Midterm project report for the Natural Language Understanding Systems course

Anonymous ACL submission

Abstract

This work provides a concept tagging tool for queries related to movies, such as "who is the director of thor". The final version of the script enables tweaking of various learning parameters to better evaluate which setting is the best for the situation. A number of possible solutions are discussed and provided in the tool itself to show the steps taken to reach the highest scoring version that uses pre-processing of the training data. It is determined that accurate analysis of both the training and test set is required to increase accuracy in specific scenarios.

1 Introduction

What is the best way to estimate the concept expressed by words in a sentence using Weighted Finite State Transducers? An incremental approach is taken into consideration, starting from the simplest of methods (a single transducer), adding elements that could improve performance in some way at each step. Doing so results in three main branches to explore: increase the number of transducers to better include the additional features, increase the complexity of the learned concept model and the manipulation of the test set.

2 Data set analysis

First and foremost, an analysis of the dataset is required to implement any of the possible solutions. The dataset is subdivided into a training set and a test set. These sets are in a word per line format, with the sentences being separated by an empty line. Multiple columns represent the features of the specific word (such as part-of-speech tag and the lemma) and the correct concept tag.

2.1 Zipf's law

The first analysis performed on the training data is to check whether or not Zipf's law is verified. Due to the nature of the data, a bias towards words typical of the movie industry was expected. This can be seen clearly in the word frequency distribution, with "movies" and "movie" occupying respectively the second and fifth spot. (with "movies" appearing more frequently than "of")

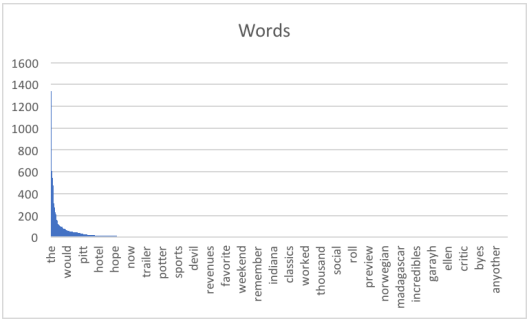

Figure 1: Zipf's law

2.2 Concept distribution

The most frequent concept is "movie.name" by a wide margin, which means this plus out-of-span tags make up 86% of the tags present in the training dataset.

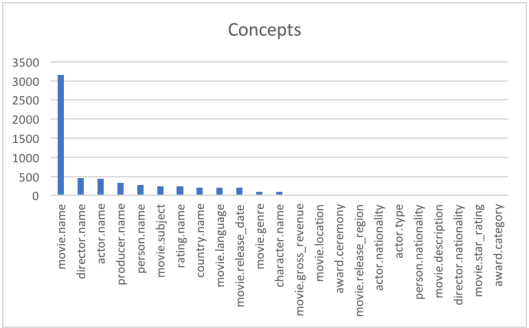

Figure 2: Concept distribution

3 Tools used

To train the language models used in every solution except the baseline, OpenGRM (Gorman, OpenGRM) was used. It provides an implementation of various smoothing methods and supports any n-gram order during training. Every parameter has been tweaked to find the best score for each proposed solution.

In addition, OpenFST (Gorman, OpenFST) was necessary to create the Weighted Finite State Transducers used in all the solutions.

The final tool is mostly written in Python, with Bash scripts that automate the steps required to do a full concept tagging of a test corpus.

4 Baseline solution

The case of a single transducer from word to concept will be taken in consideration as the baseline for all the proposed solutions. In addition, no concept model was learned to improve performance in this case.

```
accuracy:  67.32%
precision:  20.15%
recall:  54.72%
FB1:  29.45
```

The resulting F1-Score is very low as expected.

5 Direct solution

The simplest and most direct method is obtained by simply composing the transducer from the previous step with the concept model trained on the training data. To do this the training data was converted from word-per-row to a concept sentence-per-row format. These sentences have been used to train a wide array of language models, by changing the order of the n-grams and the smoothing method.

It can be observed that using a language model trained on 2-grams achieves better performance than one trained on 3-grams.

```
Smoothing method: witten_bell
Order: 3-grams
Cut-Off: none
accuracy:  92.62%
precision:  76.58%
recall:  74.61%
FB1:  75.58
```

```
Smoothing method: witten_bell
Order: 2-grams
Cut-Off: none
accuracy:  92.68%
precision:  78.51%
recall:  74.34%
FB1:  76.37
```

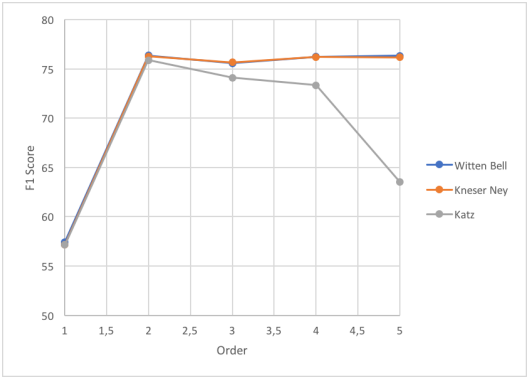

Figure 3: F1 Score related to smoothing method and order

6 Solution using extra features

The next experiment was to try and include the extra features provided by the dataset to improve the F1 Score. At a first glance the chaining of multiple transducers, each passing from one feature to the next, seemed like a good idea. Because of this 3 transducers were built and composed: word to lemma, lemma to part of speech, part of speech to concept tag. By composing the result with the concept model it is shown to be even worse than the baseline:

```
Smoothing method: witten_bell
Order: 2-grams
Cut-Off: none
accuracy:  72.81%
precision:  27.35%
recall:  6.14%
FB1:  10.03
```

An explanation can be found by considering that this method is reducing the input space of the final transducer from the size of the vocabulary to just 38 pos-tags.

Because of this, a second method which keeps the first transducer, while replacing the second and third with a lemma-to-concept transducer was developed. The second transducer takes into consideration the pos-tags in the calculation of the weights.

```
Smoothing method: witten_bell
```

```
Order: 3-grams
Cut-Off: none
accuracy:  92.34%
precision:  76.06%
recall:  73.97%
FB1:  75.00
```

The performance of this method is comparable to the version without extra features.

7 Solution with generalization

As the inclusion of the extra features does not seem to improve performance this method does not use them.

The idea was to create two transducers, one that transforms words into a generalization (a word class), while the second transduces from a word class to a concept. The method was implemented by pre-processing the training set, adding a "word class" column created as a derivative of the concept tag. Because of this, including the fact that the only generalization that provides a decent score does nothing more than remove the IOB part of the tags, the performance is lower than the standard method:

```
Smoothing method: witten_bell
Order: 3-grams
Cut-Off: none
accuracy:  88.70%
precision:  61.42%
recall:  61.14%
FB1:  61.28
```

8 Using frequency cut-off

Frequency cut-off can be applied to all the previous solutions, but it does nothing to improve the performance. In fact the *no extra features* concept tagger achieves slightly lower scores when used with a cut-off of 1 (removing words that appear only 1 time in the training corpus):

```
Smoothing method: witten_bell
Order: 2-grams
Cut-Off: 1
accuracy:  91.74%
precision:  77.95%
recall:  70.30%
FB1:  73.93
```

Almost 2 points lower than the same solution used without cut-off. The cut-off does remove a lot of the words that are written incorrectly such as "accumalate" and specific movie related terms such as "Dory", which may be desirable when dealing with larger amounts of data.

9 Pre-processing the training set

Having the `O` Concept tag assigned to most of the words in the training corpus meant that the concept model would learn that the `O` tag is very likely in any part of the sentence. To try and solve this problem, a pre-processing of the training set is done by replacing every `O` tag into a `O-<word>` tag, effectively increasing the total number of concept tags by the number of words. By doing so, the concept model yields more information regarding the other tags, which are lower in number, and it is possible to achieve a significant improvement on the F1-Score. The following scores have been obtained by using the *no extra features* solution described precedently with the pre-processed training corpus:

```
Smoothing method: witten_bell
Order: 3-grams
Cut-Off: none
accuracy:  93.72%
precision:  80.04%
recall:  82.68%
FB1:  81.33
```

It is interesting to note that using 2-grams with the same smoothing method degrades performance (as opposed to the solution without the pre-processing).

Cut-Off deteriorates performance, so it is still kept at 0.

```
Smoothing method: witten_bell
Order: 2-grams
Cut-Off: none
accuracy:  93.13%
precision:  78.35%
recall:  80.29%
FB1:  79.31
```

The best possible score is obtained by increasing the order of the n-grams at learning time of the concept model(and thus increasing the model's complexity):

```
Smoothing method: kneser_ney
Order: 9-grams
Cut-Off: none
accuracy:  94.39%
precision:  82.38%
recall:  83.13%
FB1:  82.76
```

10 Conclusions

The value to be used for the learning parameters depends on the distribution of the data we are using for the learning itself. Just increasing the order of the n-grams used to learn the language model can hurt performance instead of improving it. An example of this is the performance achieved using 2-grams and the witten bell smoothing method, which is higher than the same method used with 3-grams.x

Extra features don't improve the concept tagger's score as we would need those features to be available at test time to improve performance. Having only the words themselves at test time means that every single extra feature we want to use to guess the concept tag would have to be estimated from the word itself, increasing the possibility of error.

Minimum frequency cut-off seems to degrade performance for every proposed solution, even though it removes words spelled wrong and some words specific to certain movies.

Finally, the biggest improvement to performance was achieved simply by manipulating the training data in a very specific manner. The analysis of the data used for training and testing is essential to reach the desired accuracy. A small modification to the dataset allowed an F1 Score of `82,76`, without adding any extra features or transducers.

References

Kyle Gorman. OpenFST. Openfst library. http://www.openfst.org/twiki/bin/view/FST/WebHome. Visited 21/04/2017.

Kyle Gorman. OpenGRM. Opengrm libraries. http://www.opengrm.org/. Visited 21/04/2017.