ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY

CENG 463: Introduction to Natural Language Processing
N-grams and Language Models

Asst. Prof. Cagri Toraman
Computer Engineering Department
ctoraman@ceng.metu.edu.tr

14.10.2025

Words

Word Type: Distinct words in a corpus (vocabulary size)

Word Token: Total number of words or subwords in text input

Example:
“Middle East Technical University is in the Middle East”

Word types: 7
Word tokens: 9

Text Normalization: Preprocessing

Before processing in an algorithm/model, text data should be preprocessed.

Why?

Common preprocessing phases:

- Tokenization
- Normalization
- Segmentation

Text Normalization: Tokenization

Splitting minimal text units in a given text input.

Example (split by whitespace):

“University is a living city”

“University” “is” “a” “living” “city”

Can you find a better tokenization method?

What about other languages?

Text Normalization: Tokenization

Deep learning and Transformer-based models utilize overlapping text sequences as minimal text units.

We will study them later (weeks of LLMs). Some important algorithms are:

- BPE
- WordPiece

Text Normalization: Normalization

Tokens are normalized to a single form.

Example:

“FB” and “Fenerbahçe”

Case folding (lower case) is a basic method to achieve text normalization.

Is it enough to apply only case folding?

Text Normalization: Normalization

Lemmatization is another method to normalize text.

Example:
“dog” “dogs” “dog’s”
“improve” “improving” “improvement”

* Stemming is more efficient approach to get the root.

Text Normalization: Segmentation

Splitting long text into segments (mostly sentences or chunks).

Example:

“Call me. Call me, when you return to Ankara.”

“Call me.” “Call me,” “when you return to Ankara.”

What about punctuation marks?

N-grams: Next Word Prediction

“Call me. Call me, when you return to …

“ Fenerbahçe has 28 …

How do you predict? Common/syntax/domain knowledge?

N-grams: Next Word Prediction

Next word prediction based on statistics (probability of a text sequence)

Language Model (LM):

Use previous words in a text sequence to predict next word

N-grams: Next Word Prediction

How to train LM?

Vocabulary (Dictionary): A set of text units (“words”) that define the language

Corpus (Corpora): A collection of (online) documents including text and speech

N-grams: Next Word Prediction

Reminder:

Sentence: Unit of written language

Word form: the inflected form as it actually appears in the corpus

Lemma: Abstract form, shared by word forms having the same stem, part of speech, and word sense

Types: Distinct words in a corpus

Tokens: Total number of “words”

N-grams: Next Word Prediction

Question: How likely is the word “Ankara” to follow the word “to”?

Given that language has T word types.

Any ideas?

N-grams: Next Word Prediction

Question: How likely is the word “Ankara” to follow the word “to”?

Given that language has T word types.

i) Probability that next word is “Ankara”: 1/T

N-grams: Next Word Prediction

Question: How likely is the word “Ankara” to follow the word “to”?

Given that language has T word types.

Markov Assumption: The **probability** of the next element in a sequence only **depending** on a **limited history** of the sequence

ii) Probability that next word is “Ankara”:

1/P where P is estimated based on frequency of word occurrence from corpus (unigram model)

N-grams: Next Word Prediction

<u>Question</u>: How likely is the word “Ankara” to follow the word “to”?

Given that language has T word types.

iii) Probability that next word is “Ankara”:

1/P where P is estimated based on frequency of word occurrence conditioned on previous words from corpus (n-gram model: bigram, trigram, …)

How much data is required for each solution?

Probabilistic Language Models

W: "Call me, when you return to …"

Goal: Compute the probability of a sentence or sequence of words:

$P(W) = P(w_1,w_2,w_3,w_4,w_5 \ldots w_n)$

Related task: probability of an upcoming word:

$P(w_5|w_1,w_2,w_3,w_4)$

A model that computes either of these:

$P(W)$ or $P(w_n|w_1,w_2 \ldots w_{n-1})$ is called a **language model**.

Probabilistic Language Models

```
<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>
```

Here are the calculations for some of the bigram probabilities from this corpus

$P(\texttt{I}|\texttt{<s>}) = \frac{2}{3} = 0.67$ $P(\texttt{Sam}|\texttt{<s>}) = \frac{1}{3} = 0.33$ $P(\texttt{am}|\texttt{I}) = \frac{2}{3} = 0.67$

$P(\texttt{</s>}|\texttt{Sam}) = \frac{1}{2} = 0.5$ $P(\texttt{Sam}|\texttt{am}) = \frac{1}{2} = 0.5$ $P(\texttt{do}|\texttt{I}) = \frac{1}{3} = 0.33$

Probabilistic Language Models

<u>The Chain Rule:</u>

$P(x_1,x_2,x_3,\dots,x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1,x_2)\dots P(x_n|x_1,\dots,x_{n-1})$

P("its water is so transparent") =

P(its) × P(water|its) × P(is|its water)

× P(so|its water is) × P(transparent|its water is so)

What can be the main bottlenecks when you estimate P(transparent | its, water, so) ?

Probabilistic Language Models

The Chain Rule:

$$P(x_1,x_2,x_3,\ldots,x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1,x_2)\ldots P(x_n|x_1,\ldots,x_{n-1})$$

P("its water is so transparent") =

P(its) × P(water|its) × P(is|its water)

× P(so|its water is) × P(transparent|its water is so)

What can be the main bottlenecks?

Estimating P(transparent | its, water, so)?
Very small P(.) values?
Efficiency concerns?

Probabilistic Language Models

How to train your N-gram Language Model?

Use a corpus!

What about the characteristics of this corpus?

Noisy?
Small/Large?

Probabilistic Language Models

How to train your N-gram Language Model?

Your probabilistic language model reflects your corpus knowledge, it has so many biases!

“Unseen words”, “domain shift”, and “testing/evaluation” are other issues to handle!

Probabilistic Language Models

Unseen words:

What if your corpus does not involve “Ankara” at all?

Smoothing:

$$P_{\text{Laplace}}(w_i) = \frac{c_i + 1}{N + V}$$

Laplace Smoothing

$$P^*_{\text{Add-k}}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n) + k}{C(w_{n-1}) + kV}$$

Add-k Smoothing

$$\begin{aligned}\hat{P}(w_n|w_{n-2}w_{n-1}) &= \lambda_1 P(w_n) \\ &+ \lambda_2 P(w_n|w_{n-1}) \\ &+ \lambda_3 P(w_n|w_{n-2}w_{n-1})\end{aligned}$$

Interpolation

Probabilistic Language Models

Domain shift:

What happened when you train your language model with Wikipedia but test in Twitter?

Probabilistic Language Models

Evaluation:

Need to measure the ability of language model to comprehend a natural language

Can be evaluated on a test set using “perplexity”

Perplexity of a language model on a test set is the inverse probability of the test set (one over the probability of the test set), normalized by the number of words (or tokens). We normalize by the number of words *N* by taking the *N*th root.

$$\begin{aligned}\text{perplexity}(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\ &= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}\end{aligned}$$

Probabilistic Language Models

$$\begin{aligned}\text{perplexity}(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\ &= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}\end{aligned}$$

By using the Chain Rule:

$$\text{perplexity}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \ldots w_{i-1})}}$$

The lower the perplexity of a model on the data, the better the model!

Probabilistic Language Models

$$\text{perplexity}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \ldots w_{i-1})}}$$

$$\text{perplexity}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i)}}$$

For unigram

Probabilistic Language Models

$$\text{perplexity}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i)}}$$

$L = \{\texttt{red}, \texttt{blue}, \texttt{green}\}$

test set T = “`red red red red blue`”

$$\begin{aligned}\text{perplexity}_A(T) &= P_A(\texttt{red red red red blue})^{-\frac{1}{5}} \\ &= \left(\left(\frac{1}{3}\right)^5\right)^{-\frac{1}{5}} \\ &= \left(\frac{1}{3}\right)^{-1} = 3\end{aligned}$$

What if P(red)=0.8 and others=0.1?

ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY

Thanks for your participation!

Çağrı Toraman
14.10.2025