Project Report

Programming exercise :
A machine-learning algorithm for co-reference resolution

Mahbub Ul Alam (3144196) [mahbub.ul.alam.anondo@gmail.com]
Tanzia Haque Tanzi (3144251) [tanzia.haque.tanzi@gmail.com]

Introduction

In this exercise we tried to implement a machine learning classifier based on the methods described in Soon et al. (2001: A Machine Learning Approach to Coreference Resolution of Noun Phrases). We used OntoNotes, CoNLL-2012 data as training and test data. We used Support Vector Machine (SVM) classifier using scikit-learn toolkit in python to predict the output. We chose SVM because Decision Tree classifier has a tendency to become over-fitted easily and SVM is quite effective in binary boundary detection.

The whole data (input, scripts and output) can be downloaded from this link,

https://goo.gl/T0fh5a

Determination of Markables

We chose the following items as the components for each NP phrases in the input data (both test and training data),

1. Words
2. Parts of Speech
3. Name-Entity Values ('null' if not present)
4. Coreference Chain Number ('null' if not present)
5. Sentence Number

Also each begin statement of a file segment is kept for future calculation.

The files in where the markables are stored are named as follows,

1. training_markables
2. test_markables

Each line in the file represents a NP phrases except the file boundary lines. For example a file boundary line,

#begin document (wb/eng/00/eng_0009); part 000

The following line is a noun phrase line (' | ' is used as a column barrier),

Michael Jackson | NNP NNP | PERSON | 149 | 6

Words | Parts of Speech | Name-Entity Values | Co-reference Chain Number | Sentence Number

In the corpus all the name-entity valued word segments were inside the related noun phrase. So we did not check the 'Nested noun phrase extraction' because it has already been done.

As we have considered every noun phrases (denoted as NP) in our code (markable_extraction.py) so we are sure that the precision and recall will be 100%.

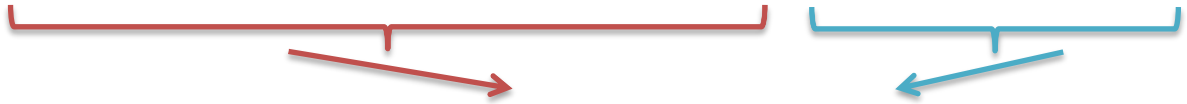
Generating Training Examples and Feature Vectors

List of positive and negative training examples (mention pairs) together with their features were generated. The following files are created,

1. training_data_Positive_chain_lines
2. training_data_Negative_chain_lines
3. training_features

The format of training_data_Positive_chain_lines and training_data_Negative_chain_lines files data is a antecedent part (as in markable format) separated by '|**|' and a anaphor part (as in markable format). For example,

the present capitalist followers | **DT JJ JJ NNS** | **null** | **null** | **1** |**| **the age** | **DT NN** | **null** | **21** | **4**

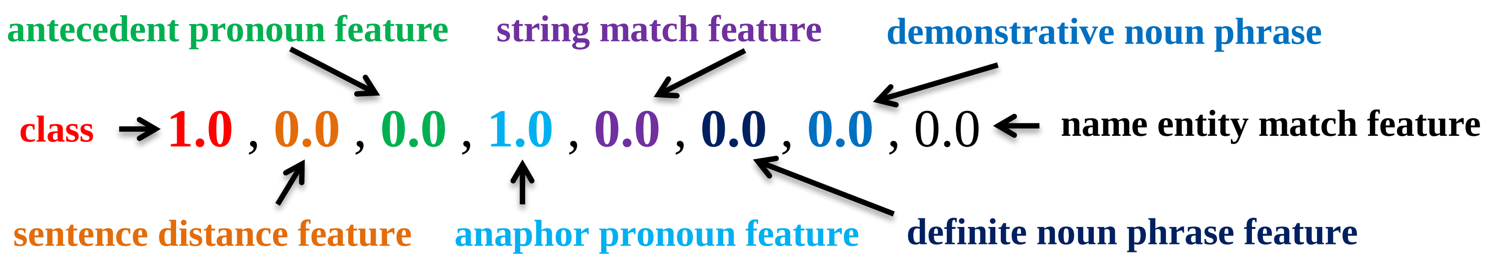


antecedent part||anaphor part**

We have selected the following features,

1. distance of mention i and j in sentences
2. antecedent is a pronoun (personal, possessive or reflexive)
3. anaphor is a pronoun (personal, possessive or reflexive)
4. string match (after removing articles and demonstrative pronouns)
5. anaphor is definite noun phrase
6. anaphor is demonstrative noun phrase
7. Name-entity recognition match

The format of the training_features file is as follows, each line represents the feature vectors of a coreferent pair (positive or negative) separated by ',' . The first column indicate the class. 0.0 means negative pair, 1.0 means positive pair. For all other vectors 1.0 means True and 0.0 means false unless it is feature number one. For example a line in training_features file,


antecedent pronoun feature
string match feature
demonstrative noun phrase
class
1.0 , 0.0 , 0.0 , 1.0 , 0.0 , 0.0 , 0.0 , 0.0
name entity match feature
sentence distance feature
anaphor pronoun feature
definite noun phrase feature

Generating Test Pairs and Feature Vectors

List of test mention pairs together with their features are generated. The files are,

1. test_data_Positive_chain_lines
2. test_data_Negaitive_chain_lines
3. test_features

We have excluded the nested markables while generating the pairs and features. The file formats and feature values are same as mentioned in section 2

Classification

Classification is performed in Support Vector Machine (SVM) classifier using scikit-learn tool-kit in python. List of classified test instances are stored in the following files,

1. test_data_predicted_classes
2. test_data_Positive_chain_lines_with_predicted_classes
3. test_data_Negaitive_chain_lines_with_predicted_classes

In the file number 1, the value of each class prediction (1.0 or 0.0) is stored in each line. Format of the data in file 2 and 3 is as follows, each line contains the pair (antecedent and anaphor) with the predicted value (1.0 or 0.0 as true or false) separated by '|**|'. For example,

I | PRP | null | 7 | 1 || I | PRP | null | 7 | 3 |**| 1.0**

Antecedent line||Anaphor line|**|Predicted Value**

Evaluation

We have evaluated the predicted data using sklearn.metrics module and obtained the following results

- Accuracy = 96.1963771458%
- Precision = 84.0616966581%
- Recall = 28.7671232877%
- F1-Score = 42.8651685393%

The information is stored in the following file,

1. Evaluation

Discussions

Accuracy and precision is quite good but the recall is very bad. So this classifier is returning very few correct results, but most of its predicted labels are correct when compared to the training labels. We presume that it may be due to our feature value selection process. If we can include the ‘number’. ‘gender’ and ‘semantic role’ features then we think the result will be improved vastly. The other classifiers are also need to be checked for comparison of the result; as it takes a huge amount of time so we could not do it.

Individual Contribution

Mahbub Ul Alam –

1. markable_extraction.py
2. feature_extraction.py (starting to before 'feature_extraction_main(..)' function)
3. Documentation

Tanzia Haque Tanzi-

1. feature_extraction.py (from 'feature_extraction_main(..)' function to end)
2. svm-classification.py
3. run_steps.sh

-*-