. 3.** Length Distribution of Full Names

- *Character occurrences in organization names.* The distribution of commonly used 3500 Chinese characters is calculated from the organization list, which shows the user preferences to name organizations. In particular, we collect positive characters that occur frequently with organizations, and negative characters which never appear in organization names. For instance, negative words such as "篡 (tamper)" and "瘟 (pestilence)" have no occurrences in the organization list. A language model is built using these characters to predict the probability whether the string is an organization. If the probability is very low, it can be treated as a non-organization of high confidence.
- *Length Difference of items in a list.* If items in a same list are organization names, their lengths should not differ too much. In contrast, the word lengths might vary a lot. When the length difference exceeds 6 characters, we can safely remove it.

We extract competitors from a list-type CDS as follows. First, we find all lists in the CDS of a prospectus. Then we remove those parallel structures in text paragraphs whose precedent words do not contain "竞争对手(competitor)". After that, the first string after each sequence number is extracted as a competitor name candidate. We further remove names from those candidates which do not follow any of the above three rules. Finally, the remaining ones are returned as competitors, which are used as seeds for further processing.

**Competitive Table Column Detection** The following observations are used to detect competitors in table-type CDSs.

- While the formats of tables are diverse in real world, tables in prospectuses are much simpler and table headers always refer to columns. Even for complex tables which have nested headers across multiple columns, we can still find headers strictly aligned to one column as our targets.
- Contents in a table column have the same sort of data. For instance, if more than two cells of a column in a table contain competitor names, then all cells of that column correspond to competitors.

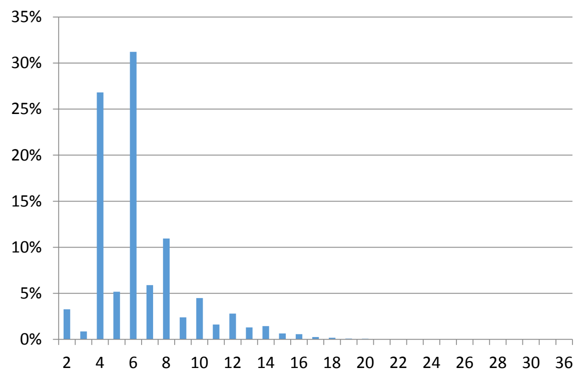

**Fig. 4.** Length Distribution of Abbreviated Names

– Table columns contain similar contents if they have similar headers. For example, the column "主要竞争对手 (major competitors)" is similar to the column titled " 竞争企业 (competitive companies)", and both column contents refer to competitors.
– Column headers are similar if they have similar contexts. The context of a column header is defined as names of all other headers in the table. For example, If a table has four columns in form of $(h1, h2, h3, h4)$, then $(h2, h3, h4)$ is the context of a header $h1$.

The process of detecting competitors from table-type CDSs is as follows.

1. Extracting tables. All tables in CDSs are extracted.
2. Finding competitive table columns. For a table column, if it has more than two cells and the contents of these cells are recognized as competitor names by results extracted from list-type CDSs as seeds, the column is a possible competitive table column.
3. Finding similar table columns. We then find all columns whose header names are similar to the names in $C_{names}$. There are three ways to calculate the similarity between two table columns. They are `Cosine Similarity`, `Edit Distance`, and `Context Similarity`. The former two captures the header name similarity while the last one is the distributional similarity between the contexts of two headers. We use the combined results of the three types of similarities to find additional competitive table column candidates. This step is inspired by the work done by Limaye et al. [?].
4. Extracting competitor names. For a column returned in the previous two steps, we check its contents to see whether they conform to the three rules of organization names. If yes, we take contents of the column as competitor names, and add these names to the seed set, and finally add the column header name to the name set of competitive columns denoted as $C_{names}$.
5. Iterating the whole process until no new column headers can be found.

The details of the three types of similarity calculations are as follows.

- **Cosine Similarity**. It is a measure of similarity between two vectors. The words occurring in the two table header names are used as features, and the times of occurrences are used as weights of features. For example, to compute the similarity between "竞争对手" and "竞争企业", after word segmentation, three features "竞争", "对手", and "企业" are selected. Then the two header names are converted into vectors as $< 1, 1, 0 >$ and $< 1, 0, 1 >$, and thus the similarity score is 0.5.
- **Edit Distance**. It is a way to quantify how two strings are similar to each other by counting the minimum number of operations (insert, delete, and substitution) required to transform one string into the other. The smaller the `Edit Distance` is, the more similar the two headers are. For example, "厂商" can be transformed to "厂家" by one substitution operation, namely "商" replaced by "家". Therefore, the `Edit Distance` is 1.
- **Context Similarity**. The context of each header is also represented as a vector, and `Cosine Similarity` is used to calculate the similarity between two header contexts. If the name of a header name appears in more than one table, the two "local" contexts are merged and the occurrences of some headers are accumulated. For example, one table column header list is $(h1, h2, h4, h5)$ and the other is $(h1, h2, h3, h4)$. The context vector of $h1$ is $< h2, h3, h4, h5 >$ by merging two local context vectors $< h2, h4, h5 >$ and $< h2, h3, h4 >$. The weight of the context is $< 2, 1, 1, 1 >$. If there are altogether eight header names for all tables, the context vector of $h1$ is $< 0, 2, 1, 1, 1, 0, 0, 0 >$.

**Distant Supervision for Competitor Patterns in Free Texts** Competitor identification on text-type CDSs requires competitor name annotations in sentences to learn patterns. These patterns are further used in other sentences to extract competitors. The quality of the extracted competitors heavily depends on the number of annotated sentences while manual annotation costs too many human efforts. Here, we leverage competitors extracted from previous steps to label free texts automatically. Such kind of distant supervision can save manual efforts of labeling sentences significantly. We first collect sentences that contain seed competitor names and label these sentences. Then we generate frequent co-occurrence word sets from the labeled corpus. Extraction patterns are further generated from the word sets and the annotated sentences. Finally, we use the generated patterns to extract new competitors from the other text-type CDSs. The whole process is iterative until there are no new patterns found. We describe the details of each step as follows:

1. **Labeling text with seeds**
   We use a triple <L, seed, R>to express the occurrences of a seed. The "L" is the left context of a seed with a few words before the seed, and the "R" is the right context having several words after the seed. We do not allow triples spanning across multiple sentences separated by punctuations like full stops, commas, and semi-commas.

Sometimes, there are more than one competitor names in a sentence and these names occur continuously. For example, “竞争对手有微软、索尼与苹果等 (competitors are Microsoft, Sony, and Apple etc..)”. Such sentences are called *multi-slot sentences*, and sentences that contain only one competitor name are called *single-slot sentences.* For the above example, if the seed is “SONY”, the triple looks like <“competitors are Microsoft”, seed, “and Apple etc..”>. Since the left side or the right side of a seed may also be an organization name, it is slightly different to generate patterns for multi-slot sentences, which will be discussed later.

2. **Generating frequent co-occurrence word sets**
   In multi-slot sentences, competitor names are delimited by “slight-pause marks”. We neglect seeds which have “slight-pause mark” directly before or after them at their left contexts or right contexts. That is to say, we only select sentences labeled with at least two seeds where the left side of the most left seed should not be a “slight-pause mark”, and the right side of the most right seed is not a “slight-pause mark” either. Back to the above example, if “微软 (Microsoft)” and “苹果(Apple)” become seeds, the whole sentence can be labeled, and the triple is <“竞争对手有 (competitors are)”, “微软、索尼与苹果 (Microsoft, Sony, and Apple)”, “等（etc..）”>.
   In this way, multi-slot sentences and one-slot sentences can be processed in the same way as follows. First, we perform word segmentation for $S$ where $S$ is the set of strings of all “L”s and “R”s in <L, seed, R>triples. Each string in $S$ is segmented into words using segmentation tools. Then we select words occurring more than 5 times as words of high frequencies. Finally, we construct the high frequency co-occurrence word sets by selecting those co-occur in <L, seed, R>triples for more than twice.
3. **Generating extraction patterns**
   Frequent co-occurrence word sets are used to generate extraction patterns. Take $WS$ as the word sets, and an element $ws$ in $WS$ would be a word set $\{w_1,w_2...w_n\}$. All sentences which contain $\{w_1,w_2...w_n\}$ are returned, and each sentence forms a distinguished pattern. Patterns are aligned and consolidated based on the order of occurrences of elements in *ws.* Some words are replaced by wildcards, others are unioned together. We also retain the boundary words of the occurrences in the merged pattern. The pattern generalization is similar to that of Snowball [**?**]. The novelty lies on the previous steps especially the first step to label sentences with seeds extracted from other types of CDS corpus automatically. The distant supervision part as well as the multi-strategy learning part have not been covered in traditional seed-based pattern learning methods.
   For example, if “企业 (Company)” and “竞争对手 (Competitor)” are within one frequent word set. Given two sentences “企业的主要竞争对手有微软与甲骨文。(The major competitors of the company are Microsoft and Oracle.)”, and “主要竞争对手企业是索尼与苹果。 (Major competitive company are SONY and Apple.)”, the distinguished pattern for each sentence is “企业*竞争对手*有 (Company*Competitor*include)”, and “竞争对手*企业*是 (Competitor *Company* are)” respectively. The merged pattern is

**Table 2.** Classification of Identified Competitors

| | Is-competitor | Not-competitor |
|---|---|---|
| Extracted | A | B |
| Not-Extracted | C | D |

"(企业*竞争对手—竞争对手*企业)*(有—是)". It is more general to cover both situations. Here $*$ denotes any number of any characters, | means the union of several characters, and boundary strings such as "有 (include)" and "是 (are)" are included in the pattern.

4. **Extracting competitor names using patterns**
Based on patterns generated in the previous step, strings are extracted as competitor candidates. Note that not all candidates refer to organizations. Thus, we reuse the above mentioned three rules to check whether a string is an organization name, and thus filter out irrelevant ones.

## 5 Experiments

### 5.1 Experiment Setup

All prospectuses used in the experiment were crawled from the Web site of Shanghai Stock Exchange (`http://www.sse.com.cn`). Although our approach is unsupervised, labeled data is required to assess the quality and the coverage of extracted results. 836 prospectuses are manually labeled, which result in 3000 competitor pairs in total. Precision and Recall are used as the evaluation metrics. As shown in Table **??**, we use A to represent the number of correctly extracted competitor names, B is the number of incorrectly extracted competitor names, C indicates the number of competitor names that are not extracted. In this way, precision can be defined as $A/(A+B)$, and Recall is $A/(A+C)$.

Precision and recall are defined at two levels: the micro level and the macro level. The micro level evaluates on prospectuses while the macro level evaluates on competitor names. If the corpus has $n$ prospectuses, and for each prospectus $d_i$, we can get the corresponding $A_i$, $B_i$, and $C_i$ as defined in Table **??**. Precision and Recall at two levels are defined as follows.

$$Micro\ Precision = \frac{\sum_1^n precison(d_i)}{n} \tag{1}$$

$$Micro\ Recall = \frac{\sum_1^n Recall(d_i)}{n} \tag{2}$$

$$Macro\ Precision = \frac{\sum_1^n A_i}{\sum_1^n A_i + \sum_1^n B_i} \tag{3}$$

$$Macro\ Recall = \frac{\sum_1^n A_i}{\sum_1^n A_i + \sum_1^n C_i} \tag{4}$$

**Table 3.** Results of Each Type

| | List | Table | Text |
|---|---|---|---|
| Micro Recall | 0.9157 | 0.9691 | 0.7440 |
| Micro Precision | 0.9734 | 0.9143 | 0.9029 |
| Macro Recall | 0.9285 | 0.9493 | 0.7497 |
| Macro Precision | 0.9814 | 0.9557 | 0.9279 |

**Table 4.** Iterations for Table-type CDSs

| | 1 | 2 | 3 |
|---|---|---|---|
| #extracted candiates | 874 | 984 | 1015 |
| #new competitor names | 869 | 98 | 3 |
| Micro Recall | 0.9030 | 0.9686 | 0.9691 |
| Micro Precision | 0.9770 | 0.9663 | 0.9143 |
| Macro Recall | 0.8868 | 0.9433 | 0.9493 |
| Macro Precision | 0.9943 | 0.9827 | 0.9557 |

### 5.2 Results Evaluation

**Results for Each Type of CDSs** Table **??** shows the evaluation results for competitor identification in the corpus of each CDS type. Here, precision and recall are calculated against the corpus of each CDS type instead of the whole corpus. For instance, regarding the recall $A/(A+C)$ for the list type, $A$ is the number of correctly extracted competitor names in list-type CDSs, and $C$ is number of competitor names that are not extracted in the list-type corpus. The other metrics can be calculated in the similar way. From the table, we can find that our approach achieves very high precision for each type of CDSs. The recall for text-type CDSs is lower than that on other type of corpus, but is still about 0.74. This is because we only capture frequent patterns but some sentences about competitors are described by ad hoc lexical patterns that seldomly occur. We also find the gap between micro-level and macro-level is small, which means our approach is stable without very poor performance on some prospectus.

Since competitor identification on table-type CDSs and text-type CDSs are both iterative, we show the results after each iteration for the two types of corpus in Table **??** and Table **??** respectively. For both types, after a small number of iterations (3 for table-type, and 2 for text-type), the whole process terminates. As shown in Table **??**, for each iteration of table-type corpus extraction, more similar table headers are found, but fewer competitor names are extracted. For example, after the second Iteration, 984 candidates are found, but only 98 of them are recognized as organization names. This is due to the fact that most candidates are from headers similar to the headers which do not represent actual competitive table columns. Recalls increase after each iteration, but the precisions might drop a bit especially for the late iterations. This is a signal to tell that we should set more strict threshold values to ensure the quality of extracted competitors. We can have similar findings in Table **??**.

**Table 5.** Iterations for Text-type CDSs

| | 1 | 2 |
|---|---|---|
| #extracted candidates | 724 | 735 |
| #new competitor names | 675 | 7 |
| Micro Recall | 0.7345 | 0.7440 |
| Micro Precision | 0.9098 | 0.9029 |
| Macro Recall | 0.7272 | 0.7497 |

**Table 6.** Multi-strategy Learning Results

| | List | Table | Text | List+Table | List+Text | Table+Text | All |
|---|---|---|---|---|---|---|---|
| Micro Recall | 0.4715 | 0.3791 | 0.3406 | 0.7838 | 0.6544 | 0.5611 | **0.8423** |
| Micro Precision | 0.9688 | 0.9147 | 0.8944 | 0.9433 | 0.9109 | 0.8807 | **0.9090** |
| Macro Recall | 0.4114 | 0.4470 | 0.3142 | 0.7892 | 0.5814 | 0.6058 | **0.8487** |
| Macro Precision | 0.9814 | 0.9557 | 0.9279 | 0.9698 | 0.9540 | 0.9444 | **0.9437** |

**Results of Multi-strategy Learning** We also carry out experiments to compare the performance using multi-strategy learning with that based on single-strategy learning. The detailed experimental results are shown in Table **??**. The column "All" represents our final results, which combines the results from different types of CDSs. The column in form of "A+B" refer to the combined results from the A-type corpus and the B-type corpus. Unlike the computation of recall values in Table **??**, recalls are computed against the whole corpus instead of one specific type of CDS corpus. This is why for the first three columns (i.e. List, Table, and Text), their recalls are much lower than those reported in Table **??**.

From the table, we can see that recall values are greatly improved through multi-strategy learning. The micro recall value for list-type, table-type and text-type CDSs is 0.4715, 0.3791 and 0.3406 respectively. They are all below 0.5. However, the combined result is about 0.8423, almost 200% increases. In addition, even we use multi-strategy learning on two types of CDS corpus, the recall value improvements are obvious. Meanwhile, the precision values are still very high. All these findings show the effectiveness of multi-strategy learning.

### 5.3 Comparison with Traditional NER-based Methods

In order to identify competitors, we can also use named entity recognition (NER) methods to find organization mentions in the CDSs. Here, we select some popular NLP tools namely NLPIR[3], FudanNLP[4], and Stanford NER[5] to extract competitors from the corpus. For the Stanford NER, we further distinguish whether it uses distributional similarity features, denoted as Stanford NER with distSim, and Stanford NER without distSim. We use these tools as the baselines to com-

[3] `http://ictclas.nlpir.org/`

[4] `http://code.google.com/p/fudannlp/`

[5] `http://nlp.stanford.edu/software/CRF-NER.shtml`

**Table 7.** The Results of Different Methods

| | NLPIR | FudanNLP | Stanford NER with distSim | Stanford NER without distSim | Our Approach |
|---|---|---|---|---|---|
| Precision | 34.21% | 28.97% | 68.02% | 63.85% | **94.37%** |
| Recall | 1.02% | 2.43% | 47.30% | 40.80% | **84.87%** |

pare with our approach. Macro precisions and macro recalls of all methods are shown in Table **??**.

From the table, NLPIR and FudanNLP perform worst with low precisions (around 30%) and pretty low recalls (between 1% and 3%). When using distributional similarity features, Stanford NER can achieve more than 5% increases in terms of precision and recall. Compared with these baselines, our approach has much more promising results. The precision almost reaches 94.37% and the recall is also higher than 80%.

## 6 Conclusions and Future Work

In this paper, we provide a multi-strategy learning approach to extract competitors from Chinese prospectuses. Different kinds of competitive description sections (list-type, table-type, and text-type) require different extraction methods and have different levels of difficulties. We extract competitors from list-type CDSs first, and the extraction results are fed as seeds to boost the extraction process from other two CDS types. Distant supervised learning is used in these processes to avoid manual labeling efforts. One benefit of our approach is that the named entity recognition (NER) step is not required to identify competitors. Experimental results show our approach achieves higher precision and recall than those of the traditional NER methods. As for the future work, we plan to try our approach on English prospectuses and then extend to other corpus like company Web sites for mining competitors.

**Acknowledgements** This work is funded by the National Key Technology R&D Program through project No. 2013BAH11F03.

## References

1. Bao, S., Li, R., Yu, Y., Cao, Y.: Competitor mining with the web. Knowledge and Data Engineering, IEEE Transactions on **20**(10) (2008) 1297–1310
2. Lappas, T., Valkanas, G., Gunopulos, D.: Efficient and domain-invariant competitor mining. In: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM (2012) 408–416
3. Ciravegna, F., Chapman, S., Dingli, A., Wilks, Y.: Learning to harvest information for the semantic web. In: The Semantic Web: Research and Applications. Springer (2004) 312–326

4. Milne, D., Witten, I.H.: Learning to link with wikipedia. In: Proceedings of the 17th ACM conference on Information and knowledge management, ACM (2008) 509–518
5. Limaye, G., Sarawagi, S., Chakrabarti, S.: Annotating and searching web tables using entities, types and relationships. Proceedings of the VLDB Endowment **3**(1-2) (2010) 1338–1347
6. Agichtein, E., Gravano, L.: Snowball: Extracting relations from large plain-text collections. In: Proceedings of the fifth ACM conference on Digital libraries, ACM (2000) 85–94
7. Banko, M., Cafarella, M.J., Soderland, S., Broadhead, M., Etzioni, O.: Open information extraction for the web. In: IJCAI. Volume 7. (2007) 2670–2676
8. Etzioni, O., Cafarella, M., Downey, D., Kok, S., Popescu, A.M., Shaked, T., Soderland, S., Weld, D.S., Yates, A.: Web-scale information extraction in knowitall:(preliminary results). In: Proceedings of the 13th international conference on World Wide Web, ACM (2004) 100–110
9. Ciravegna, F., Gentile, A.L., Zhang, Z.: Lodie: Linked open data for web-scale information extraction. SWAIE **925** (2012) 11–22
10. Hao, Q., Cai, R., Pang, Y., Zhang, L.: From one tree to a forest: a unified solution for structured web data extraction. In: Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval, ACM (2011) 775–784
11. Gulhane, P., Madaan, A., Mehta, R., Ramamirtham, J., Rastogi, R., Satpal, S., Sengamedu, S.H., Tengli, A., Tiwari, C.: Web-scale information extraction with vertex. In: IEEE 27th International Conference on Data Engineering (ICDE 2011), IEEE (2011) 1209–1220
12. He, J., Gu, Y., Liu, H., Yan, J., Chen, H.: Scalable and noise tolerant web knowledge extraction for search task simplification. Decision Support Systems **56** (2013) 156–167
13. Dalvi, N., Kumar, R., Soliman, M.: Automatic wrappers for large scale web extraction. Proceedings of the VLDB Endowment **4**(4) (2011) 219–230
14. Gentile, A.L., Zhang, Z., Ciravegna, F.: Web scale information extraction with lodie. In: 2013 AAAI Fall Symposium Series. (2013)
15. Mintz, M., Bills, S., Snow, R., Jurafsky, D.: Distant supervision for relation extraction without labeled data. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2, Association for Computational Linguistics (2009) 1003–1011
16. Roth, B., Barth, T., Wiegand, M., Singh, M., Klakow, D.: Effective slot filling based on shallow distant supervision methods. arXiv preprint arXiv:1401.1158 (2014)
17. Roth, B., Barth, T., Chrupała, G., Gropp, M., Klakow, D.: Relationfactory: A fast, modular and effective system for knowledge base population. EACL 2014 (2014) 89
18. ChenYuan, X., HaoFen, W., Bo, J., Mengjie, W., Daqi, G.: Effective chinese organization name linking to a list-like knowledge base. In: 8th Chinese Semantic Web & Web Science Conference (CSWS 2014), Springer (2014)