Deep Understanding of Financial Knowledge through Unsupervised Learning

Chang Su, Wenjia Xu and Liangliang Zhang

Abstract—**In this project, a universal information extraction method was implemented and applied to financial area, which supports aggregation and self analysis of complex information from massive correlated sources. In order to extract domain-independent relations between entities, open information extraction algorithm is used. Firstly, we actively label dataset using unsupervised learning algorithm by leveraging semantic analysis techniques based on features such as grammatical structure, semantic roles, etc. The labeled training set will then be used to train extractor model, which will be further utilized to extract relations from any text efficiently and effectively. Finally, the extracted relations will be stored in database for future query. We compared and analyzed results of different models, measured the importance of features using feature selection and also built a simple UI tool to illustrate our methods.**

Index Terms **— relation based search, supervised and unsupervised learning**

I. Introduction

SEARCH engine is one of the greatest inventions in the information era. However, traditional search engine has limited capability of providing answers to complex requests. Traditional search engines are keyword-based, thus only return pages that directly match the words in user's query without aggregating essential information from multiple resources. The limitations restrict the quality and scalability of search results, which impedes further organization and usage of the searched results. Nowadays, large web corpus themselves has large set of data with numerous relations, but we're still in shortage of tools to fully leverage these valuable resources to improve searching experience.

Aiming at smarter searching with deeper understanding of relations between entities(words/relations) in user's query, it is important to build up relations between search keywords, enrich search results by providing users with not only directly related articles, but also indirect sources.

There are several reports on unsupervised learning algorithms that gather script like information without referring to previously labeled key words or predefined language structures. In this project, a universal information extraction method was implemented and was applied to financial area, which supports aggregation and self analysis of complex information from massive correlated sources. Moreover, the proposed algorithm will be able to find the chain of articles between seemingly unrelated keywords.

Chang Su, Wenjia Xu and Liangliang Zhang are with the department of Electrical Engineering, Stanford University, CA, USA.
(authors' e-mail address: changsu@stanford.edu, wenjiaxu@stanford.edu, lianglia@stanford.edu).

II. Method and Algorithm

In order to extract domain-independent relations between entities/words, open information extraction algorithm will be used. Initially, we will label dataset using unsupervised learning algorithm by leveraging semantic analysis techniques based on features such as grammatical structure, semantic roles, etc. The labeled training set will then be used to train extractor model, which will be further utilized to extract relations from any text efficiently and effectively. Finally, the extracted relations will be stored in form of inverted index table for future query.

We apply this method to financial area as an example, since financial topics are generally very hot and highly correlated, which makes the data resource easier to retrieve and the general result more interesting to users. While it can obviously be applied to other areas easily since it is a domain-independent model.

The tasks include：

1. Collect seed data from NYtimes API;
2. Generate Tree Structure using Stanford NLP Parser and label data as positive/negative and Collect seed data;
3. Generate Features, phrase chunk, part-of-speech tag, etc;
4. Train Extractor based on different models, including Naive Bayes, SVM, logistic regression, decision tree and boosting;
5. Use extractor on seed dataset collected and predict;
6. Export samples and predicted labels to database;
7. Build UI client.

Below, we will provide the detailed description of the tasks.

A. ***Collect Seed Data:***

We have built a php script that supports pulling data from New York Times through Article Search API. There are numerous fields and facets can be extracted or searched by, and we tailor the usage to our application by extracting title, url and head paragraph. Due to the fact that the API does not provide full article body search, we crawl their web pages based on url to get first page of the full article. The output of the script are files in JSON format,

B. ***Build Application Framework:***

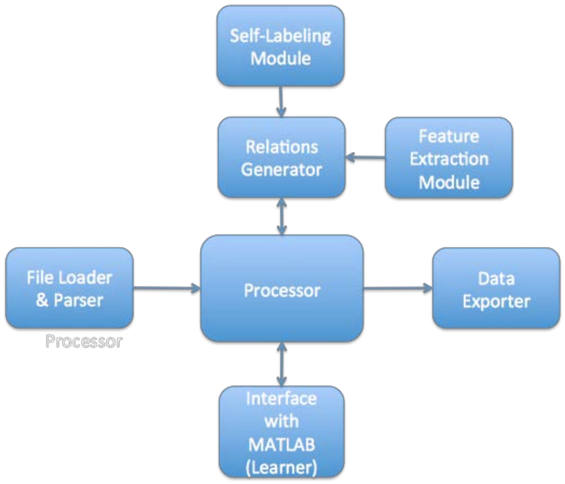


Fig. 1. Implementation of the framework.

We have build a Java application framework that partially implemented the whole unsupervised learning method as shown in Figure 1.

----- Processor

Process is the core part of the framework in terms of functionality, it imports and parses files, transforms articles into corpus of relations, namely samples that will be the input of our machine learning model and also saves samples into database for future query.

------ Relations Generator

Relations generator consumes articles passed by the processor and generate relations stemming from each article, and finally return the corpus back to processor. It will trigger two core models of the methods: self-labelling module and feature extraction module.

------ Self-labeling Module

This module utilized set of heuristic rules on each candidate relation and label it as positive or negative sample, please refer to point 4 “Apply Partial Rules/Heuristics in Self-Labeling” for details.

------ Feature Extraction Module

In order to realize relation-independent model, we extracted several features that do not depend on syntactic or semantic analysis at extraction time. Please refer to pint 3 “Extract Partial Features” for details.

------ Interface with MATLAB

Output: This module will output labeled relations into matlab as samples, where we will implement all kinds of learning algorithms and evaluation.

Input: This module will also input the predicted labels of new samples generated by our learned model.

C. ***Relation-independent Feature Extraction:***

Given a candidate relation consists of two noun phrases (E1, E2) and the whole sentence.

- words between E1 and E2
- number of words between E1 and E2
- number of stop words between E1 and E2
- number of punctuations between E1 and E2
- Entity Type of E1 and E2
- Number of phrases between E1 and E2
- POS to left of E1
- POS to right of E2

There are also other features that may or may not useful for the model, we will test out using feature selection in training.

D. ***Apply Partial Rules/Heuristics in Self-Labeling:***

The algorithm we used for separation and labeling positive and negative (to determine if the provided sentence could be classified in one of the certain relationships) is based on the one proposed in Dr. Michele Banko’s thesis “Open Information Extraction for the Web”.

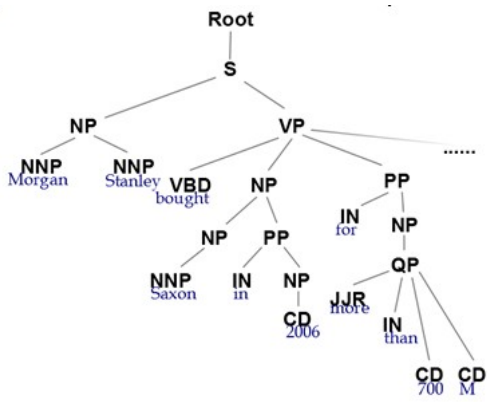


Fig. 2(a). Illustration of tree parser.

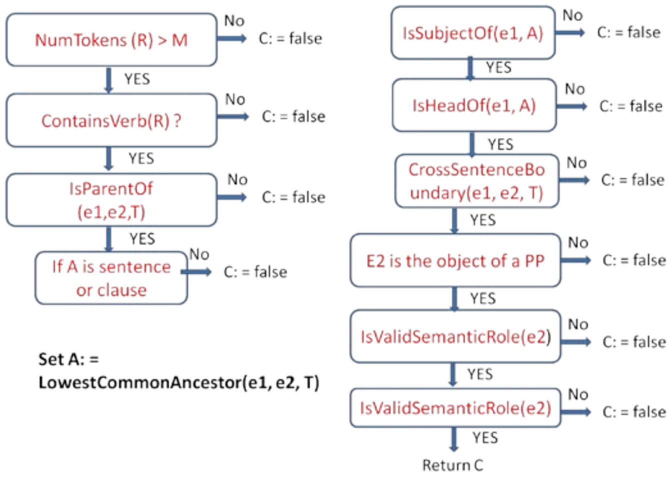


Fig. 2 (b). The algorithm to label the sentence, which is implemented using language parser[1]. The algorithm is a modified version of the one in ref [2].

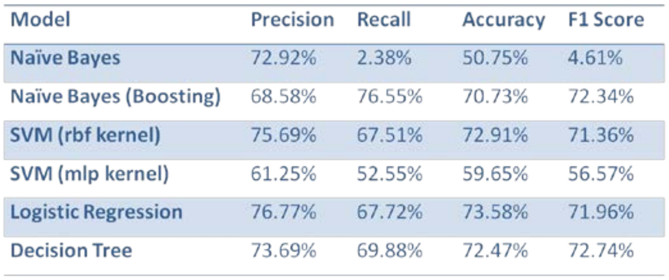
The input of the algorithm is a parsed tree of a sentence with the first node labeled 'S'. It first finds and stores the first NP (noun-phrase) E1 and another NP E2 that appears after E1. The algorithm then judges if

1) there is a verb in the given sentence;
2) if E1 is parent of E2;
3) if E1 is the subject and head of the sentence;
4) if the relation crossed the boundary of the sentence;
5) if E1 and E2 are within the correct semantic role; etc.

Then a true or false label is returned which indicated that the sentence belongs to one of the relationships.

III. Result and analysis:

With label and features at hand, we applied several classifiers for the binary classification, as shown in Fig.3.



Model	Precision	Recall	Accuracy	F1 Score
Naïve Bayes	72.92%	2.38%	50.75%	4.61%
Naïve Bayes (Boosting)	68.58%	76.55%	70.73%	72.34%
SVM (rbf kernel)	75.69%	67.51%	72.91%	71.36%
SVM (mlp kernel)	61.25%	52.55%	59.65%	56.57%
Logistic Regression	76.77%	67.72%	73.58%	71.96%
Decision Tree	73.69%	69.88%	72.47%	72.74%

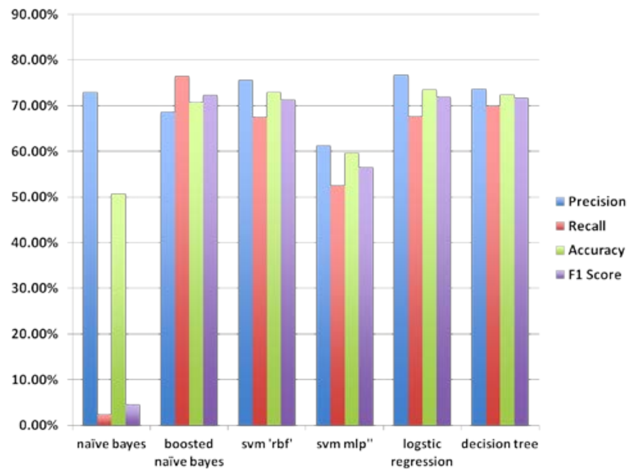


Fig. 3. The comparison of the Pecision, Recall, Accuracy and F1 score by using different models.

Taking the building and fixing approach, we first applied Naive Bayes classifier because it is a quick (though dirty) way to get a sense of variable distribution and also feature correlations. In Fig. 3, it is observed that the precision we got about Naive Bayes is around 73% while the accuracy and especially the recall is very poor. The classifier almost classifies all ground truth data as false and performs similar to a random classifier in terms of accuracy. The results indicate that the features we extracted are not independent to each other, in which case Naive Bayes model is not a good fit.

Considering the model performs a little bit better than random guesser, we then applies Adaboosting to construct a strong classifier based on these weak classifiers. In Fig. 3, we can see that Adaboosting improved the performance of the model considerably. Though dropped a little bit in metric of precision, boosted Naive Bayes classifier increase 70 percent in terms of Recall and F1 score, and 20% in accuracy. We also applied SVM considering that it's widely used, adaptive to features due to the fact that different kernels can be applied to map features to higher and more separable dimension space. In SVM, some of the kernels like linear and quadratic can't converge in expected time. Then we tried other kernels including Gaussian Radial Basis Function (RBF) kernel and Multilayer Perceptron kernel.(MLP). From the figure we can conclude that RBF kernel performs better than MLP and the dataset is non-linear separable.

Additionally, considering the fact that our feature space contains both scale and also category features, we then apply decision tree which can be fit in this case. In decision tree model, we apply three different kinds of split criterion, Gini's diversity index, twoing rule, and maximum deviance reduction(cross entropy) and choose the minimum one as the final result shown in Fig. 3 as well. Finally, we apply another common classifier logistic regression and get result similar to decision tree and svm. However, logistic regression runs faster and is computationally efficient.

Overall, comparing all models, we can see that apart from Naive Bayes which performs like a random classifier, all other models Boosted Naive Bayes, Decision Tree, SVM(rbf), logistic regression can achieve precision around 74%, recall around 68%, accuracy and F1 score around 71%. SVM(mlp) performs relatively poor compared with the four above, but can still achieve accuracy around 60%. In further observation, among all models except boosted Naive Bayes, all of them have higher precision than recall, which means that the classifier is relatively more likely to ignore a real relation than include a false relation. The property is expected in our case in the sense that tons of relations can be generated among large web corpus of articles and we're more in need of accurate and well-filtered relations rather than spans of non-relevant relations.

Features eliminated	accuracy
All feature	**0.71652**
Eliminate num stop words	**0.72162**
Eliminate e2 entity	**0.72604**
Eliminate num cap words	**0.72842**
Eliminate e1 entity	**0.71618**
Eliminate num nps btw	**0.7087**
Eliminate num words	**0.70666**
Eliminate num puncs	**0.70156**
Eliminate right e2 pos	**0.68831**

Table. 1. Comparison of the accuracy of all features.

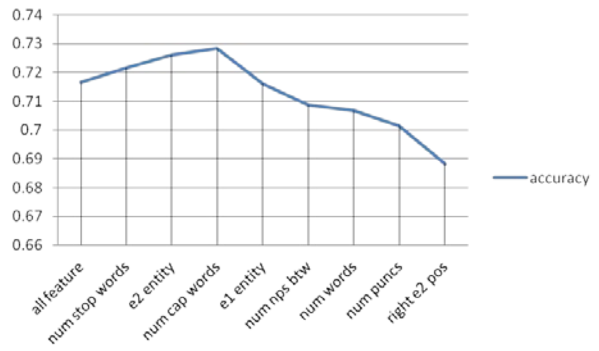
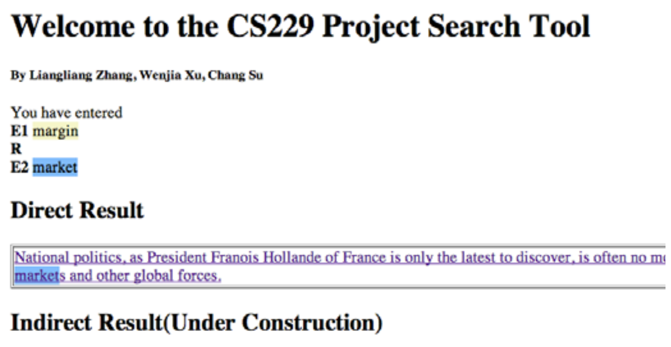


Fig. 4.The accuracy of feature selection.

To investigate the influence of different features, we applied backwards feature selection which eliminates one feature at a time to illustrate its influence. The feature eliminated each time is the one which made the least influence to accuracy. Due to the performance we choose 'logistic regression' as the model.

We plot the accuracy with remaining features along with the procedure in Fig. 4. The x-axis indicates the feature we eliminated each round and the y-axis indicates the accuracy. From the figure we can see among all features, feature 'num cap words', 'e1 entity' and 'right e2 pos' count the most.

IV. Application:


Welcome to the CS229 Project Search Tool

By Liangliang Zhang, Wenjia Xu, Chang Su

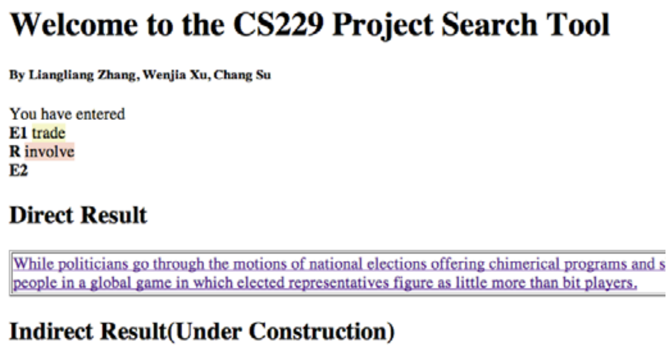
You have entered
E1 margin
R
E2 market

Direct Result

National politics, as President Franois Hollande of France is only the latest to discover, is often no m
markets and other global forces.

Indirect Result(Under Construction)

Fig. 6. the demonstration of the use case 1.


Welcome to the CS229 Project Search Tool

By Liangliang Zhang, Wenjia Xu, Chang Su

You have entered
E1 trade
R involve
E2

Direct Result

While politicians go through the motions of national elections offering chimerical programs and s
people in a global game in which elected representatives figure as little more than bit players.

Indirect Result(Under Construction)

Fig. 6.The demonstration of the use case 2.

Combing all part of the system (data collection, unsupervised labeling, supervised learning), we finally built a simple UI tool[3] to better illustrate and visualize our goal.

Basically, the tool is targeted to support search based on relations. Three parts compose the query E1, E2 (usually nouns), R (usually verb). Though you can use all combinations of the three criterions, we expect to issue queries in form of E1 & E2, E1 & R or R & E2 because it unveils the essence of query based on relations. Fig. 5 and Fig. 6 illustrate some examples use cases. It should be also noted that since the tool is used just as a simple illustration of our methods, the current relation database is generated by only 100 articles collected from New York times related to finance. In addition, many relations are eliminated by our trained model. Therefore, the data pool is relatively small and you should not be surprised with the fact that you may get no result 8 out 10 times or so. For detail usage, please refer to our documentation in the git repository[4].

V. Conclusion and Future work

In this project, we leverage the open information extraction algorithm to actively label data, then train domain-independent models that can filter and extract useful relations from articles automatically and efficiently. The next big step to go is that since we have generate many relations, we can build a huge knowledge graph that provide us with the capability of graph search, which kind of stimulates and facilitates reasoning. In this case, given two entities, we can not only return the result if they appear in one document simultaneously, it's also possible for us to traverse the graph and find path from entity1 to entity 2, namely, chains of relations (articles) they may indicate deeper relation between entities.

Acknowledgment

The authors would like to thank the useful discussion and suggestion from Mr. Botao Hu.

References

[1] the Stanford parser: http://nlp.stanford.edu/software/lex-parser.shtml.
[2] M. Bancho, "Open Information Extraction for the Web," *Ph.D. thesis*, 2009.
[3] http://www.stanford.edu/~changsu/cgi-bin/cs229/CS229/UI_tool/
[4] https://github.com/changsu/CS229.git