A writing assistant using language models derived from the Web

Sven Karlsson
Department of Computer Science
Lund University
sk@svenk.nu

Abstract

The web is the largest amount of text ever available to man, and the search engines has classified and counted words on a large portion of it. This give us access to huge copra in a number of languages. This article tries to se how the different measurement methods behave and what possible use they might have. The measurements used in this article are uni/bigrams, bi/trigrams, mutual information (Church & Hanks 1990), T-scores (Church & Mercer 1993), and log likelihood (Dunning 1993). A lot of work has been done to make a visual display that makes it easy to compare the differ measurements

1 Credits

This project is a part of the course DAT171 Language Processing and Computational Linguistics at Lund University. The idea of using the web as a source for statistics came from senior lecturer Pierre Nugues.

2 Introduction

Language was first grammar came second. The set that is described by grammar is not the same set as the modern language. These are the key points why I think statistic analyzing and the brute force approach has been so successful. Though I could not fully comprehend Wittgenstein's language theory I sympathizes with the basic idea that language is a game governed by a set of commonly agreed set of rules. The rules change over time and between groups. Think of football, American football, rugby, Wales football and football originate from the same set of rules and they have their likeness but being good at one of the those not mean that you are good at all of them. Now that we have access to huge amounts of corpora we need find ways to use it. We have looked at uni/bigrams, bi/trigrams, mutual information, T-scores, and log likelihood as measurements for word collocation. The Google API didn't give us any possibility to use a window for the words so had to make do with regular bi & trigrams. There are of course other statistical test we could have used such as t-scores and $\chi 2$. The criticism about t-scores has been that is assume normal distribution and is therefore not suitable for limited copra, so would been a candidate but time did not allow that.

3 Background

Being dyslectic, I produce a tremendous amount of errors and I haven't found a speller that can find them all. My dream is to make a speller that would catch more errors and help you before you make them. Today I usually let someone proofread what I write, but there are four problems with this:

1. I'm running out of friends because it can be quite time consuming.
2. The proofreader needs domain specific knowledge.
3. Some persons are better proofreaders than others.

4. I need to understand and be able to work together with proofreader.

So if I hand out this paper to ten people and ask them and to proofread and correct it for me we would probably end up with ten different documents. Every person has is view of language should look like and all of them would or could be grammatically correct. Grammar alone we will not catch awkward expirations or faulty implications, so we I need tools that can handle this.

4 Practical uses

I have mentioned spellers before. I think it would work great as a complement finding words that have been misspelled in a way that they are correctly spelled words but not the intended word. I also think it will get the suggestions for the replacement word up to a level of a human user. It can also pick up problems with sentencing that is missed by the normal grammar checker, but it won't be able to give any clues to what is wrong in this case. In Swedish, there is a special problem with split compounds that probably also could be solved. There are more areas than spelling and proofreading that will benefit from this. To simplify languages is a big market for those who write manuals and technical documents or other literature where it is important that the reader can understand. It does also make things easier when you want to translate a text, especially if it will be machine translated. It could also help people who write in a non-native language. Here I think we can go a step further and create a tool that will be able to suggest word and actually help the writer in an early stage. There are products like these on the market Co:Writer® by Don Johnston inc. This program is for kids and uses very restricted topic dictionaries but I think that there will come much more advanced tools in a near future.

5 The statistic measurements

The five methods compared in this paper range from very simple to complex. The common feature of the methods is the use of the count of uni, bi and trigrams. In all the methods we used the scores and the resulting ranking and discarded level of significance.

5.1 Uni/Bigram and Bi/Trigram

These are the two simplest measurements methods. Here we look at the quotient between unigram and bigram or bigram and trigram. The idea is to counter the effect of commonly used words by looking at the quotient, which means if a certain combination of word is likely it's also likely that a smaller part of it also will be likely. The problem here is that we still do not compensate for extremes in occurrence. But still is an easy and fast way that gives very interesting results for some problems. The big difference between Uni/Bi- and Bi/Tri-grams and the other methods are that they are merely balanced counts as opposed to the other methods, which are based on significance.

5.2 Mutual Information

Fano (1961: 27–28) defined mutual information between particular events x and y, in our case the occurrence of particular words. Pointwise Mutual Information was presented by Church & Hanks in 1990 and is the method we used. Mutual Information measurements are better at compensating for rare word then simple Uni/Bigram and Bi/Trigram. This mean that we find rare word that when used their used together. When Church & Hanks wrote their article they saw a use for tool for lexicographers so they automate the processing of copra. The formula we used looks like this:

$$I\left(w^1, w^2\right) = \log_2 \frac{NC\left(w^1, w^2\right)}{C\left(w^1\right)C\left(w^2\right)}$$

5.3 T-Scores

T-Scores is standard statistical test that looks at the difference between the observed and expected means. T-Scores according to Church & Mercer 1993 better picking out grammatical patterns or combinations of very frequent word such as "and of" or "and the". The formula looks like this

$$T\left(w^1, w^2\right) = \log_2 \frac{C\left(w^1, w^2\right) - \frac{1}{N} C\left(w^1\right)C\left(w^2\right)}{\sqrt{C\left(w^1, w^2\right)}}$$

5.4 Log likelihood

Log likelihood (Ted Dunning 1993) is much more complex than the other measurements we used and claims that it is more accurate seen from a statistical viewpoint. The main advantage with this method is that it can be used on small corpora that are not binomially distributed. In our case we can assume that we have a binomially distributed corpora as we have huge or might I say ridiculously large corpora.

6 Implementation

The implementation consists of tree parts the web connection (Google API), calculations and GUI/displaying (see Appendix A). As the web transactions take a lot of time, every look up take up to 3 sec. and the average length of a sentence is ten words and as we access it 3 time once for unigrams, bigrams, and trigrams so it can take up to 2 minutes. At times when load is heavy, it can time out and fail a search completely and there is not much to do about the time it takes to do a lookup.

6.1 Google API

Google provides a SOAP[1] API with example code in Java. The API makes it possible to access the Google search engine from your program and gives an easy way to retrieve the result. A search via the API should give you the same result as normal web search, but it doesn't as we can se from Table 1 & 2. If you don't use language restriction as in Table 1 the errors are so grave that the result must be impaired.

Word	Google	api	%
The	3670000000	failed	
Of	2440000000	1530000000	0,627049
And	2390000000	1510000000	0,631799
To	2320000000	1460000000	0,62931
A	2240000000	1410000000	0,629464
You	213000000	426000000	2
Was	20100000	245000000	12,18905
An	16000000	300000000	18,75

[1] SOAP, or the Simple Object Access Protocol, is an XML based protocol for accessing remote objects over the network.

Table 1. Google & api no restriction.

Word	Google	api	%
The	18300000	12700000	0,693989
Of	16700000	11600000	0,694611
And	17400000	12100000	0,695402
To	17200000	12100000	0,703488
A	17800000	12400000	0,696629
You	16100000	11200000	0,695652
Was	16200000	11300000	0,697531
An	15800000	11300000	0,71519

Table 2. Google & api language restriction.

What causes these results is unclear. One theory is that it is different languages or control characters that could cause these problems. This may explain some of the grave errors in Table 1 but it doesn't explain the 30% difference between a normal Google search and an API search. I restricted the search to include only English and only the body text in Table 2. A difference that remains and can cause problems is the three percent variation in lookup between words. **And** and **To** has a difference in excess to 200000 when we use a restricted Google search, but when we use the API we get the exact same answer. There is a possibility to check if number of hits is an estimate or if it's an exact value, but when we are dealing with English almost everything is an estimate so it is of no greater use.

When restricting a search, you choose from 28 different languages and 240 domains. It is also possible to restrict a search to just text or links. To even further restrict a search you can put on date restriction or search just one site.

6.2 Scaling

To get the scale right was a time consuming job. And as discussed before we are only interested in the scores and the resulting ranking.
The scaling model can be set in different ways linear, logarithmical or preset scale; they can either be self-adjusting or have fix min and max. Each problem needs is own scaling so that result really shines through the noise. It would also be good if the users could make their own adjustments.

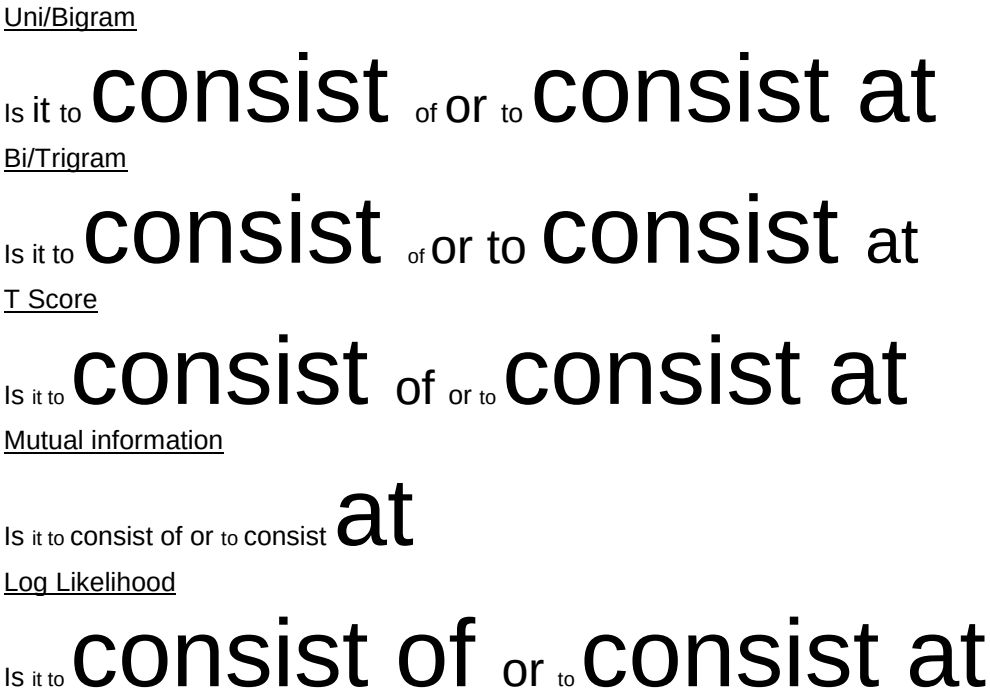

Figure 1. consist at consist of.

7 Results

We start with a favorite example *Will it consist of or will it consist at*.

Here we can see (Figure 1.) the expected difference between **of** and **at** in all 5 of the measurements. We can also se that the word *consist* scores as very unlikely alternative except with the mutual information. This could be explained by the fact that *consist* is a rare word compared to the rest of the words. The behavior that mutual information shows is exactly what we want. A divergence that is harder to explain is the result from log likelihood. Any idea mail me!

In Table 3, we can see the scores from the Goggle API. *Consist* sticks out and is a tenth as common as any of the other words are.

Unigram count	
Is	11600000
it	10900000
to	11100000
consist	1200000
of	11600000
or	11000000
at	10500000

Table 3. Unigrams.

If we look at the bigrams, we find that even without any calculation it is obvious that **consist of** is the correct form. We can also see that all the bigrams that contain *consist* are much lower than the others.

Bigram count	
Is it	5880000
it to	7110000
to consist	123000
consist of	976000
of or	3440000
or to	6790000
to consist	123000
consist at	2130

Table 4. Bigrams

When we look at the trigrams it is not so obvious **to consist at** is wrong because we have **or to consist** that scores equally low. But what we can

see is that the proper way to use **consist** in is the phrase **to consist of.**

Trigram count	
Is it to	318000
it to consist	1090
to consist of	107000
consist of or	5660
of or to	32800
or to consist	192
to consist at	299

Table 5. Trigrams.

Word	T-Score	Mutual	Log Like
it	0,506897	2424,219	11,86122
to	0,652294	2665,84	12,07437
consist	0,010165	350,2004	9,416546
of	0,820168	987,7525	12,46565
or	0,296552	1853,864	11,07464
to	0,617273	2605,124	11,99476
consist	0,010165	350,2004	9,416546
at	0,00179	42,76772	3,769503

Table 6. T-scores Mutual information Log Likelihood

This is for the person who wants to go to the motions and see that I did the math correctly.

8 Further work

- Finding more good examples to try out.
- Tweak the scales and make it to do simple math operation between the ranks of the different measurements.
- Test it other languages and specific domains.
- Implement methods for smoothing N-grams for sparse data.
- Test other measurements like Z-score and χ^2.
- Building a large local database of uni, bi and trigram would be the first step this would cut execution drastically time and make evaluations of methods, measurements and problem a more reasonable task.
- See if it is possible to make word prediction

9 Conclusions

The conclusion must be that it is possible to use the web as base for collecting language data. There seem to be great possibilities to make a lot of different tools out of the data available and the amount of data makes it easier and produces more reliable data. The sparse testing that we have done so far shows good promises for the future.

References

Ted E. Dunning 1993 *Accurate Methods for the Statistics of Surprise and Coincidence*. Computational Linguistics : Vol. 19, No. 1, pp.61-74

Christopher D. Manning and Hinrich Schütze, 2000. *Foundations of Statistical Natural Language Processing.* , MIT Press, Cambridge Mass., 1999.

Ken. W. Church and P. Hanks. 1990. *Word Association Norms, Mutual Information, and Lexicography*, Computational Linguistics, Vol. 16, No. 1, pp. 22-29

Ken. W. Church and R. Mercer. 1993. Introduction to the special issue on Computational Linguistics using large corpora. *Computational Linguistics*, 19 (1): 1--24.

Appendix A. My program