NLP - Master Notes

June 11, 2021

1 Language Models

1.1 N-Gram Models

- Language (prediction) models which make the *Markov assumption* for an $(n-1)^{th}$ order Markov Model; i.e. that only the previous n-1 words have a probabilistic dependence on the current word.
- Probability of words 1 to n: $P(w_1^n) = \prod_{k=1}^{n} P(w_k|w_{k-N+1}^{k-1})$
  General steps for creating an n-gram model:
  1. choose a vocabulary
  2. add $< s >$ and $< /s$ symbols
  3. replace unknown words in the training corpus with $< UNK >$
  4. calculate probabilities (on an as needs basis?)
  5. calculate most probably words in order (until reaching end of sentence symbol) OR evaluate perplexity of test corpus using above formulas.
- *example:*

  Text:
  One cat sat. Three **cats sat**. Eight **cats sat**. The **cats** had nine lives.
  P(sat | cats sat) $= \frac{C(cats,sat)}{C(cat)} = \frac{2}{3}$

- **Definition of a language model**: a model which assigns probabilities to sentences, based on the training corpus. The sum of all the probabilities of all possible sentences (of arbitrary length) should equal 1.

  Trivial example:
  - Training corpus contains two sentences:
    1. "a b",
    2. "b a"
  - Append $< sos >$ and $< /sos >$ to each:
    1. "s a b /s"
    2. "s b a /s"

– generate the probabilities of a bigram (N=2) language model:

$P(a|s) = \frac{1}{2}$
$P(b|s) = \frac{1}{2}$
$P(b|a) = \frac{1}{2}$
$P(a|b) = \frac{1}{2}$
$P(/s|a) = \frac{1}{2}$
$P(/s|b) = \frac{1}{2}$

– To calculate the probability of ALL possible sentences, we take the prob of all sentences of length 1, all sentences of length 2, etc. Note when calculating this, the TEST sentences need to include $< s > and < /s >$

– For example: P(a) = P($< s > a < /s >$) $= P(a| < s >) * P(< /s > |a) = 1/4$

– Probability is same for b, so the sum of all probabilities of sentence length 1 $= \frac{1}{2}$
The sum of all sentences will be the infinite series:

$$P(all) = \frac{1}{2} + \sum_{i=2}^{\infty} \frac{i!}{(i-2)!} \frac{1}{2}^{i+1}$$

Does this sum to 1? A proof would be cool.

- Sentence Generation: until you produce a $< /s >$ symbol, continually generate words using: $argmax_{(}w_k)\frac{C(w_{k-n+1},...,w_k)}{C(w_{k-n+1},...,w_{k-1})}$

- Perplexity: how well a model fits the data $PP(W) = P(w1, ...w_N)^{-\frac{1}{N}}$, for N words in the test corpus. A perplexity of 1 would be the lowest possible.

- Smoothing:

  – Laplace (add-one): $P(W_n|w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1}w)+1}{c(w_{n-N+1}^{n-1})+V}$, where V is the vocabulary size

  – add δ, normalise by δV

- Interpolation: creating a linear combination of n-gram models of varying n: $\hat{P}(w_i|w_{i-1}, ...w_{i-n}) = \sum_{j=2}^{n} \lambda_j * P(w_i|w_{i-1}, ..., w_{i-1-j})$, where $\sum_j \lambda_j = 1$

- Back-off: back-off to lower n models until data is available (does this mean you can have arbitrary n?)