NATURAL LANGUAGE PROCESSING

CHAPTER 23

Natural Language Processing

- The idea behind NLP: To give computers the ability to process human language.
- To get computers to perform useful tasks involving human language:
  - Dialogue systems
    - Cleverbot
  - Machine translation
  - Question answering (Search engines)

Natural Language Processing

- Knowledge of language.
- One of the important information about a language: Is a string a valid member of a language or not?
  - Word
  - Sentence
  - Phrase
  - …

Language Models

- Formal grammars (e.g. regular, context free) give a hard "binary" model of the legal sentences in a language.
- For NLP, a ***probabilistic*** model of a language that gives a probability that a string is a member of a language is more useful.
- To specify a correct probability distribution, the probability of all sentences in a language must sum to 1.

Uses of Language Models

- Speech recognition
  - "I ate a cherry" is a more likely sentence than "Eye eight uh Jerry"
- OCR & Handwriting recognition
  - More probable sentences are more likely correct readings.
- Machine translation
  - More likely sentences are probably better translations.
- Generation
  - More likely sentences are probably better NL generations.
- Context sensitive spelling correction
  - "Their are problems wit this sentence."

Completion Prediction

- A language model also supports predicting the completion of a sentence.
  - Please turn off your cell _____
  - Your program does not _____
- *Predictive text input* systems can guess what you are typing and give choices on how to complete it.

N-Gram Models

- Estimate probability of each word given prior context.
  - P(phone | Please turn off your cell)
- Number of parameters required grows exponentially with the number of words of prior context.
- An N-gram model uses only N–1 words of prior context.
  - Unigram: P(phone)
  - Bigram: P(phone | cell)
  - Trigram: P(phone | your cell)
- The ***Markov assumption*** is the presumption that the future behavior of a dynamical system only depends on its recent history. In particular, in a ***kth-order Markov model***, the next state only depends on the *k* most recent states, therefore an N-gram model is a (N–1)-order Markov model.

N-Gram Model Formulas

- Word sequences

$$w_1^n = w_1 \ldots w_n$$

- Chain rule of probability

$$P(w_1^n) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1^2)\ldots P(w_n \mid w_1^{n-1}) = \prod_{k=1}^{n} P(w_k \mid w_1^{k-1})$$

- Bigram approximation

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

- N-gram approximation

$$P(w_1^n) = \prod_{k=1}^{n} P(w_k \mid w_{k-N+1}^{k-1})$$

Estimating Probabilities

- N-gram conditional probabilities can be estimated from raw text based on the ***relative frequency*** of word sequences.

Bigram: $$P(w_n \mid w_{n-1}) = \frac{C(w_{n-1} w_n)}{C(w_{n-1})}$$

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

- To have a consistent probabilistic model, append a unique start (<s>) and end (</s>) symbol to every sentence and treat these as additional words.

N-gram character models

- One of the simplest language models: $P(c_1^N)$
- Language identification: given the text determine which language it is written in.
- Build a trigram character model of each candidate language: $P(c_i \mid c_{i-2:i-1}, l)$
- We want to find the most probable language given the Text:

$$
\begin{aligned}
l^* &= \operatorname*{argmax}_l P(l \mid c_1^N) \\
&= \operatorname*{argmax}_l P(l) P(c_1^N \mid l) \\
&= \operatorname*{argmax}_l P(l) \prod_{i=1}^{N} P(c_i \mid c_{i-2:i-1}, l)
\end{aligned}
$$

Train and Test Corpora

- We call a body of text a corpus (plural corpora).
- A language model must be trained on a large corpus of text to estimate good parameter values.
- Model can be evaluated based on its ability to predict a high probability for a disjoint (held-out) test corpus (testing on the training corpus would give an optimistically biased estimate).
- Ideally, the training (and test) corpus should be representative of the actual application data.

Unknown Words

- How to handle words in the test corpus that did not occur in the training data, i.e. ***out of vocabulary*** (OOV) words?
- Train a model that includes an explicit symbol for an unknown word (<UNK>).
  - Choose a vocabulary in advance and replace other words in the training corpus with <UNK>.
  - Replace the first occurrence of each word in the training data with <UNK>.

Perplexity

- Measure of how well a model "fits" the test data.
- Uses the probability that the model assigns to the test corpus.
- Normalizes for the number of words in the test corpus and takes the inverse.

$$Perplexity(W_1^N) = \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

- Measures the weighted average branching factor in predicting the next word (lower is better).

Sample Perplexity Evaluation

- Models trained on 38 million words from the Wall Street Journal (WSJ) using a 19,979 word vocabulary.
- Evaluate on a disjoint set of 1.5 million WSJ words.

	Unigram	Bigram	Trigram
Perplexity	962	170	109

Smoothing

- Since there are a combinatorial number of possible word sequences, many rare (but not impossible) combinations never occur in training, so MLE incorrectly assigns zero to many parameters.
- If a new combination occurs during testing, it is given a probability of zero and the entire sequence gets a probability of zero (i.e. infinite perplexity).
- In practice, parameters are ***smoothed*** to reassign some probability mass to unseen events.
  - Adding probability mass to unseen events requires removing it from seen ones (***discounting***) in order to maintain a joint distribution that sums to 1.

Laplace (Add-One) Smoothing

- The simplest type of smoothing.
- In the lack of further information, if a random Boolean variable X has been false in all n observations so far then the estimate for P(*X*=*true*) should be 1/(n+2).
- He assumes that with two more trials, one might be false and one false.
- Performs relatively poorly.

Advanced Smoothing

- Many advanced techniques have been developed to improve smoothing for language models.
  - Interpolation
  - Backoff

Model Combination

- As N increases, the power (expressiveness) of an N-gram model increases, ***but*** the ability to estimate accurate parameters from sparse data decreases (i.e. the smoothing problem gets worse).
- A general approach is to combine the results of multiple N-gram models of increasing complexity (i.e. increasing N).

Interpolation

- Linearly combine estimates of N-gram models of increasing order.

Interpolated Trigram Model:

$$\hat{P}(w_n \mid w_{n-2,} w_{n-1}) = \lambda_3 (w_n \mid w_{n-2,} w_{n-1}) + \lambda_2 P(w_n \mid w_{n-1}) + \lambda_1 P(w_n)$$

Where: $\sum_i \lambda_i = 1$

- λ_i Can be fixed or can be trained.
- λ_i Can depend on the counts: if we have a high count of trigrams then we weigh them relatively more otherwise put more weight on the bigram and unigrams.

Backoff

- Only use lower-order model when data for higher-order model is unavailable (i.e. count is zero).
- Recursively back-off to weaker models until data is available.

$$P_{katz}(w_n \mid w_{n-N+1}^{n-1}) = \begin{cases} P^*(w_n \mid w_{n-N+1}^{n-1}) & \text{if } C(w_{n-N+1}^{n}) > 1 \\ \alpha(w_{n-N+1}^{n-1}) P_{katz}(w_n \mid w_{n-N+2}^{n-1}) & \text{otherwise} \end{cases}$$

Where P* is a discounted probability estimate to reserve mass for unseen events and α's are back-off weights (see text for details).

Summary

- Language models assign a probability that a sentence is a legal string in a language.
- They are useful as a component of many NLP systems, such as ASR, OCR, and MT.
- Simple N-gram models are easy to train on unsupervised corpora and can provide useful estimates of sentence likelihood.
- MLE gives inaccurate parameters for models trained on sparse data.
- Smoothing techniques adjust parameter estimates to account for unseen (but not impossible) events.