Natural Language Models

Floris de Bruin - 5772583
Jasper van Eck - 6228194
Gijs van Horn - 10070370

May 17, 2013

Abstract

The application of language and distortion (or task) models, as suggested by Shannon's noisy channel model, on different tasks involving natural language processing, is demonstrated here. These models are used to calculate probabilities for words given their precedents, whole sentences, and PoS-tags. An outline of how these models are built, smoothed and applied, is used to describe problems and how possible solutions, that arise in the field of natural language processing and specifically when using probability based methods. The accuracy of the resulting system is high but can be improved.

1 Introduction

A desciption of how we built a system for natural language processing will form the guideline of this paper. The main goal of this system is to predict (or calculate a probability for) a word, sentence or PoS-tag. This is achieved using large training corpora, which are analyzed using n-grams, n-grams being n-long sequences of words, and how many times they occur in the training corpus. These counts are used to calculate the relative frequencies of words and their precedent. One of many problems which arose is that not all words occur in the training corpus (and certainly not all combinations), smoothing techniques are applied to handle these 0-counts. The discussed smoothing techniques are add-λ (add-one in our case) and Good-Turing smoothing. Finally a method for adding PoS-Tags to sentences is described and evaluated.

1.1 Calculating the probability of a word given its precedents

Given a sentence $\{w_1, w_2, w_3\}$, what is the probability that w_3 occurs given that w_1 and w_2 precede it? Or generally speaking, what is $P(w_n|w_{n-k}, ..w_{n-1})$? Where k is the amount of preceding words taken into account. This probability is applicable in for example, a spelling correction system. When a wrongly spelled word, meaning no match is found in a dictionary, is recognized, several possible corrections are suggested,using methods to correct the most made typing mistakes: deletion, insertion, substitution and transposition, and then these corrections are sorted based on their probability given the previous words. Another application would be in the word suggestion found on most mobile phones, a technique that tries to predict the next word the user wants to type based on their history where the training corpus would be all previously entered text by the user. Further information about the implementation of the calculation is given in section 2.1

1.2 Smoothing the n-gram counts

A first look at smoothing techniques suggests using add-one smoothing, also called naive smoothing. This method increments all seen n-grams counts by one and increments the total amount possibilities by the amount of seen n-grams. Thus transforming the original formula for relative frequency:

$$P_{rf}(e) = \frac{C(e)}{N}$$

into

$$P_\lambda(e) = \frac{C(e) + \lambda}{N + V}$$

where e is an n-gram, $C(e)$ is its count, N is the total amount of possible n-grams given the vocabulary (all $w \times w$ combinations) and V is the total amount of n-grams already seen in training. This method increments all n-grams with the same amount which is undesirable because increasing the count for n-grams that occur often seems unnessesary and increasing all n-grams with count 1 to 2 is very counter-intuitive, they have a very low count for a reason. As an improvement, the Good-Turing method is presented. This method estimates the amount of unseen events (n-grams) based on the amount of event with count one. This mass is then taken from the known events based on their counts. All this is to ensure that the probability for all events (including the zero counts) is 1. More details about the implementation in section 2.2.
Smoothing

1.3 Part Of Speech Tagging

With Part of Speech tagging, the lexical and task models come into play. They will provide a new way to make predictions on which words will occur, based on precedents. With Part of Speech(PoS) tagging, we assign a PoS tag to each word. Examples of PoS tags are: determiners(DT), proper noun(PN) and verb(vb). The two models serve different functions, when determining the probabilities for sentences. The lexical model determines which PoS tag will chosen next, based on the preceding tags. In a way, it is the same as we calculated a probability before. The task determines which tag, has the highest probability of occuring for a certain word. With these two models, we can calculate the probabilities of given sentences, by first determining the most likely PoS tags for the words in the sentence, and then calculating the probability of those PoS tags succeeding eachother.

2 Outline of the System

The implemented system consists of 7 java classes, a Main class which calls ProbabilityCalculator, which in turn starts the training process by calling the Smoothing class with the training corpus. The Smoothing class instantiates two NGrams, differing in their n-size by one.

The system can be compiled and run with:

```
javac *.java
java ProbabilityCalculator
```

After completion, three files are created: precision.txt, recall.txt and resultsOfTagger.txt. These contain respectively the precision of the tagger, recall of the tagger and the sentences that have been tagged with both the original tag sequence and new tag sequence.

2.1 Calculating probabilities

The probabilities of PoS tag n-grams, are calculated after Good-Turing smoothing has been applied on not only the counts of the n-grams, but also on the task model counts. This way, whenever there is an unknown occurence, the probability of a sentence, will not end up being zero. The probabilities, or relative frequencies(r.f), of the n-grams are calculated, by dividing the count of a n-gram by the count of the n-1-gram.

$$r.f = \frac{C(t_1, t_2, t_3)}{C(t_1, t_2)}$$

Above is shown what the calculation looks like. t_1 through t_3 are tags. $C()$ is the count of that sequence of tags.

For determining the r.f. of the task model, we need to calculate:

$$P(w_i|t_i) = \frac{C_t(w)}{C_t}$$

Here w_i is the i^t*hwordinasentence*, *and*t$_i$ is the i^t*htaginthesentence*.C$_t$() is the total amount of times a tag t occurs. $C_t(w)$ is the total amount of times occurs with tag t.

These two formulas, allow us to calculate the probabilities of our task and lexical models.

2.2 Smoothing

As mentioned above, we have implemented Good-Turing smoothing on both the lexical and the task model. This is done, to prevent any zero probability from happening, but in return, it does assign very low probabilities to the non-occuring.

$$r^* = (r+1)\frac{N_{r+1}}{N_r}$$

Above the formula for Good-Turing smoothing is displayed. r^* is the new adjusted count for events originally occuring r times. N_r is the frequency of frequency r. This formula works for both the lexical model and task model.

2.3 Viterbi

To actually calculate the probability of sentence, we need to utilise the Viterbi algorithm. With the Viterbi algorithm, the most probable PoS tag sequence is determined, and the probability of that sequence with it. It does this, by maximizing it's probability over the new tag, given the preceding tags. The new tag is determined by the word in the sentence. We use different tags linked to the word, as try outs for the next tag in the sequence.

$$\gamma_t(state_i) = P(w_t|state_i) \times max_{state_j}(P(state_i|state_j) \times \gamma_{t1}(state_j)$$

$\gamma_t(state_i)$ is the current Viterbi function at $state_i$. $P(w_t|state_i)$ is the probability for word at position t given $state_i$. This is multiplied with the maximized probability of $P(state_i|state_j)$. And $_{t-1}(state_j)$ is the probability of the previous state.

3 Results

In figure 1 the precision and recall values, outputted by the program, are displayed. These are the results when tagging only sentences that are 15 words long or shorter. Tests show that the tagger works almost as well on all the other sentences but results have been omitted because the excercise did not demand these. These values vary greatly between different tags, e.g. precision for LS is 100% whereas the precision for PDT is only 6.6%. The high precision and recall for LS is due to the fact that LS is a list item marker, which is always a number. The same explanation for the high scores for EX and UH can be given, they stand out so much in the training and test set that they are always evaluated properly. For a list of POS-tags and their meaning, see appendix A.

4 Discussion

The results show us, that smoothed PoS tagging, performs quite well. Automatic sentence prediction, with just PoS tagging, is not the best solution, but it performs very reasonable. Certainly when taking in consideration, that doing it manually takes a quite a bit of time.

Precision for certain tags could possibly be improved by using the morphology of the words. For example, our precision for NNPS (proper noun plural) is only 50%, probably due to the fact that they words tagged as NNP (proper noun singular) if they are unknown. Using the morphology of the unknown word, the fact that it ends with an s, might give some advantage, and provide better tagging precision.

Furthermore, we believe, that smoothed PoS tagging, and even our system, could be used for several applications, such as spelling correction, and predicting the next word, one might want to type on their mobile phone. Due to the relatively good results we obtained, the system could perform reasonably well on both applications.

Eventhough, our system would not be able to correct individually misspelled words. It would be able to tell a user, that the used word does not fit in that sentence structure. This would be rather valueable,

Precision Values per Tag

```
JJ --- 97.4155069582505%
RB --- 97.9746835443038%
DT --- 99.61538461538461%
TO --- 94.14893617021278%
RP --- 34.78260869565217%
RBR --- 42.30769230769231%
RBS --- 33.33333333333333%
LS --- 100.0%
JJS --- 95.23809523809523%
FW --- 16.666666666666664%
JJR --- 65.95744680851064%
NN --- 99.8086124401914%
NNPS --- 50.0%
VBN --- 93.37349397590361%
VB --- 95.47169811320755%
PDT --- 6.666666666666667%
VBP --- 93.45238095238095%
PRP --- 95.09433962264151%
MD --- 98.9795918367347%
WDT --- 27.27272727272727%
VBZ --- 98.9010989010989%
WP --- 45.45454545454545%
IN --- 99.58620689655172%
EX --- 100.0%
VBG --- 96.93877551020408%
POS --- 95.55555555555556%
VBD --- 98.33333333333333%
UH --- 100.0%
NNS --- 99.58419958419958%
CC --- 96.53179190751445%
NNP --- 99.54476479514416%
CD --- 93.7037037037037%
WRB --- 100.0%
```

Recall Values per Tag

```
JJ --- 83.47529812606473%
RB --- 83.58531317494601%
DT --- 96.40198511166254%
TO --- 94.14893617021278%
RP --- 32.0%
RBR --- 47.82608695652174%
RBS --- 33.33333333333333%
LS --- 100.0%
JJS --- 86.95652173913044%
FW --- 40.0%
JJR --- 68.88888888888889%
NN --- 88.16568047337277%
NNPS --- 33.33333333333333%
VBN --- 87.57062146892656%
VB --- 89.08450704225352%
PDT --- 25.0%
VBP --- 84.86486486486487%
PRP --- 98.4375%
MD --- 88.9908256880734%
WDT --- 27.27272727272727%
VBZ --- 92.15017064846417%
WP --- 71.42857142857143%
IN --- 95.5026455026455%
EX --- 100.0%
VBG --- 81.19658119658119%
POS --- 82.6923076923077%
VBD --- 86.76470588235294%
UH --- 100.0%
NNS --- 87.40875912408758%
CC --- 92.77777777777779%
NNP --- 80.5896805896806%
CD --- 82.95081967213115%
WRB --- 100.0%
```

Figure 1: The precision and recall percentages for each tag, as outputted by the program

when writing in a language one does not master fully. As it allows the user to switch to a word with the correct semantic meaning.

In addition, since we used the Wall Street Journal corpus, the vocabulary does not include colloquialisms. This will have a positive effect on spelling correction for the user, when writing a formal document.

On the other hand, when a user wants to use vernacular terms in their writing, the system will not be able to properly tag those words, as they do not occur in our corpus. This problem goes for both possible applications, though the mobile application will probably be more heavily influenced, as people are more likely to use colloquialisms, when writing on their mobile machines.

Concluding, our system has quite a good performance, for just smoothed PoS tagging. But due to corpus limitations, it might not be the suitor for spelling correction and word prediction. With the addition of colloquialisms to the corpus, it might suit its tasks better.

Appendix A: List of POS-tags

CC - Coordinating conjunction
CD - Cardinal number
DT - Determiner
EX - Existential there
FW - Foreign word
IN - Preposition or subordinating conjunction
JJ - Adjective
JJR - Adjective, comparative
JJS - Adjective, superlative
LS - List item marker
MD - Modal
NN - Noun, singular or mass
NNS - Noun, plural
NNP - Proper noun, singular
NNPS - Proper noun, plural
PDT - Predeterminer
POS - Possessive ending
PRP - Personal pronoun
RB - Adverb
RBR - Adverb, comparative
RBS - Adverb, superlative
RP - Particle
SYM - Symbol
TO - to
UH - Interjection
VB - Verb, base form
VBD - Verb, past tense
VBG - Verb, gerund or present participle
VBN - Verb, past participle
VBP - Verb, non-3rd person singular present
VBZ - Verb, 3rd person singular present
WDT - Wh-determiner
WP - Wh-pronoun
WRB - Wh-adverb

Source: `http://bulba.sdsu.edu/jeanette/thesis/PennTags.html`