Name Entity Recognition
Project Report
CS263 Data And Knowledge Bases

by
Magnus Settemsli Mogstad and John-Olav Storvold

December 2014

Contents

List of Tables

1 Abstract

For the project in CS263 - Data and Knowledge Bases we chose to look closer at a Name Entity Recognition (NER) system. We chose this because it appealed as interesting project and it would teach us more about machine learning techniques
The meaning of a name entity regognition system is to classify words into different entities. In this project we chose to support and classify the entities; Organization, Location, Time, Person and Other. To solve this problem we approached the problem using The Viterbi Algorithm, which uses a hidden markov model to find the given entity probabilities based on the previous entities in the sentence. With the name enitity recognition system we aim to answer which kind of entity each word is and give statistics of how accurate our implementation is once we are finished.

2 Introduction

At the beginning of the project we looked into making the Name Entity Recognition system using a sequence model after reading that this was the approach Stanford used in their NER system. However, after more in-depth reading on the subject we quickly realized that there was not many research papers out there which gave a good idea how to go about it as well as give a fairly good description on how to implement the actual parts of the system. After some consideration and searching into other ways to do name entity recognition we found two good sources on the Viterbi Algorithm and decided to implement it in java. The Viterbi Algorithm uses hidden markov model to find the highest probability of a word being a certain entity given the previous entities. The algorithm can be used in a scope of a whole document. However, we decided to set the scope on the individual sentences instead.

3 Background

To implement The Viterbi Algorithm, we had to look into Hidden Markov Model as none of us have had any significance background implemention systems which uses these models into practise. After some time we got eased into how they were made up and we began the project by implementing start probability model, emission probability model and a transition matrix model which probabilities are based on the training data set. By using these probabilities we can use the hidden markov model to find the highest probability of a word being any of the five entities we aim to classify. Our goal is to get the algorithm working before we start to find or make training sets which is of decent size, this is because we think the hardest part of the project is to make the hidden markov model work. To check if the algorithm is working we will only be using a small training set which we have created, the downsize of using a small training data set is that you will get terrible results. It is important to remember to have training data

sets be built up by sentences because it will help the algorithm to classify based on the "flow" in the entities. Our hope is to get a accuracy of around 80% for our classifications made by our name entity recognition system.

4 Body of report

At the start of the project we looked closer at doing a sequence model name entity recognition system, but after reading some papers and turning up empty in the search for an algorithm to follow. Then we chose to chase down the hidden markov model approach to name entity recognition, after reading paper reading we ended up using The Viterbi Algorithm.

4.1 Training Data Set

To begin with we agreed on a way how a file is structured to read words and their classification. The java class implementation of this was used to read from the training data set(s). The training data sets needed to be segmented into word sequences because only real sentences will be the only way to effectivly improve our prediction when we want to classify more advanced texts such as articles and blog posts. We went ahead with a word and classification seperated with space between them. This means that the data in the text files can be written in two different ways. Either you can have one word and its entity in one line, or you can have multiple words and their entity in one line see figure 1. Our training data set looks like the dataset you see on the right hand side of the figure, it also contains 2518 classified words which we have classified manually. This means that some words in the training data may have one or more entity like New which is a location when it is in context with New York or it may be other when it is just by itself when no particular context is given.

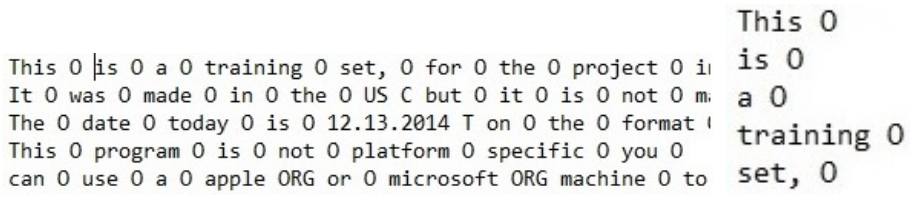

Figure 1: Two ways to write training data set

We have made a enum java class which holds the different entities the program has to classify, and in the file which reads in the training data set we use both hashmap and arraylists to keep track of the words and their entity. The reasons for making so many lists is that they make it easier to calculate the different probabilities used by the hidden markov model even if they hold redundant information as well as ease the programming for ourselves.

4.2 Viterbi

When the reading of the training data set is done and all the data is in hashmaps or lists, we started with The Viterbi Algorithm. This algorithm uses the start probability, emission probability matrix and transmission probability matrix in the Hidden Markov Model to find the best probability for a words entity.

4.2.1 Guava

Guava is a project by Google and contains some extended datastructures built on the core of the java language as well as some of the java core rewritten. For our project we are using the collections part in guava created by Google. This makes it easier to store the different values in the emission and transmission probability matrix. The data structure we used is called Table and value3 is given by the key values in value1 and value2.

```
Table<value1, value2, value3> table = HashBasedTable.create();
table.row(value1); //returns a mapping of the different value2 to value3
table.column(value2); //returns a mapping of the different value1 to value3
table.get(value1,value2); //returns value3
```

We found this library most useful because we needed a good way to store the probability model which is decided by two key variables, and it is used in both the emission, transmission probability matrix and as well as the hidden markov model to store the updated values for each iteration step the algorithm makes.

4.2.2 Start Probability

To find the start probability we need to find the entity of the starting word in each sentence. Then divide this entities number by the total of sentences.

$$\text{Start probability} = \frac{\text{number of start words with entity}}{\text{Toal number of sentences}} \tag{1}$$

This gives us the start probability will look somthing like this
Entities = {Other, Organization, Person, Time, Location}

Other	Organization	Person	Time	Location
$\frac{6}{12}$	$\frac{1}{12}$	$\frac{2}{12}$	$\frac{1}{12}$	$\frac{2}{12}$

Table 1: Start probability

4.2.3 Emission Probability Matrix

The emission probability matrix is the probability of a specific word being a specific entity. If you look at table 2 and the word Obama has the probability of $\frac{1}{15}$ for person and zero for every other entity. This tells us that the word Obama is most likely a person.

$$\text{Emission probability} = \frac{\text{number of specific word with entity}}{\text{Number of words with that entity}} \tag{2}$$

This means that the emission probability matrix will look like this only much bigger because you have to calculate the probability for each distinct word. To store The Emission Probability Matrix we use the table data structure from Google's guava library.

	Hi	This	Is	Obama	From	France	...
Other	$\frac{1}{15}$	$\frac{2}{15}$	$\frac{1}{15}$	$\frac{0}{15}$	$\frac{1}{15}$	$\frac{0}{15}$	$\frac{2}{15}$
Organization	$\frac{0}{15}$	$\frac{0}{15}$	$\frac{0}{15}$	$\frac{0}{15}$	$\frac{0}{15}$	$\frac{0}{15}$	$\frac{0}{15}$
Person	$\frac{0}{15}$	$\frac{0}{15}$	$\frac{0}{15}$	$\frac{1}{15}$	$\frac{0}{15}$	$\frac{0}{15}$	$\frac{0}{15}$
Time	$\frac{0}{15}$	$\frac{0}{15}$	$\frac{0}{15}$	$\frac{0}{15}$	$\frac{0}{15}$	$\frac{0}{15}$	$\frac{0}{15}$
Location	$\frac{0}{15}$	$\frac{0}{15}$	$\frac{0}{15}$	$\frac{0}{15}$	$\frac{0}{15}$	$\frac{1}{15}$	$\frac{0}{15}$

Table 2: Emission probability

4.2.4 Transmission Probability Matrix

The Transmission Probability Matrix is the probability that a word is followed by a word with a given entity. If you have a entity Other then the probability tells you that the next word is most likely a word with the entity Other(look at table 3).

$$\text{Transmission probability} = \frac{\text{Number of words from first entity to second entity}}{\text{Total number of first entity in text}} \tag{3}$$

The transmission probability matrix is in this case 5x5(number of entities*number of entities), which is because it shows the probability of the next word in a sentence given the current words entity. To make the Transmission Probability Matrix we use the table data structure from Google's guava library.

	Other	Organization	Person	Time	Location
Other	$\frac{11}{15}$	$\frac{1}{15}$	$\frac{1}{15}$	$\frac{1}{15}$	$\frac{1}{15}$
Orgainization	$\frac{1}{2}$	$\frac{0}{2}$	$\frac{0}{2}$	$\frac{0}{2}$	$\frac{0}{2}$
Person	$\frac{1}{1}$	$\frac{0}{1}$	$\frac{0}{1}$	$\frac{0}{1}$	$\frac{0}{1}$
Time	$\frac{2}{2}$	$\frac{0}{2}$	$\frac{0}{2}$	$\frac{0}{2}$	$\frac{0}{2}$
Location	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{0}{2}$	$\frac{0}{2}$	$\frac{0}{2}$

Table 3: Transmission Probability

4.2.5 Hidden Markov Model

We started by looking into the Baum-Welch Algorithm, but did not quite understand how we could implement it, so we went ahead with The Viterbi Algorithm to tackle the Hidden Markov Model problem. To calculate the probability for each words entity in a sentence we are using a Hidden Markov Model. The Hidden Markov Model uses the Start Probability, Emission Probability and the Transmission Probability as input to calculate the entities for each word in a sentence. Since the training data set also contains sentences from the real world which means that the algorithm is trained to learn how a sentence is built up by specific words and entity sequences. Since our implementation of the hidden markov model identifies entities based on sentences the first word in every sentence is a special case. Note that none of the words probabilities can be zero as it needs to be 0.0000000001, because we use the logarithm of 2 to transfer stacking probabilies from a multiply operation into a simply add operation. Logarithm of zero gives us a negative infinity number which gives problems in certain parts of the code. Then transfer it back once we are done with all the various operations. We also presurve better decimal accuracy by doing it this way.

$$P_{\text{entity}}(word, 1) = \log_2(startProbFor(Entity)) - \log_2(emissionprobability(Word)) \quad (4)$$

The words entity is given by which entity that has the highest probability by the hidden markov model or the entity which has the closest number to zero in our case. To get the entities for the rest of the words in the sentence, we need to use the earlier words in the sentence entities. Based on the previously entities we need to find the entity path with the highest overall probability, to get the next words entity you use this equation:

$$\begin{aligned} P_{\text{entity}}(word, n) = P_{\text{emission}}(Word, entity) + \\ max(P_{\text{entity}}(word, n-1) + P_{\text{transmission}}(entity, sameEntity), \\ P_{\text{otherEntity}}(word, n-1) + P_{\text{transmission}}(entity, otherEntity)) \end{aligned} \quad (5)$$

In our case the max function will have many more values, we will also have to calculate this probability for every entity we are trying to classify. The table 4

	Hi	Apple	Santa Barbara	Project
Other	**-1.2**	-2.567	-3.455	**-1.33**
Organization	-2.34	**-1.98**	-3.45	-3.44
Location	-2.33	-4.55	**2.33**	-4.55
Person	-3.21	-3.42	-3.34	-7.55
Time	-4.33	-4.22	-4.12	-8.34

Table 4: Table of words entity with Hidden Markov Model

shows how it chooses which entity each word should have the entity with the highest probability is bold to show how the entity path is through a sentence.

4.3 Result

Our result is a Name Entity Recognition system which identifies the different words entity based on previously words entity the sentence, by hidden markov model which bases the decisions on the start, emission and transmission probability.
By using regex expressions we have tried to help the system identify the correct entity for the different words, but only so when the viterbi algorithm gives a word other as classification. Because when the algorithm gives the entity other it is because it is the entity which most frequently occur in texts. Hopefully the regex expressions we have made can help us to identify more words with the right entity. We have made regex expressions to help us identify time, persons and Location entities.

4.4 Statistics

The statistics show both the error and correctness rate of our implementation of the viterbi algorithm.

$$\text{correct rate} = \frac{\text{number of correct}}{\text{total number of words}} \tag{6}$$

$$\text{error rate} = \frac{\text{number of wrong entities}}{\text{total number of words}} = 1 - \text{Correct rate} \tag{7}$$

	Article 1	Article 2	Article 3	Article 4
Correct rate	0.841	0.902	0.846	0.880
Error rate	0.159	0.098	0.154	0.120

Table 5: Correct and Error rate from first run

By looking at the correct and error rate we can see that all of our articles got a better result than we hoped for when we started. We consider us happy with how the system runs and gives out results.

5 Evaluation

We both strongly agree with each other that we should have made the choice to either ask for help by the teaching assistants or to make the switch from the sequence modeling approach earlier in the project, because we found that the time frame was a bit short when we got started working with the project so late. Due to the late start we did not have much time to tweak the algorithm and program for better performance. We could also probably have made a cleaner code but since we experienced trouble with the time we prioritized to get the system up and running and working as it should, rather than making the code cleaner and optimizing it, but overall we are happy with our project since all of our classifications are higher then we hoped when we started doing the project, and we both belive we have learned a lot by doing this project.

6 Conclusion

We have learned a lot about how to use a Hidden Markov Model to help us find highest probability based on earlier observations. We have also learned that you should ask for help if you are stuck with a problem and can not get started with a project within reasonable time as it only leads to long nights before the due date. The program can be improved more by getting more training data set(s) from various sources and to make the training data set larger and make some more models which is run on the text to classify. Maybe use cross checking between the classification results if you classify by using more than one model and find the most likely entity based on the classification results. You can also improve the program by improving on the some of our regex expressions.

7 Bibliography

References

[1] Stanford Named Entity Recognizer
`http://nlp.stanford.edu/software/CRF-NER.shtml`

[2] Sudha Morwal, Nusrat Jahan and Deepti Chopra *Named Entity Recognition using Hidden markov model (HMM)* International Journal on Natural Language Computing (IJNLC) Vol. 1, No.4, December 2012
`http://airccse.org/journal/ijnlc/papers/1412ijnlc02.pdf`

[3] Guava from google
`https://code.google.com/p/guava-libraries/`

[4] Baum-Welch algorithm
`http://en.wikipedia.org/wiki/Baum-Welch_algorithm`

[5] Jie Liu *Chinese named entity recognition algorithm based on the improved hidden markov model* School of Mathematics and Computer Science, Shaanxi University of Technology, Hanzhong, Shaanxi, China
`http://jocpr.com/vol6-iss7-2014/JCPR-2014-6-7-1474-1478.pdf`

[6] Dan Jurafsky, Christopher Manning *Stanford Natural Language Processing*
`https://class.coursera.org/nlp/lecture/59`

[7] HMM : Viterbi algorithm - a toy example
`http://homepages.ulb.ac.be/ dgonze/TEACHING/viterbi.pdf`

8 Program Listings

To run this name entity recognition system you will need a IDE like eclipse, and you will also need to import the libraries guava-18.0.jar and jsoup-1.8.1.jar(you will get these jar files when you download the project from github https://github.com/johnolos/classify or when you receive it).
To use the imported libraries follow these steps(in eclipse):

1. Import project to IDE
2. Right click on project in package explorer
3. Click on build path → Configure build path
4. Then click button called import external jar(s)
5. Go to folder with project
6. Click on jars called guava-18.0.jar and jsoup-1.8.1.jar and click add.
7. Program should now be working.

9 Appendix

```
Run classification on test document: 1
Statistics for classification of file: testDocs/testArticle.txt
Correct Rate: 0.8415492957746479
Error Rate: 0.15845070422535212
Run classification on test document: 2
Statistics for classification of file: testDocs/testArticle2.txt
Correct Rate: 0.9027027027027027
Error Rate: 0.0972972972972973
Run classification on test document: 3
Statistics for classification of file: testDocs/testArticle3.txt
Correct Rate: 0.8464285714285714
Error Rate: 0.15357142857142858
Run classification on test document: 4
Statistics for classification of file: testDocs/testArticle4.txt
Correct Rate: 0.8801571709233792
Error Rate: 0.11984282907662082
```

Figure 2: Print Screen from first run

```
Run classification on test document: 1
Statistics for classification of file: testDocs/testArticle.txt
Correct Rate: 0.8415492957746479
Error Rate: 0.15845070422535212
Run classification on test document: 2
Statistics for classification of file: testDocs/testArticle2.txt
Correct Rate: 0.9027027027027027
Error Rate: 0.0972972972972973
Run classification on test document: 3
Statistics for classification of file: testDocs/testArticle3.txt
Correct Rate: 0.8464285714285714
Error Rate: 0.15357142857142858
Run classification on test document: 4
Statistics for classification of file: testDocs/testArticle4.txt
Correct Rate: 0.8801571709233792
Error Rate: 0.11984282907662082
```

Figure 3: Print screen from second run

As you can see from figure 2 and 3 the results are the same in both runs. Project rapport was made using shareLatex