T-61.5020 Statistical Natural Language Processing

Answers 8 — N-gram language models
Version 1.0

1. In the table below, there are example human estimates (first column) and statistical estimates (second column) on how probable it would be that one of the words would follow the words "tuntumaan jo".

word	statistics/trigram	human/trigram	human/sentence
ja	0.00	0.00	0.00
hyvältä	1.00	0.18	0.40
kumisaapas	0.00	0.00	0.00
keväältä	0.00	0.23	0.50
ilman	0.00	0.05	0.05
päihtyneeltä	0.00	0.20	0.00
turhalta	0.00	0.23	0.05
koirineen	0.00	0.00	0.00
öljyiseltä	0.00	0.11	0.00
Turku	0.00	0.00	0.00

Table 1: *Human vs. statistics, trigram estimates*

We can see that the estimates given by a human are somewhat adrift and statistics badly so.

Statistics were estimated from a corpus of 30 million words. In that corpus, none of the trigrams occured even once. If the words were stemmed, 11 sentences were found that had the trigram "tuntua jo hyvä". The estimate needs badly some smoothing, and it will not be very good even after that.

Also the estimates given by our human can be doubted, as some quite possible senteces have a probability of zero. One example might be "Kyllä alkaa tuntumaan jo kumisaapas jalassa" (free translation: *"Now the rubber boots are beginning to feel in my feet"*), something that one might say after a long walk.

When the full sentence was given to the test person, we got quite good estimates (third column). To get even close with a computer, the language model would need understand grammatical syntax of Finnish as well as sematic meaning of words (e.g. that February is in the end of winter season).

2. a) The maximum likelihood estimates can be calculated as

$$P(w_i|w_{i-1}, w_{i-2}, \dots) = \frac{P(w_i, w_{i-1}, w_{i-2}, \dots)}{P(w_{i-1}, w_{i-2}, \dots)} \approx \frac{C(w_i, w_{i-1}, w_{i-2}, \dots)}{C(w_{i-1}, w_{i-2}, \dots)},$$

where function C tells the number of occurrences in the training set.

In the unigram estimates we forgot everything about the previous words, brigram estimates depend on just one previous word, and trigram estimates use a history of two words.

For unigrams

$$P(w_i) = \frac{C(w_i)}{C_0},$$

where C_0 is number of samples in the training set. These estimates are independent of the word histories.

$$\begin{aligned}
P(\text{'olla'}) &= \frac{5}{98} = 0.051 \\
P(\text{'leuto'}) &= \frac{1}{98} = 0.001 \\
P(\text{'gorilla'}) &= \frac{0}{98} = 0.000
\end{aligned}$$

In the bigram estimates we use the previous word, i.e.

$$\begin{aligned}
P(w_i|w_{i-1}) &= \frac{C(w_i, w_{i-1})}{C(w_{i-1})} \\
P(\text{'olla'}|\text{'olla'}) &= \frac{0}{5} = 0.000 \\
P(\text{'leuto'}|\text{'olla'}) &= \frac{1}{5} = 0.200 \\
P(\text{'gorilla'}|\text{'olla'}) &= \frac{0}{5} = 0.000 \\
P(\text{'olla'}|\text{'vaikuttaa'}) &= \frac{0}{1} = 0.000 \\
P(\text{'leuto'}|\text{'vaikuttaa'}) &= \frac{0}{1} = 0.000 \\
P(\text{'gorilla'}|\text{'vaikuttaa'}) &= \frac{0}{1} = 0.000
\end{aligned}$$

None of the word combinations that did not occur in the training data are possible for this model.

b) Laplace estimation corresponds to a prior assumption that all words have uniform probabilities. In practise, it is assumed that each word is seen already once:

$$P(w_i|w_{i-1}, w_{i-2}, \ldots) = \frac{C(w_i, w_{i-1}, w_{i-2}, \ldots) + 1}{C(w_{i-1}, w_{i-2}, \ldots) + N},$$

where N is the size of the vocabulary.

So we get the following estimates for unigrams:

$$\begin{aligned} P(\text{'olla'}) &= \frac{5+1}{98+64000} = 9.3 \cdot 10^{-5} \\ P(\text{'leuto'}) &= \frac{1+1}{98+64000} = 3.1 \cdot 10^{-5} \\ P(\text{'gorilla'}) &= \frac{0+1}{98+64000} = 1.6 \cdot 10^{-5} \end{aligned}$$

And bigrams:

$$\begin{aligned} P(\text{'olla'}|\text{'olla'}) &= \frac{1}{5+64000} = 1.6 \cdot 10^{-5} \\ P(\text{'leuto'}|\text{'olla'}) &= \frac{1+1}{5+64000} = 3.1 \cdot 10^{-5} \\ P(\text{'gorilla'}|\text{'olla'}) &= \frac{1}{5+64000} = 1.6 \cdot 10^{-5} \\ P(\text{'olla'}|\text{'vaikuttaa'}) &= \frac{1}{1+64000} = 1.6 \cdot 10^{-5} \\ P(\text{'leuto'}|\text{'vaikuttaa'}) &= \frac{1}{1+64000} = 1.6 \cdot 10^{-5} \\ P(\text{'gorilla'}|\text{'vaikuttaa'}) &= \frac{1}{1+64000} = 1.6 \cdot 10^{-5} \end{aligned}$$

Take notice that the uniform prior affects the estimates much. All words have almost the same probability.

c) In Lidstone estimate we can control how much we trust in the uniform prior. We assume that all the words are seen λ times before the training corpus:

$$P(w_i|w_{i-1}, w_{i-2}, \ldots) = \frac{C(w_i, w_{i-1}, w_{i-2}, \ldots) + \lambda}{C(w_{i-1}, w_{i-2}, \ldots) + \lambda N},$$

where we were asked to set $\lambda = 0.01$. The estimates are:

$$\begin{aligned} P(\text{'olla'}) &= \frac{5+0.01}{98+0.01 \cdot 64000} = 6.8 \cdot 10^{-3} \\ P(\text{'leuto'}) &= \frac{1+0.01}{738} = 1.4 \cdot 10^{-4} \\ P(\text{'gorilla'}) &= \frac{0.01}{738} = 1.4 \cdot 10^{-5} \end{aligned}$$

Bigrams:

$$\begin{aligned}
P(\text{'olla'}|\text{'olla'}) &= \frac{0.01}{645} = 1.6 \cdot 10^{-5} \\
P(\text{'leuto'}|\text{'olla'}) &= \frac{1+0.01}{645} = 1.6 \cdot 10^{-3} \\
P(\text{'gorilla'}|\text{'olla'}) &= \frac{0.01}{641} = 1.6 \cdot 10^{-5} \\
P(\text{'olla'}|\text{'vaikuttaa'}) &= \frac{0.01}{641} = 1.6 \cdot 10^{-5} \\
P(\text{'leuto'}|\text{'vaikuttaa'}) &= \frac{0.01}{641} = 1.6 \cdot 10^{-5} \\
P(\text{'gorilla'}|\text{'vaikuttaa'}) &= \frac{0.01}{641} = 1.6 \cdot 10^{-5}
\end{aligned}$$

Here the training data has more clear control on the estimates. A suitable value for λ can be found by taking part of the training data out of the actual training and using it to optimize the parameter.

3. In absolute discounting, we reduce the observed number of occurrences with a constant value (here $D = 0.5$). The discounting can naturally be done only if there were some occurrences. When the discounting is done to the bigram estimates, some probability mass is left out, and that can be used as an interpolation weight (or alternatively back-off weight) with the unigram probabilities. Thus, combining absolute discounting and interpolation, we get the following bigram estimate:

$$P(w_i|w_{i-1}) = \frac{\max(C(w_i, w_{i-1}) - D, 0)}{C(w_{i-1})} + \gamma(w_{i-1})\frac{C(w_i)}{C(0)}$$

Here $\gamma(w_{i-1})$ is a coefficient that normalizes the distribution to sum up to one, and depends on the bigram history. Its value can be estimated from how many different following words the history had:

$$\gamma(w_{i-1}) = |\{(w_{i-1}, w_i) : C(w_i, w_{i-1}) > 0\}| \cdot \frac{D}{C(w_{i-1})}$$

Let's start by calculating the interpolation coefficients for the histories "olla" and "vaikuttaa". The first one had five different right contexts (following words) and it has occurred five times. The second one has occurred one once and thus with one right context.

$$\begin{aligned}
\gamma(\text{olla}) &= 5 \cdot \frac{0.5}{5} = 0.5 \\
\gamma(\text{vaikuttaa}) &= 1 \cdot \frac{0.5}{1} = 0.5
\end{aligned}$$

Now we can estimate the interpolated bigram probabilities:

$$
\begin{aligned}
P(\text{'olla'}|\text{'olla'}) &= \frac{0}{5} + 0.5\frac{5}{98} = 0.025 \\
P(\text{'leuto'}|\text{'olla'}) &= \frac{1}{5} + 0.5\frac{1}{98} = 0.205 \\
P(\text{'gorilla'}|\text{'olla'}) &= \frac{0}{5} + 0.5\frac{0}{98} = 0.0 \\
P(\text{'olla'}|\text{'vaikuttaa'}) &= \frac{0}{1} + 0.5\frac{5}{98} = 0.025 \\
P(\text{'leuto'}|\text{'vaikuttaa'}) &= \frac{0}{1} + 0.5\frac{1}{98} = 0.005 \\
P(\text{'gorilla'}|\text{'vaikuttaa'}) &= \frac{0}{1} + 0.5\frac{0}{98} = 0.0
\end{aligned}
$$

Notice that the method does not give any probability mass to words that did not occur in the training data, such as “gorilla”. This can be fixed by using some additional smoothing method for the unigram. Another alternative would be to continue the interpolation with an uniform distribution (sometimes called “zero-gram”).

Interpolation or back-off to lower order n-grams is not always very reliable. As an example, think about the bigram “San Francisco”. As the name is quite commonly used, the estimate $P(\text{'Francisco'}|\text{'San'})$ would be very reliable. So if the previous word was “San”, no problems there. But what if it wasn't? In this case, the probability of “Francisco” should be clearly lower than it is if we use the unigram distribution without any idea of the previous word. Based on this idea, *Kneser-Ney smoothing* estimates the lower order distributions by calculating how many times the observed word to occurs in a *new* context. The current state-of-the-art smoothing technique is modified Kneser-Ney interpolation, that applies this kind of type counts and absolute discounting with three separately optimized discounts.

4. Let's derive the formula of perplexity so that we can directly use logarithmic probabilities:

$$
\begin{aligned}
perp(w_1, w_2, \ldots, w_N) &= \prod_{i=0}^{N} P(w_i|w_{i-1}, \ldots, w_1)^{-\frac{1}{N}} \\
&= \prod_{i=0}^{N} 10^{-\frac{1}{N}\log(P(w_i|w_{i-1},\ldots,w_1))} \\
&= 10^{-\frac{1}{N}\sum_{i=0}^{N}\log(P(w_i|w_{i-1},\ldots,w_1))}
\end{aligned}
$$

Now we can count the sum of the logarithmic probabilities:

$$\begin{aligned}
&\sum_{i=0}^{N} \log(P(w_i|w_{i-1},\ldots,w_1)) \\
= \; &\overbrace{-4.1763}^{\text{kielen}} \; \overbrace{-2.1276}^{\text{oppiminen}} \; \overbrace{-0.4656}^{\text{on}} \; \overbrace{-0.001-4.2492}^{\text{monimutkainen}} \; \overbrace{-0.8876}^{\text{ja}} \; \overbrace{+0.0495-4.1804}^{\text{huonosti}} \\
&\overbrace{-0.1415-0.1652-5.2195}^{\text{ymmärretty}} \\
= \; &-21.5644
\end{aligned}$$

For those words that had no trigram probabilities, we had to use both back-off coefficients and bigram probabilities. If neither bigram was found, we had to back-off again.

Inserting the result to the expression of perplexity:

$$perp(w_1, w_2, \ldots, w_N) = 10^{\frac{-21.5644}{-7}} \approx 1200$$

This can be thought as that the model corresponds to one that must choose between 1 200 equally probable words each time. Word "tapahtumaketju" was not in the 64 000 most common words and thus not included in the model. So the out-of-vocabulary rate for the model is $\frac{1}{8} \approx 13\%$.