# Tutorial 12: Statistical language models

## Some more on grammars

1. Give three examples of each of the following word classes:
2. count noun: cats, birds, people
3. mass noun: water, air, fire
4. proper noun: John, Frank, China
5. determiner: that, this, these
6. preposition: in, on, under
7. intransitive verb: smile, sit, arrive, go, lie, die
8. transitive verb: kill, give, catch
9. ditransitive verb: pass, give, tell, read, allow, bring
10. Give three examples of each of the following phrase types:
11. none phrase

a box, huge wave, beautiful paint

1. verb phrase

check email, call me, put something somewhere

1. prepositional phrase

under the table, in the box, over the sea

## Linear probabilistic language models

6. n-gram models

1. Explain how an n-gram model can be trained to predict the next word in a corpus  of text.

For a given corpus, we compute the probability distribution of P {Wi, Wi+1 ... **Wi+n-1**}

(for n- gram), scanning through the whole article. Then for testing, for predicting some word W, we compute the conditional probability of P {X | Wi, Wi+1, Wi+n-2}, and we select the one that has max value.

1. What problems arise if we build an n-gram model with a small n? (E.g. n = 1 or 2?)

With small n, when n = 1, we just compute the prior probability of each individual word. And the testing will appear that we just pick up the highest frequency word.

When n = 2, the connection is just the word and its previous potion word. The whole text generated will not make too much sense.

1. What problem arises if we build an n-gram model with a large n?

When n is large, we will meet the sparse data problem which means we don’t have enough data to estimate some vary rare words. And this problem is particular severe because words are distributed according to Zipf’s law: some few words appears much frequently than other words.

1. How can an n-gram model be used to resolve word sense ambiguities?

Similar to use n-gram to predict next word, we can build our n-gram to train the ambiguity. During training, we need annotate those ambiguous words with right meaning, and compute the probability P {W1, W2, … Wn(meaning)}. During the testing, we select the max value of P{Wn(X=meaning) | W1, W2, … Wn-1}.

8. Linear models of language with neural network

1. What is an Elman network? How does it differ from a regular back-prop network? Sketch it and explained how it trained?

An Elman network is simple recurrent network in which the internal state of the network is also depended on its previous state.

A regular back-prop network is back-prop the error of output to its previous layer. So the input unit which are connected with strongest weight will take more responsibility for the blame. So we can train the multiple-layer network. But the network is still a feed forward neural network which its internal state is not depended on its previous state.

1. How is the language model learned by Elman network similar to an n-gram model? How does it differ from n-gram model?

When using Elman network to do sentence generation, the last layer before out put which is the next word is a probability distribution of all the words in the dictionary (using Soft-max function). Then the output is the word with highest probability. After the training, the hidden layer in Elman network learned to represent the aspects of previous n words that are useful to predict the next word. It is basically learning P (next word | previous n word), similar to n-gram. However, the n is not fix in Elman network training and the training is automatic without teaching.