Relation Extraction

CSE 628 Assignment 2

Due Date: Apr 14th 2015 at 5:00 pm.

This is a long assignment. Please read carefully and ask questions on Piazza if anything is unclear.

In this assignment you will build a relation extractor to identify institution relation instances from Wikipedia sentences. An institution relation institution(x, y) indicates that a person x studied in the institution y.

Your task is to build a suite of relation extractors as described below and investigate the performance of each set of features. You will be given training and test sentences that are annotated for the relation instances.

Manual Rule Based Extractor

Write a rule-based extractor. Specify **five** regular expression patterns (word-based) to identify "institution" relations. Use the training set to develop your patterns and report the performance on the test set.

e.g., If sentence matches "(.*?)[space]graduate.*[space]in" then output "Yes" for the relation otherwise output "No".

See evaluation section on what to report.

(20 points)

Bonus (5 points):

> Improve the coverage of your patterns by specifying regexes over word + POS tags.

Supervised Classification

As we saw in class, the institution relation extraction can be viewed as a supervised classification task, where the input is an entity pair, and a sentence or a paragraph containing the entities and the output is a Yes/No label to determine if the institution relation holds between the entities or not.

At a high-level the following steps are involved:

1) Convert the training and test instances into feature vectors + the label.
2) Use a binary classification algorithm such as SVM to learn a classifier on the training data.

3) Use the trained classifier to make predictions on the test data.
4) Evaluate the predictions.

In this assignment you will try a few different features for the task:

1) ***Bag-of-words*** – Every word that occurs in the span between the PERSON mention and the INSTITUTION mention is used as a feature in a supervised classification setup. The classifiers typically expect you to assign a unique id to each word in the vocabulary and represent the text via these ids. For example the following instances:

Entity Pair	Sentence	Label
(Bill Gates, Harvard)	Bill Gates **enrolled in** Harvard.	Yes
(Larry Page, Stanford)	Larry Page **was enrolled at** Stanford	Yes

can be represented as:

- 2,1,Yes
- 3,2,4,Yes

with an Id map {enrolled -> 1, in ->2, was->3, at->4}.

Veronica has created a simple script (generate_arff.py) that takes in the training and test files to generate BOW feature vectors for each labeled instance. The feature vectors are in the input format expected by Weka. This feature generator treats each unique word (after lowercasing) as a feature and assigns a unique id.

Use the training feature vectors to train an SVM classifier (see Classifier section below for details) and make predictions on the test. Evaluate the performance of the classifier with these BOW features.

(30 points)

Bonus (5 points):

The tokenization is quite naïve and relies on the python “split” function. Replace this tokenization with NLTK’s Tokenizer to get a cleaner set of Tokens. Evaluate extraction performance with the naïve split tokenizer and NLTK’s tokenizer.

2) ***Clustering Features*** – Using words directly as features often leads to sparsity. A standard technique to overcome sparsity is to use a generalized representation of words via clustering. The idea is to first cluster words based on their usage and represent each word by the cluster to which it belongs.

Brown Clustering is a widely used for clustering words. You can use Percy Liang's code here to generate hierarchical word clusters. The algorithm takes in a text file as input and a cluster parameter (c) that controls the number of clusters. The output includes a path file that contains the cluster ID (which is a path on the hierarchical (binary) clustering).

Follow these steps to create clusters:

1. Download Liang's code and "make" the executables.
2. Extract the sentences in train.tsv and test.tsv and output them to a new file (say sentences.txt).
3. Run the clustering code as follows:

```
./wcluster --text sentences.txt --c 50
```

The output will include a path file. Here is a section of the Brown clusters obtained for this collection:

Cluster ID	Word	Frequency
11110101	teaches	48
11110101	taught	430
11110101	lectured	9
11110101	speaks	14
11110101	read	80
11110101	practised	18
11110101	practiced	49
11110101	**studied**	1394
11110101	**matriculated**	48
11110101	**enrolled**	132
11110101	flew	9

The words that have similar meanings and usages are clustered together. For example, the words shown in blue are all indicative of the INSTITUTION relation. Note that there are also other possibly unrelated words (e.g., teaching) that get the same cluster ID.

Instead of the unique word ID, represent each word using the cluster ID as is (e.g., studied -> 11110101). In the BOW approach every word had a unique ID. With this approach, many words will get the same ID. Evaluate the classifier using this set of features.

(10 points)

Bonus (5 points):

Pick an appropriate prefix length instead of using the full cluster ID. For example, if you choose prefixes of length 5, studied -> 11110).

If you use the full ID then the clustering is fine grained and fewer words would be grouped together. This may reduce the ability to cluster certain types of synonyms. On the other hand, using a short prefix would mean the clustering is coarse grained. This will lead to merging words that may not necessarily be synonyms or related.

You can manually inspect the quality of various prefix lengths and choose one that you think is at the right level of granularity. Show the extractor performance with the full cluster ID and with the prefix length that you chose.

3) ***Dependency Features*** – Mere presence of words that indicate relations isn't adequate.
e.g.,

   Mario was born to Roger, **an alumnus of** Harvard, and to Mary **an alumnus of** Yale.

   Even though alumnus is a strong indicator of the studied in relation, the syntactic structure clearly shows that alumnus is not directly related to Mario.

   Process the sentences using Stanford Dependency parser. Code **three** syntactic dependency-based features that ensure that the relation words are connected to the entity mentions.

   (30 points)

4) ***Kitchen Sink*****:** Combine all features including the manual regular expression patterns.

   (10 points)

Classifier

Use an SVM-based classifier. If you are using Veronica's script then it is convenient to use the implementation in Weka (see here). There are also other implementations including libLinear, libSVM and SVMLight.

I recommend using either libLinear or the implementation in Weka with the linear kernel. The SVM's typically require you to specify a regularization parameter (typically called 'C'). You can set this to any fixed value. The goal of this assignment is for you to see how adding the different feature sets affect performance. The ML magic required to get a high performance is not the focus.

Bonus (5 points):

You can try a few values in the range C = {0.001, 0.01, 0.1, 1, 10, 100} on the training dataset to figure out the value that gives you the best performance.

Evaluation

1. For each group of features report the Precision/Recall and F1 on the institution class.

2. For each group of features report the most egregious errors – i.e., instances where there isn't a relation but the classifier assigns a high score to the relation.

   Show **ONE** example for each feature group and provide an explanation for why the classifier made a mistake on this example. You can inspect the features used to form ideas.

What to turn in:

1. Code – Any code your write. The feature generation code.
2. Report:
   a. **Feature Descriptions:** Include a description of all the features you generated and a line about how you generated them. The description should be clear enough for someone else who reads it to understand and implement that feature.

   b. **Results and Analysis:** Use tables to clearly show the performance of each feature subset.

Feature Group	Precision	Recall	F1
Manual Regex			
BOW			
Brown Clusters			
Dependency			
Kitchen Sink			

Provide plausible reasons for the observed performance of the different methods.