Lecture 15

N-gram models

N-grams

- Sequence of N tokens in a sentence.
- Also called a “phrase” containing a sequence of N words
- Phrase containing a sequence of 2 words : 2- gram (bigram)
- Phrase containing a sequence of 3 words : 3- gram (trigram)
- Phrase containing a sequence of 1 word : 1- gram (unigram)

and so on….

- We have already used N-gram before for feature engineering for Text Classification (N-gram features arranged along columns)

N-gram models

- Next, we will see how N-gram models are used for constructing N-gram language models (LM) from a given input text (training data)
- LM used for language generation
- Language modelling is a type of statistical modelling
- Probability based calculations for predicting one word at a time
- Therefore, text is generated, one word at a time
- Applications of language generation: generating story/script, text summarization (extractive, abstractive), IPL cricket summary (AI journalist), machine translation, chatbots (Seq2Seq models) etc…

Calculations

- Probability of the Nth word given the previous N-1 words

(bigram)
$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

(N-gram)
$$P(w_n|w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1}w_n)}{C(w_{n-N+1}^{n-1})}$$

max probability→winner among all candidate words=next word

- Probability of the whole sentence containing n words (eg. for bigrams)

$$P(w_1^n) \approx \prod_{k=1}^{n} P(w_k|w_{k-1})$$

Smoothing (optional)

- Laplace smoothing (of last slide probability formula)
- To prevent 0/0 situation
- Add 1 to numerator and V to the denominator

$$p_i^* = \frac{c_i + 1}{N + V}$$

V=size of the vocabulary (denominator)

Application of N-gram models –one example

- NMT (Neural Machine Translation)
- Machine translation is the task of converting one sequence to another
- Eg. Translating English to German
- NMT uses neural networks to achieve Machine Translation
- Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." In *Advances in neural information processing systems*, pp. 3104-3112. 2014.

http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf

Perplexity: How good is your N-gram model

- Lower the perplexity, the better your model fits the data
- So you can decide whether a bigram model fits your training data better or a trigram model
- If N= number of words in your test document

$$PP(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$