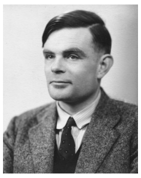
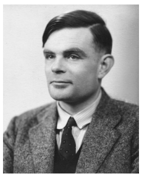
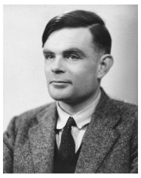
Word prediction performance of n-gram models applied to essentially different corpora

GROUP 34

Sofia Broomé	Jeremy Krebs	Valentin Geffrier	Erik Fredriksen
901210	BIRTHDATE2	BIRTHDATE3	BIRTHDATE4
sbroome@kth.se	MAIL2@kth.se	MAIL3@kth.se	MAIL4@kth.se
			

Abstract

Bla hej bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla

NOTE

- The following sections are arranged in the order they would appear in a scientific paper. We think that these sections need to be there and written. However, these are only guidelines and if you think that some of these sections or subsections are irrelevant to you, please feel free to remove them. Similarly, if you want to include more sections or subsections please go ahead. Also feel free to rearrange them according to your convenience, but keeping some common sense (eg. Introduction cannot come after Conclusions).
- *Introduction, Related Works, Experimental Results, Discussions, Summary* are sections that MUST be contained.
- In the section of your *Method*: please do not list your project as log book entries, please talk about the final method you want to present to us. Talk about the method scientifically or technically and not as "I did this..." "Then I tried this..." "this happened...." etc.
- Do not paste any code unless it is very relevant!
- The section *Contributions* is a place to express any difference in contributions. The default assumption is that you all agree that all of you had an equal part to play in the project.
- We suggest that you try to write this as scientifically as possible and not simply like a project report. Good Luck!
- Please remove **this** NOTE section in your final report.

1 Introduction (1–2 pages)

Being able to dissect, classify, analyze and reproduce language is a highly relevant task for various fields. In the realm of artificial intelligence, we want to give language to our agents by means of communicating with them. When we deal with natural language processing we say that we make language models. Seen as there is no finite set of rules that can describe, say, the entire English language in a complete sense, for pragmatic reasons our best option seems to be basing our models on probabilistic observations - regardless of Noam Chomsky's contempt[11] for the notion of probability of a sentence.

At the foundation of every language model that wants to predict words is the concept of n-grams, a method based on probabilistic distributions over length n combinations of subsequent words. An n-gram is a Markov chain of degree n-1. This quite simple construct can capture many patterns in sentences. Even though it doesn't consider grammar explicitly, grammar will inevitably be built in. For instance, an adjective will in many cases be followed by a noun, or a pronoun by a verb, and thus a bigram composed of those two grammatical types in the mentioned order will score high in probability.

An n-gram gives us context for words, albeit not the full one. Gao and Suzuki[7] explore long distance dependency for words through word clusters and the linguistically motivated *function word skipping* method where function words such as "has", "a", "in", "and", "the", etc, are skipped in favor of more significant words, called head words. In our experiments however, we will not delve further into this subject.

N-grams can also be used in a meta-sense - for instance it's common for part-of-speech-taggers to use n-gram models where they tag the current word based on the last word's tag.

There are some practical issues with the classical n-gram model. What do we do with the n-grams that aren't in our training set and thus have zero probability assigned? This is where techniques of so called smoothing comes in so that our model doesn't fail on encountering a previously unseen word in the test set. In case we are dealing with a higher-order n-gram and we find it has no probability mass , we might want to "back off" from the higher order and estimate the probability for a conditioned unigram, meaning we temporarily look at a smaller portion of a word's history.

Furthermore, what kinds of test sets does our training set allow us to perform well on? One should train on a corpus which is representative of the domain of the intended use. And what happens to our model when we apply it to languages with a higher degree of inflection like Swedish, Basque or German?

From the above examples we see that in many cases just using the n-gram model in itself will not suffice. Over the years researchers in natural language processing have added a lot of tweaks to the original idea such as linear combinations of n-gams, cache language models, LSA-based language models and maximum entropy models, to name a few.

In what follows we will explore n-gram models of varying degrees on dito corpora and grammar to see which results are obtained under which circumstances.

1.1 Contribution

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla

1.2 Outline

Bla bla bla bla bla bla bla Section 2 bla bla bla bla bla bla bla bla bla Section 3 bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla Section 4 bla bla bla bla bla bla Section 5 bla bla bla bla bla

2 N-gram models

2.1 Theory

The first mathematical tool needed in all natural language processing experiment is n-gram models. Speech taggers, smoothing and other methods may not be implemented at first, but n-gram models are needed to predict the next likely words of a sentence. These models are quite easy to understand: if n is an integer fixed, a n-gram model works as a Markov chain to predict the next word. A model is trained on a corpora C, for instance a book or a set of books, so that the probability of each n-gram in the language is learned by the model. The bigger the corpora the more accurate and reliable will be the model as it will have more data to compute probabilities and More precisely, the probability p that the word w_n follows the group of words $w_1 w_2 ... w_{n-1}$ is given by the following formula:

$$p = p(w_n | w_1 .. w_{n-1}) = \frac{|(w_1, .., w_{n-1}, w_n) \in C|}{|(w_1, .., w_{n-1}, x) \in C|}$$

With this formula, it is possible to know the more likely next word of a sentence, but also to generate the end of the sentence, each new word selected at random using this probability distribution. Therefore, a n-gram which appeared a lot in a corpora is more likely to appear in the sentence generator. However, this also tells us that heuristically, we should expect different results from one corpora to another. For instance using corpora from Shakespeare, the generated sentences are likely to be more erratic than with a novel since Shakespeare's syntax and grammar is more complicated.

2.2 Experiments

2.2.1 Parameter n

The first experiment we did was to try to understand the influence of n in our word predictor and how we could tune this parameter. In these experiments, only the corpora and n is changed. There is no use of speech tagger or smoothing yet. The figure **??** shows multiple iterations of prediction of the end of the sentence "Alice was looking for" with n-gram models for n between 1 and 4. The corpora used was the novel "Alice in Wonderland" from Lewis Carroll, 1865. We should note that a 1-gram model is not a Markov model. It just predicts a word according to its frequency in a book, regardless of the previous words. As punctuation symbols are considered as words in this corpora, that explains why it predicts so many commas and apostrophes. Using 2-gram models will only predict a word based according to the last one. This explains why after words like "the" and "a" there are often adjectives or nouns. However, the prediction is still not perfect since 2-gram models seems a bit shallow. With 3-gram models and 4-gram models one can see that the sentences are a bit more grammatically correct but there is still some issues since the English grammar and syntax are not used in these models. This is why this is necessary to implement taggers and use a model that takes into consideration the grammar and the syntax of a language. Another thing with 4-gram models is that we can see the predictions are quite close for different attempts. This is because the more words are fixed, the less freedom there is for the next word: there is less 2-gram starting with "for" than 4-grams starting with "was looking for" in the corpora.

If n is too low, the predictor works really bad because one or two words might not be enough to predict a likely next word. However if n is too big, the corpora need to be really big as well and contain enough different n-gram so that the predictor is more diversified.

3 My method

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla

3.1 Implementation

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla

Bla bla	Bla bla	Bla bla
42	42	42
42	42	42

Table 1: A description that makes browsing the paper easy and clearly describes what is in the table.

bla bla bla bla bla bla bla bla bla bla bla

4 Experimental results

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla

4.1 Experiemntal setup

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla

4.2 Experiment ...

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla

Bla bla bla bla bla Figure 1 bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla

Bla bla bla bla bla Table 1 bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla

5 Summary and Conclusions

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla

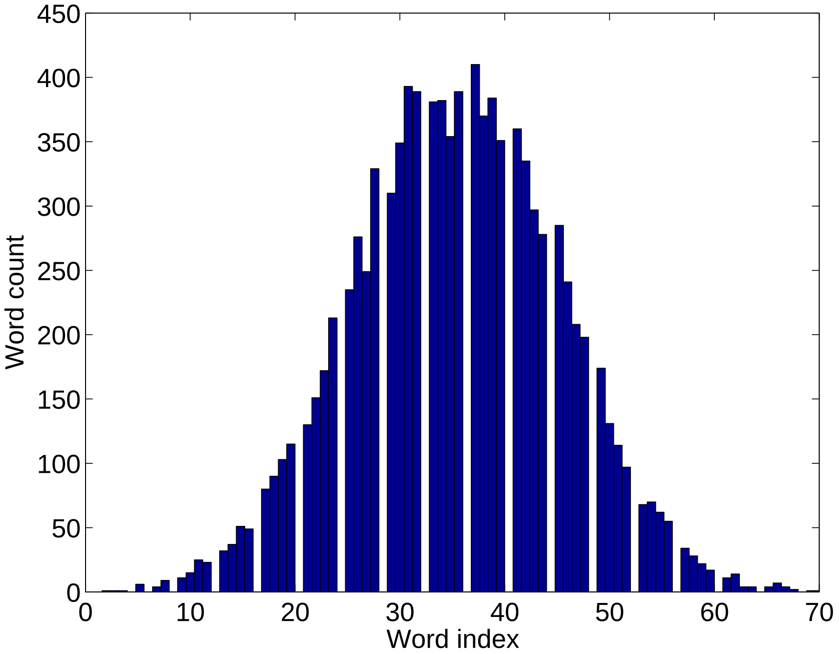


Figure 1: A description that makes browsing the paper easy and clearly describes what is in the picture. Make sure that the text in the figure is large enough to read and that the axes are labelled.

6 Contributions

We the members of project groupXX unanimously declare that we have all equally contributed toward the completion of this project. (PLEASE CHANGE THIS SUITABLY WITH DETAILS, IF IT IS NOT TRUE)

References

[1] Adam L Berger, Vincent J Della Pietra, and Stephen A Della Pietra. A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):39–71, 1996.

[2] Steffen Bickel, Peter Haider, and Tobias Scheffer. Predicting sentences using n-gram language models. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 193–200. Association for Computational Linguistics, 2005.

[3] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, 2009.

[4] Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479, 1992.

[5] John Eng and Jason M Eisner. Informatics in radiology (info rad) radiology report entry with automatic phrase completion driven by language modeling 1. *Radiographics*, 24(5):1493–1501, 2004.

[6] Goodman Joshua Cao Guihong Gao, Jianfeng and Hang Li. Exploiting headword dependency and predictive clustering for language modeling. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 248–256. Association for Computational Linguistics, 2002.

[7] Jianfeng Gao and Hisami Suzuki. Long distance dependency in language modeling: an empirical study. In *Natural Language Processing–IJCNLP 2004*, pages 396–405. Springer, 2005.

[8] Nestor Garay-Vitoria and Julio Abascal. Text prediction systems: a survey. *Universal Access in the Information Society*, 4(3):188–203, 2006.

[9] Peter A Heeman. Pos tags and decision trees for language modeling. In *Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 129–137, 1999.

[10] Rukmini M Iyer and Mari Ostendorf. Modeling long distance dependence in language: Topic mixtures versus dynamic cache models. *Speech and Audio Processing, IEEE Transactions on*, 7(1):30–39, 1999.

[11] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2000.

[12] Daniel Jurafsky, Chuck Wooters, Jonathan Segal, Andreas Stolcke, Eric Fosler, G Tajchaman, and Nelson Morgan. Using a stochastic context-free grammar as a language model for speech recognition. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 189–192. IEEE, 1995.

[13] Johannes Matiasek, Marco Baroni, and Harald Trost. Fastya multilingual approach to text prediction. In *Computers helping people with special needs*, pages 243–250. Springer, 2002.

[14] Stuart J. Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach.* Number ISBN 978-0-13-207148-2. Pearson Education, 3rd edition, 2010.