

Carl Bildt Tweets: A comparison of regular and constrained Markov chain for text generation

Group Ain't intelligent

Viktor Björkholm	Jesper Bränn	Daniel Duberg	Jakob Tideström
92-11-17	92-09-30	93-01-17	90-10-04
viktorbj@kth.se	jbrann@kth.se	dduberg@kth.se	jakobti@kth.se



Abstract

Skriv sist, nr vi vet vad vi har skrivit om (y)

1 Introduction (1–2 pages)

This paper aims to develop an understanding on refining natural language text generation. Natural Language Generation (NLG) is an area of research within the field of Artificial Intelligence. The aim is to generate text that is semantically correct in order to make communication with computer systems more natural and understandable for users.

Within this paper we show the difference in quality of two different approaches to text generation. One of these approaches is using Markov chains or more commonly known as n-grams. These n-grams take n words in sequence and uses a corpus of text to guess what the most probable next word is. Using a larger n means that more text is copied straight from the corpus, however this also means that there is a higher likelihood that the text being generated is meaningful. We will contrast this method with using constrained Markov chains. The main constraint of the Markov chains is that two words following each other will have the same sequence of part-of-speech as the corpus. Part-of-speech is a concept within NLG that divides a text into the different linguistical categories of the words within it.

We aim to show that using this constraint upon the Markov chain, sentences will have a greater diversity but still be as semantically correct as just using a n-gram. Since the problem with larger n:s within n-grams is that text is copied straight from the corpus, the constraint will help us create semantically correct sentences without taking word sequences straight from the corpus.

To be able to show differences in these two approaches we generate Twitter messages, so called tweets. We build upon the work by Barbieri et al., 2012 to implement our own constraints.

1.1 Contribution

We have implemented a unigram of part-of-speech on to a bigram in order to observe the difference in the result. As mentioned above a problem with n-grams with too high n:s is that they will simply copy parts of the corpus if said corpus does not contain a large variation of similar sentences, so that a given start of n words does not automatically lead to one sentence finish. i.e. a larger corpus is needed for larger n:s. To be able to keep a smaller and diverse corpus we applied the unigram on top of the n-gram to allow the

program to select words of a common word type order but perhaps not words that have occurred after each other naturally in the corpus.

Our main contribution to the field is that we have tried putting this unigram constraint upon the regular Markov chain and doing it for short message generation. The research area we base this paper on focus on generating text that fits a theme, or rhythm whereas we generalize the concept.

A lot new research involves the constraining of Markov chains to produce text that fits different molds than just text generated from a corpus. It is important to be able to generate text that is

1.2 Outline

We bring up the relation of our work to some other work in the field Barbieri et al. [2012] in section 2 and explain how that research has affected ours. In section 3 we then go through the details on how the algorithm works, we also give examples to explain in detail what the difference between the two methods we are comparing. This is complemented with details regarding our specific implementation of the algorithm in section 3.1. The data from running the algorithm is explained, reviewed and evaluated in section 4.

2 Related work (1–3 pages)

In Shannon [1948] Shannon goes through the first basic steps in how to naively generate text from a corpus with Markov chains. He explains in depth how Markov chains can be used to generate real sounding English words from letters and also sentences from words. By creating differently sized n-grams for both letters and words he shows how the results improve when n grows. The technique is building up a transition matrix where each row and column index are the words or letters that represent the states. The values of the rows in a column represent the probability of moving from the state of the column to the state of the row. The probabilities of the transitions between the different states is taught to the transition matrix by iterating through the corpus. In a unigram the states could be every single different word and the probability of moving to one word from another is determined only by looking at the preceeding one. However, in a bigram the different states grows exponentially as they could be every two combinations of represented sequences of word, since every new word in a bigram is dependent on the two preceeding ones. This is then followed through to generate the text. His approach was novel at the time but it is a staple in modern text generation. The base Markov chain used in this paper is generated in a similar way to what Shannon describes. It builds what is the ground of the algorithm which we build on to.

In the work by del Socorro Bernardos Galindo and de Cea [2001], they describe the process of building a corpus for a natural language generation system. Different steps are presented which span over the very basic demands such as the structure of the content of the corpus that is supposed to be examples of output. To clarify, the content of the corpus should be examples of what we want the output from our system to immitate. We applied this demand by populating our corpus with content in the same style that we want to generate. They also describe an early step of deciding a form of constraints (user requerments) of the corpus, if the users want the output to maintain a specific format or if they want the output to exclude specific content. As we play the role of our own users, we made sure to specify what kind of content we want to exclude that was naturally included in the corpus, such as hash-tags and image links in our case with tweets. Lastly it describes the step of applying the user requests on the collected data that we want to populate our corpus with. It involves modifying, simplifying and removing data that we can not use in its original form. We wrote a script that downloaded tweets in to our corpus that applied our requests on the format of the outputs on the data downloaded.

Our work on actually constraining Markov models is built upon Barbieri et al. [2012] work, where the authors generated lyrics from different artists using Markov models with constraints. These were to be generated in a specific style and with a rhythm, to simulate real lyrics from a large span of artists. They start in a standard fashion by building up a corpus of content that they want to imitate. In their case they built up a corpus consisting of the collected work of each artist. So when generating lyrics for bob dylan as an example, the corpus consisted of 7600 unique words, a total of 96 089 words and 12 408 verses. They proceeded with applying constraints on a bigram to meet their demands on the generated text to have a certain style and to rhyme. Some of the constraints they apply on their bigram is that some words need to have the same meter as in the original song, but also that the text generated needs to fit a part-of-speech template and a rhythmic template that comes from the song the newly generated lyrics is to be based on. The templates are not generated by their algorithm, but rather handcrafted to get a proper result. Our take from this paper was to be able to apply constraints upon a bigram, or rather a Markov chain in general. Similarly we utilize a part-of-speech template of sorts, but we diverge a bit from this paper.

3 My method (1–4 pages)

Our method that forms the basis of this paper is generating Twitter messages with the use of both Markov chains and constrained Markov chains and to compare the two methods with each other. The theory is that constraining an Markov chain will yield better and more diverse text being generated. In section 4 where we explain our experiments and data we try out different orders of the Markov chain, for the sake of explanation we assume a first order Markov chain is used to explain how the method works. In a limited corpus such as this one:

“Rolf has a dog.”	Noun → verb → singular quantifier → noun
“Rolf owns a dog.”	Noun → verb → singular quantifier → noun
“Rolf can not walk his dog.”	Noun → modal → adverb → verb → pronoun → noun

This corpus generates a Markov chain that looks like figure 1a, with the edges being the probabilities for the specific transitions. In the figure the specific part-of-speech is included in the states beneath the words from the corpus. If we then add constraints from a transition matrix, built up from POS-analysis of the same corpus we are given the new chain that is seen in figure 1b.

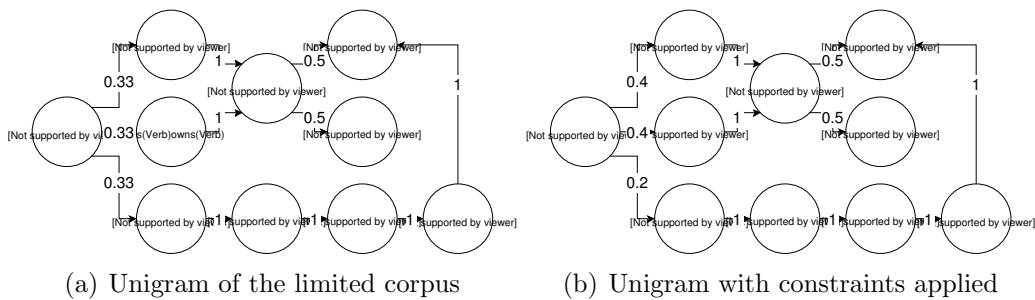


Figure 1: Unigrams

We can see in figure 1b that since both “has” and “owns” are verbs they are more probable to occur than “can”, this is reflected in the new edges. This happens because the part-of-speech “verb” is more likely to follow NNP according to our transition matrix, and thus “has” and “owns”, who are both verbs become even more likely to follow NNP (Rolf). This method can then be further applied to a bigram, a trigram or any n-grams that follows. Our method however will only have an unigram for the transition matrix, even if the Markov chain of words from the corpus is longer, the transitions are only

observed with one previous state in consideration.

3.0.1 Calculating uniqueness

We originally wanted to analyse the semantic correctness of our tweets but were unable to find a method of doing so mechanically. In lieu of other options we decided to instead analyse how unique the tweets were in terms of how much it just copied straight out of the corpus.

Since we use a bigram it will always take three successive words but anything above that compromises uniqueness.

For example purposes we added these sentences to the limited corpus mentioned earlier:

“Mufasa is a dog with a purple tail.”

“Mufasa owns a yellow cat.”

“Jessica can not eat a yellow dog with a pink tail.”

“Steve has a shirt with a purple tail.”

Example 1: “Rolf can not eat a yellow cat.”

It first matches “can not eat a yellow” since it is the largest continuous match, this reduces uniqueness by 2. What is left is “Rolf” and “cat” since both are shorter than three words we stop here.

Thus this sentences uniqueness is $\frac{9-2}{9} \approx 78\%$.

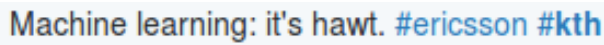
Example 2: “Rolf has a shirt with a yellow dog with a pink tail.”

The first match is “a yellow dog with a pink tail”, reducing uniqueness by 4. Remaining is “Rolf has a shirt with” which matches “has a shirt with”, further reducing uniqueness by 1. Only “Rolf” is left over thus we stop.

Thus this sentence has a uniqueness of $\frac{12-(4+1)}{12} \approx 58\%$.

3.0.2 Corpus

In order to generate tweets, we first discussed the problem regarding the semantical corectness of a generalized tweet. The average level is according to our own experience far from sematically correct, which isn’t a problem in understadabillity for an experienced Twitter user, but is a problem for our POS who would not recognize the words. Even if it would give it an “unknown“-tag, we would not be able to predict any kind of results.



Machine learning: it's hawt. #ericsson #kth

Figure 2: An example tweet

To solve this problem approximately, we decided to generate tweets for a specific user who mostly uses correct grammar and semantics when tweeting and tweets in English. Our option fell on Carl Bildt, former foreign minister of Sweden, because of his active use of Twitter, that he tweets in English and that most of his tweets are in grammatically correct English.

3.1 Implementation (0–2 pages)

The implementation relied on a Part-of-Speech tagger from Stanford's Natural Language Processing group.

We start by collecting data, in the form of tweets, by a target person, Carl Bildt, to be used as the corpus of our tweet generator. The data is filtered to remove unwanted elements such as hashtags and web addresses.

Each tweet is run through Stanford's speech tagger to get the lexical class of each word depending on context. We then use these lexical classes as states for the transition matrix, more specifically they're used to see what lexical class usually follows any given lexical class or terminates a sentence.

While iterating through the gathered data we store what word follows any given two words, including sentence terminating symbols such as '.' or '!' and store it as a bigram.

When generating tweets two methods are used, first the bigram gets to select which words to use entirely on its own digression (Markov chain). Then the process is repeated but this time the bigram is constrained by which lexical class of words it is allowed to choose from, that class being whichever one the transition matrix has decided should follow (constrained Markov chain).

4 Experimental results (1–4 pages)

4.1 Experimental setup

We generate 10000 tweets with a length of between 30 and 140 characters using both the Markov chain and the constrained Markov chain.

Once generated we analyse the tweets uniqueness by counting how many of the used words are from the same part of the corpus. However, since we use a bigram three of the words have to appear in succession and thus we only count if four or more of the words are from the same part of the corpus. We then divide this by the number of words to get an average for the whole tweet.

4.2 Experiment

4.2.1 Uniqueness based on words in tweet

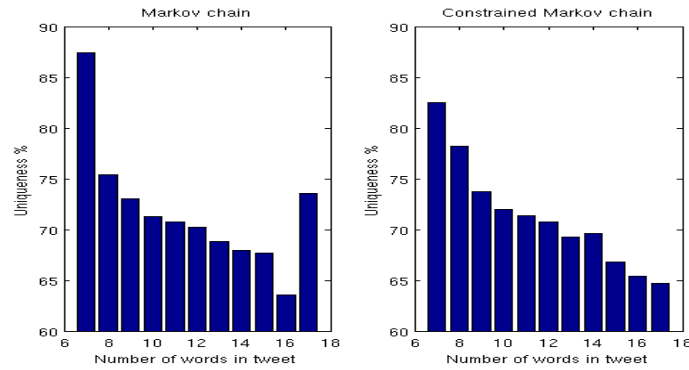


Figure 3: Uniqueness based on words in tweet

Uniqueness drops as the number of words in the tweet increases, both Markov chain and the constrained Markov chain decrease at approximately the same rate however. This is because of how our algorithm for uniqueness works, namely that if the tweet has 6 words then at worst it can take all 6 words from the same part of the corpus. This will result in a uniqueness of 50 percent, since this is a worst case scenario it cannot be lower than this.

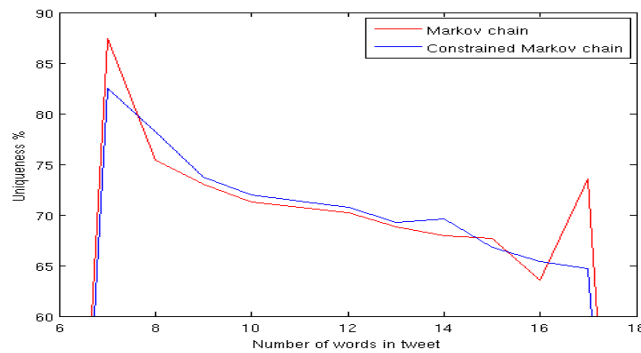


Figure 4: Uniqueness based on words in tweet (Comparison)

On average the constrained Markov Chain produces tweets that are 71.04 percent unique and the Markov Chain has an average uniqueness of 70.29 percent. This means the constrained Markov chain constructs, on average, tweets that are 1.01 percent more unique.

4.2.2 Number of tweets based on uniqueness

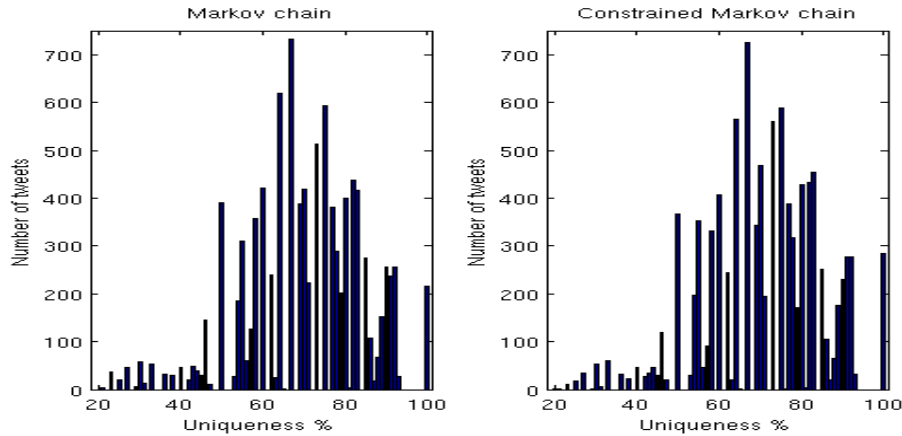


Figure 5: Number of tweets based on uniqueness

Here too both the Markov chain and the constrained Markov chain have similar results as a whole but the Markov chain has more tweets with less uniqueness than the constrained Markov chain. The opposite is also true, the constrained Markov chain has more tweets with more uniqueness than the Markov chain.

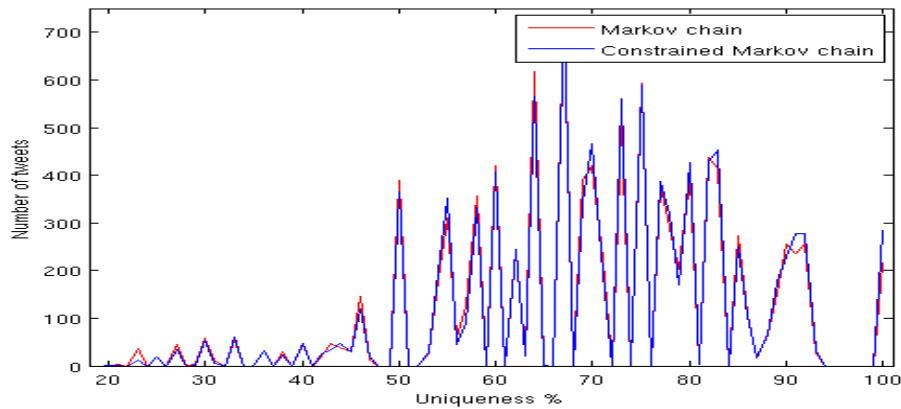


Figure 6: Number of tweets based on uniqueness (Comparison)

Figure 6 makes this more obvious, if you look at the lower end the Markov chain has visible spikes on the lower end (namely around 23 and 46 percent) and the constrained Markov chain has the same on the higher end (82, 91 and 100 percent).

4.2.3 Actual content of tweet

Even though we could not mechanically analyse semantic correctness of our tweets we still looked at some of the tweets it generates.

1. With NGram [23 ms]: Of risk of situation in Ukraine. By Pyongyang standards. With TM [56 ms]: Russia is good.
2. With NGram [19 ms]: Victory Day has a great time! With TM [46 ms]: Truly alarming development in Bosnia.
3. With NGram [142 ms]: And clear messages from Brussels after two days. The final rally of the. With TM [161 ms]: Started the meeting in Brussels. Istanbul after a couple of grey suits as well.
4. With NGram [15 ms]: Third capital today. Symbolic that US wins over Russia. With TM [97 ms]: Advance in Kobane must certainly be stopped. Her contribution on behalf of EU.
5. With NGram [40 ms]: Well, Cold War it was? For a day in the clowns". With TM [605 ms]: PM's Reinfeldt of Sweden. Most of our world. Looks worse than a dog.

5 Summary and Conclusions (0.5–1 page)

We compared generating natural language with a native bigram from our corpus with using a constrained markov chain with a bigram to generate tweets seeming to origin from a specific user. In the result we can see that using a constrained markov chain has a slight advantage over simply using a bigram in generating a diversity of content. We were able to show that with a slight change in implementation we could slightly change the outcome of results. The constrained matrix only uses a unigram, if it would use a bigram for it's POS constraints, perhaps the outcome would have been different. Or less diversity. What do we know. We need logic here. Using i bigram for constraints would perhaps give a more stable result and thus beeing more similar to only using a bigram. But since it's POS, i have no idea.s

References

- Gabriele Barbieri, Franois Pachet, Pierre Roy, and Mirko Degli Esposti. Markov constraints for generating lyrics with style. 242:115–120, 2012. URL <http://dblp.uni-trier.de/db/conf/ecai/ecai2012.html#BarbieriPRE12>.
- Maria del Socorro Bernardos Galindo and Guadalupe Aguado de Cea. A new approach in building a corpus for natural language generation systems. In Alexander F. Gelbukh, editor, *CICLing*, volume 2004 of *Lecture Notes in Computer Science*, pages 216–225. Springer, 2001. ISBN 3-540-41687-0. URL <http://dblp.uni-trier.de/db/conf/cicling/cicling2001.html#GalindoC01>.
- Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, July, October 1948. URL <http://cm.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf>.