# Convote / SOTU topic modeling

Anton Malko

March 6, 2018

## 1 Convote. Trying out different numbers of topics.

### 1.1 Implementation

In this section I am using the data from Convote dataset (all files inside "data_stage_three" directory). The topic model is trained using Mallet with the following parameters of interest:

- Number of topics: 3, 5, 10 25, 50, 100, 200, 300
- Number of iterations per model: 500

The effectiveness of models is estimated by the Log-Likelihood/Token statistic that Mallet outputs to the console during training. To get a better sense of how the models behave with random initialization, each model is run 10 times, then these values are averaged and their standard deviation is calculated (an even better way would be to repeat this 10-times-running a bunch of times, and get the mean of the sampling distribution of the means and standard errors, but the simulations would require quite some time, so I didn't do it).

In order for the results to be replicable, each of the 10 chains per each model has a seed associated with it; the seeds don't change between the models with different number of topics.

The simulation is controlled from an R script, which sets the parameters (number of topics, number of iterations per model, seeds for chains, output files etc.), calls Mallet, collects its output from the console[1], and produces a plot with averaged log-likelihoods/token for the models with different number of topics. R was chosen because I am more familiar with its plotting system.

See more details on the script in the README file.

### 1.2 Results discussion

#### 1.2.1 Log-likelihood/token scores

Below is the plot of the LL/token means (Fig.1). Notice that the y-axis is inverted, thus the lower the bar (i.e. closer to 0), the better the model performance.

From the figure, it appears that the models performs best with a medium number of topics (50 topics model is the best in terms of LL). One way to interpret that would be to think that with few topics, the model is forced to lump together words which are quite heterogeneous, and thus don't really make a good topic, and with too many topics, the model struggles to select really fine-grained topics, which might be too specific to be useful.

[1]The idea was taken from here: https://gist.github.com/benmarwick/4559589, but I have re-written and expanded the code

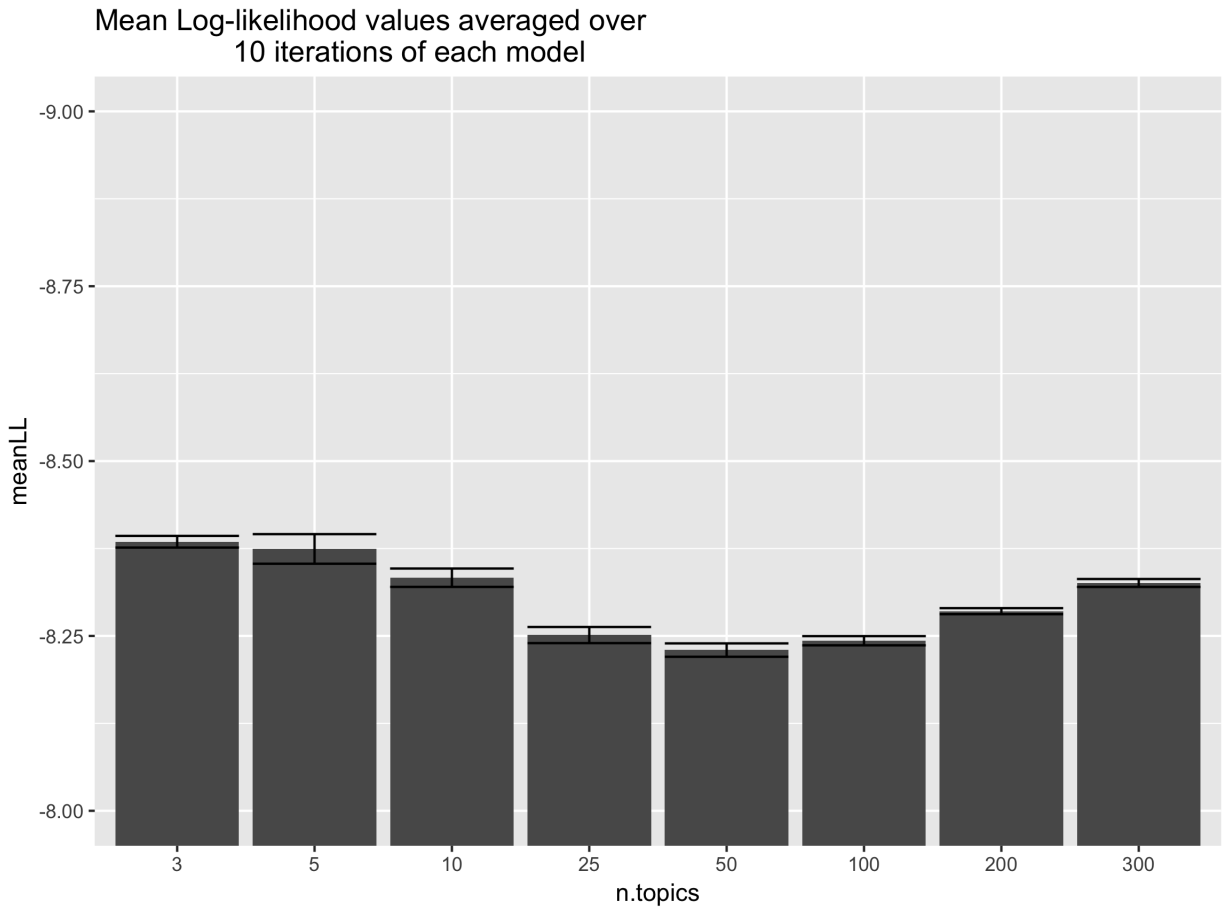


Figure 1: Log-likelihoods/token for the models in Q1. Error bars represent standard deviations

### 1.2.2 Topics composition

The results of Chang et al.(2009) suggest that a good performance of the model in terms of a statistic does not guarantee that the topics it came up with will be judged as good by humans. Below is a short discussion of what topics look like for different models.

As I mentioned, for the previous section each model was trained 10 times with different initializations. For this section, for each number of topics, I select one model out of 10 trained with the highest LL/token number.

The general impressions seem to align well with the LL numbers. For models with few topics (3,5, even 10) it's quite hard to figure out what the commonality between the words in a topic is; there can be several distinct threads within one topic, or vice versa, topics can have a large overlap in terms of words.

For models with medium number of topics (25, 50) the topics become more interpretable and more narrow; of course, not all of them, but for others it is relatively easy to summarize a topic in one or two words.

For models with higher number of topics it's becoming challenging to manually go through the words and evaluate it (I can definitely see why the work on automatic assession of topic goodness and even topic labeling is useful).

Below I give some examples of 20 most frequent words for some topics in some models. The size of

the word corresponds to its proportion in the topic[2]

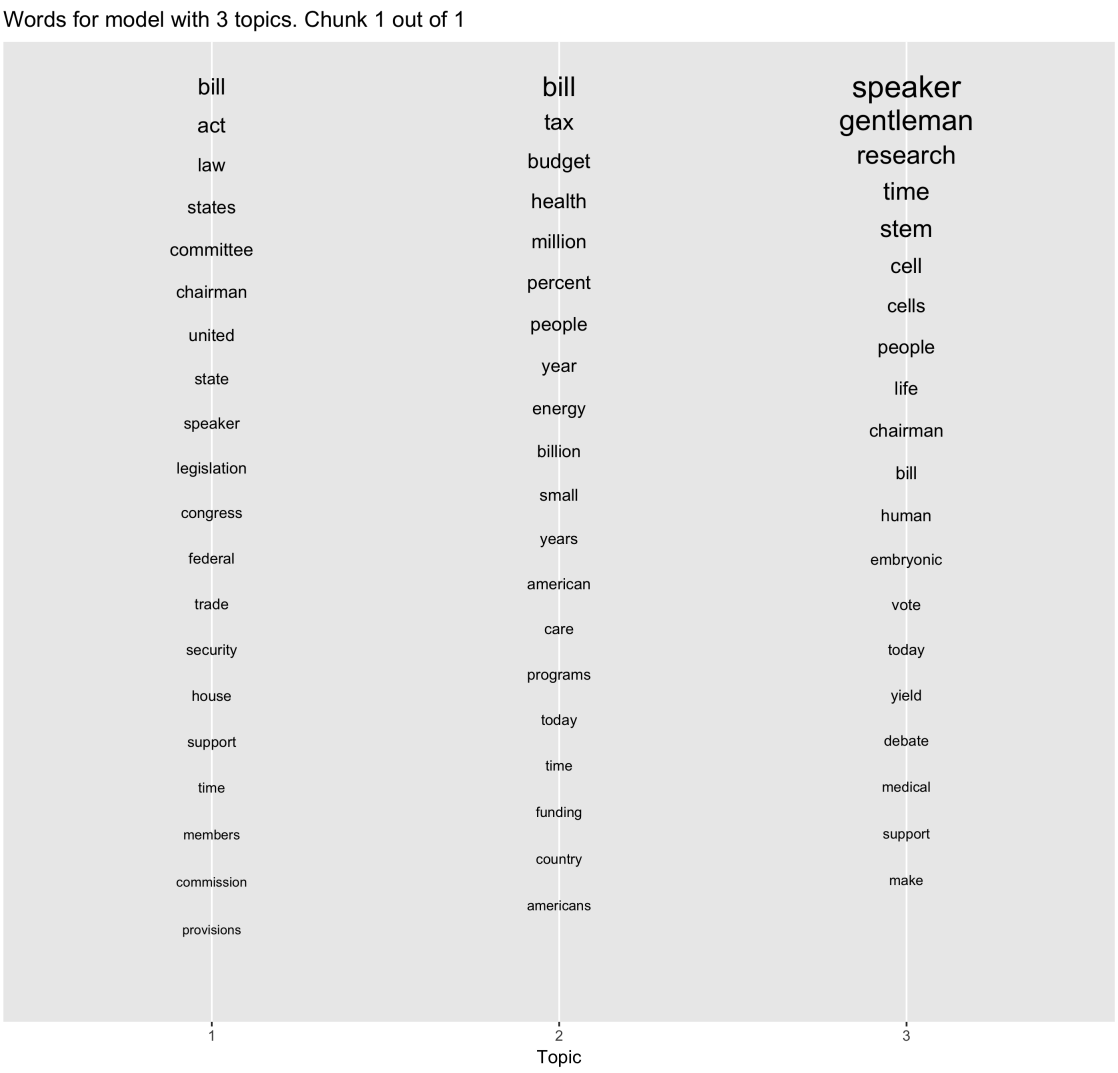


Figure 2: Model with 3 topics

**Model with 3 topics:** If one squints hard enough, topic 2 might be said to relate to money and finance in a *very* broad sense; topic 3 seems to contain some words related to stem cells research (this topic will become much clearer in bigger models), but they are mixed with other clearly unrelated words, e.g. "'speaker" or "today".

[2]The idea is taken from here: https://de.dariah.eu/tatom/topic_model_visualization.html; the code in the link is in Python, I re-implemented the same idea in R. The size of the word depends on it's proportion within a topic in the following way: $\frac{45}{sqrt(proportion)}$. 45 is just a hand-picked scaling constant; sqrt function ensures that small proportions are translated to reasonably big font sizes, and that the proportions which would be too far apart if plotted on original scale are smooshed together a little bit.

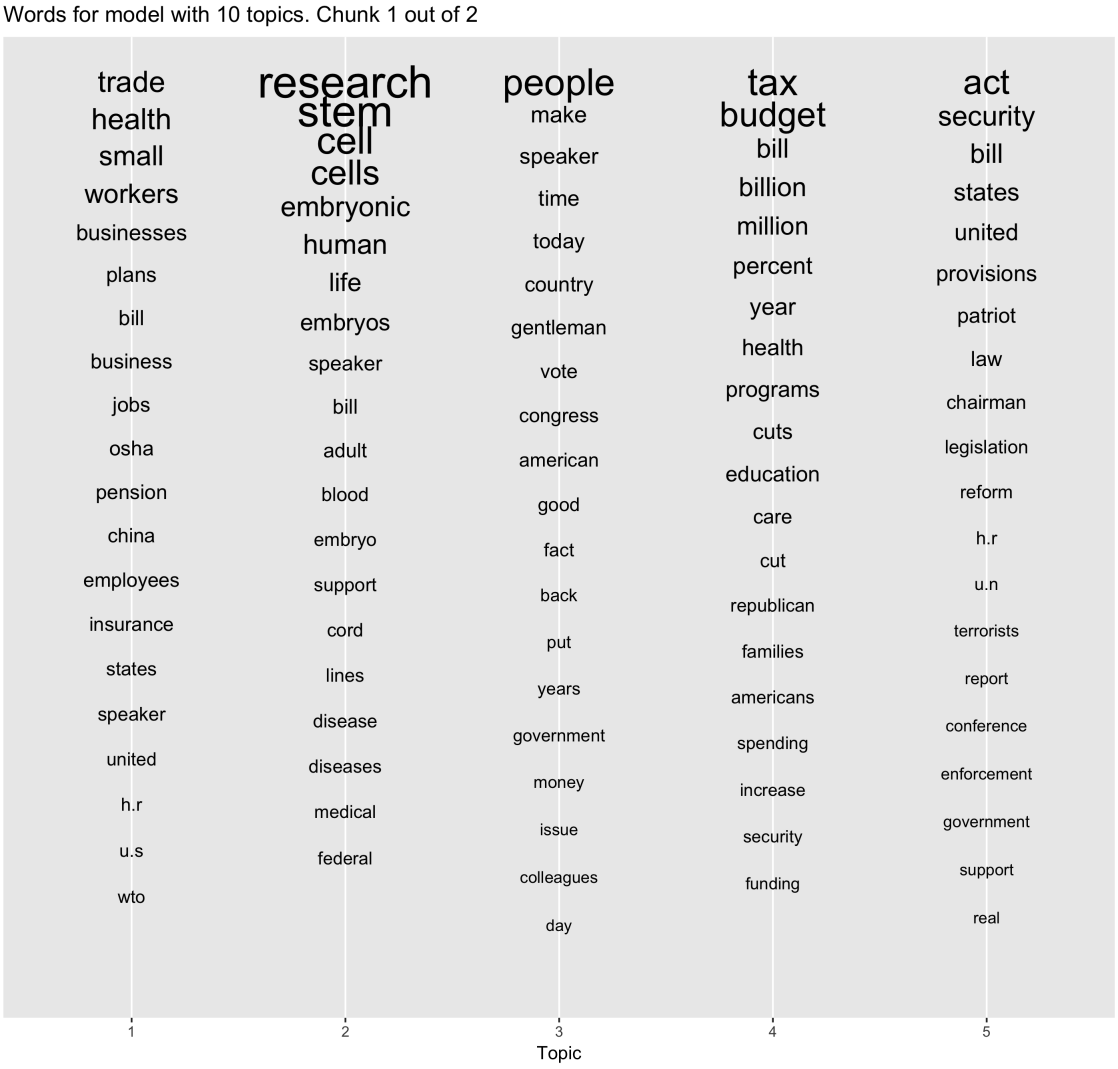


Figure 3: Subset of topics from model with 10 topics

**Model with 10 topics:** The topics become more clearly shaped. E.g. topic 2 is quite clearly relates to medicine and health, and topic 4 is more clearly about budget and finance compared to topic 2 from the model with 3 topics.

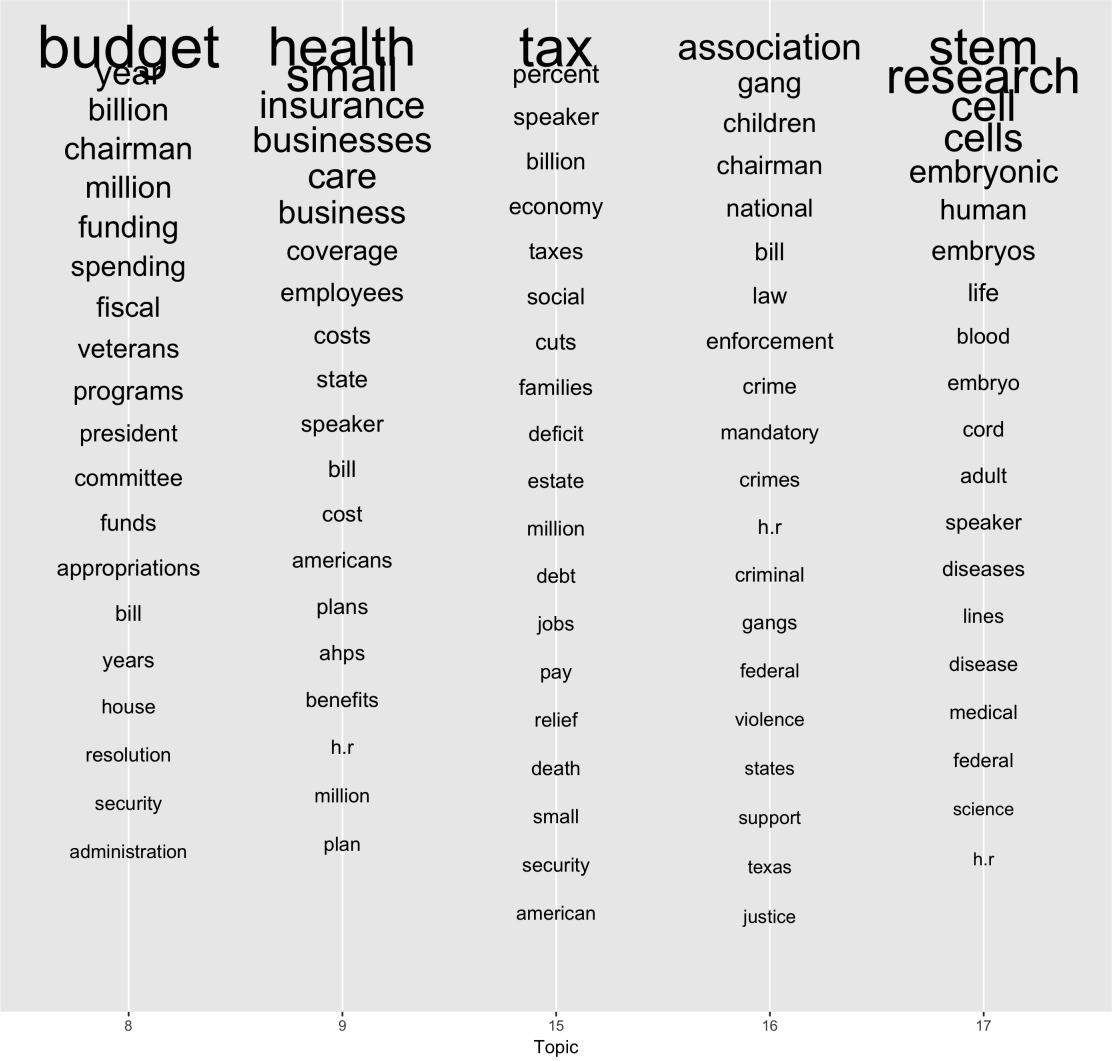


Figure 4: Subset of topics from model with 25 topics

**Model with 25 topics:** The topics are quite clearly shaped, and they also start to narrow down: e.g. cf. topics 9 and 17, which both connect to health (like topic 2 from the model with 10 topics), but are more specialized: topic 9 seems to be more about health insurance, and topic 17 is more about stem cells research. Similarly for topics8 and 15: both seem to be about finances, like topic 4 from the model with 10 topics, but they are specialized too: topic 8 seems to be about budget in general and topic 15 seems to be specifically about taxes.

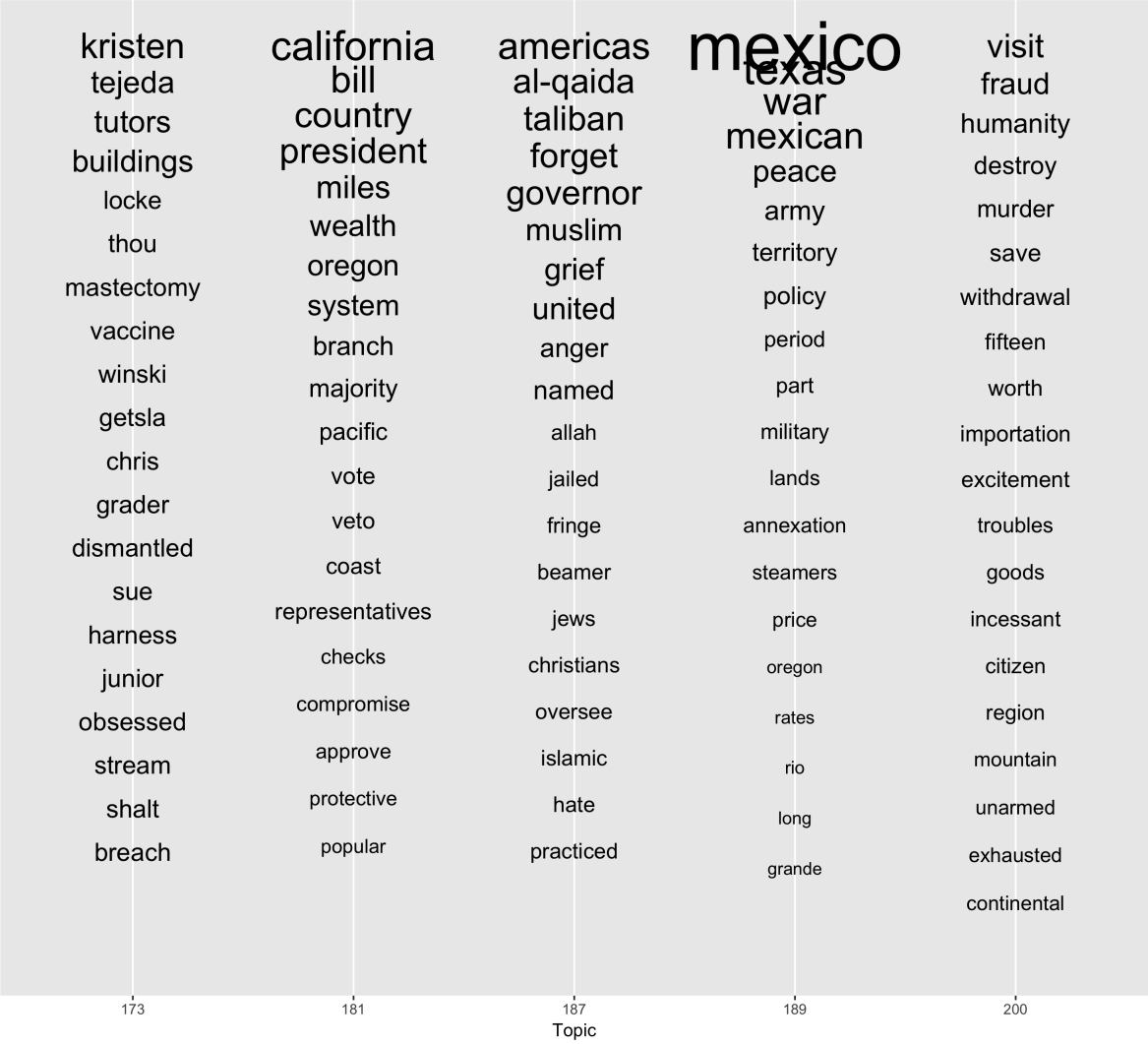


Figure 5: Subset of topics from model with 200 topics

**Model with 200 topics:** Topics start becoming quite narrow, sometimes too narrow for confident interpretation. For some it's still possible to give quite a broad generalization (e.g. topic 187 clearly is about religious issues); for others, it's not quite easy to figure out what the topic is about (in descending order of clarity: 189 clearly has something to do with Mexico and apparently armed conflicts, but it's not immediately clear what real-world events are behind this; 181 might be about California, but other words in the topic don't clarify the picture too much; in 200 the words maybe could be imagined to be connected in some ways (destroy-murder-unarmed-save(?); goods-importation), but the overall theme is not clear; 173 - not at all clear what this topic represents)

## 2 SOTU. The influence of phrases

In this section I am using the SOTU dataset, mainly because it's not lowercased, thus easier to perform named entity recognition upon. I use Python and Spacy to pre-process the data (recognize and mark phrases), and then I use the pipeline from part 1, i.e. calling Mallet from an R script and collecting results from the console.

To get a sense of baseline performance and to pick a good number of topics, I again train models with different number of topics, fitting each model 10 times. Fig. 6 shows the results. Surprisingly, the models with fewer topics show better LL/token scores than bigger models, the model with 3 topics being the best. I am not sure how to explain this. One possibility would be that the speeches are very homogeneous, and there are only a few topics addressed. This may be a little bit of a stretch, since the speeches span a couple of hundred years and it's not that likely that topics don't change in this time.

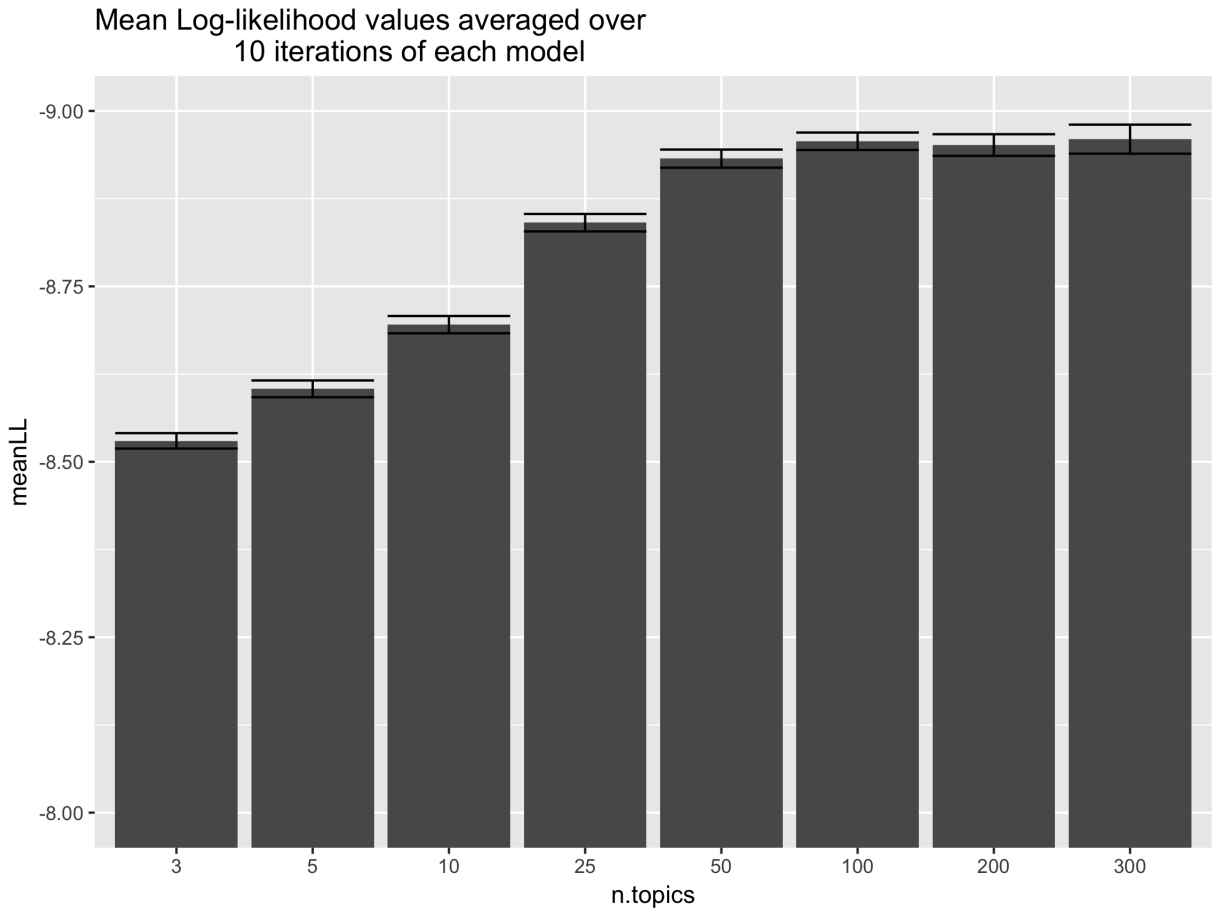


Figure 6: Log-likelihoods/token for the models with different number of topics in Q2. Error bars represent standard deviations

For the following experimentation, I pick the model with 10 topics. It's intermediate in terms of LL/token points, and the number of topics is big enough to make the topic composition analysis interesting.

## 2.1 Data preparation

I train model on 6 variants of SOTU corpus:

1. Baseline: no substantial preprocessing, except converting for the format that Mallet takes in (one file with one document per line).

2. Bigrams: The texts are tokenized, lowercased; stopwords, punctuation and numbers are removed; bigrams are found and sorted by the likelihood ratio statistic. (NLTK is used, and the code is

mostly taken from Assignment 2). Top 15 % percent of the bigrams are converted to tokens (the choice of 15 % is arbitrary).

3. NER: the texts are fed to Spacy, which performs Named Entity Recognition; the entities are converted to tokens.

4. NER_cleaned: same as above, but before being fed to Spacy, the texts are cleaned from stopwords, punctuation and numbers.

5. NP: the texts are fed to Spacy, which parses them and discovers NPs, which are converted to tokens.

6. NP_cleaned: same as above, but before being fed to Spacy, the texts are cleaned from stopwords, punctuation and numbers.

## 2.2 General picture

Each model is run 10 times to determine the average LL/token score; the results are given in Fig. 7. Surprisingly, the baseline model is the best, closely followed by NER and NER cleaned models; other models are quite a bit worse. This result is unexpected to me, since I would think that pre-processing texts to mark collocations would improve models results. I don't have a good sense of why it can be the case.

## 2.3 Topics composition

The influence of preprocessing on topics is surprisingly small. Very few collocations make it to the top 20 words per topic, regardless of the preprocessing regime. The exception to this is NP model, in which the topic include nouns with determiners instead of bare nouns.

One phrase which reliably (and not surprisingly) makes it to the top list in all preprocessing regimes, is "United states".

One reason for why the results are so unimpressive may be a small number of topics. The results from Part 1 suggest that a model with 10 topics might be too small to produce interpretable results; however, the situation is very similar with the model which uses 25 topics. The topics themselves do become more well defined, but the average LL/token scores look almost exactly the same as for the models with 10 topics (Fig. 8: the pattern is exactly the same, but the values are slightly different); and again, the collocations don't almost make it to the top 20 words, except in the case of NP model.

It is possible that for this dataset even 25 topics is too small a number and simulations with bigger number of topics would give different results.

Another possibility is that merging separate words into collocations makes the data more sparse, and if most discovered collocations are pretty rare, the model might be expected to be performing worse (the data got sparser in terms of individual words, and new collocations didn't bring any advantage to the table). If this is the case, then we might see a different pattern on the texts in which many collocations do occur frequently. (Being smarter about collocations filtering, e.g., as suggested in the assignment, only extracting two word NPs even if the identified NPs are longer and potentially filtering these by frequency, might work in a similar direction, leaving only relatively frequent collocations in the texts.) This explanation seems to be plausible for the NP models: the identified NPs were not in any way filtered, so they are likely to have many unique or very infrequent members (e.g. in cases where several adjectives modify a noun) .

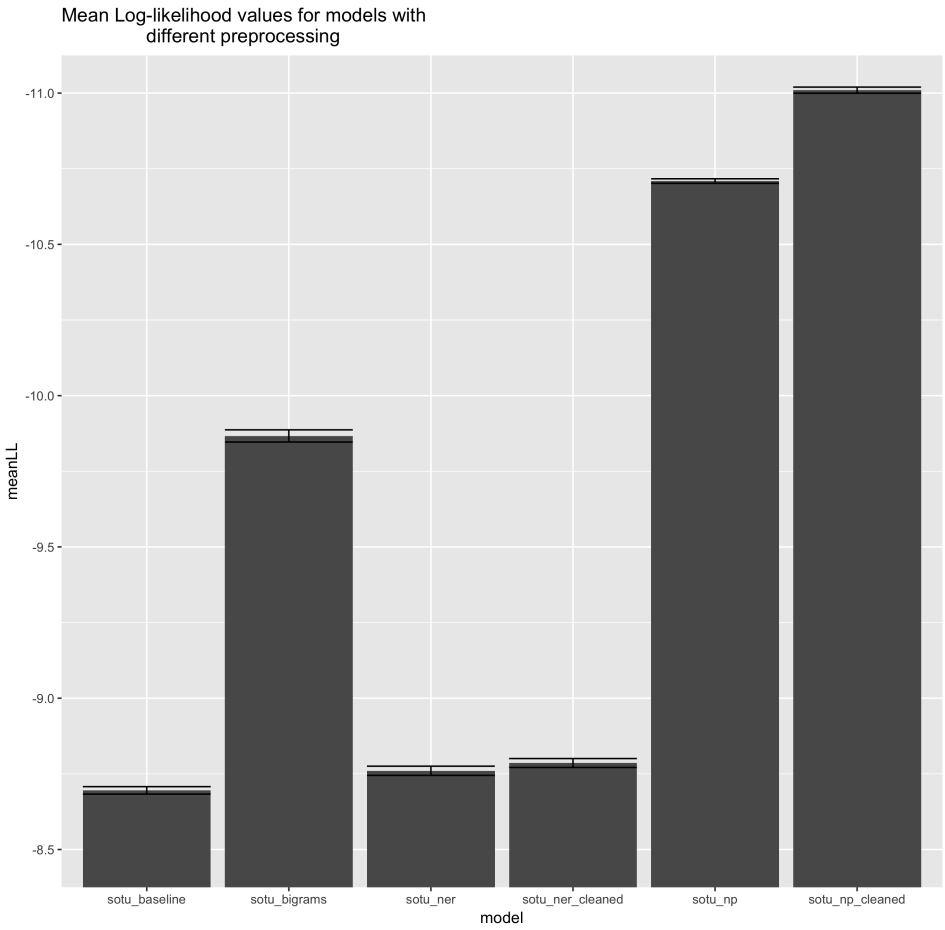


Figure 7: Log-likelihoods/token for the models with different preprocessing in Q2. Error bars represent standard deviations

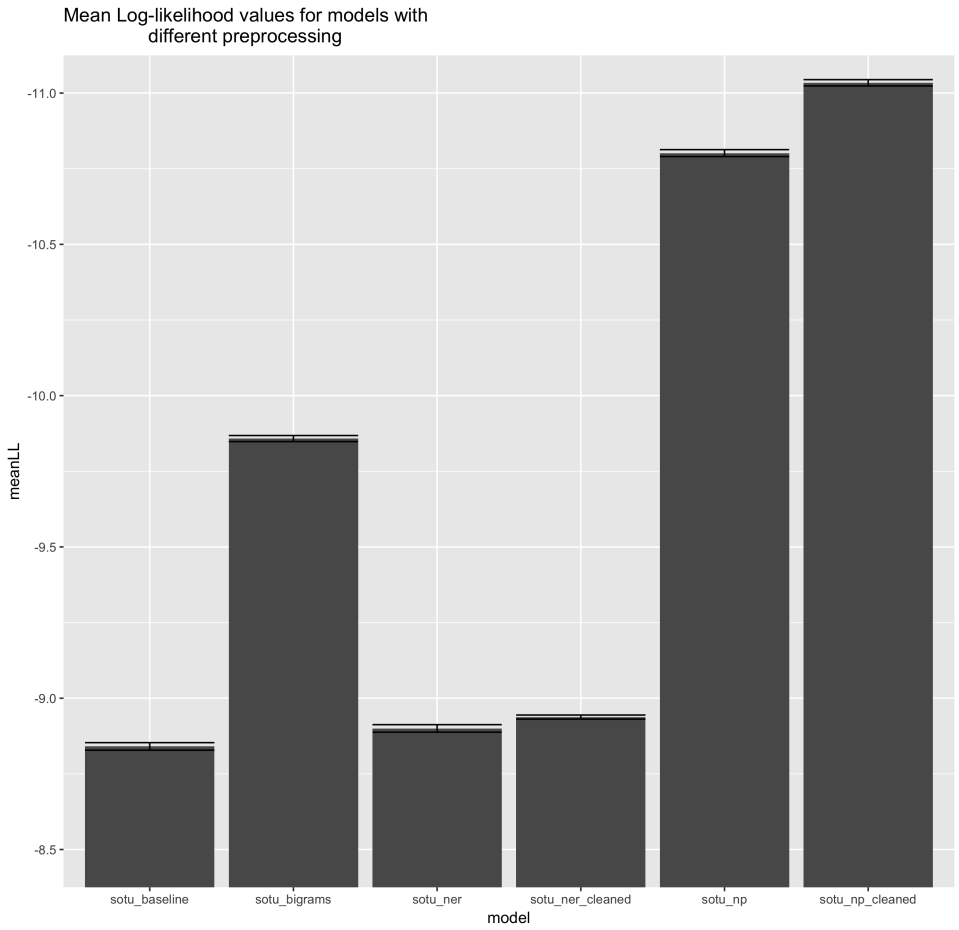


Figure 8: Log-likelihoods/token for the models with different preprocessing in Q2. Error bars represent standard deviations