NLP UNIT II

Unsmoothed N-grams

An **N-gram** is a sequence of **N** consecutive items (words, letters, or syllables) from a given text or speech sample. They are the fundamental building blocks of statistical language models.

- **Unigram (1-gram):** A single word. The probability is simply its frequency in the corpus.
- **Bigram (2-gram):** A two-word sequence. The probability of a word is conditioned on the previous word.
- **Trigram (3-gram):** A three-word sequence. The probability of a word is conditioned on the two previous words.

The primary goal of an N-gram model is to predict the next word in a sequence. We calculate the probability of a word w_n given the preceding n-1 words:

$P(wn|w1,w2,\ldots,wn-1)$

For simplicity, we use the **Markov Assumption**, which states that the probability of the next word depends only on a limited number of prior words.

- **Bigram Model:** $P(w\_n|w\_n-1)$
- **Trigram Model:** $P(w\_n|w\_n-2,w\_n-1)$

Maximum Likelihood Estimation (MLE)

The probability of an N-gram is calculated using **Maximum Likelihood Estimation (MLE)**, which is essentially counting and dividing.

For a bigram (w_{n-1}, w_n):

$P(wn|wn-1)=Count(wn-1)Count(wn-1,wn)$

Example: Consider the sentence "the cat sat on the mat".

- Count("the") = 2
- Count("the", "cat") = 1
- P("cat" | "the") = Count("the", "cat") / Count("the") = 1/2 = 0.5

The Zero-Frequency Problem

A major issue with unsmoothed N-grams is that if a specific N-gram never appeared in the training corpus, its probability will be **zero**. This is problematic because it implies the sequence is impossible, which is often untrue. This leads to poor model performance on unseen data.

Evaluating N-grams

The quality of a language model is judged by how well it predicts a new, unseen set of test data. The standard evaluation metric is **Perplexity**.

Perplexity (PP)

Perplexity measures how well a probability model predicts a sample. A lower perplexity score indicates a better language model. It can be thought of as the **weighted average branching factor** of a language.

For a test set W with N words (w_1,w_2,dots,w_N), the perplexity is calculated as:

PP(W)=NP(w1,w2,…,wN)1

A low perplexity means the model is less "surprised" by the test data, assigning it a higher probability.

Smoothing

Smoothing (or discounting) is a set of techniques used to address the zero-frequency problem. It "borrows" probability mass from N-grams that were seen in the training data and distributes it to the N-grams that were not seen.

1. Laplace (Add-one) Smoothing

This is the simplest smoothing technique. It adds **one** to every N-gram count before normalizing.

The formula for a bigram model becomes:

PLaplace(wn|wn−1)=Count(wn−1)+VCount(wn−1,wn)+1

where V is the vocabulary size (the number of unique words in the corpus).

- **Advantage:** Simple to implement and guarantees no zero probabilities.

- **Disadvantage:** Often overestimates and gives too much probability mass to unseen events.

2. Interpolation

Interpolation combines different N-gram models (e.g., trigram, bigram, and unigram) by mixing their probability estimates.

In **linear interpolation**, we calculate the probability by taking a weighted average of the MLE estimates from each model.

$$Pinterp(wn|wn-2,wn-1)=\lambda 1P(wn)+\lambda 2P(wn|wn-1)+\lambda 3P(wn|wn-2,wn-1)$$

where the weights (lambda_1,lambda_2,lambda_3) are learned from a validation set and sum to 1 (sumlambda_i=1).

This approach allows the model to leverage higher-order N-grams when there is enough data, while still relying on lower-order N-grams for context.

3. Backoff

Backoff is similar to interpolation but is more of a "fall-back" strategy. It uses the trigram probability if the evidence is sufficient (i.e., non-zero count). If not, it "backs off" to the bigram model, and if that also fails, it backs off to the unigram model.

General Idea:

- If Count(w_{n-2}, w_{n-1}, w_n) > 0, use the trigram probability.
- Else if Count(w_{n-1}, w_n) > 0, use the bigram probability.
- Else, use the unigram probability.

Unlike interpolation, backoff doesn't blend the models; it chooses one based on data availability.

Word Classes and Part-of-Speech (PoS) Tagging

Word Classes are categories of words that have similar grammatical properties. The most common type of word class is the **Part-of-Speech (PoS)**.

PoS Tagging is the process of assigning a grammatical tag (like noun, verb, adjective, etc.) to each word in a sentence.

Example:

- **Sentence:** The cat sat on the mat.
- **Tagged Sentence:** The/**DT** cat/**NN** sat/**VBD** on/**IN** the/**DT** mat/**NN**.

Common PoS Tags (Penn Treebank Tagset):

- **NN:** Noun, singular
- **NNS:** Noun, plural
- **VB:** Verb, base form
- **VBD:** Verb, past tense
- **JJ:** Adjective
- **DT:** Determiner
- **IN:** Preposition
- **PRP:** Personal pronoun

Methods for PoS Tagging

1. Rule-based Tagging

This method uses a set of hand-crafted linguistic rules to assign PoS tags. A two-stage architecture is common:

1. **Dictionary Lookup:** Assign possible tags to each word from a dictionary.
2. **Rule Application:** Use rules to disambiguate and choose the correct tag.

Example Rule: "If a word ends in '-ing' and the previous word is a form of 'be', tag it as a present participle (VBG)."

- **Pros:** High precision, captures specific linguistic knowledge.
- **Cons:** Low recall, requires extensive manual effort, difficult to maintain.

2. Stochastic (Probabilistic) Tagging

This is the most common approach. It uses a trained probabilistic model to find the most likely sequence of tags for a given sentence. The goal is to find the tag sequence T=(t_1,t_2,dots,t_n) that maximizes P(T|W), where W is the word sequence.

Using Bayes' Theorem, this is equivalent to maximizing:

$P(T|W) \propto P(W|T) \times P(T)$

- P(W|T): **Emission Probability** (Probability of a word given a tag).
- P(T): **Transition Probability** (Probability of a tag sequence).

This approach forms the basis of HMM taggers.

3. Transformation-based Tagging (Brill Tagger)

This is an error-driven learning method that combines aspects of rule-based and stochastic approaches.

Process:

1. **Initial Tagging:** Start by assigning each word its most frequent tag from the training corpus.
2. **Learn Transformation Rules:** The system compares the initial tags with the correct tags (ground truth) and learns a set of transformation rules that best fix the errors.
3. **Iterative Application:** The learned rules are applied sequentially to improve the tagging accuracy.

Example Rule: "Change the tag from **NN** (noun) to **VB** (verb) if the previous word is 'to'."

- **Pros:** Rules are learned automatically and are often easy to understand. It's more compact than stochastic models.
- **Cons:** Can be slower than HMM-based taggers during tagging.

Issues in PoS Tagging

1. **Ambiguity:** Many words can have multiple PoS tags depending on the context. For example, "book" can be a noun ("read the **book**") or a verb ("**book** a flight").
2. **Unknown Words:** Models need a strategy to handle words not seen in the training data (out-of-vocabulary words). This is often done by looking at word features like prefixes, suffixes, and capitalization.
3. **Tagset Granularity:** The choice of tagset can impact performance. A very fine-grained tagset is more descriptive but harder to learn accurately.

Hidden Markov and Maximum Entropy Models

These are two powerful statistical models used extensively for sequence labeling tasks like PoS tagging.

Hidden Markov Models (HMM)

An HMM is a statistical model where the system being modeled is assumed to be a Markov process with **unobserved (hidden)** states. For PoS tagging, the **words are the observations**, and the **PoS tags are the hidden states**.

An HMM is defined by:

1. **States (Q):** A set of N hidden states (the PoS tags).
2. **Observations (O):** A set of M possible observations (the vocabulary).
3. **Transition Probabilities (A):** The probability of moving from one state (tag) to another. A=a_ij=P(t_j|t_i).
4. **Emission Probabilities (B):** The probability of an observation (word) being generated from a state (tag). B=b_j(k)=P(w_k|t_j).
5. **Initial State Probabilities (pi):** The probability of starting in a particular state. pi_i=P(t_itextatstart).

Diagram of an HMM for PoS Tagging:

```
   Transition Probabilities (A)

    <------>

  t(i-1) ------> t(i) ------> t(i+1)    <-- HIDDEN STATES (TAGS)

    |          |          |

    |          |          |

 Emission Probabilities (B)

    |          |          |

    V          V          V

  w(i-1)       w(i)       w(i+1)    <-- OBSERVATIONS (WORDS)
```

Goal: Given a sequence of observations (words), find the most likely sequence of hidden states (tags). This is solved efficiently using the **Viterbi Algorithm**.

Maximum Entropy Models (MaxEnt)

A Maximum Entropy (MaxEnt) model, also known as a log-linear model, is a **discriminative** model, unlike the **generative** HMM. It doesn't model the joint probability P(T,W); instead, it directly models the conditional probability P(T|W).

Core Principle: Of all the models that fit the training data, choose the one that makes the fewest assumptions—that is, the one with the **maximum entropy**.

Key Components:

- **Features:** A feature is a binary function f(c,d) that connects a context c (information about the word and its surroundings) with a decision d (the tag).
  - **Example Feature:** f_1(c,d)=1 if current_word is "run" and d is "VB"; 0 otherwise.
  - **Example Feature:** f_2(c,d)=1 if previous_tag is "DT" and d is "NN"; 0 otherwise.
- **Weights (lambda):** Each feature f_i is assigned a weight λ_i which is learned during training.

The probability of a decision d given a context c is calculated as:

P(d|c)=Z(c)1exp(i∑λifi(c,d))

where Z(c) is a normalization factor that ensures all probabilities sum to 1.

Advantages over HMM:

- **Flexibility in Features:** MaxEnt allows for the inclusion of a very rich and diverse set of overlapping features (e.g., word prefixes, suffixes, capitalization, previous words, next words), which is much harder to incorporate into a standard HMM. This often leads to higher accuracy.