Semantic Relations for Actor Entities

Benny Longwill

June 2019

Abstract

Relationship extraction plays a vital role in many Natural Language Processing applications. This system was designed to test the effects of extractor type, sentence selection criteria, and classifier type on the precision, recall, and accuracy of semantic relation extractions in the implementation of a distantly supervised algorithm. Previous information extraction systems that measured precision have reported positive results for both extraction through dependency path parsing methods and simple collection from an available online database. Furthermore, previous researchers also achieved high results using an info-box heuristic to generate distantly labeled training data in conjunction with Naive Bayes, CRF, or Logistic classifiers. Thus, the current study examined the effects of a dependency tree parse semantic relation extractor in conjunction with a Wikipedia info-box heuristic to label training data used in a Naive Bayes classifier in an attempt to confirm and add clarity to the effect. Results indicated that these methods do improve precision and total average recall; however, modification was needed in order to properly compare results with previous systems.

1 Background

In recent years there has been great influx of research into the field of Information Extraction (IE) in order to find new ways to solve the fundamental problem of retrieving structured information from many pieces of naturally unstructured text. More specifically relationship extraction, extracting semantic relations in the form of a tuple (i.e., t = (**arg$_1$**, **semRel**, **arg$_2$**) where **arg** represents entities and **semRel** is the relation between them), has been found to have vital applications in natural language processing. More and more research has been devoted toward semantic relation extractions in IE because of its particularly high utility in creating machine-accessible and diverse knowledge bases that can be used in ontologies for complex artificial intelligence and spoken dialogue systems and IE in search applications.

The most effective semantic relation extractions systems are widely dominated by supervised learning algorithms that traditionally require humanly annotated training data. This approach to obtaining training data is both tedious

and expensive. That is to say that annotators require both monetary payment and also take a long time to annotate large amounts of data. For this reason, the current paper investigates an alternative distantly supervised approach.

Researchers in this area have been examining the efficacy of systems that utilize distantly supervised machine learning for precision and recall. In these studies, the method of extraction, method of obtaining training data, and method of classification were reported and results were compared. Knowing how these methods in the machine learning pipeline affect the quality and quantity of semantic relation extractions may indicate the value of these methods in predicting the efficacy of an overall system for obtaining semantic relations.

Previous studies have reported mixed results for the effect of using different distantly supervised machine learning methods in semantic relation extraction. For example in Banko et al. (2007), the 'Text Runner' system used a small collection of parsed self-labeled samples to train a "trust-worthy tuple" binary Naive Bayes classifier that classified a tuple as a good or bad candidate to extract training data. Then it walks through any large corpus (i.e., the web) and in a single pass extracts relations using a simple pattern extractor that extracts all text between noun phrases and ranks it's confidence based off text frequency. The extractor did not use a parser but rather part of speech to obtain relation tuples. Results presented by Wu and Weld (2010) showed that 'TextRunner' achieves an F-measure from between 30% to 40% across three large corpuses.

Furthermore, Mintz et al. (2009) achieved higher results using Freebase, an online semantic relation database was utilized to generate distantly labeled training data. Because the semantic relations were human-created and already accessible, no extractor was needed. The algorithm traversed a large unlabeled corpus, and for each semantic relation, all sentences for which the pair of entities appeared were selected and both syntactic and lexical features were extracted and used toward training data for a multi-class logistic classifier optimized using L-BFGS with Gaussian regularization. The classifier took the entity pair and a feature vector as its input and returned a semantic relation name. Results showed that the P@100 was highest at .69 for both syntactic and lexical together and for P@1000 was highest at .68 for just the syntactic features.

Finally, Wu and Weld (2010) received conflicting results. They implemented an open IE extractor that utilized a self-supervised learning method similar to 'TextRunner' but with the addition of a Wikipedia info-box heuristic. With this heuristic, their system selected attributes from the info-box to the right side of any Wikipedia entry and passed through the database text selecting sentences for training data that contain the attribute and the subject of the Wikipedia entry. This system used one of two parsers to train a classifier. The first, $\text{WOE}_{\text{parse}}$, used features from dependency-parse trees to train a pattern learner to classify whether the shortest dependency path between two noun phrases indicated a semantic relation. The second, WOE_{pos}, used shallow features like POS tags to train a conditional random field to output relational text between noun phrases. Neither of the two extractors utilized individual words or lexical features for training data. Both extractors outperformed the 'TextRunner' system. $\text{WOE}_{\text{parse}}$ achieved an F-measure around 40-45% ; whereas, $\text{WOE}_{\text{parse}}$

achieved around 60-65% but completed the task at a much slower rate.

2 Approach

The purpose of the current study was to investigate how the extractor, selection criteria for labeling, and classifier affect the precision, recall, and accuracy of semantic relation extractions in the implementation of a distantly supervised algorithm. Based on the results of Wu and Weld (2010) $\text{WOE}_{\text{parse}}$, I hypothesize that the use of the Textacy parser in order to extract semantic relations from sentences will produce higher recall than the previous systems that extract based on shallow features alone. Although their system took 30 times longer to run, according to Wu and Weld (2010), "dependency-parse features are highly informative when performing unlexicalized extraction." Thus given the smaller-scale nature of the current system, results could be extrapolated to give insight on the computational cost of dependency structure parsing on a IE system.

Also based on the results of Wu and Weld (2010), I hypothesize that implementing the Wikipedia info-box heuristic into the current system will increase the average precision and recall higher than in systems without this heuristic. The knowledge base on Wikipedia, in the form of an info-box label to the right of most Wikipedia entries, can be leveraged to generate fact informed labels for training data by gathering candidate sentences in any text that contain both the info-box attribute, and title for the Wikipedia entry. (e.g., if the Wikipedia info-box says Robin Williams died in Paradaise, California then all sentences in another text that contain some derivation of Robin Williams and Paradise will be counted as a candidate sentence and labeled with the relation 'died'. This hypothesis is to replicate the previous results of the Wikipedia info-box heuristic to confirm it's efficacy in distantly supervised systems. Results could be generalized to allow for investigation of this heuristic in other domains outside of the Wikipedia online database.

Finally, based on the results of Banko et al. (2007), I hypothesize that if the current semantic relation extractor utilizes a Naive Bayes classifier to first run a binary classification, the average precision and recall reported will be significantly higher than previous systems. This is due simply to the efficient and robust- to-noise nature of Naive Bayes. According to (McCallum and Nigam, 1998), "In empirical results on five real-world corpora we find that the multinomial [Naive Bayes] model reduces error by an average of 27%, and sometimes by more than 50%." Thus, Naive Bayes is a strong candidate to be utilized for good precision and efficiency on large scale applications.

3 Methodology

3.1 Phase 1

In the current study the first phase was devoted to text processing, training candidate generation, and featurization in order to create training data used

in classification. To begin, a list of nine million, three hundred six thousand, three hundred nine names were parsed from a file of famous participant facts obtained from a file *name.basics.tsv.gz* file downloaded from the IMDB database website[4]. These names were separated into two lists in order to hold out the testing data source for later evaluation. For the training list of names, candidate sentences were accessed from HTML extracted using the library BeautifulSoup[7] and sentences were generated from utilizing the Wikipedia info-box heuristic with parentheses and bracketed contents removed. If a name was unable to be accessed then it was simply skipped and the next name on the list was attempted. Unlike Wu and Weld (2010), The current system utilized a first-come first-serve method of selection for most classes in which only one candidate could be selected and the first sentence encountered that contained some iteration of both arg_1 and arg_2 was selected as the candidate sentence for that class. The 'occupation' class, on the other hand, was allowed to select more than one candidate due to the appearance of a greater number of positive relation contexts.

3.1.1 Featurization

Each sentence was featurized by creating a bag of words that used real token count values after tokenizing, removing stop words, lowercasing, stemming, removing proper nouns, cardinal digits, adjectives, prespositions, words that aren't alpha numerical, and words that contain either of the two semantic relation arguments.

3.1.2 Training Data Set

Overall, six-hundred-eighty multi-class training vectors were generated from corresponding Wikipedia entries using the info-box heuristic and were distantly tagged for each of five classes (i.e., died, born, occupation, spouse(s), education). These same vectors were also separately recorded with the binary label 'present' to be used to be used as training data for a binary classifier. Finally for each 'present' binary label created, a random sentence from a random Wikipedia page was accessed using the link, 'https://en.wikipedia.org/wiki/Special:Random' in order to generate a corresponding 'absent' vector as negative training data. In general, cases where content was not present in a candidate sentence, no vector would be produced resulting in a disparity of class counts. For this reason, the multi-class vector list was trimmed in order to ensure uniform vector counts. The resulting training vectors were used to create a binary and a multi-class language model.

3.2 Phase 2

Phase two of the current system required extracting semantic relations from test sentences and using a binary Naive Bayes classifier to first detect if a relation contained any form of the target relations. Relations that were classified as

having one of the target relations present were input to a second multi-class classifier that returns the predicted class label.

3.2.1 Extractor

The Textacy.extract `subject_verb_object_triples()` method extracts an ordered sequence of subject-verb-object (SVO) triples from a spacy-parsed doc and was used as a foundation for the current system's extractor. This method was modified to allow prepositional phrase constituents to be extracted as part of the relation. Furthermore, this method was also modified to output binary relations in cases of intransitive verbs and an issue was fixed with auxiliar verbs in order for them and the main verb to be counted as one unit (e.g., 'was burried' would be counted as a single constituent)

3.2.2 Classifier

The binary and multi-class classifiers chosen in the current study was the Multinomial Naive Bayes Classifier from McCallum and Nigam (1998). Features used were real value numbers of occurrences and included add-one smoothing in order to return a smooth confidence percentage for each classification. Unknown words were skipped when calculating classification. The following is the formula used for classification:

$$classify(d_i) = argmax_c P(c) \prod_{k=1}^{|V|} P(w_k|c)^{N_{ik}}$$

where class c prior probability is :

$$P(c_i) = \frac{1 + Cnt(c_i)}{|C| + \sum_i Cnt(c_i)}$$

and the word w conditional probability is:

$$P(w_t|c_j) = \frac{1 + \sum_{i=1}^{|D|} N_{it} P(c_j|d_i)}{|V| + \sum_{s=1}^{|V|} \sum_{i=1}^{|D|} N_{is} P(c_j|d_i)}$$

where V represents the vocabulary set and D represents document set

4 Evaluation

4.1 Naive Bayes Classifier Evaluation

In the current study, the efficacy of two Naive Bayes classifiers were evaluated using human annotated test vectors derived from a held out set of Wikipedia articles. Using the same method to generate training data, newline separated candidate sentences were written to a test output file, raw without featurization or bag of words. Next, any distantly supervised labels previously added were

stripped in order for the extractor to output a list of test extractions to a file. Two humans annotated these relations creating a gold standard.

Figures 2-4 show confusion matrices that indicate the accuracy, precision, recall, and F1 scores that were automatically calculated for the binary Naive Bayes Classifier across three level (i.e., 10, 25, and 50). Looking at the pattern of results displayed in Figures 2-4, it appears that relations, in general, were able to be correctly classified above 79% percent of the time across all levels. More specifically, accuracy levels begin around 85% at k=10 but fall until .79% at k=50.

Furthermore, Figures 5-7 show confusion matrices for the mutli-class Naive Bayes classifier and also indicate the accuracy, precision, recall, and F1 scores across the same three levels. Total Accuracy in general appears stable, beginning at 84% at k=10, falling to 82.4% at k=25, then rising again to 86% at k=50. However, under further investigation of precision across classes, it appears that there is a statistically significant drop in precision and increase of recall for one particular across all levels. That is to say that all relations, except the 'occupation' relation, maintain a precision 80-100% across all levels; whereas, the 'occupation' relation's precision remains significantly lower at 40-44% across all levels.

4.2 Extractor Recall

The efficacy of the current study's extractor design was also evaluated. One hundred sixty test sentences from vectors previously used in the Naive Bayes evaluation were sent through the `run_semantic_extraction` pipeline. This method accepts a list of sentences as input, extracts semantic relations, classifies them using the Naive Bayes classifiers, and outputs the classification results and a count file that indicates how many total raw relations were extracted from each sentence. Furthermore, semantic relations were also separately extracted from these same test sentences using OpenIE to be used for comparison. OpenIE is a publicly available and well known semantic relation extractor that consistently outperforms state-of-the-art systems on benchmark data sets; for this reason, it's total count of semantic relations by sentence was used as the gold standard for evaluating the current system's recall. Using the recall formula with an imposed cieling of 1.0:

$$R = \frac{\texttt{total current system extractions}}{\texttt{total OpenIE extractions}}$$

the recall was calculated per sentence and in Figure 8 total average recall for the 180 sentence is reported as 70%.

5 Conclusion

This study was designed to investigate how the extractor, candidate selection criteria, and classifier type affect the precision, recall, and accuracy metrics of a

distantly supervised semantic relation extraction system. I predicted that that the use of a dependency path parser to make extractions, that implementing the Wikipedia info-box heuristic into content selection, and utilizing a Naive Bayes classifier would increase the average precision and recall higher than in systems without these implementations. The results of this study indicated that these methods do positively affect accuracy, precision, and recall; however, there are several points of inconsistency that must be considered and fixed in further research before a true comparison with previous systems could be solidified.

First and foremost, it must be made clear that the design of current system has five target relations solely about IMDB entities, and has been tested using only Wikipedia. In contrast, the previous systems worked on a much larger scale and are able to extract an indeterminate amount of relation types from a variety of sources. For example, the 'Text-Runner' system, from Banko et al. (2007), runs by traversing pages on the web. This inconsistency limits the current system's semantic relation extractions and could be patched by incorporating more diverse machine accessible database fact-boxes to obtain training data and by directing the system access web pages non-specific to Wikipedia for gathering training data.

Furthermore, there was also an extraneous variable within data collection that prevented obtaining full accuracy in both Naive Bayes classifiers. The multi-class Naive Bayes classifier had mixed precision values across classes with the 'occupation' class consistently receiving low precision from 40-44% across levels. Given this inconsistency, it is quite clear that data collection for the 'occupation' training vectors is biased in some way. It may be the case that allowing the system to gather additional candidate sentences during Phase 1, while all other target relations only accept a single candidate sentence per Wikipedia page has created additional noise for the particular class. This extraneous variable in training vectors may also negatively affect the binary classifier given that the vectors are identical but with different tags. As a result, in future research this feature that allows for a single Wikipedia entry to produce multiple candidate occupation sentences must be removed.

Finally, the method for calculating total average recall may have been oversimplified. In Wu and Weld (2010), the method for annotation of gold standard test data involved both their human grader and also those of Mechanical Turk. For this reason they had a very robust test data set that was not derived using another system to create the gold standard. The current study utilized OpenIE to create a gold standard and evaluated recall based on total count of relations. In general, this disregards the quality of extraction because it counts any output relation as identical to the gold standard even though this may not be the case. In future research if human annotation is not available due to the high cost, a method of comparison that includes quality of extraction (e.g. unigram comparison between tuples) must be implemented to obtain a more accurate total average recall.

This study examined the effects of using a dependency path based extractor, a first-come first-serve candidate sentence selector, and the use of a Naive Bayes classifier on semantic relation extraction precision and recall. These results in-

dicated that the methods chosen provide significant increase in these metrics; however, there are some inconsistencies in design that must be reconsidered in order to properly compare this system with previous work. The results of this study may benefit future investigations that explore the utility of these modules or modules similar to these in order to improve semantic relation extraction quality.

References

[1] Bird, Steven, Edward Loper and Ewan Klein (2009), Natural Language Processing with Python. O'Reilly Media Inc.

[2] Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In Proceedings of the 20th international joint conference on Artifical intelligence (IJCAI'07), Rajeev Sangal, Harish Mehta, and R. K. Bagga (Eds.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2670-2676.

[3] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. 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 (ACL '09), Vol. 2. Association for Computational Linguistics, Stroudsburg, PA, USA, 1003-1011.

[4] "name.basics.tsv.gz File Contains Information for IMDB Actors." IMBD Internet Movie Data Base.

[5] Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 1003–1011, Suntec, Singapore, 2-7 August 2009. c 2009 ACL and AFNLP

[6] Stanovsky, Gabriel Michael, Julian Zettlemoyer, Luke Dagan, Ido. (2018). Supervised Open Information Extraction. 885-895. 10.18653/v1/N18-1081.

[7] Vineeth G. Nair. 2014. Getting Started with Beautiful Soup. Packt Publishing.

[8] "Wikipedia The Free Encyclopedia." Wikipedia, www.wikipedia.com/.

[9] Wu, F., Weld, D. S. (2010, July). Open information extraction using Wikipedia. In Proceedings of the 48th annual meeting of the association for computational linguistics (pp. 118-127). Association for Computational Linguistics.

Appendix

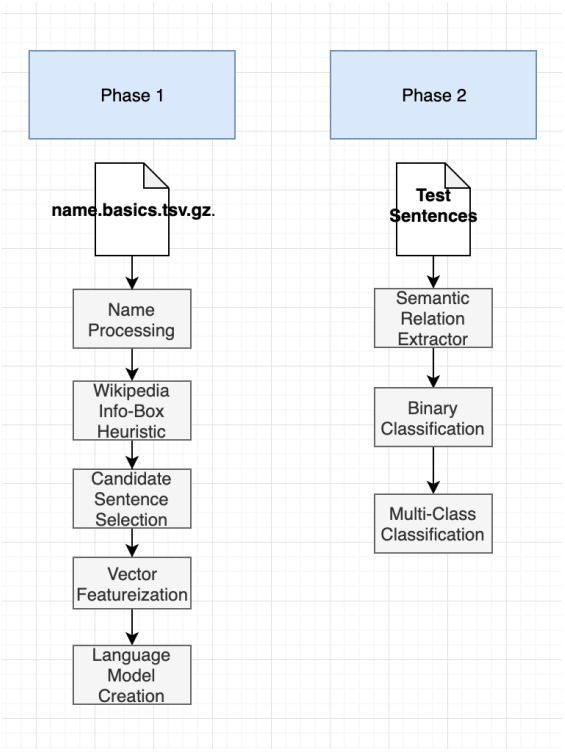

Figure 1: System Architecture

	abs	pre				
abs:	9	1	abs:	P=0.9	R=0.81818	F1=0.85714
pre:	2	8	pre:	P=0.8	R=0.88889	F1=0.84211
	Test Accuracy=.85					

Figure 2: Binary Classifier k=10

	abs	pre				
abs:	21	4	abs:	P=0.84	R=0.84	F1=0.84
pre:	4	21	pre:	P=0.84	R=0.84	F1=0.84
	Test Accuracy=.84					

Figure 3: Binary Classifier k=25

	abs	pre				
abs:	34	16	abs:	P=0.68	R=0.87179	F1=0.76405
pre:	5	45	pre:	P=0.9	R=0.7377	F1=0.81081
	Test Accuracy=.79					

Figure 4: Binary Classifier k=50

	bor	die	edu	occ	spo				
bor:	10	0	0	0	0	bor:	P=1.0	R=0.76923	F1=0.86957
die:	0	8	0	1	1	die:	P=0.8	R=0.88889	F1=0.84211
edu:	0	0	10	0	0	edu:	P=1.0	R=1.0	F1=1.0
occ:	3	1	0	4	2	occ:	P=0.4	R=0.8	F1=0.53333
spo:	0	0	0	0	10	spo:	P=1.0	R=0.76923	F1=0.86957
	Test Accuracy=0.84								

Figure 5: Multi-Class Classifier k=10

	bor	die	edu	occ	spo				
bor:	24	0	1	0	0	bor:	P=0.96	R=0.8	F1=0.87273
die:	0	23	0	0	2	die:	P=0.92	R=0.88462	F1=0.90196
edu:	1	1	22	0	1	edu:	P=0.88	R=0.95652	F1=0.91667
occ:	4	2	0	10	9	occ:	P=0.4	R=1.0	F1=0.57143
spo:	1	0	0	0	24	spo:	P=0.96	R=0.66667	F1=0.78689
	Test Accuracy=0.824								

Figure 6: Multi-Class Classifier k=25

	bor	die	edu	occ	spo				
bor:	49	0	1	0	0	bor:	P=0.98	R=0.84483	F1=0.90741
die:	0	47	1	0	2	die:	P=0.94	R=0.92157	F1=0.93069
edu:	1	1	48	0	0	edu:	P=0.96	R=0.90566	F1=0.93204
occ:	7	3	3	22	15	occ:	P=0.44	R=1.0	F1=0.61111
spo:	1	0	0	0	49	spo:	P=0.98	R=0.74242	F1=0.84483
	Test Accuracy=0.86								

Figure 7: Multi-Class Classifier k=50

Total Avg. Recall	0.70020585

Figure 8: OpenIE comparison Average Recall k=160 sentences