Text Mining and Language Analytics

Lecture 3

N-grams

Dr Stamos Katsigiannis

2023-24

Durham University

Probabilistic Language Models

- **Goal** → Assign a probability to a sentence (sequence of words)

- But why?
  - Probabilistic classification models (e.g. Naïve Bayes)
  - Machine translation
    - e.g. P(the **tall** giraffe) > P(the **high** giraffe)
  - Spelling correction
    - e.g. I will be there in **twety** minutes
      - P(in **twenty** minutes) > P(in **twety** minutes)
  - Speech recognition
    - e.g. P(I saw a van) >> P(eyes awe Evan)
  - …

Durham University

Probabilistic Language Modelling

- **Goal** → Assign a probability to a sentence (sequence of words)

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

- **Related task** → Probability of an upcoming word

$$P(w_n \mid w_1, w_2,\ldots, w_{n-1})$$

- A model that computes $P(W)$ or $P(w_n \mid w_1, w_2,\ldots, w_{n-1})$ is called a **Language Model**

Durham University

Remember the chain rule

- From the definition of conditional probabilities:

$$P(B|A) = \frac{P(A,B)}{P(A)} \quad \Rightarrow \quad P(A,B) = P(A) \cdot P(B \mid A)$$

- For more variables:

$$P(A,B,C,D) = P(A) \cdot P(B \mid A) \cdot P(C \mid A,B) \cdot P(D \mid A,B,C)$$

- Chain rule:

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

Durham University

Computing P(W)

- How to compute the joint probability of the following sentence?

  The old dog was very friendly

- Apply the chain rule:

  P(The old dog was friendly) =

  P(The)·P(old | The)·P(dog | The old)

  ·P(was | The old dog)·P(friendly | The old dog was)

Durham University

Computing the probability of upcoming words (I)

- How to compute the probability $P(w_n \mid w_1, w_2, \ldots, w_{n-1})$?

- Count the occurrences of sentence $(w_1, w_2, \ldots, w_{n-1}, w_n)$ in the corpus
- Count the occurrences of sentence $(w_1, w_2, \ldots, w_{n-1})$ in the corpus

- Maximum Likelihood Estimation (MLE):

$$P(w_n \mid w_1, w_2, \ldots, w_{n-1}) = \frac{\mathrm{count}(w_1, w_2, \ldots, w_{n-1}, w_n)}{\mathrm{count}(w_1, w_2, \ldots, w_{n-1})}$$

Durham University

Computing the probability of upcoming words (II)

- Compute P(but | The old dog was very friendly)

- Could we just count and divide?

$$P(\text{but}|\text{The old dog was friendly}) = \frac{\text{count}(\text{The old dog was friendly but})}{\text{count}(\text{The old dog was friendly})}$$

- **NO !**
  - Too many possible sentences!
  - We'll never have enough data to estimate these

The Markov assumption (I)

- Let's make a simplification assumption:

$$P(\text{but} \mid \text{The old dog is friendly}) \approx P(\text{but} \mid \text{friendly})$$

- Or

$$P(\text{but} \mid \text{The old dog is friendly}) \approx P(\text{but} \mid \text{is friendly})$$

- In other words, assume that the probability of an upcoming word depends only on the previous or a few previous words

Durham University

The Markov assumption (II)

Andrey A. Markov
1856-1922

- The Markov assumption
  - **When predicting the future, the past doesn't matter, only the present**

- Extending to sentences
  - **When predicting the next word in a sequence, the previous words don't matter, only the current word**

- Expanding a little…
  - **When predicting the next word in a sequence, not all the previous words matter, only a few of them**

The Markov assumption (III)

- Markov assumption: **Limited history**

- There is a $k \geq 0$ such that for all upcoming words w_n

$$P(w_n \mid w_1, w_2, \ldots w_{n-1}) \approx P(w_n \mid w_{n-k}, \ldots, w_{n-1})$$

- k^{th} order Markov assumption:
  - $k=0 \rightarrow P(w_n \mid w_1, w_2, \ldots w_{n-1}) \approx P(w_n)$
  - $k=1 \rightarrow P(w_n \mid w_1, w_2, \ldots w_{n-1}) \approx P(w_n \mid w_{n-1})$
  - $k=2 \rightarrow P(w_n \mid w_1, w_2, \ldots w_{n-1}) \approx P(w_n \mid w_{n-2}, w_{n-1})$
  - $k=K \rightarrow P(w_n \mid w_1, w_2, \ldots w_{n-1}) \approx P(w_n \mid w_{n-K}, \ldots, w_{n-2}, w_{n-1})$

Durham University

The Markov assumption (IV)

- The order of the Markov assumption is defined by the length of its history
  - 0^{th} order: No previous words used
  - 1^{st} order: One previous word used
  - k^{th} order: k previous words used

- 0^{th} order: $P(w_1, \ldots, w_n) \approx P(w_1) \prod_{i=1}^{n-1} P(w_{i+1}) \approx P(w_1)P(w_2) \ldots P(w_n)$

- 1^{st} order: $P(w_1, \ldots, w_n) \approx P(w_1) \prod_{i=1}^{n-1} P(w_{i+1}|w_i)$

- 2^{nd} order: $P(w_1, \ldots, w_n) \approx P(w_1)P(w_2|w_1) \prod_{i=2}^{n-1} P(w_{i+1}|w_{i-1}, w_i)$

- …

Durham University

N-grams

- Given a sequence of N-1 words, an N-gram model predicts the most probable word that might follow this sequence

- Probabilistic model trained on a corpus of text
  - **Large corpus needed!**

- k^{th} order Markov assumption → (k+1)-gram
  - 1-gram: Unigram
  - 2-gram: Bigram
  - 3-gram: Trigram
  - …

N-grams: Unigrams

- Unigrams (1-grams) → $P(w_n) = \frac{\text{count}(w_n)}{\text{Total words}} = \frac{\text{count}(w_n)}{\sum_{i=1}^{V} \text{count}(w_i)}$

- Consider the following text
  - I saw a red table, a red car, and a red box. Then, I bought a red table.
  - Vocabulary of size V=10: {I, saw, a, red, table, car, and, box, Then, bought}
  - Total words: **18**

- Examples:
  - $P(\text{table}|\text{a red}) \approx P(\text{table}) = \frac{\text{count(table)}}{\text{Total words}} = \frac{2}{18} \approx 0.111$
  - $P(\text{red}|\text{a}) \approx P(\text{red}) = \frac{\text{count(red)}}{\text{Total words}} = \frac{4}{18} \approx 0.222$

Unigram	Count
I	2
saw	1
a	4
red	4
table	2
car	1
and	1
box	1
Then	1
bought	1

SUM=18

Durham University

N-grams: Bigrams (I)

- Bigrams (2-grams) can be modelled as a table containing all bigrams and their count
- What about the start and end of a sentence?
  - We can insert artificial terms <s> and </s> respectively

$$P(w_n|w_{n-1}) = \frac{count(w_{n-1}, w_n)}{count(w_{n-1})}$$

- Consider the following text
  - <s> I saw a red table, a red car, and a red box </s> <s> Then, I bought a red table </s>
  - Vocabulary of size V=12: {<s>, I, saw, a, red, table, car, and, box, </s>, Then, bought}
  - Total words: **22**

- Examples:
  - $\mathrm{P(red|a)} = \frac{\mathrm{count(a\ red)}}{\mathrm{count(a)}} = \frac{4}{4} = 1$
  - $\mathrm{P(box|table)} = \frac{\mathrm{count(table\ box)}}{\mathrm{count(table)}} = \frac{0}{2} = 0$
  - $\mathrm{P(I|<s>)} = \frac{\mathrm{count(<s>\ I)}}{\mathrm{count(<s>)}} = \frac{1}{2} = 0.5$

Bigram	Count
<s> I	1
<s> red	0
saw a	1
a red	4
red table	2
table box	0
…	…

Unigram	Count
<s>	2
I	2
saw	1
a	4
red	4
table	2
…	…

Zero-count bigrams can be omitted from the table

Durham University

N-grams: Trigrams

- Trigrams (3-grams) can be modelled as a table containing all trigrams and their count
  - Bigrams table also needed!

- Consider the previous text

- Example:
  - $\text{P}(\text{table}|\text{a red}) = \frac{\text{count}(\text{a red table})}{\text{count}(\text{a red})} = \frac{2}{4} = 0.5$
  - $\text{P}(\text{red}|\text{saw a}) = \frac{\text{count}(\text{saw a red})}{\text{count}(\text{saw a})} = \frac{1}{1} = 1$

Trigram	Count
<s> table box	0
red I car	0
saw a red	1
a red table	2
red table a	1
table a red	1
…	…

Bigram	Count
<s> I	1
<s> red	0
saw a	1
a red	4
red table	2
table box	0
…	…

$$P(w_n|w_{n-2}, w_{n-1}) = \frac{count(w_{n-2}, w_{n-1}, w_n)}{count(w_{n-2}, w_{n-1})}$$

N-grams: Unknown words

- What about words that don't exist in the training corpus?

- **Closed vocabulary** systems don't suffer from this issue
  - Test sets can only contain words from the system's vocabulary
- **Open vocabulary** systems → Out of vocabulary (OOV) words (unknown words) exist in test sets
  - Modelled by pseudo-word <UNK>

- **Approach 1**
  - Select a predefined vocabulary
  - Convert any word from the training corpus to <UNK> if not in vocabulary
  - Treat <UNK> as a regular word
- **Approach 2**
  - Convert to <UNK> any word with a count less than C (C=small number)
  - Treat <UNK> as a regular word
- **Approach 3**
  - Select a vocabulary size V'
  - Choose the V' most frequent words and replace the rest with <UNK>
  - Treat <UNK> as a regular word

Durham University

N-grams: Zero counts (I)

- Consider a **bigram** model trained on the following text:

<s> Yesterday I drove to the supermarket </s> <s> It was Tesco </s>

- **Problem:** P(I drove to Tesco)=0
  - P(I|<s>)=0, P(Tesco|to)=0
  - Bigrams (<s>,I) and (to,Tesco) don't exist in corpus

- Zero-count N-grams are called **unseen events**

- **But the sentence "I drove to Tesco" is valid!**

- How do we estimate its probability?

Durham University

N-grams: Zero counts (II)

The sparsity problem

- As N increases, the number of N-grams is greater
  - Vocabulary size V → V^N N-grams

- The chance that all 2-grams exist in a training corpus is small
- Smaller probability for 3-grams, 4-grams, ….

- **Is more data the answer?**
  - Any given corpus has a few very frequent words but many infrequent ones
  - Having a reasonably sized corpus is important, but there will always be zero-count N-grams
  - Smoothing can be applied → Eliminate zero counts

Durham University

N-grams: Smoothing

- **Smoothing** → Increasing the probability of infrequent events while slightly reducing the probability of frequent events

- Laplace smoothing → Increase all counts by 1
  - Unigrams: $\mathrm{P}(\mathrm{w_n}) = \frac{\mathrm{count}(\mathrm{w_n})+1}{\sum_{\mathrm{i}=1}^{\mathrm{V}}[\mathrm{count}(\mathrm{w_i})+1]} = \frac{\mathrm{count}(\mathrm{w_n})+1}{\mathrm{V}+\sum_{\mathrm{i}=1}^{\mathrm{V}}\mathrm{count}(\mathrm{w_i})}$, Bigrams: $\mathrm{P}(\mathrm{w_n}|\mathrm{w_{n-1}}) = \frac{\mathrm{count}(\mathrm{w_{n-1}},\mathrm{w_n})+1}{\mathrm{count}(\mathrm{w_{n-1}})+\mathrm{V}}$
  - N-grams: $\mathrm{P}(\mathrm{w_n}|w_1,\dots,\mathrm{w_{n-1}}) = \frac{\mathrm{count}(\mathrm{w_1},\dots,\mathrm{w_n})+1}{\mathrm{count}(w_1,\dots,\mathrm{w_{n-1}})+\mathrm{V}}$

- Add-λ method → Increase all counts by $0 < \lambda \leq 1$
  - Unigrams: $\mathrm{P}(\mathrm{w_n}) = \frac{\mathrm{count}(\mathrm{w_n})+\lambda}{\sum_{\mathrm{i}=1}^{\mathrm{V}}[\mathrm{count}(\mathrm{w_i})+\lambda]} = \frac{\mathrm{count}(\mathrm{w_n})+\lambda}{\lambda\mathrm{V}+\sum_{\mathrm{i}=1}^{\mathrm{V}}\mathrm{count}(\mathrm{w_i})}$
  - N-grams: $\mathrm{P}(\mathrm{w_n}|w_1,\dots,\mathrm{w_{n-1}}) = \frac{\mathrm{count}(\mathrm{w_1},\dots,\mathrm{w_n})+\lambda}{\mathrm{count}(w_1,\dots,\mathrm{w_{n-1}})+\lambda\mathrm{V}}$

- More sophisticated methods exist (e.g. Good-Turing, Kneser-Nay)

Durham University

N-grams: Interpretation

- Consider a corpus of 1,000,000 total words
  - The word cat occurs 400 times
  - $P(\text{cat}) = \frac{400}{1000000} = 0.0004$

- Consider a text of 1,000,000 total words

- What is the probability that a random word selected from the text is cat?
  - The maximum likelihood estimation (MLE) of its probability is 0.0004
  - Is 0.0004 the best possible estimate? → **NO**
  - Maybe in this text the word cat is very rare or very common

- This probability means that it is **most likely** that the word cat will occur 400 times in a 1,000,000 word corpus

Durham University

N-grams: Advantages & Disadvantages

- Advantages
  - Simple, easy, cheap
  - Availability over the internet
  - Useful in many applications
  - Well defined (and understood) math

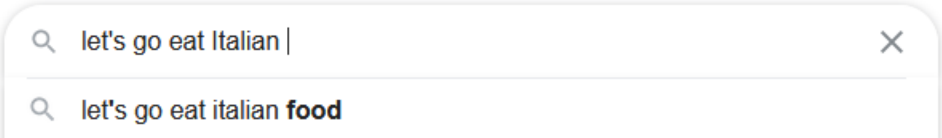
- Disadvantages
  - **Language:** They don't capture non-local word dependencies
  - **Sparsity:** The majority of counts is zero
  - **Assumptions:** Markov assumption might be too strong
  - **Data hungry:** Extremely huge corpora required for large N

Durham University

N-grams: Word autocomplete (I)

- Consider that the following sentence is typed:

  Let's go eat Italian



- How can N-grams be used to suggest the next word w_{next}?
  - Unigrams: Suggest w_{next} with the highest $P(w_{next})$
  - Bigrams: Suggest w_{next} with the highest $P(w_{next} \mid \text{Italian})$
  - Trigrams: Suggest w_{next} with the highest $P(w_{next} \mid \text{eat Italian})$
  - 4-grams: Suggest w_{next} with the highest $P(w_{next} \mid \text{go eat Italian})$
  - …

- Multiple suggestions can be made, ranked by the N-grams' probability

N-grams: Word autocomplete (II)

- Consider the following N-grams:

Unigram	Count
food	60
beaches	30
eat	20
I	100
pasta	80
Italian	50

Bigram	Count
I eat	30
Italian food	80
Italian language	60
Italian beaches	150
go eat	60
eat Italian	100

Trigram	Count
eat Italian pasta	120
eat Italian food	150
eat Italian bread	90
sunny Italian beaches	100
go eat Italian	120
go out of	300

- Predict the next word: Let's go eat Italian ______
  - **Unigrams:** Let's go eat Italian I
  - **Bigrams:** Let's go eat Italian beaches
  - **Trigrams:** Let's go eat Italian food

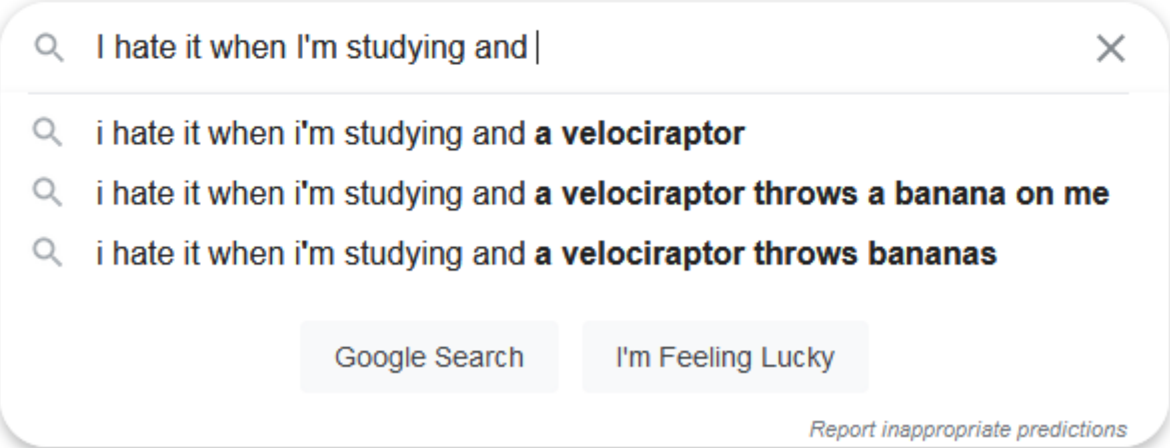
Predictions can differ

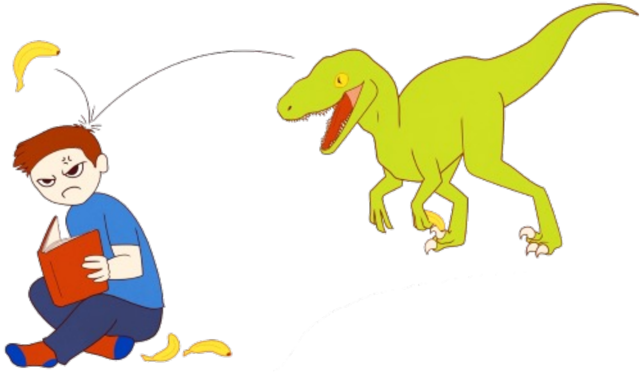
Larger N typically leads to better predictions

Durham University

N-grams: Training corpora matter!

Word predictions using N-grams depend on the corpora used for training the N-gram models!





*Searched on 19/11/2020 at https://www.google.com

N-gram availability

- Various N-gram collections are available online

- Example:
  - Google English N-grams dataset
  - Corpus size > 1 trillion tokens
  - Up to 5-grams
  - https://ai.googleblog.com/2006/08/all-our-n-gram-are-belong-to-you.html

Google AI Blog

The latest news from Google AI

All Our N-gram are Belong to You

Thursday, August 3, 2006

Posted by Alex Franz and Thorsten Brants, Google Machine Translation Team

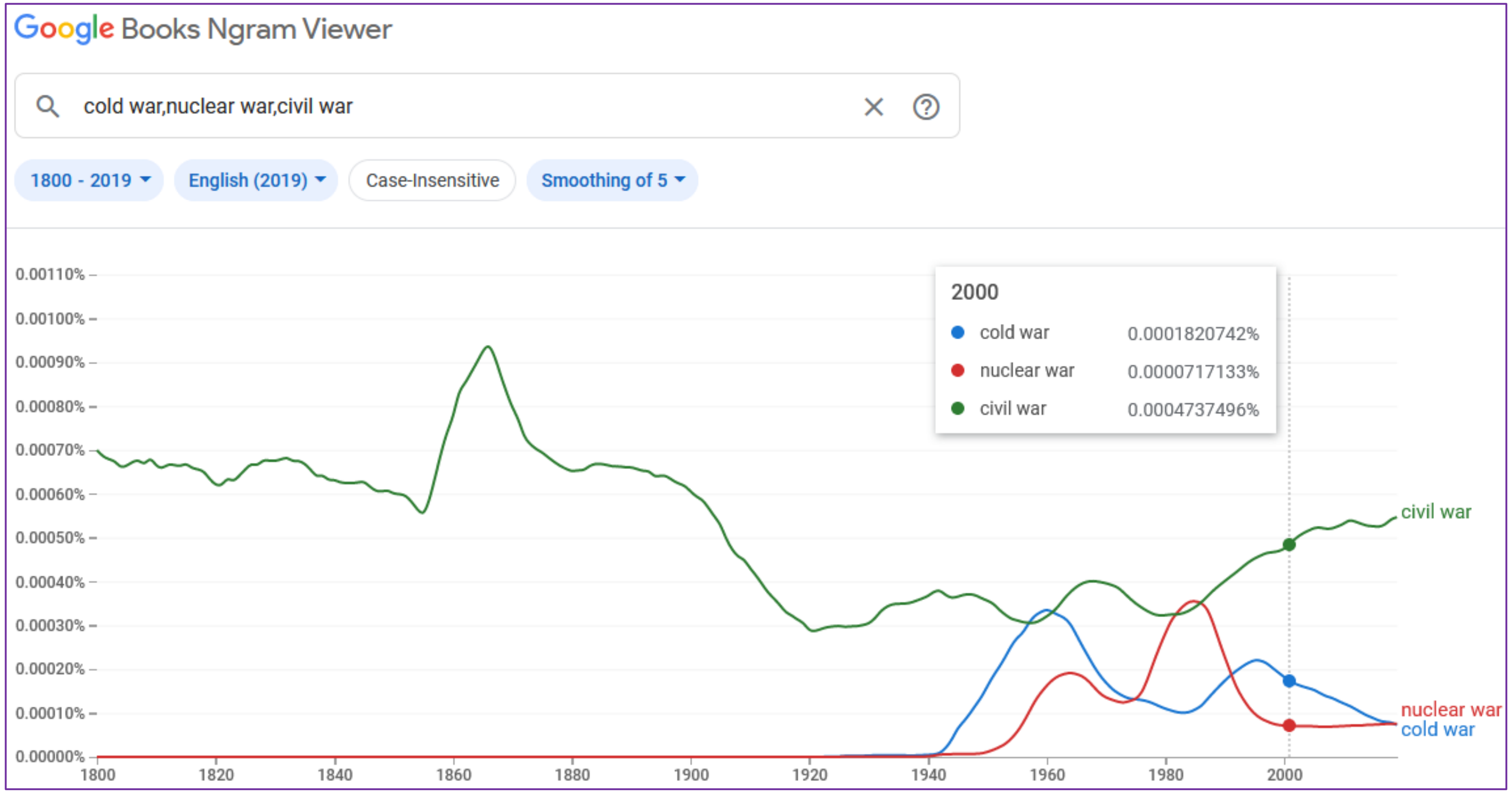
```
File sizes: approx. 24 GB compressed (gzip'ed) text files

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
```

Durham University

Google Books N-gram viewer

Check the probabilities of various N-grams in Google books: https://books.google.com/ngrams/



*Searched on 19/11/2020 at https://books.google.com/ngrams/

Durham University

Bag of Words with N-grams

N=1: This is a sentence — Unigrams:

- This
- is
- a
- sentence

N=2: This is a sentence — Bigrams:

- This is
- is a
- a sentence

N=3: This is a sentence — Trigrams:

- This is a
- is a sentence

Consider N-grams as tokens

Convert to numerical vectors

Durham University

Questions?

Durham University