Efficiently Extracting Relationships From Natural Language

James Jolly
jolly@cs.wisc.edu

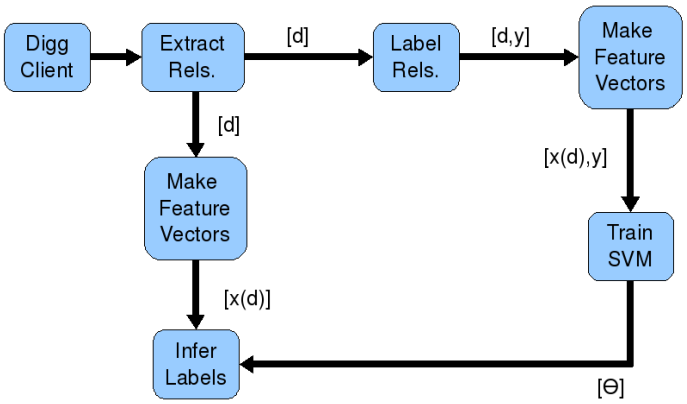
Abstract

Minerva aims to quickly extract relationship tuples of the form ⟨object, relationship, object⟩ from large corpora such that they can later be explored by a user or data-mining program. A custom noun-phrase chunker allows the system to identify candidate relationships. These candidates are then either accepted or rejected by a classifier trained using manually-labeled relationships. Minerva was evaluated using a dataset consisting of headlines taken from recently popular articles on Digg.com.

Introduction

A large body of ⟨object, relationship, object⟩ tuples extracted from a corpus may make it possible to answer certain queries more easily than keyword search. Examples include discovering an unknown relationship between two known objects (i.e. *How are Linux and UNIX related?*), or finding unknown objects related to a known object in a particular way (i.e. *What is Lawrence Lessig a proponent of?*).

Minerva's approach to extraction is similar to that of TextRunner (Etzioni et al. 2008). Both systems share the philosophy of performing 'best-effort' extraction on a large volume of data in a 'single-pass'. Notably, TextRunner 'bootstraps' itself by automatically generating data of reasonable quality to train a classifier to grade candidate relationships. Having been developed in just a few weeks, Minerva simply uses manually-labeled data. Another interesting contrast is Minerva's use of SVM to classify relationships instead of naive Bayes.



Architecture Overview

First, text was downloaded and a list of candidate relationships extracted from it ($[d]$). Then, a user labeled each relationship tuple as good or bad, creating a training set ($[d, y]$). A function ($x()$) was designed to represent a tuple so a SVM's parameters ($[\theta]$) could be trained to recognize good relationships.

Dataset Choice

Headlines from Digg.com were selected to test Minerva's capabilities. These consist of natural language and are associated with stories rated by users (in an effort to surface quality content within Digg). Higher rated stories tended to have descriptions that were easier to parse and learn from. Digg's API allowed Minerva to easily fetch an arbitrarily-sized corpus of posts.

Extraction Approach

First, for each headline, sentence segmentation was performed using NLTK's (Bird 2006) Punkt sentence segmenter (Kiss and Strunk 2006) implementation. Most headlines consisted of a single phrase or sentence, so this was generally not an important step. Afterward, tokenization and POS tagging (using NLTK's *pos_tag*) was performed on the words in each sentence. Next, a custom noun-phrase chunker looked for contiguous blocks of adjectives and nouns that could be an object.

Once these objects were located, the text between any pair of them within a sentence was considered a possible relationship. For example, the following tokenized and tagged sentence...

```
('The','DT'),('FAA','NNP'),('wants','VBZ'),
('a','DT'),('new','JJ'),('traffic','NN'),
('control','NN'),('system','NN')
```

... contains two possible objects...

```
* ('FAA','NNP')
* ('new','JJ'),('traffic','NN'),
  ('control','NN'),('system','NN')
```

... with the potential relationship text...

```
* ('wants','VBZ'),('a','DT')
```

Relationship Tuple Classification

Each candidate relationship tuple was then converted into a feature vector for the mlpy SVM implementation to operate on. Unfortunately, picking a good feature vector representation that can distinguish between genuine relationships and sentence fragments is challenging. After some experimentation, Minerva settled on:

```
1  Does the relationship end with an
   'IN' word? (bool)
2  Does the relationship start with
   a verb? (bool)
3  Does the relationship start with a 'TO'
   word followed by a 'VB' word? (bool)
4  Does the relationship start with
   a stop-word? (bool)
5  What is the ratio of stop-words to
   other words in the objects? (float)
```

Note that the stop-word list was a collection of 659 commonly-used words.

Evaluation

Experiments were performed on 3 weeks of headlines taken from the 'World & Business Container' with 'over 100 diggs', 835 stories in total. 'World & Business' was selected as its headlines tended to be simple and declarative in nature. From this, 1063 candidate relationships were extracted.

Of these, 404 ($\sim$ 38%) were manually marked as good using the labeler. A linear SVM kernel was then trained on these marked tuples using a 10-fold cross-validation scheme in an effort to gauge its prediction accuracy. Its performance was mediocre, attaining $\sim$ 75% accuracy (a mere $\sim$ 21% improvement to the majority classifier).

Challenges

Initially, naive Bayes was used to perform classification, but its performance was worse than SVM, possibly due to lack of training data given the large space of values the feature vector can take on.

One cause of the high incidence of bad relationship tuples is incorrect tagging. From 20 randomly-drawn headlines, it appears $\sim$ 88% of the tokens are tagged correctly. This sounds good, but not all errors are equal. Particularly common in the 12% is mistaking a verb for a noun or vice-versa. Is 'fires' the act of laying someone off, or many things burning independently? These type of mistakes result in bad object delineation, which the system doesn't degrade well under (in terms of accuracy). A better POS tagger could very well make the classifier's job much easier.

Finding useful features that can aid classification is a difficult task. Many boolean features tried initially simply looked for the presence of particular tags in the tuple. Other general features, such as the length of texts within the tuple were also used. It turns out there's little to be learned from these features. It now appears the more useful features may involve the presence of certain tag sequences in certain positions within the relationship tuple. It would be interesting to conduct an automated analysis on labeled relationship tuples in an effort to discover these more complex features that would assist the classification process.

Conclusion

Minerva is a fast but inexact way to extract relationships between objects discussed on Digg. Future work will focus on further improving the quality of its extractions by enhancing its parser, tagger, and feature vector representation. Simple approaches to relationship storage and retrieval will also be addressed.

References

Bird, S. 2006. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, 69–72. Morristown, NJ, USA: Association for Computational Linguistics.

Etzioni, O.; Banko, M.; Soderland, S.; and Weld, D. S. 2008. Open information extraction from the web. *Commun. ACM* 51(12):68–74.

Kiss, T., and Strunk, J. 2006. Unsupervised multilingual sentence boundary detection. *Comput. Linguist.* 32(4):485–525.