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, när 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 purpose of NLG 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 in an n-gram 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 hopefully help us create semantically correct sentences with a greater diversity 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 Shannon [1948] for the theory of implementing an n-gram, a description on generating and parsing corpuses by [del Socorro Bernardos Galindo and de Cea, 2001] and the work by Barbieri et al., 2012 to implement our own constraints on the n-gram.

1.1 Contribution

We have implemented a unigram of part-of-speech (POS) on to a bigram in order to observe the difference in the result. As mentioned above a problem with n-grams with to 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 of POS tags on top of the n-gram to allow the program to select words of a common word type order and with a greater diversity than naturally occurs 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 by creating text that is not constrained by a pre-written sequence of POS. Our approach is that we let the model be trained with two types of information, thus our contribution to the field lies in examining how the combination of these two approaches improves the general natural language generation.

1.2   Outline

We bring up the relation of our work to some other work in the field. We explain how [Shannon, 1948], [del Socorro Bernardos Galindo and de Cea, 2001] and 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)

Shannon [1948] 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. We implemented n-grams and tried different orders of them which according to Shannon would generate output more similar to the content of the corpus with the higher orders. The difference between the different orders of Markov chains lies in how many preceding states that are taken in consideration. Our Markov chain was built up with a transition matrix where 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. This concept is introduced by Shannon and proven to work in his paper. His approach was novel at the time however it is a staple in modern text generation.

In the in-proceedings by del Socorro Bernardos Galindo and de Cea [2001], they describe the process of building a corpus for a natural language generation system with different user requests and demands that constrain the corpus. The corpus consists of example output that the NLG will imitate and the user requests should be applied through modifying and simplifying the corpus according to them. Since the corpus should consist of example output, and we want to generate text that should imitate tweets, we populate our corpus with tweets. As we play the role of our own users since we are the ones with demands on the generated text and we need to generate a corpus that is usable by our algorithms. Thus, as [del Socorro Bernardos Galindo and de Cea, 2001] describe we remove things from our corpus that we do not want to use when generating, such as hash-tags and URLs.

Our work on actually constraining the Markov chain is built upon Barbieri et al. [2012] work, where the authors generated lyrics from different artists using Markov chains 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, which is achieved by applying constraints on a bigram, a Markov chain of order 2. They use POS templates to fit the correct poetic metre, these 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. Similarily we utilize a part-of-speech template of sorts, but we diverge a bit from this paper by not handcrafting them and instead letting our algorithm generate

them based on the corpus.

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.
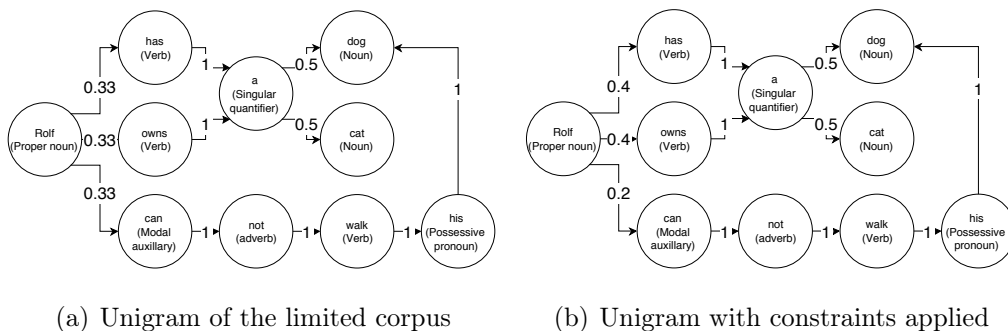

(a) Unigram of the limited corpus   (b) Unigram with constraints applied

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 a personal noun according to our transition matrix, and thus "has" and "owns", who are both verbs become even more likely to follow a personal noun. 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 approximatily, 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 Nat-
ural Language Processsing 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
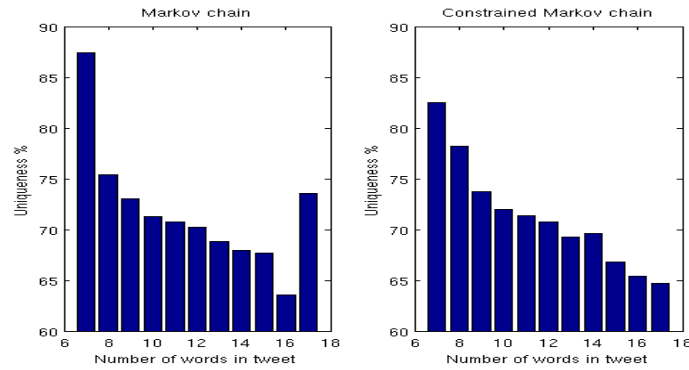
4.2.1   Uniqueness based on words in tweet

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 approximatively 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 that this.
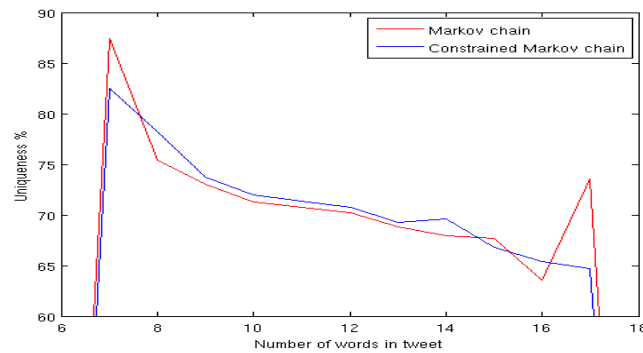

Figure 4: Uniqueness based on words in tweet (Comparison)

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.

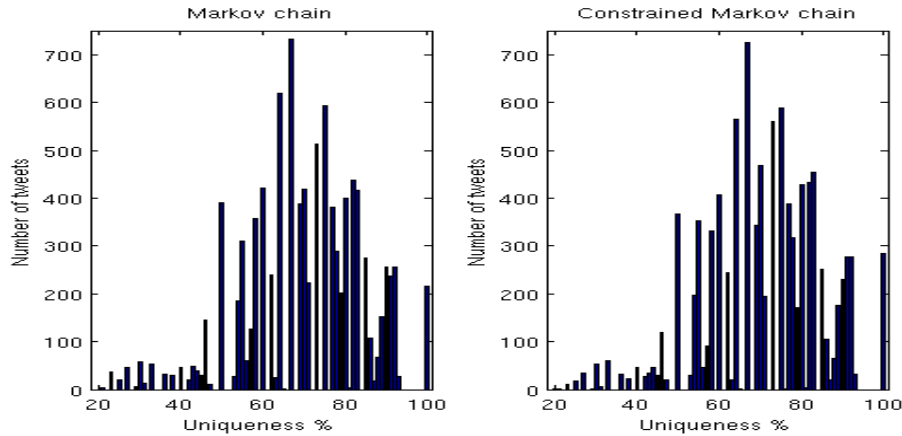### 4.2.2   Number of tweets based on uniqueness

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.
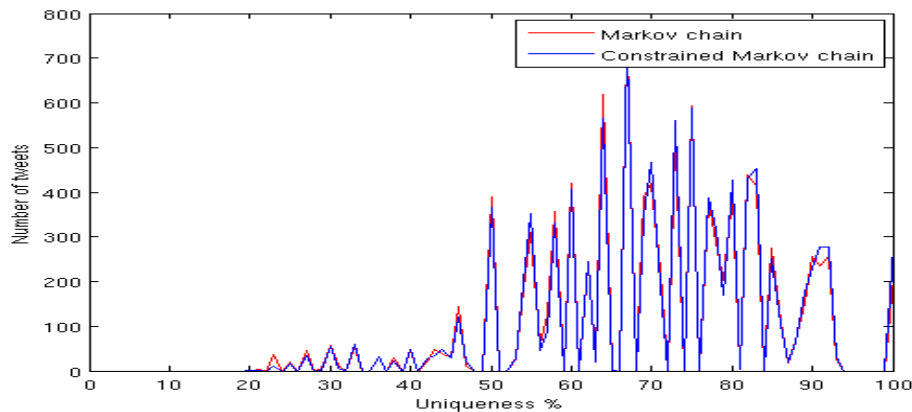

Figure 6: Number of tweets based on uniqueness (Comparison)

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. We generated 5 example tweets, using both methods, for analysis.

Markov chain

"Of risk of situation in Ukraine. By Pyongyang standards."
"Victory Day has a great time!"
"And clear messages from Brussels after two days. The final rally of the."
"Third capital today. Symbolic that US wins over Russia."
"Well, Cold War it was? For a day in the clowns"."

Constrained Markov chain

"Очевидно обострение экономической войны Russia is good."
"Truly alarming development in Bosnia."
"Started the meeting in Brussels. Istanbul after a couple of grey suits as well."
"We have every reason to follow the different parts of Eastern Ukraine."
"PM's Reinfeldt of Sweden. Most of our world. Looks worse than a dog."

Occasionally the beginning and end of our generated tweets are semantically incorrect, this is caused by our corpus filtering as the preceding/subsequent element has been removed from the corpus.

We do not filter languages so sometimes we get long strings of, for example, Russian that we cannot determine if they are semantically correct or not.

Otherwise the subjects of the generated tweets all seem to revolve around politics and foreign relations and could therefore possibly be from a foreign minister.

Regardless of the method used to generate them the tweets are similar.

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 of order 2 to generate tweets seeming to origin from a specific person, which in our experiment was Carl Bildt. The tweets should thus be semantically correct and have a somewhat meaningful content. 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. However, there was seemingly no difference at all in the quality of the content. They did seem to be from Carl Bildt, since the topics of the tweets were affected by the words in his tweets. A lot of the generated tweets had a political theme where conflicts were discussed and political situations in different regions of the world.

We were able to show that applying constraints on a bigram made little difference rather than just using a bigram. We show that our approach adds a lot of work with small gain, and the gain gotten results in mostly nonsensical text. When the diversity in the generated tweet increased so did the quality decrease.

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. The outcome would probably be affected by it, however it is hard to say whether it would improve the results a lot. If the Markov chain was constrained by another property than the sequence of POS in Carl's tweets the results could possibly be different. An alternative could be to generate a template with topics that divide the tweet into parts and let that be the constraint of the Markov chain instead.

References

Gabriele Barbieri, François 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`.