A Machine learning Filter for Relation Extraction

Kevin Lange Di Cesare,
Michel Gagnon
Polytechnique Montréal
Montreal, Canada
{kevin.lange-di-cesare,
michel.gagnon}@polymtl.ca

Amal Zouaq
University of Ottawa
Ottawa, Canada
azouaq@uottawa.ca

Ludovic Jean-Louis
Netmail Inc.
Montreal, Canada
ludovicj@netmail.com

ABSTRACT

The TAC KBP English slot filling track is an evaluation campaign that targets the extraction of 41 pre-identified relations related to specific named entities. In this work, we present a machine learning filter whose aim is to enhance the precision of relation extractors while minimizing the impact on recall. Our approach aims at filtering relation extractors' output using a binary classifier based on a wide array of features including syntactic, lexical and statistical features. We experimented the classifier on 14 of the 18 participating systems in the TAC KBP English slot filling track 2013. The results show that our filter is able to improve the precision of the best 2013 system by nearly 20% and improve the F1-score for 17 relations out of 33 considered.

Keywords

Information Extraction; Relation Extraction; Slot Filling

1. INTRODUCTION

In this paper, we focus on the TAC KBP English slot filling (ESF) track, an evaluation campaign that targets the extraction of 41 pre-identified Wikipedia info-box relations (*title, spouse*, etc.) related to specific named entities (persons and organizations) [6]. A named entity (the query entity) and a relation (the slot) are submitted to the system, which must find every other entity (the filler) that is linked to this entity with this particular relation, and must return a textual segment that justifies this result[1] [6]. In 2013, the top performing system, Relation Factory, achieved a recall of 33.2%, precision of 42.5% and F1 of 37.3% [5]. We propose to complement the output of a relation extraction system with a filtering step. Our aim is to tackle the following research questions:

(1) *Which features can be used for filtering the output of relation extractors?*

(2) *Can we increase the precision of slot filling systems without impacting their F1-score?*

[1] Examples of filtered responses, the list of relations and further results are available at:
http://westlab.herokuapp.com/extractionfilter/appendix

WWW'16 Companion, April 11–15, 2016, Montréal, Québec, Canada.
ACM 978-1-4503-4144-8/16/04.
http://dx.doi.org/10.1145/2872518.2889397 .

2. METHODOLOGY

Our approach consists in using a machine learning filter trained on the output of the ESF participating systems. For each relation, a binary classifier is appended to the end of a relation extractor's pipeline. It eliminates responses which it classifies as *wrong* with the goal to improve its precision. The filter cannot increase recall scores as it does not generate additional responses. Reponses from the relation extractors are passed to the part-of-speech (POS) tagger and syntactic analysis parser which return the POS tags and the syntactic dependency tree [2]. Our approach is based on a wide selection of generic features which are listed in Table 1 ranging from statistical, lexical to syntactic features. As we are using syntactic-based features, we only focus on intra-sentence relations. To enhance our classifier, we captured the context between the query entity and the filler at the lexical, the morpho-syntactic and the syntactic levels. The contexts are extracted from the raw justification and from the POS/syntactic analysis output. To limit the number of contexts, we retained the most frequent subsets of part-of-speech tags between the query entity and the filler for each class (*correct/wrong*) by using the Apriori algorithm after separating *wrong* instances from *correct* ones, for each relation [1]. The Apriori algorithm allows us to retain the desired subsets with reduced computational resources, by first identifying individual items, in this case a single POS tag, for which the support is greater than the minimum support and second, by extending them to larger item sets by retaining only the subsets which have a support greater than a minimum support, 15% in this case. Similarly, we used the Apriori algorithm to retain the most frequent subsets of words within the sentence (excluding stop-words & named entities) as well as syntactic dependencies and {POS tag, syntactic dependency & relation} tuples within the syntactic dependency tree. Each retained subset is translated into a boolean feature indicating the presence of the subset. For each relation, we experimented various classifiers and kept the one with the highest F1 score on the *correct* class. Models are selected from one of the following base classifiers: RandomForest, SMO, NBTree, DecisionTable, J48 and K* [3]. The evaluation is done on the training set using a 10 fold cross-validation. To avoid over-fitting, we do not filter relations for which the number of training instances is inferior to 15 for either class (*correct/wrong*). To prevent favoring the *wrong* class, for each relation where the number

Table 1: Full set of generic features used for filtering

Name	Description
Statistical features	
Sentence length	Number of tokens in the sentence
Query/filler length	Number of tokens within the query and filler (strings in the justification)
Entity order	Order of appearance of query and filler
#tokens left/between/right	Number of tokens left/right or between the query and the filler in the sentence
Confidence score	Score given by the relation extractor [6]
Lexical/POS features	
POS fractions	Proportion of nouns, verbs, adjectives and others (left/between/right/sentence) [4]
POS subsets	Most frequent subsets of POS tags between the query and the filler
Word subsets	Most frequent subsets of words in the sentence (excluding stop-words and named entities)
Syntactic features	
Distance between entities	Distance between the query and the filler in the syntactic dependency tree
Entity level difference	Level difference in the syntactic dependency tree between the query and the filler
Entities are ancestors	The query and the filler are ancestors in the syntactic dependency tree
Syntactic dependencies subsets	Syntactic dependencies in the syntactic dependency tree between the query and the filler
Multilevel subsets	{POS tag, syntactic dependency & relation} tuples in the syntactic dependency tree

of *wrong* instances is more than double the number of *correct* instances, we down-sample the subset of *wrong* instances to five subsets randomly selected. Each subset contains the same number of instances as the *correct* class subset. For relations which have been rebalanced, we train six classifiers for the five subsets of the majority class and retain the classifier/subset pair which has the greatest F1 score on the *correct* class.

3. EXPERIMENTS & EVALUATION

We excluded 4 systems out of 18 as the offsets they provided did not allow us to extract the justifications correctly. Our training data is obtained by merging all systems outputs. The experiment was executed for each of the 14 systems by holding out its own output from the training data. Our filter increases the precision score for every tested relation extractor compared with their original performance. The filter results in an average increase in precision of 8.8% for the 14 systems[1]. We tested different combinations of features. Table 2 reports our results. We used as baseline Relation Factory's highest F1 run, on which we applied our filter for all the feature sets. We also consider its highest precision run for comparison. The combination of statistical, lexical/POS and syntactic features render the greatest performance: precision increases from 42.5% to 61.6% for a 19.1% increase. The F1 is increased from 37.2% to 37.7%. The precision achieved is also 10.7% higher than Relation Factory's highest precision run. From the 33 filtered relations, the precision was increased for 26 relations and F1 was increased for 17 relations. Following the filtering, F1 and precision were decreased for only four relations. Performances could not be measured for three relations due to a lack of test data.[1]

Table 2: Results obtained by Relation Factory after filtering when using different feature sets

Feature Set	R	P	F1
Relation Factory (best F1)	33.2	42.5	37.3
Relation Factory (best precision)	25.9	50.9	34.3
Statistical	25.6	57.4	35.4
Statistical + Lexical/POS	26.6	61.4	37.1
Statistical + Syntactic	25.3	58.2	35.2
Statistical + Lexical/POS + Syntactic	**27.1**	**61.6**	**37.7**

4. CONCLUSION

We present a generic method to increase the precision of relation extractors by appending a machine-learning-based filter to the relation extractor's pipeline. The filter utilizes a wide scope of features, including statistical, lexical/POS and syntactic features. The features used to train the classifiers are generic and could be used for classifying any pre-defined relation. Our filter has allowed an increase in precision of nearly 20% for the best performing system in the 2013 slot filling track. We believe that the performance of the filter can be improved by exploiting more training samples of better quality. We are working on improving the features using various feature selection approaches.

5. ACKNOWLEDGMENTS

This research was partially funded by the Industrial Innovation Scholarships Program.

6. REFERENCES

[1] R. Agrawal, R. Srikant, et al. Fast Algorithms for Mining Association Rules. In *20th VLDB*, 1994.

[2] M.-C. De Marneffe, B. MacCartney, C. D. Manning, et al. Generating Typed Dependency Parses from Phrase Structure Parses. In *LREC*, 2006.

[3] M. Hall, E. Frank, G. Holmes, and al. The Weka Data Mining Software: An Update. *ACM SIGKDD*, 2009.

[4] M. Mintz, S. Bills, and al. Distant Supervision for Relation Extraction without Labeled Data. ACL, 2009.

[5] B. Roth, T. Barth, M. Wiegand, M. Singh, and D. Klakow. Effective Slot Filling Based on Shallow Distant Supervision Methods. *CoRR*, 2014.

[6] M. Surdeanu. Overview of the TAC 2013 Knowledge Base Population Evaluation: English Slot Filling and Temporal Slot Filling. In *TAC 2013*, 2013.