Language Models

Dereje T. Abzaw

Recap

- Word Normalization
- Case Folding
- Stemming
- Lemmatization

Outline

- Tokenization again
- N-Gram Language Model
- Introduction to Language Models Evaluation and Perplexity

Let’s get back to Tokenization, Once more …

- In NLP, tokenization is a particular kind of document segmentation.
- Segmentation breaks up text into smaller chunks or segments, with more focused information content.
- Segmentation can include breaking a
    - Document into paragraphs,
    - Paragraphs into sentences,
    - Sentences into phrases, or
    - Phrases into tokens (usually words) and punctuation.

Contd …

Building Blocks of NLP	**Computer Language Equivalence**
Tokenizer	Scanner, Lexer, Lexical Analyzer
Vocabulary	Lexicon
Parser	Compiler
Token, Term, Word, N-gram	Symbol / Terminal Symbol

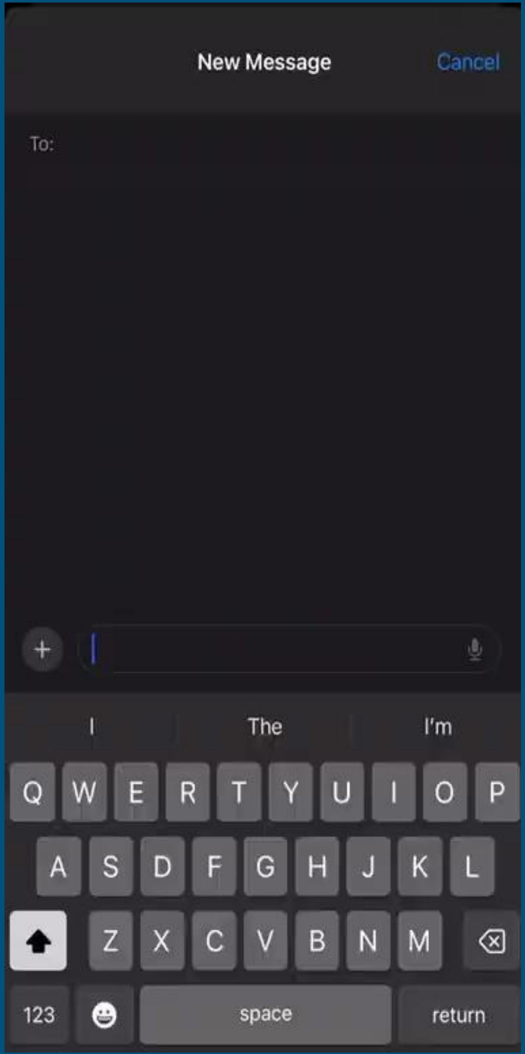
Contd ...

- Tokenization is the first step in an NLP pipeline, so it can have a big impact on the rest of your pipeline.
- A tokenizer breaks unstructured data, natural language text, into chunks of information that can be counted as discrete elements.
- What do you think is the simplest way to tokenize a sentence?
    - Use **whitespace** within a string as the “delimiter” of word
- In Python, we have a standard library to achieve that
    - **split()**

N-Gram Language Models

- Predicting is difficult
- Predicting something that seems much easier, like the next few words someone is going to say
- How do you think the following ideas work?
  - Suggestions in Messengers
  - Spelling Correction
  - Email Reply Suggestions
- What do you expect after the following sentence

  Eg. “Have you turned in …”


New Message
Cancel
To:
I
The
I'm
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
123
space
return

Contd ...

- Generally, we formalize this intuition by introducing models that assign a **probability** to each possible next word.
- Models that assign probabilities to upcoming words, or sequences of words in general, are called l**anguage models** or **LMs**.
- Language models can also assign a probability to an **entire sentence.**
- Why do we need to predict words or even sentences?
  - It is to choose **better, over less-appropriate**.

Contd ...

- N-Gram is the simplest language model
- It is a sequence of n words:
  - 2-gram (**bigram**) - a two-word sequence
  - 3-gram (**trigram**) - a three-word sequence
- Generally, “n-gram” is to mean a probabilistic model that can estimate the probability of a word given the n-1 previous words, and thereby assign probabilities to sequences.

N-Grams Explained

- Word Counts
- Estimating Probabilities
- Chain Rule
- Conditional Probability
- Markov Assumption - Markov Models
- Maximum Likelihood Estimation (MLE)

Evaluating Language Models

- The best way to evaluate the performance of a language model is to embed it in an application and measure **how much the application improves**.
- Such end-to-end evaluation is called **extrinsic evaluation**.
- Extrinsic evaluation is the only way to know if a particular improvement in the language model (or any component) is really going to help the task at hand.
- An Objective Evaluation is needed

Contd ...

- Unfortunately, running big NLP systems end-to-end is often very expensive.
- Instead, it’s helpful to have a metric that can be used to quickly evaluate potential improvements in a language model.
- An **intrinsic evaluation** metric is one that measures the quality of a model **independent of any application - Concept of Perplexity.**

Extrinsic Evaluation

- In order to evaluate any machine learning model, we need to have at least three distinct datasets:
  - Training,
  - Development,
  - Test Sets

Contd …

- The training set is the data we use to learn the parameters of our model
- The test set is a different, held-out set of data, not overlapping with the training set, that we use to evaluate the model.
- We need a separate test set to give us an **unbiased estimate** of how well the model we trained can generalize when we apply it to some new unknown dataset.
- We measure the quality of an n-gram model by its performance on **unseen test set**

How should we choose sets?

➔ The test set should reflect the language we want to use the model for:

- If we’re going to use our language model for speech recognition of chemistry lectures, the test set should be text of **chemistry lectures.**
- If we’re going to use it as part of a system for translating hotel booking requests from Chinese to English, the test set should be text of **hotel booking requests.**
- If we want our language model to be general purpose, then the test test should be drawn from a **wide variety of texts.**

Fitting the test set…

- If we are given a corpus of text and want to compare the performance of two different n-gram models, we divide the data into training and test sets, and train the parameters of both models on the **training set**. We can then compare how well the two trained models **fit the test** set.
- whichever language model assigns a higher probability to the test set means it more accurately predicts the test set , so it is a **better model**

Intrinsic Evaluation

- In practice we don’t use raw probability as our metric for evaluating language models, but a function of probability called **perplexity**.
- Perplexity is used to evaluate neural language models.
- The perplexity (sometimes abbreviated as PP or PPL) of a language model on a test set is the **inverse probability** of the test set , normalized by the number of words.
- For this reason it’s sometimes called the **per-word perplexity**.

Reading Assignment

- Read further on the concept of **Overfitting**
- “Training on Test Set”
- Development Set
- Perplexity