NLP – N-Gram Language Modeling

Winter Semester 2023/2024

October 19, 2023

Prof. Dr.-Ing. Christian Bergler, Prof. Dr. Patrick Levi | OTH Amberg-Weiden

POS, NER, BoW, and TF-IDF

Short Recap...

Ostbayerische Technische Hochschule Amberg-Weiden

Part-Of-Speech (POS)

- Assigning language-related grammar- and word-specific “roles” (part-of-speech tags, e.g. noun, verb, adjective, etc.) to individual words in order to derive syntactic structures, essential for text understanding

POS, NER, BoW, and TF-IDF

Short Recap...

Part-Of-Speech (POS)

- Assigning language-related grammar- and word-specific “roles” (part-of-speech tags, e.g. noun, verb, adjective, etc.) to individual words in order to derive syntactic structures, essential for text understanding

Named Entity Recognition (NER)

- Identification of “named entities” (predefined categories, e.g. locations, persons, organizations, etc.) together with the respective text-based morpheme/word assignment

→ Both methods realize a contextual summarization and lead to a (categorical) reduction of the original textual complexity

POS, NER, BoW, and TF-IDF

Short Recap...

Bag-of-Words (BoW)

- Determination of the vocabulary and associated word frequencies across a set of documents (text pieces of varying size, e.g. paragraphs, single pages), which results in a matrix representation (documents $\times$ vocabulary size) denoting individual word counts

POS, NER, BoW, and TF-IDF
Short Recap...

Bag-of-Words (BoW)

- Determination of the vocabulary and associated word frequencies across a set of documents (text pieces of varying size, e.g. paragraphs, single pages), which results in a matrix representation (documents $\times$ vocabulary size) denoting individual word counts

Term Frequency-Inverse Document Frequency (TF-IDF)

- Identification of all the word-specific frequencies, referred to as "term frequency" (TF), in addition to the number of occurrences per word across all the documents, while the word importance decreases with an increasing cross-document appearance

POS, NER, BoW, and TF-IDF

Short Recap...

Bag-of-Words (BoW)

- Determination of the vocabulary and associated word frequencies across a set of documents (text pieces of varying size, e.g. paragraphs, single pages), which results in a matrix representation (documents $\times$ vocabulary size) denoting individual word counts

Term Frequency-Inverse Document Frequency (TF-IDF)

- Identification of all the word-specific frequencies, referred to as "term frequency" (TF), in addition to the number of occurrences per word across all the documents, while the word importance decreases with an increasing cross-document appearance
- Inverse Document Frequency (IDF) $= log(\frac{1+N}{1+df(word)}) + 1$ with N as the number of documents and $df(word)$ as the word-specific document frequency $\rightarrow$ TF $\times$ IDF

BoW, and TF-IDF

Shortcomings and Downsides

BoW and TF-IDF

- Both concepts rely on vocabulary-related word frequencies (unweighted & weighted)

BoW, and TF-IDF
Shortcomings and Downsides

BoW and TF-IDF

- Both concepts rely on vocabulary-related word frequencies (unweighted & weighted)
- By default, no contextual (surrounding) word information → Single-word approach

BoW, and TF-IDF
Shortcomings and Downsides

BoW and TF-IDF

- Both concepts rely on vocabulary-related word frequencies (unweighted & weighted)
- By default, no contextual (surrounding) word information → Single-word approach
- Plain word frequency-based techniques are often used for "sentiment analysis" & "topic recognition", however with a lot of space for improvements:
  - "The football game of FC Bayern Munich was great and not boring" → Positive statement
  - "The football game of FC Bayern Munich was not great and boring" → Negative statement

→ Identical vocabulary & word count!

BoW, and TF-IDF

Shortcomings and Downsides

BoW and TF-IDF

- Both concepts rely on vocabulary-related word frequencies (unweighted & weighted)
- By default, no contextual (surrounding) word information → Single-word approach
- Plain word frequency-based techniques are often used for “sentiment analysis” & ”topic recognition“, however with a lot of space for improvements:
  - “The football game of FC Bayern Munich was great and not boring” → Positive statement
  - “The football game of FC Bayern Munich was not great and boring” → Negative statement

  → Identical vocabulary & word count!
- Position and contextual information in text is often very important, e.g. speech recognition, machine translation, spell correction, question answering, summarization, ...

Scope of this Lecture...

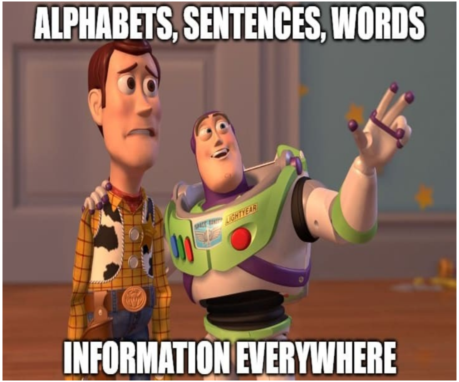
“What is next?”

“Let’s assign a probability to a sentence...”

- “Probability of a sentence”?

  → How likely is it, that this sentence occurs in reality (natural language) !

  → Contextual information involved!



Source: https://www.linkedin.com/pulse/natural-language-processing-begin-learning-naturally-kumar

Scope of this Lecture...

“What is next?”

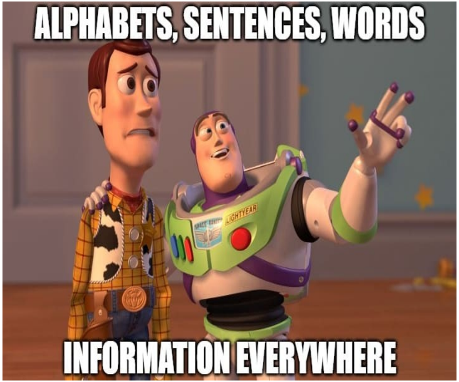
“Let’s assign a probability to a sentence...”

- “Probability of a sentence”?

  → How likely is it, that this sentence occurs in reality (natural language) !

  → Contextual information involved!

- Machine Translation
  - $P(\text{mixed martial arts}) > P(\text{varied martial arts})$



Source: https://www.linkedin.com/pulse/natural-language-processing-begin-learning-naturally-kumar

Scope of this Lecture...

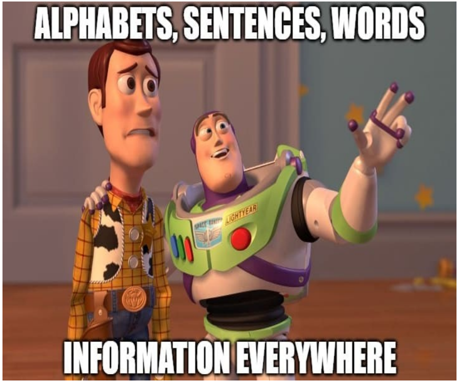
“What is next?”

“Let’s assign a probability to a sentence...”

- “Probability of a sentence”?

  $\rightarrow$ How likely is it, that this sentence occurs in reality (natural language) !

  $\rightarrow$ Contextual information involved!
- Machine Translation
  - $P(\text{mixed martial arts}) > P(\text{varied martial arts})$
- Spell Correction
  - $P(\text{i enjoy deep learning}) > P(\text{i enjoy diip learning})$



Source: https://www.linkedin.com/pulse/natural-language-processing-begin-learning-naturally-kumar

Scope of this Lecture...

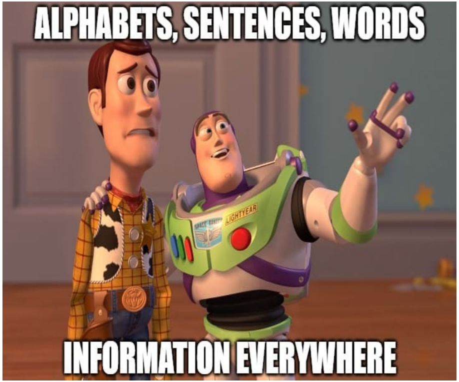
"What is next?"

"Let's assign a probability to a sentence..."

- "Probability of a sentence"?

  $\rightarrow$ How likely is it, that this sentence occurs in reality (natural language) !

  $\rightarrow$ Contextual information involved!
- Machine Translation
  - P(mixed martial arts) $>$ P(varied martial arts)
- Spell Correction
  - P(i enjoy deep learning) $>$ P(i enjoy diip learning)
- Automatic Speech Recognition (ASR)
  - P(ready for robotics) $>>$ P(eddy four optics)

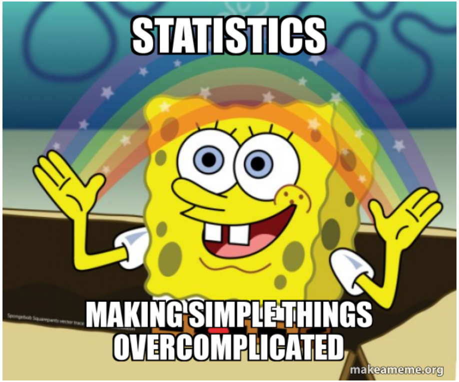


Source: https://www.linkedin.com/pulse/natural-language-processing-begin-learning-naturally-kumar

Probabilistic Language Modeling

We need Statistics!

Goal: Calculate probability of a sequence of words (= Sentence)



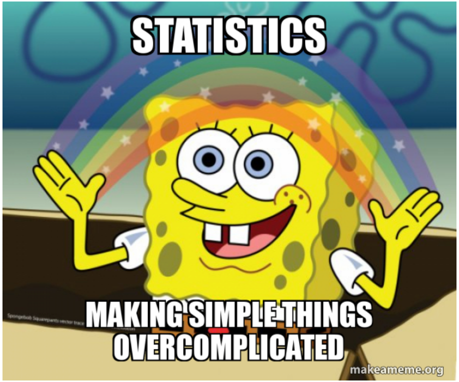
Source: https://makeameme.org/meme/statistics-making-simple

Probabilistic Language Modeling

We need Statistics!

Goal: Calculate probability of a sequence of words (= Sentence)

- Joint probability: $P(\vec{w}) = P(w_1, w_2, w_3, ..., w_m)$ or something related/similar



Source: https://makeameme.org/meme/statistics-making-simple

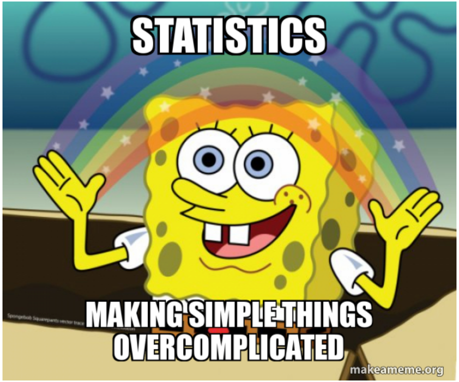
Probabilistic Language Modeling

We need Statistics!

Goal: Calculate probability of a sequence of words (= Sentence)

- Joint probability: $P(\vec{w}) = P(w_1, w_2, w_3, ..., w_m)$ or something related/similar
- Conditional probability: $P(w_m | w_1, w_2, w_3, ..., w_{m-1})$

$\rightarrow$ A trained model computing one of these probabilities P is called Language Model (LM) (also referred to as Grammar)



Source: https://makeameme.org/meme/statistics-making-simple

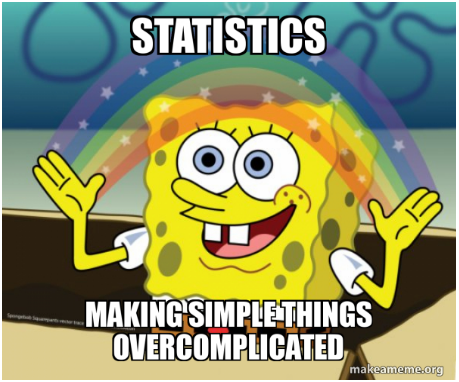
Probabilistic Language Modeling

We need Statistics!

Goal: Calculate probability of a sequence of words (= Sentence)

- Joint probability: $P(\vec{w}) = P(w_1, w_2, w_3, ..., w_m)$ or something related/similar
- Conditional probability: $P(w_m | w_1, w_2, w_3, ..., w_{m-1})$

$\rightarrow$ A trained model computing one of these probabilities P is called Language Model (LM) (also referred to as Grammar)



$\rightarrow P(\vec{w}) = P(\text{How, do, I, compute, this, probability})$???

Source: https://makeameme.org/meme/statistics-making-simple

Probabilistic Language Modeling

Even more Statistics!

Chain Rule...

- Recall the definition of:

Probabilistic Language Modeling

Even more Statistics!

Chain Rule...

- Recall the definition of:
  - Conditional Probability: $P(W|Y) = \frac{P(Y,W)}{P(Y)}$

Probabilistic Language Modeling

Even more Statistics!

Chain Rule...

- Recall the definition of:
  - Conditional Probability: $P(W|Y) = \frac{P(Y,W)}{P(Y)}$
  - Joint Probability: $P(Y, W) = P(W|Y)P(Y) = P(Y|W)\,P(W)$

Probabilistic Language Modeling
Even more Statistics!

Chain Rule...

- Recall the definition of:
  - Conditional Probability: $P(W|Y) = \frac{P(Y,W)}{P(Y)}$
  - Joint Probability: $P(Y,W) = P(W|Y)P(Y) = P(Y|W)\,P(W)$
  - Bayes Theorem: $P(W|Y) = \frac{P(Y|W)\,P(W)}{P(Y)} = \frac{P(Y,W)}{P(Y)}$

Probabilistic Language Modeling

Even more Statistics!

Chain Rule...

- Recall the definition of:
  - Conditional Probability: $P(W|Y) = \frac{P(Y,W)}{P(Y)}$
  - Joint Probability: $P(Y,W) = P(W|Y)P(Y) = P(Y|W)\,P(W)$
  - Bayes Theorem: $P(W|Y) = \frac{P(Y|W)\,P(W)}{P(Y)} = \frac{P(Y,W)}{P(Y)}$
  - Posterior $P(W|Y)$, Likelihood $P(Y|W)$, Prior $P(W)$, Evidence $P(Y)$

Probabilistic Language Modeling

Even more Statistics!

Chain Rule...

- Recall the definition of:
  - Conditional Probability: $P(W|Y) = \frac{P(Y,W)}{P(Y)}$
  - Joint Probability: $P(Y,W) = P(W|Y)P(Y) = P(Y|W)\,P(W)$
  - Bayes Theorem: $P(W|Y) = \frac{P(Y|W)\,P(W)}{P(Y)} = \frac{P(Y,W)}{P(Y)}$
  - Posterior $P(W|Y)$, Likelihood $P(Y|W)$, Prior $P(W)$, Evidence $P(Y)$
- Automatic Speech Recognition: Acoustic Model $P(Y|W)$, Language Model $P(W)$

Probabilistic Language Modeling

Even more Statistics!

Chain Rule...

- Recall the definition of:
  - Conditional Probability: $P(W|Y) = \frac{P(Y,W)}{P(Y)}$
  - Joint Probability: $P(Y,W) = P(W|Y)P(Y) = P(Y|W)\,P(W)$
  - Bayes Theorem: $P(W|Y) = \frac{P(Y|W)\,P(W)}{P(Y)} = \frac{P(Y,W)}{P(Y)}$
  - Posterior $P(W|Y)$, Likelihood $P(Y|W)$, Prior $P(W)$, Evidence $P(Y)$
- Automatic Speech Recognition: Acoustic Model $P(Y|W)$, Language Model $P(W)$
- More Variables: $P(w_1, w_2, w_3, w_4) = P(w_1)\,P(w_2|w_1)\,P(w_3|w_1, w_2)\,P(w_4|w_1, w_2, w_3)$

Probabilistic Language Modeling

Even more Statistics!

Chain Rule...

- Recall the definition of:
  - Conditional Probability: $P(W|Y) = \frac{P(Y,W)}{P(Y)}$
  - Joint Probability: $P(Y,W) = P(W|Y)P(Y) = P(Y|W)\,P(W)$
  - Bayes Theorem: $P(W|Y) = \frac{P(Y|W)\,P(W)}{P(Y)} = \frac{P(Y,W)}{P(Y)}$
  - Posterior $P(W|Y)$, Likelihood $P(Y|W)$, Prior $P(W)$, Evidence $P(Y)$
- Automatic Speech Recognition: Acoustic Model $P(Y|W)$, Language Model $P(W)$
- More Variables: $P(w_1, w_2, w_3, w_4) = P(w_1)\,P(w_2|w_1)\,P(w_3|w_1, w_2)\,P(w_4|w_1, w_2, w_3)$
- General Chain Rule (Sentence $= \hat{w}$):

$$P(\vec{w}) = P(w_1, w_2, w_3, \ldots, w_m) = P(w_1)\,P(w_2|w_1)\,P(w_3|w_1, w_2)\ldots P(w_m|w_1, \ldots, w_{m-1})$$

Probabilistic Language Modeling

Chain Rule to Compute Joint Word Probability!

Ostbayerische Technische Hochschule Amberg-Weiden

Chain Rule...

- $P(\vec{w}) = P(w_1, w_2, w_3, ..., w_m) = \prod_i P(w_i | w_1, w_2, ..., w_{i-1})$

Probabilistic Language Modeling

Chain Rule to Compute Joint Word Probability!

Chain Rule...

- $P(\vec{w}) = P(w_1, w_2, w_3, ..., w_m) = \prod_i P(w_i | w_1, w_2, ..., w_{i-1})$
- $P(\text{How, do, I, compute, this, probability}) =$

  $P(\text{How}) \times$
  $P(\text{do} \mid \text{How}) \times$
  $P(\text{I} \mid \text{How ,do}) \times$
  $P(\text{compute} \mid \text{How, do, I}) \times$
  $P(\text{this} \mid \text{How, do, I, compute}) \times$
  $P(\text{probability} \mid \text{How, do, I, compute, this})$

Probabilistic Language Modeling

Chain Rule to Compute Joint Word Probability!

Chain Rule...

- $P(\vec{w}) = P(w_1, w_2, w_3, ..., w_m) = \prod_i P(w_i | w_1, w_2, ..., w_{i-1})$
- $P(\text{How, do, I, compute, this, probability}) =$

  $P(\text{How}) \times$
  $P(\text{do} \mid \text{How}) \times$
  $P(\text{I} \mid \text{How ,do}) \times$
  $P(\text{compute} \mid \text{How, do, I}) \times$
  $P(\text{this} \mid \text{How, do, I, compute}) \times$
  $P(\text{probability} \mid \text{How, do, I, compute, this})$
- How to calculate these probabilities?

Probabilistic Language Modeling

Chain Rule to Compute Joint Word Probability!

Ostbayerische Technische Hochschule Amberg-Weiden

Chain Rule...

- $P(\vec{w}) = P(w_1, w_2, w_3, ..., w_m) = \prod_i P(w_i | w_1, w_2, ..., w_{i-1})$
- $P(\text{How, do, I, compute, this, probability}) =$

  $P(\text{How}) \times$
  $P(\text{do} \mid \text{How}) \times$
  $P(\text{I} \mid \text{How ,do}) \times$
  $P(\text{compute} \mid \text{How, do, I}) \times$
  $P(\text{this} \mid \text{How, do, I, compute}) \times$
  $P(\text{probability} \mid \text{How, do, I, compute, this})$
- How to calculate these probabilities?
- By-the-way: How about removing stop words or other words with less information value?

Probabilistic Language Modeling

How to calculate these probabilities?

- Calculating relative frequencies $\rightarrow$ Probability estimation

Probabilistic Language Modeling

How to calculate these probabilities?

- Calculating relative frequencies $\rightarrow$ Probability estimation
- $P(x_i) = \frac{C(x_i)}{M}$ with $C(x_i)$ as count of event x_i and M as the total number of events/items in the dataset

  $\rightarrow$ Maximum-Likelihood Estimation (MLE)

Probabilistic Language Modeling

How to calculate these probabilities?

- Calculating relative frequencies $\rightarrow$ Probability estimation
- $P(x_i) = \frac{C(x_i)}{M}$ with $C(x_i)$ as count of event x_i and M as the total number of events/items in the dataset

  $\rightarrow$ Maximum-Likelihood Estimation (MLE)
- Transfer to a single word ($\vec{w}_{1\times 1}$) or word-phrase/sentence ($\vec{w}_{1\times M}$) $\rightarrow P(\vec{w}) = \frac{C(\vec{w})}{C(\vec{w})_{ref}}$ with $C(\vec{w})$ and $C(\vec{w})_{ref}$ as total word/sequence and reference count

Probabilistic Language Modeling

How to calculate these probabilities?

Ostbayerische Technische Hochschule Amberg-Weiden

- Calculating relative frequencies $\rightarrow$ Probability estimation
- $P(x_i) = \frac{C(x_i)}{M}$ with $C(x_i)$ as count of event x_i and M as the total number of events/items in the dataset

  $\rightarrow$ Maximum-Likelihood Estimation (MLE)
- Transfer to a single word ($\vec{w}_{1\times 1}$) or word-phrase/sentence ($\vec{w}_{1\times M}$) $\rightarrow P(\vec{w}) = \frac{C(\vec{w})}{C(\vec{w})_{ref}}$ with $C(\vec{w})$ and $C(\vec{w})_{ref}$ as total word/sequence and reference count
- Single word: $P(\vec{w}_{1\times 1}) = P(\text{How}) = \frac{C(\vec{w})}{C(\vec{w})_{ref}} = \frac{\#(How)}{|V|}$ with $|V|$ as the vocabulary size

Probabilistic Language Modeling

How to calculate these probabilities?

- Calculating relative frequencies $\rightarrow$ Probability estimation
- $P(x_i) = \frac{C(x_i)}{M}$ with $C(x_i)$ as count of event x_i and M as the total number of events/items in the dataset

  $\rightarrow$ Maximum-Likelihood Estimation (MLE)
- Transfer to a single word ($\vec{w}_{1\times 1}$) or word-phrase/sentence ($\vec{w}_{1\times M}$) $\rightarrow P(\vec{w}) = \frac{C(\vec{w})}{C(\vec{w})_{ref}}$ with $C(\vec{w})$ and $C(\vec{w})_{ref}$ as total word/sequence and reference count
- Single word: $P(\vec{w}_{1\times 1}) = P(\text{How}) = \frac{C(\vec{w})}{C(\vec{w})_{ref}} = \frac{\#(How)}{|V|}$ with $|V|$ as the vocabulary size
- Phrase/Sentence: $P(\vec{w}_{1\times M}) = P(\text{this} \mid \text{How, do, I, compute}) = \frac{\#(How\ do\ I\ compute\ this)}{\#(How\ do\ I\ compute)}$

Probabilistic Language Modeling

How to calculate these probabilities?

Challenges

- What is the problem if the word phrase or sentence $\vec{w}_{1 \times M}$ is getting longer and longer ($M >> 1$)?
- In practice, it is simply not feasible, since there are too many possibilities (combinatorial diversity), particularly with a growing word phrase/sentence size $\rightarrow$ Sparse Data!
- Never enough training material to observe all of them in significant large numbers
- What happens to events which are never seen in training?

Probabilistic Language Modeling

Markov Assumption

- Simplification of $P(\vec{w}) = P(w_1, w_2, w_3, ..., w_m) = \prod_i P(w_i | w_1, w_2 \dots w_{i-1})$

Probabilistic Language Modeling

Markov Assumption

- Simplification of $P(\vec{w}) = P(w_1, w_2, w_3, ..., w_m) = \prod_i P(w_i | w_1, w_2 \dots w_{i-1})$
- Markov Assumption: $P(\vec{w}) = P(w_1, w_2, w_3, ..., w_m) \approx \prod_i P(w_i | w_{i-n}, ..., w_{i-1})$, leading to an approximation of the original joint probability

Probabilistic Language Modeling

Markov Assumption

- Simplification of $P(\vec{w}) = P(w_1, w_2, w_3, ..., w_m) = \prod_i P(w_i | w_1, w_2 \dots w_{i-1})$
- Markov Assumption: $P(\vec{w}) = P(w_1, w_2, w_3, ..., w_m) \approx \prod_i P(w_i | w_{i-n}, ..., w_{i-1})$, leading to an approximation of the original joint probability
- $P(\text{probability} \mid \text{How, do, I, compute, this}) \approx P(\text{probability} \mid \text{this})$

Probabilistic Language Modeling

Markov Assumption

- Simplification of $P(\vec{w}) = P(w_1, w_2, w_3, \ldots, w_m) = \prod_i P(w_i | w_1, w_2 \ldots w_{i-1})$
- Markov Assumption: $P(\vec{w}) = P(w_1, w_2, w_3, \ldots, w_m) \approx \prod_i P(w_i | w_{i-n}, \ldots, w_{i-1})$, leading to an approximation of the original joint probability
- $P(\text{probability} \mid \text{How, do, I, compute, this}) \approx P(\text{probability} \mid \text{this})$
- Estimation of $P(\vec{w})$ based on smaller contextual information (n steps in the past)

Probabilistic Language Modeling

Markov Assumption

- Simplification of $P(\vec{w}) = P(w_1, w_2, w_3, ..., w_m) = \prod_i P(w_i | w_1, w_2 \dots w_{i-1})$
- Markov Assumption: $P(\vec{w}) = P(w_1, w_2, w_3, ..., w_m) \approx \prod_i P(w_i | w_{i-n}, ..., w_{i-1})$, leading to an approximation of the original joint probability
- $P(\text{probability} \mid \text{How, do, I, compute, this}) \approx P(\text{probability} \mid \text{this})$
- Estimation of $P(\vec{w})$ based on smaller contextual information (n steps in the past)
- How many of the previous values do I want to consider and how do I consequently choose a meaningful history n?

  $\rightarrow$ The concept of N-Grams (Markov model of order N-1) !

Probabilistic Language Modeling

Markov Assumption

- Simplification of $P(\vec{w}) = P(w_1, w_2, w_3, ..., w_m) = \prod_i P(w_i | w_1, w_2 \dots w_{i-1})$
- Markov Assumption: $P(\vec{w}) = P(w_1, w_2, w_3, ..., w_m) \approx \prod_i P(w_i | w_{i-n}, ..., w_{i-1})$, leading to an approximation of the original joint probability
- $P(\text{probability} \mid \text{How, do, I, compute, this}) \approx P(\text{probability} \mid \text{this})$
- Estimation of $P(\vec{w})$ based on smaller contextual information (n steps in the past)
- How many of the previous values do I want to consider and how do I consequently choose a meaningful history n?

$\rightarrow$ The concept of N-Grams (Markov model of order N-1) !

Predicting the probability of a particular pattern (e.g. words, morphemes), based on the history of n previous patterns ("grams")

Probabilistic Language Modeling

N-Gram

- Simplest form refers to a Unigram with a history of $n = 0$ (word counts only – see BoW & TF-IDF) $\rightarrow P(w_i | w_1, w_2, ..., w_{i-1}) \approx P(w_i)$
  - Joint probability sequence: $P(w_1, w_2, w_3, ..., w_m) \approx P(w_1) \prod_{i=1}^{m-1} P(w_{i+1})$

Probabilistic Language Modeling

N-Gram

- Simplest form refers to a Unigram with a history of $n = 0$ (word counts only – see BoW & TF-IDF) $\rightarrow P(w_i|w_1, w_2, ..., w_{i-1}) \approx P(w_i)$
  - Joint probability sequence: $P(w_1, w_2, w_3, ..., w_m) \approx P(w_1) \prod_{i=1}^{m-1} P(w_{i+1})$
- Probability is only based on the previous word $i - 1$, also referred to as Bigram, with a history of $n = 1 \rightarrow P(w_i|w_1, w_2, ..., w_{i-1}) \approx P(w_i|w_{i-1})$
  - Joint probability sequence: $P(w_1, w_2, w_3, ..., w_m) \approx P(w_1) \prod_{i=1}^{m-1} P(w_{i+1}|w_i)$

Probabilistic Language Modeling

N-Gram

- Simplest form refers to a Unigram with a history of $n = 0$ (word counts only – see BoW & TF-IDF) $\rightarrow P(w_i|w_1, w_2, ..., w_{i-1}) \approx P(w_i)$
  - Joint probability sequence: $P(w_1, w_2, w_3, ..., w_m) \approx P(w_1) \prod_{i=1}^{m-1} P(w_{i+1})$
- Probability is only based on the previous word $i - 1$, also referred to as Bigram, with a history of $n = 1 \rightarrow P(w_i|w_1, w_2, ..., w_{i-1}) \approx P(w_i|w_{i-1})$
  - Joint probability sequence: $P(w_1, w_2, w_3, ..., w_m) \approx P(w_1) \prod_{i=1}^{m-1} P(w_{i+1}|w_i)$
- Probability focuses on the two previous words $i - 1$ and $i - 2$, also referred to as Trigram, with a history of $n = 2 \rightarrow P(w_i|w_1, w_2, ..., w_{i-1}) \approx P(w_i|w_{i-2}, w_{i-1})$
  - Joint probability sequence: $P(w_1, w_2, w_3, ..., w_m) \approx P(w_1)P(w_2|w_1) \prod_{i=2}^{m-1} P(w_{i+1}|w_{i-1}, w_i)$

Probabilistic Language Modeling
N-Gram

- Simplest form refers to a Unigram with a history of $n = 0$ (word counts only – see BoW & TF-IDF) $\rightarrow P(w_i|w_1, w_2, ..., w_{i-1}) \approx P(w_i)$
  - Joint probability sequence: $P(w_1, w_2, w_3, ..., w_m) \approx P(w_1) \prod_{i=1}^{m-1} P(w_{i+1})$
- Probability is only based on the previous word $i-1$, also referred to as Bigram, with a history of $n = 1 \rightarrow P(w_i|w_1, w_2, ..., w_{i-1}) \approx P(w_i|w_{i-1})$
  - Joint probability sequence: $P(w_1, w_2, w_3, ..., w_m) \approx P(w_1) \prod_{i=1}^{m-1} P(w_{i+1}|w_i)$
- Probability focuses on the two previous words $i-1$ and $i-2$, also referred to as Trigram, with a history of $n = 2 \rightarrow P(w_i|w_1, w_2, ..., w_{i-1}) \approx P(w_i|w_{i-2}, w_{i-1})$
  - Joint probability sequence: $P(w_1, w_2, w_3, ..., w_m) \approx P(w_1)P(w_2|w_1) \prod_{i=2}^{m-1} P(w_{i+1}|w_{i-1}, w_i)$

$\rightarrow$ Higher-order N-Grams than $n = 3$ are possible, but? What trade-off has to be kept in mind (keyword: data sparsity)?

Probabilistic Language Modeling

N-Gram Assumption and Shortcomings

- N-Grams assume stochastic independence, since the probability of a pattern (e.g. word) following a sequence of previous patterns (history) is restricted to a fixed number N

Probabilistic Language Modeling

N-Gram Assumption and Shortcomings

- N-Grams assume stochastic independence, since the probability of a pattern (e.g. word) following a sequence of previous patterns (history) is restricted to a fixed number N

  $\rightarrow$ N-Gram assumption could also be misleading, e.g. Trigram – $P(\text{match}|\text{the football})$:
  - $P(\text{match}|\text{I was not hurt during the football})$
  - $P(\text{match}|\text{I try to attend the football})$
  - $P(\text{match}|\text{I scored twice during the football})$

Probabilistic Language Modeling
N-Gram Assumption and Shortcomings

- N-Grams assume stochastic independence, since the probability of a pattern (e.g. word) following a sequence of previous patterns (history) is restricted to a fixed number N

  $\rightarrow$ N-Gram assumption could also be misleading, e.g. Trigram – $P(\text{match}|\text{the football})$:
  - $P(\text{match}|\text{I was not hurt during the football})$
  - $P(\text{match}|\text{I try to attend the football})$
  - $P(\text{match}|\text{I scored twice during the football})$

- Curse of dimensionality $\rightarrow |V|^n$ with $|V|$ as the size of the vocabulary and n as the N-Gram order (not enough data!!!)

Probabilistic Language Modeling

N-Gram Assumption and Shortcomings

- N-Grams assume stochastic independence, since the probability of a pattern (e.g. word) following a sequence of previous patterns (history) is restricted to a fixed number N

  $\rightarrow$ N-Gram assumption could also be misleading, e.g. Trigram – $P(\text{match}|\text{the football})$:
  - $P(\text{match}|\text{I was not hurt during the football})$
  - $P(\text{match}|\text{I try to attend the football})$
  - $P(\text{match}|\text{I scored twice during the football})$

- Curse of dimensionality $\rightarrow |V|^n$ with $|V|$ as the size of the vocabulary and n as the N-Gram order (not enough data!!!)
- To counteract the data sparsity different NLP-techniques are used to categorize text information (lemmatization, POS, NER, etc.) $\rightarrow$ Careful, information loss – Trade off!

Probabilistic Language Modeling

N-Gram Assumption and Shortcomings

- N-Grams assume stochastic independence, since the probability of a pattern (e.g. word) following a sequence of previous patterns (history) is restricted to a fixed number N

  $\rightarrow$ N-Gram assumption could also be misleading, e.g. Trigram – $P(\text{match}|\text{the football})$:
  - $P(\text{match}|\text{I was not hurt during the football})$
  - $P(\text{match}|\text{I try to attend the football})$
  - $P(\text{match}|\text{I scored twice during the football})$

- Curse of dimensionality $\rightarrow |V|^n$ with $|V|$ as the size of the vocabulary and n as the N-Gram order (not enough data!!!)
- To counteract the data sparsity different NLP-techniques are used to categorize text information (lemmatization, POS, NER, etc.) $\rightarrow$ Careful, information loss – Trade off!
- Every unseen event (not in the training data!) has a probability of zero ("A gorilla eats hamburger" – unlikely but... we will see later how to handle this!)

Probabilistic Language Modeling

N-Gram Example

Sentence ($\vec{w}$)

- “<s> I was not able to pass this lecture module without preparation </s>” $\rightarrow P(\vec{w})$?

Probabilistic Language Modeling

N-Gram Example

Sentence ($\vec{w}$)

- "<s> I was not able to pass this lecture module without preparation </s>" $\rightarrow P(\vec{w})$?
- Unigram: $P(\vec{w}) = P(< s >) \cdot P(I) \cdot (\ldots) \cdot P(preparation) \cdot P(< s/ >)$

Probabilistic Language Modeling

N-Gram Example

Sentence ($\vec{w}$)

- "<s> I was not able to pass this lecture module without preparation </s>" $\rightarrow P(\vec{w})$?
- Unigram: $P(\vec{w}) = P(< s >) \cdot P(I) \cdot (\dots) \cdot P(preparation) \cdot P(< s/ >)$
- Bigram: $P(\vec{w}) = P(< s >) \cdot P(I| < s >) \cdot P(was|I) \cdot (\dots) \cdot P(< /s > |preparation)$

Probabilistic Language Modeling

N-Gram Example

Sentence ($\vec{w}$)

- "<s> I was not able to pass this lecture module without preparation </s>" $\rightarrow P(\vec{w})$?
- Unigram: $P(\vec{w}) = P(< s >) \cdot P(I) \cdot (\ldots) \cdot P(preparation) \cdot P(< s/ >)$
- Bigram: $P(\vec{w}) = P(< s >) \cdot P(I| < s >) \cdot P(was|I) \cdot (\ldots) \cdot P(< /s > |preparation)$

How to estimate these probabilities $P(w_i)$

Probabilistic Language Modeling

N-Gram Example

Sentence ($\vec{w}$)

- "<s> I was not able to pass this lecture module without preparation </s>" $\rightarrow P(\vec{w})$?
- Unigram: $P(\vec{w}) = P(< s >) \cdot P(I) \cdot (\ldots) \cdot P(preparation) \cdot P(< s/ >)$
- Bigram: $P(\vec{w}) = P(< s >) \cdot P(I| < s >) \cdot P(was|I) \cdot (\ldots) \cdot P(< /s > |preparation)$

How to estimate these probabilities $P(w_i)$

- Recap: $P(\vec{w}) = \frac{C(\vec{w})}{C(\vec{w})_{ref}}$ with $C(\vec{w})$ and $C(\vec{w})_{ref}$ as total word/sequence and reference count $\rightarrow$ Maximum-Likelihood Estimation (MLE)

Probabilistic Language Modeling

N-Gram Example

Sentence ($\vec{w}$)

- "<s> I was not able to pass this lecture module without preparation </s>" $\rightarrow P(\vec{w})$?
- Unigram: $P(\vec{w}) = P(< s >) \cdot P(I) \cdot (\ldots) \cdot P(preparation) \cdot P(< s/ >)$
- Bigram: $P(\vec{w}) = P(< s >) \cdot P(I| < s >) \cdot P(was|I) \cdot (\ldots) \cdot P(< /s > |preparation)$

How to estimate these probabilities $P(w_i)$

- Recap: $P(\vec{w}) = \frac{C(\vec{w})}{C(\vec{w})_{ref}}$ with $C(\vec{w})$ and $C(\vec{w})_{ref}$ as total word/sequence and reference count $\rightarrow$ Maximum-Likelihood Estimation (MLE)
- Unigram: $P(w_i) = \frac{C(w_i)}{|V|}$

Probabilistic Language Modeling

N-Gram Example

Sentence ($\vec{w}$)

- "<s> I was not able to pass this lecture module without preparation </s>" $\rightarrow P(\vec{w})$?
- Unigram: $P(\vec{w}) = P(< s >) \cdot P(I) \cdot (\ldots) \cdot P(preparation) \cdot P(< s/ >)$
- Bigram: $P(\vec{w}) = P(< s >) \cdot P(I| < s >) \cdot P(was|I) \cdot (\ldots) \cdot P(< /s > |preparation)$

How to estimate these probabilities $P(w_i)$

- Recap: $P(\vec{w}) = \frac{C(\vec{w})}{C(\vec{w})_{ref}}$ with $C(\vec{w})$ and $C(\vec{w})_{ref}$ as total word/sequence and reference count $\rightarrow$ Maximum-Likelihood Estimation (MLE)
- Unigram: $P(w_i) = \frac{C(w_i)}{|V|}$
- Bigram: $P(w_i|w_{i-1}) = \frac{C(w_{i-1}, w_i)}{C_{ref}(w_{i-1})}$

Probabilistic Language Modeling

N-Gram Example

Sentence ($\vec{w}$)

- "<s> I was not able to pass this lecture module without preparation </s>" $\rightarrow P(\vec{w})$?
- Unigram: $P(\vec{w}) = P(< s >) \cdot P(I) \cdot (\ldots) \cdot P(preparation) \cdot P(< s/ >)$
- Bigram: $P(\vec{w}) = P(< s >) \cdot P(I| < s >) \cdot P(was|I) \cdot (\ldots) \cdot P(< /s > |preparation)$

How to estimate these probabilities $P(w_i)$

- Recap: $P(\vec{w}) = \frac{C(\vec{w})}{C(\vec{w})_{ref}}$ with $C(\vec{w})$ and $C(\vec{w})_{ref}$ as total word/sequence and reference count $\rightarrow$ Maximum-Likelihood Estimation (MLE)
- Unigram: $P(w_i) = \frac{C(w_i)}{|V|}$
- Bigram: $P(w_i|w_{i-1}) = \frac{C(w_{i-1}, w_i)}{C_{ref}(w_{i-1})}$
- Trigram: $P(w_i|w_{i-2}, w_{i-1}) = \frac{C(w_{i-2}, w_{i-1}, w_i)}{C_{ref}(w_{i-2}, w_{i-1})}$

Probabilistic Language Modeling

N-Gram Example – Unigram $P(w_i)$ – "Word Salad"

beginning by, very Alice but was and?
reading no tired of to into sitting
sister the, bank, and thought of without
her nothing: having conversations Alice
once do or on she it get the book her had
peeped was conversation it pictures or
sister in, 'what is the use had twice of
a book''pictures or' to

$P(\texttt{of}) = 3/66$ $P(\texttt{to}) = 2/66$ $P(\texttt{,}) = 4/66$

$P(\texttt{Alice}) = 2/66$ $P(\texttt{her}) = 2/66$ $P(\texttt{'}) = 4/66$

$P(\texttt{was}) = 2/66$ $P(\texttt{sister}) = 2/66$

Source: https://courses.grainger.illinois.edu/cs447/fa2020/Slides/Lecture03.pdf – Slide 26,27

Probabilistic Language Modeling

N-Gram Example – Bigram $P(w_i|w_{i-1})$ – "Proper Text Syntax"

Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, 'and what is the use of a book,' thought Alice 'without pictures or conversation?'

$P(w^{(i)} = \text{of} \mid w^{(i-1)} = \text{tired}) = 1$

$P(w^{(i)} = \text{of} \mid w^{(i-1)} = \text{use}) = 1$

$P(w^{(i)} = \text{sister} \mid w^{(i-1)} = \text{her}) = 1$

$P(w^{(i)} = \text{beginning} \mid w^{(i-1)} = \text{was}) = 1/2$

$P(w^{(i)} = \text{reading} \mid w^{(i-1)} = \text{was}) = 1/2$

$P(w^{(i)} = \text{bank} \mid w^{(i-1)} = \text{the}) = 1/3$

$P(w^{(i)} = \text{book} \mid w^{(i-1)} = \text{the}) = 1/3$

$P(w^{(i)} = \text{use} \mid w^{(i-1)} = \text{the}) = 1/3$

Source: https://courses.grainger.illinois.edu/cs447/fa2020/Slides/Lecture03.pdf – Slide 26,27

Ostbayerische Technische Hochschule Amberg-Weiden

N-Gram Table

- General joint (sentence) probability $P(\vec{w}) = P(w_1, w_2, w_3, ..., w_m)$ (sequence of words):

$$P(\vec{w}) = P(w_1, w_2, w_3, ..., w_m) \approx P(w_1) \prod_{i=1}^{m-1} P(w_{i+1} | w_{i-k+1}^{i})$$

with $w_{i-k+1}^{i} = (w_{i-k+1}, \ldots, w_i)$ and $(i - k + 1) \leq 0 \rightarrow = 1$, next to $k = N - 1$

Probabilistic Language Modeling

N-Gram Lookup

N-Gram Table

- General joint (sentence) probability $P(\vec{w}) = P(w_1, w_2, w_3, ..., w_m)$ (sequence of words):

$$P(\vec{w}) = P(w_1, w_2, w_3, ..., w_m) \approx P(w_1) \prod_{i=1}^{m-1} P(w_{i+1}|w_{i-k+1}^{i})$$

with $w_{i-k+1}^{i} = (w_{i-k+1}, \ldots, w_i)$ and $(i - k + 1) \leq 0 \rightarrow = 1$, next to $k = N - 1$

- N-Gram table contains counts (or probabilities) for the respective N-Grams, including $N = 1$ (Unigram), $N = 2$ (Bigram), ... , $N = k + 1$ (k order of the Markov model):

$$P(w_i|w_{i-k}, \ldots, w_{i-1}) = \frac{C(w_{i-k}, \ldots, w_{i-1}, w_i)}{C(w_{i-k}, \ldots, w_{i-1})}$$

Probabilistic Language Modeling

Bigram-Table – Estimation of Probabilities

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

- Unigram-Counts: i$=2,533$, want$=927$, to$=2,417$, eat$=746$, chinese$=158$, food$=1,093$, lunch$=341$, spend$=278$

Source: https://web.stanford.edu/ jurafsky/slp3/slides/LM_4.pdf, Slide 18

Probabilistic Language Modeling

Bigram-Table – Estimation of Probabilities

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

- Unigram-Counts: i$= 2,533$, want$= 927$, to$= 2,417$, eat$= 746$, chinese$= 158$, food$= 1,093$, lunch$= 341$, spend$= 278$
- $P(to|want) =?$, $P(eat|i) =?$, $P(chinese|want) =?$

Source: https://web.stanford.edu/ jurafsky/slp3/slides/LM_4.pdf, Slide 18

Probabilistic Language Modeling

Ostbayerische Technische Hochschule Amberg-Weiden

Bigram-Table – Estimation of Probabilities

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

- Unigram-Counts: i$= 2,533$, want$= 927$, to$= 2,417$, eat$= 746$, chinese$= 158$, food$= 1,093$, lunch$= 341$, spend$= 278$
- $P(to|want) =?$, $P(eat|i) =?$, $P(chinese|want) =?$
- Do i really need to compute all tables for each N-Gram ?

Source: https://web.stanford.edu/ jurafsky/slp3/slides/LM_4.pdf, Slide 18

Probabilistic Language Modeling

Bigram-Table – Estimation of Probabilities

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

- Unigram-Counts: i$= 2,533$, want$= 927$, to$= 2,417$, eat$= 746$, chinese$= 158$, food$= 1,093$, lunch$= 341$, spend$= 278$
- $P(to|want) = 0.66$, $P(eat|to) = 0.0027$, $P(chinese|want) = 0.0065$

Source: https://web.stanford.edu/ jurafsky/slp3/slides/LM_4.pdf, Slide 18

Probabilistic Language Modeling

Bigram-Table – Estimation of Probabilities

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

- Unigram-Counts: i$= 2,533$, want$= 927$, to$= 2,417$, eat$= 746$, chinese$= 158$, food$= 1,093$, lunch$= 341$, spend$= 278$
- $P(to|want) = 0.66$, $P(eat|to) = 0.0027$, $P(chinese|want) = 0.0065$
- Data sparsity, relative frequencies & chain rule lead to very small probabilities $\rightarrow$ log-transformation: $log(p_1 \cdot p_2 \cdot p_3 \cdot p_4) = log\ p_1 + log\ p_2 + log\ p_3 + log\ p_4$

Source: https://web.stanford.edu/ jurafsky/slp3/slides/LM_4.pdf, Slide 18

Probabilistic Language Modeling

N-Gram – Sentence Boundaries

- Is it a good idea to use the entire text at once within a single string object?

Probabilistic Language Modeling

N-Gram – Sentence Boundaries

- Is it a good idea to use the entire text at once within a single string object?
- Better: provide a vector of sentences to counteract N-Grams "crossing over sentence boundaries"

Probabilistic Language Modeling

N-Gram – Sentence Boundaries

- Is it a good idea to use the entire text at once within a single string object?
- Better: provide a vector of sentences to counteract N-Grams "crossing over sentence boundaries"
- Sentence boundary tokens are denoted via $< s >$ (BOS, start) and $< /s >$ (EOS, end)
  $\rightarrow$ "<s> <s> I start with double BOS, why? </s>"
  $\rightarrow$ Think about $P(< s >) \cdot P(< s > | < s >) \cdot P(I| < s >< s >) \cdot P(start| < s > I)$

Probabilistic Language Modeling

N-Gram – Sentence Boundaries

- Is it a good idea to use the entire text at once within a single string object?
- Better: provide a vector of sentences to counteract N-Grams "crossing over sentence boundaries"
- Sentence boundary tokens are denoted via $< s >$ (BOS, start) and $< /s >$ (EOS, end)
  $\rightarrow$ "<s> <s> I start with double BOS, why? </s>"
  $\rightarrow$ Think about $P(< s >) \cdot P(< s > | < s >) \cdot P(I| < s >< s >) \cdot P(start| < s > I)$
- Extend joint (sentence) probability $P(w_0, w_1, w_2, w_3, ..., w_m, w_{m+1})$ (sequence of words):

$$P(w_0, w_1, w_2, w_3, ..., w_m, w_{m+1}) \approx P(w_0) \prod_{i=0}^{m} P(w_{i+1}|w_{i-k+1}^{i})$$

with $w_{i-k+1}^{i} = (w_{i-k+1}, \ldots, w_i)$ and $(i - k + 1) \leq 0 \rightarrow = 0$, next to $k = N - 1$, $P(w_o) =< s >$, as well as $P(w_{m+1}) =< /s >$

Probabilistic Language Modeling

N-Gram – Language Modeling

- Vocabulary V with a given number of $|V|$ words (e.g. $|V| = 1,000$) $\rightarrow$ Unigram $|V|^1 = 1,000$, Bigram $|V|^2 = 1,000,000$, Trigram $|V|^3 = 1,000,000,000$

Probabilistic Language Modeling

N-Gram – Language Modeling

- Vocabulary V with a given number of $|V|$ words (e.g. $|V| = 1,000$) $\rightarrow$ Unigram $|V|^1 = 1,000$, Bigram $|V|^2 = 1,000,000$, Trigram $|V|^3 = 1,000,000,000$
- $L \subseteq V^*$, with L as the language and V^* as the associated parametric complexity (N-Gram patterns), defines a possibly infinite set of strings, drawn from a (finite) vocabulary V

Probabilistic Language Modeling

N-Gram – Language Modeling

- Vocabulary V with a given number of $|V|$ words (e.g. $|V| = 1,000$) $\rightarrow$ Unigram $|V|^1 = 1,000$, Bigram $|V|^2 = 1,000,000$, Trigram $|V|^3 = 1,000,000,000$
- $L \subseteq V^*$, with L as the language and V^* as the associated parametric complexity (N-Gram patterns), defines a possibly infinite set of strings, drawn from a (finite) vocabulary V
- A language model $P(L) = P(V^*)$ should specify a single parametric distribution V^*, summing up to $= 1$ across all strings in $L \subseteq V^*$, irrespective of the chosen length N: $P(L) = P(V) + P(V^2) + P(V^3) + \ldots + P(V^n) = 1$
- Language models are described as a probability distribution across the entire sentences or texts $\rightarrow$ Add End-Of-Sentence (EOS) token $(V \cup \text{EOS})$

Probabilistic Language Modeling

N-Gram – Language Modeling

- Probabilistic models usually make an independence assumption
  - Markov assumption – word sequences are typically not stochastically independent $P(X, Y) = P(X)P(Y)$, but are treated as such $\rightarrow$ Significant parametric reduction, but ...
  - Independence assumptions are only rough estimations, applied during training $\rightarrow$ models are also significantly more error-prone

Probabilistic Language Modeling

N-Gram – Language Modeling

- Probabilistic models usually make an independence assumption
  - Markov assumption – word sequences are typically not stochastically independent $P(X,Y) = P(X)P(Y)$, but are treated as such $\rightarrow$ Significant parametric reduction, but ...
  - Independence assumptions are only rough estimations, applied during training $\rightarrow$ models are also significantly more error-prone
- In general there exist two individual steps to build a probabilistic language model
  - Specifying the model (choose N)
  - Train the model in order to estimate the parameters ($=$ training/learning phase)

Probabilistic Language Modeling

N-Gram – Data Corpora and Partitioning

- Large bodies of textual information, also referred to as corpora, are required to robustly train a model

Probabilistic Language Modeling

N-Gram – Data Corpora and Partitioning

- Large bodies of textual information, also referred to as corpora, are required to robustly train a model
- The entire dataset needs to be split into an independent training, validation, and completely unseen testing partition to learn, adjust, and fine-tune parameters $\rightarrow$ Done on the training and validation set!

Probabilistic Language Modeling

N-Gram – Data Corpora and Partitioning

- Large bodies of textual information, also referred to as corpora, are required to robustly train a model
- The entire dataset needs to be split into an independent training, validation, and completely unseen testing partition to learn, adjust, and fine-tune parameters $\rightarrow$ Done on the training and validation set!
- The final and trained model is applied to the unseen test material, in order to verify and judge the final performance $\rightarrow$ Generalization (under-/overfitting)

Probabilistic Language Modeling

N-Gram – Data Corpora and Partitioning

- Large bodies of textual information, also referred to as corpora, are required to robustly train a model
- The entire dataset needs to be split into an independent training, validation, and completely unseen testing partition to learn, adjust, and fine-tune parameters $\rightarrow$ Done on the training and validation set!
- The final and trained model is applied to the unseen test material, in order to verify and judge the final performance $\rightarrow$ Generalization (under-/overfitting)
- Shakespeare data corpus with $884,647$ tokens and a total vocabulary size of $|V| = 29,066$ results in $|V|^{N=2} = 844,832,356$ Bigrams $\rightarrow$ recognized only $300,000$!

Probabilistic Language Modeling

N-Gram – Data Corpora and Partitioning

- Large bodies of textual information, also referred to as corpora, are required to robustly train a model
- The entire dataset needs to be split into an independent training, validation, and completely unseen testing partition to learn, adjust, and fine-tune parameters $\rightarrow$ Done on the training and validation set!
- The final and trained model is applied to the unseen test material, in order to verify and judge the final performance $\rightarrow$ Generalization (under-/overfitting)
- Shakespeare data corpus with $884,647$ tokens and a total vocabulary size of $|V| = 29,066$ results in $|V|^{N=2} = 844,832,356$ Bigrams $\rightarrow$ recognized only $300,000$!
- In total, $99.96\,\%$ of the different N-Gram (Bigram) patterns, as part of the N-Gram table, belong to unseen (but still possible!) events with a probability of zero!

Probabilistic Language Modeling

N-Gram – Data Corpora and Partitioning – Google

We believe that the entire research community can benefit from access to such massive amounts of data. It will advance the state of the art, it will focus research in the promising direction of large-scale, data-driven approaches, and it will allow all research groups, no matter how large or small their computing resources, to play together. That's why we decided to share this enormous dataset with everyone. We processed 1,024,908,267,229 words of running text and are publishing the counts for all 1,176,470,663 five-word sequences that appear at least 40 times. There are 13,588,391 unique words, after discarding words that appear less than 200 times.

Watch for an announcement at the Linguistics Data Consortium (LDC), who will be distributing it soon, and then order your set of 6 DVDs. And let us hear from you - we're excited to hear what you will do with the data, and we're always interested in feedback about this dataset, or other potential datasets that might be useful for the research community.

Update (22 Sept. 2006): The LDC now has the data available in their catalog. The counts are as follows:

```
File sizes: approx. 24 GB compressed (gzip'ed) text files

Number of tokens:    1,024,908,267,229
Number of sentences:    95,119,665,584
Number of unigrams:         13,588,391
Number of bigrams:         314,843,401
Number of trigrams:        977,069,902
Number of fourgrams:     1,313,818,354
Number of fivegrams:     1,176,470,663
```

Source: https://blog.research.google/2006/08/all-our-n-gram-are-belong-to-you.html

Probabilistic Language Modeling

N-Gram – Data Corpora and Partitioning

- In general, N-Gram models perform only well in case the training and test corpus possess similar characteristics $\rightarrow$ Often not the case, resulting in many unseen events!

Probabilistic Language Modeling

N-Gram – Data Corpora and Partitioning

- In general, N-Gram models perform only well in case the training and test corpus possess similar characteristics $\rightarrow$ Often not the case, resulting in many unseen events!
- Training set:
  - "... write a letter"
  - "... write a proposal"
  - "... write a report"

Probabilistic Language Modeling

N-Gram – Data Corpora and Partitioning

- In general, N-Gram models perform only well in case the training and test corpus possess similar characteristics $\rightarrow$ Often not the case, resulting in many unseen events!
- Training set:
  - "... write a letter"
  - "... write a proposal"
  - "... write a report"
- Test set:
  - "...write a dissertation"
  - "...write a note

$\rightarrow$ P(dissertation|write a) =0

Probabilistic Language Modeling

N-Gram – Data Corpora and Partitioning

- In general, N-Gram models perform only well in case the training and test corpus possess similar characteristics $\rightarrow$ Often not the case, resulting in many unseen events!
- Training set:
  - "... write a letter"
  - "... write a proposal"
  - "... write a report"
- Test set:
  - "...write a dissertation"
  - "...write a note

  $\rightarrow$ P(dissertation|write a) =0
- As already mentioned, data sparsity causes a lot of N-Gram paradigms which have a probability of zero (avoid to model longer sequences $N >> 1$)!
- How to handle unknown/unseen words?

Probabilistic Language Modeling

N-Gram – Unseen/Unknown Words

- Closed vocabulary (word portfolio restricted to a certain domain, e.g. air traffic control) versus open vocabulary (“the actual real-world situation”)

Probabilistic Language Modeling

N-Gram – Unseen/Unknown Words

- Closed vocabulary (word portfolio restricted to a certain domain, e.g. air traffic control) versus open vocabulary ("the actual real-world situation")
- Unseen/Unknown words are not part of the training data corpus, however, they appear in the unseen test data and are known as Out of Vocabulary (OOV) words

Probabilistic Language Modeling

N-Gram – Unseen/Unknown Words

- Closed vocabulary (word portfolio restricted to a certain domain, e.g. air traffic control) versus open vocabulary ("the actual real-world situation")
- Unseen/Unknown words are not part of the training data corpus, however, they appear in the unseen test data and are known as Out of Vocabulary (OOV) words
- OOV rate refers to the number of OOV words in the test partition

Probabilistic Language Modeling

N-Gram – Unseen/Unknown Words

- Closed vocabulary (word portfolio restricted to a certain domain, e.g. air traffic control) versus open vocabulary ("the actual real-world situation")
- Unseen/Unknown words are not part of the training data corpus, however, they appear in the unseen test data and are known as Out of Vocabulary (OOV) words
- OOV rate refers to the number of OOV words in the test partition
- Solution: model an "open vocabulary" by adding the pseudo-word $< UNK >$

Probabilistic Language Modeling

N-Gram – Unseen/Unknown Words

- Closed vocabulary (word portfolio restricted to a certain domain, e.g. air traffic control) versus open vocabulary ("the actual real-world situation")
- Unseen/Unknown words are not part of the training data corpus, however, they appear in the unseen test data and are known as Out of Vocabulary (OOV) words
- OOV rate refers to the number of OOV words in the test partition
- Solution: model an "open vocabulary" by adding the pseudo-word $< UNK >$
- Two way of training scenarios using the $< UNK >$ pattern:
  - Choose a fixed vocabulary and map any unknown word in the training set to the $< UNK >$ token (text normalization) and compute the probabilities as for any traditional word

Probabilistic Language Modeling

N-Gram – Unseen/Unknown Words

Ostbayerische Technische Hochschule Amberg-Weiden

- Closed vocabulary (word portfolio restricted to a certain domain, e.g. air traffic control) versus open vocabulary ("the actual real-world situation")
- Unseen/Unknown words are not part of the training data corpus, however, they appear in the unseen test data and are known as Out of Vocabulary (OOV) words
- OOV rate refers to the number of OOV words in the test partition
- Solution: model an "open vocabulary" by adding the pseudo-word $< UNK >$
- Two way of training scenarios using the $< UNK >$ pattern:
  - Choose a fixed vocabulary and map any unknown word in the training set to the $< UNK >$ token (text normalization) and compute the probabilities as for any traditional word
  - Vocabulary is created based on the training data, while replacing words with only very few occurrences by the $< UNK >$ tag and train the system as usual

Probabilistic Language Modeling

Bigram-Table – Zero Values!

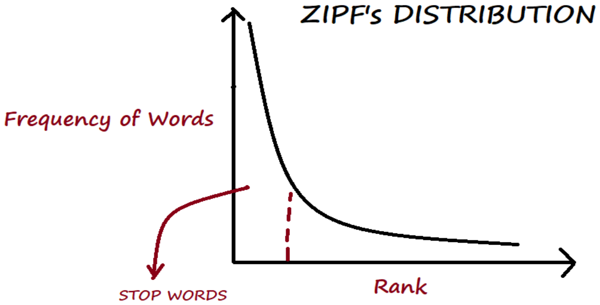
	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

- Are zero values a problem during model deployment?

Source: https://web.stanford.edu/ jurafsky/slp3/slides/LM_4.pdf, Slide 18

N-Gram – Zero Probabilities

- Zero probabilities severely affect the model performance and generalization $\rightarrow$ In case a specific N-Gram is unseen ($= 0$), the entire product of the Markov assumption is zero

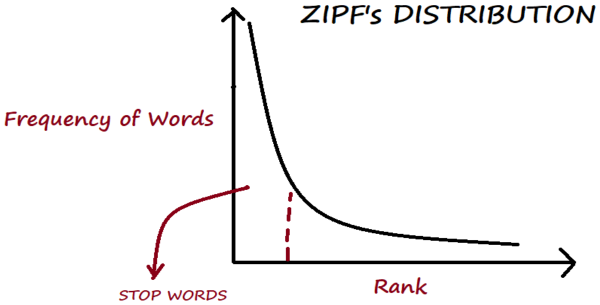


Source: https://www.kaggle.com/code/vishynair/zipf-s-law-validation-with-word-frequency

Probabilistic Language Modeling

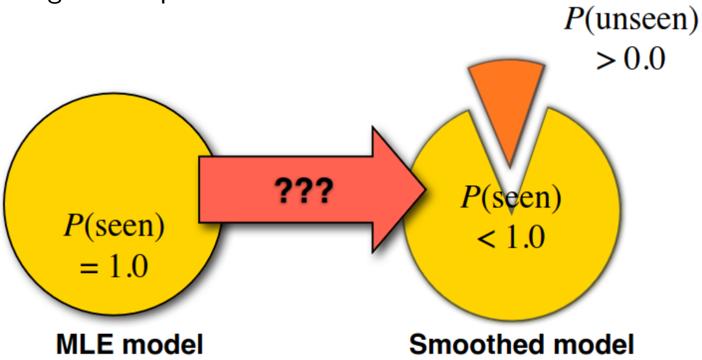
N-Gram – Zero Probabilities

- Zero probabilities severely affect the model performance and generalization → In case a specific N-Gram is unseen ($=0$), the entire product of the Markov assumption is zero



- Even more data? → Zipf's Law $(\frac{1}{Rank}) \cdot C(w_i)$, with $C(w_i)$ as the word-specific count

Source: https://www.kaggle.com/code/vishynair/zipf-s-law-validation-with-word-frequency

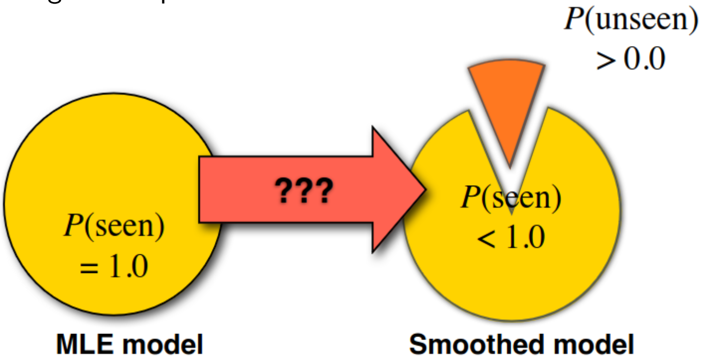


- Smooth existing probability distribution and redistribute the overall probability mass ($= 1$) to also cover unseen events

Source: https://courses.grainger.illinois.edu/cs447/fa2020/Slides/Lecture03.pdf, Slide 46

Probabilistic Language Modeling

N-Gram – Smoothing Techniques



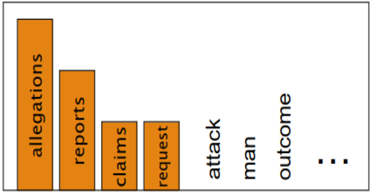
- Smooth existing probability distribution and redistribute the overall probability mass $(= 1)$ to also cover unseen events
- Try to fill the "gaps" in $|V|$ (zero count elements in N-Gram table), while trying to maintain the original distribution as much as possible

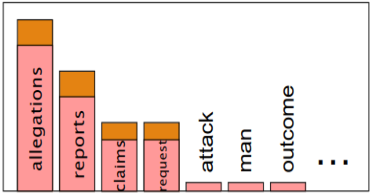
Source: https://courses.grainger.illinois.edu/cs447/fa2020/Slides/Lecture03.pdf, Slide 46

Probabilistic Language Modeling

N-Gram – Smoothing Techniques

Key Concept: Every event (e.g. uni-, bi-, trigram) occurs λ-times more frequent than it actually really does





Source: https://web.stanford.edu/ jurafsky/slp3/slides/3_LM_Jan_08_2021.pdf, Slide 50

Probabilistic Language Modeling

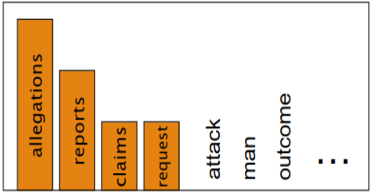
N-Gram – Smoothing Techniques

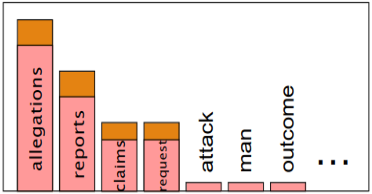
Key Concept: Every event (e.g. uni-, bi-, trigram) occurs λ-times more frequent than it actually really does

Probability Discount and Redistribution: block a certain amount of probability mass p_{unk} for the unseen events $\rightarrow$ Discounting!

- How to properly discount p-mass?
- How to properly redistribute p-mass?
- How to combine model estimates and use complementary strengths of different models (Interpolation)?

$\rightarrow$ Smoothing offers a wide repertoire of different techniques!



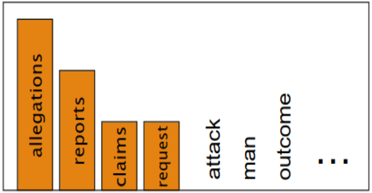


Source: https://web.stanford.edu/ jurafsky/slp3/slides/3_LM_Jan_08_2021.pdf, Slide 50

Probabilistic Language Modeling

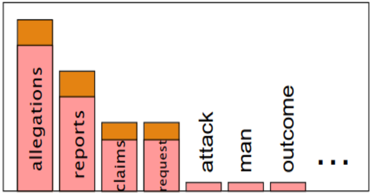
N-Gram – Smoothing Techniques

Key Concept: Every event (e.g. uni-, bi-, trigram) occurs λ-times more frequent than it actually really does



Probability Discount and Redistribution: block a certain amount of probability mass p_{unk} for the unseen events $\rightarrow$ Discounting!

- How to properly discount p-mass?
- How to properly redistribute p-mass?
- How to combine model estimates and use complementary strengths of different models (Interpolation)?



$\rightarrow$ Smoothing offers a wide repertoire of different techniques!
Laplace, Absolute Discounting, Additive/Lidstone, Good-Turing, Katz‘ backoff, Kneser-Ney, Witten-Bell, Jelinek-Marcer, . . . (see Literature Goodman et al. *“An empirical study of smoothing techniques for language modeling”*)

Source: https://web.stanford.edu/ jurafsky/slp3/slides/3_LM_Jan_08_2021.pdf, Slide 50

Probabilistic Language Modeling

N-Gram – Intrinsic versus Extrinsic Evaluation

- Two strategies to evaluate:
  - Intrinsic Evaluation: describes how well the model captures the underlying and required probability information $\rightarrow$ Evaluation metric: Perplexity

Probabilistic Language Modeling

N-Gram – Intrinsic versus Extrinsic Evaluation

- Two strategies to evaluate:
  - Intrinsic Evaluation: describes how well the model captures the underlying and required probability information $\rightarrow$ Evaluation metric: Perplexity
  - Extrinsic Evaluation: describes a task-driven evaluation scenario, measuring how the model perform on a specific task $\rightarrow$ Evaluation metric: Word Error Rate (WER)

Intrinsic Evaluation

- Evaluation metric (scoring) to measure the similarity between model prediction and ground truth (real text)
- Model training using an independent training & validation set (seen data corpora)
- Model testing using a completely unseen test set (unseen data corpora)

Ostbayerische Technische Hochschule Amberg-Weiden

Intrinsic Evaluation

- Evaluation metric (scoring) to measure the similarity between model prediction and ground truth (real text)
- Model training using an independent training & validation set (seen data corpora)
- Model testing using a completely unseen test set (unseen data corpora)

Perplexity Metric

$$PP(w_1, \ldots, w_m) = \sqrt[m]{\frac{1}{P(w_1, \ldots, w_m)}} = exp\left(\frac{1}{m}\sum_{i=1}^{m} logP(w_i|w_{i-1}, \ldots, w_{i-n+1})\right)$$

- Perplexity specifies the normalized inverse (joint) probability of the unseen test set
- The lower the Perplexity the better the model performance, because of a larger $P(w_1, \ldots, w_m)$ $\rightarrow$ two LMs only comparable when $N_{LM1} = N_{LM2}$

Ostbayerische Technische Hochschule Amberg-Weiden

Extrinsic (Task-Base) Evaluation

- Perplexity is an indicator which of the LM-models performs better on the unseen test corpora $\rightarrow$ No performance indicator regarding the final task!
- Train LM-models A & B, apply it to the same task T (unseen test data), and compare performance metrics

Probabilistic Language Modeling

N-Gram – Extrinsic Evaluation & WER

Ostbayerische Technische Hochschule Amberg-Weiden

Extrinsic (Task-Base) Evaluation

- Perplexity is an indicator which of the LM-models performs better on the unseen test corpora $\rightarrow$ No performance indicator regarding the final task!
- Train LM-models A & B, apply it to the same task T (unseen test data), and compare performance metrics

Word-Error-Rate (WER)

$$WER = \frac{Insertions + Deletions + Substituions}{Number\ of\ Words\ in\ Reference}$$

- Designed for Automatic Speech Recognition (ASR)
- Difference between the predicted word sequence (model hypothesis) and ground truth sequence of words

Probabilistic Language Modeling

N-Gram – Pros and Cons

Advantages

- Straight-forward, simple, and (computationally) cheap
- Useful across a wide variety of applications (auto-completion, sentiment analysis, text classification, text generation, etc.)
- Availability of statistics over the internet
- Underlying math well understood

Probabilistic Language Modeling

N-Gram – Pros and Cons

Advantages

- Straight-forward, simple, and (computationally) cheap
- Useful across a wide variety of applications (auto-completion, sentiment analysis, text classification, text generation, etc.)
- Availability of statistics over the internet
- Underlying math well understood

Disadvantages

- Language: do not capture non-local and long-term dependencies ("Since I am a child, as i said yesterday in our meeting, I love to run")
- Data sparsity: not enough data to estimate large ($N > 3$) language structures
- Markov assumption might be an oversimplified hypothesis and constraint

Let's switch to Jupyter and get hands on N-Grams in Python...

Source: https://www.activestate.com/blog/top-10-coding-mistakes-in-python-how-to-avoid-them/

Further Questions?

https://www.oth-aw.de/hochschule/ueber-uns/personen/bergler-christian/

Source: https://emekaboris.medium.com/the-intuition-behind-100-days-of-data-science-code-c98402cdc92c

References

Used Literature...

- Daniel Jurafsky, James H. Martin , Speech and Language Processing, Copyright © 2023. All rights reserved. Draft of January 7, 2023
- https://web.stanford.edu/ jurafsky/slp3/ed3book_jan72023.pdf
- https://web.stanford.edu/ jurafsky/slp3/slides/3_LM_Jan_08_2021.pdf
- https://courses.grainger.illinois.edu/cs447/fa2020/Slides/Lecture03.pdf
- https://staff.fnwi.uva.nl/k.simaan/D-Courses2013/D-NLMI2013/college3.pdf