MANCHESTER 1824
The University of Manchester

COMP34711
Week 2

(Simple)

Language Models and Representations

Goran Nenadic

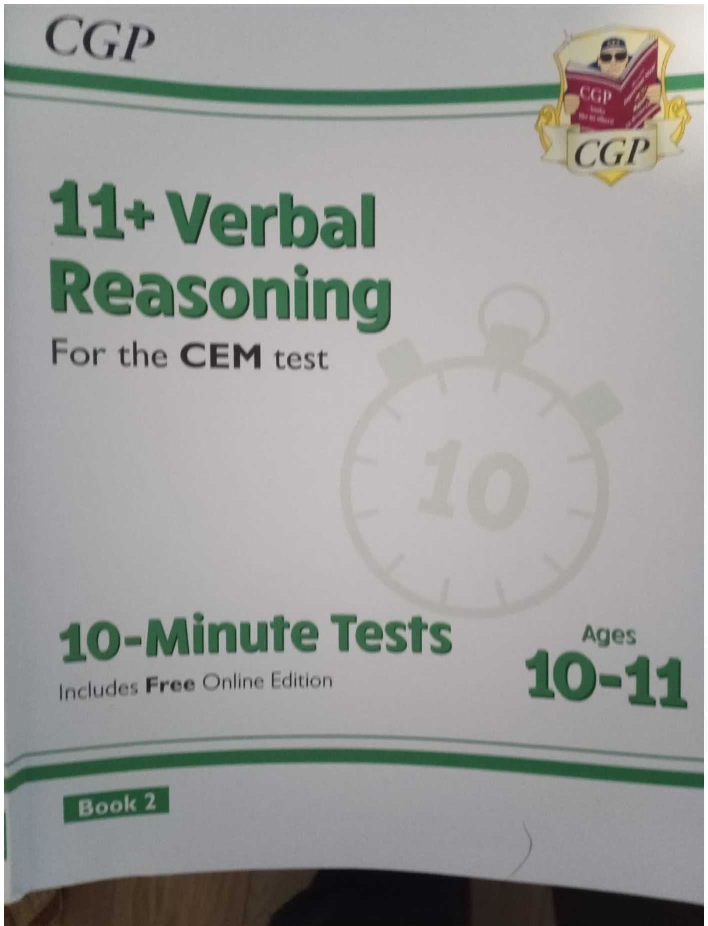
MANCHESTER
1824
The University of Manchester

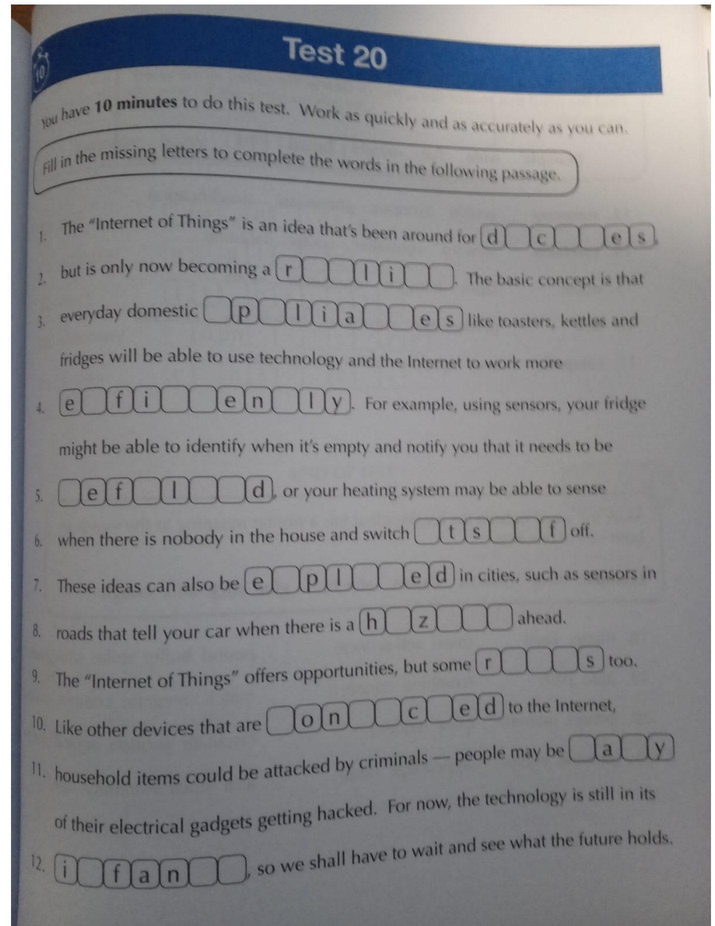
“Learning” language

The “Internet of Things” is an idea that’s been around for __________, but is only now becoming a __________ . The basic concept is that everyday domestic __________ like toasters, kettles and fridges will be able to use technology and the Internet to work more _____________. For example, using sensors, your fridge might be able to identify when it’s empty and notify you that it needs to be _________, or your heating system may be able to sense when there is nobody in the house and switch ________ off. These ideas can also be _________ in cities, such as sensors in roads that tell your car when there is a ________ ahead. The “Internet of Things” offers opportunities, but some _________ too.

MANCHESTER
1824
The University of Manchester

“Learning” language





Test 20

You have **10 minutes** to do this test. Work as quickly and as accurately as you can.

Fill in the missing letters to complete the words in the following passage.

1. The "Internet of Things" is an idea that's been around for d _ c _ _ e s
2. but is only now becoming a r _ _ l i _ _. The basic concept is that
3. everyday domestic _ p _ l i a _ _ e s like toasters, kettles and fridges will be able to use technology and the Internet to work more
4. e _ f i _ _ e n _ l y. For example, using sensors, your fridge might be able to identify when it's empty and notify you that it needs to be
5. _ e f _ l _ _ d, or your heating system may be able to sense
6. when there is nobody in the house and switch _ t s _ _ f off.
7. These ideas can also be e _ p l _ _ e d in cities, such as sensors in
8. roads that tell your car when there is a h _ z _ _ _ ahead.
9. The "Internet of Things" offers opportunities, but some r _ _ _ s too.
10. Like other devices that are _ o n _ _ c _ e d to the Internet,
11. household items could be attacked by criminals — people may be _ a _ y of their electrical gadgets getting hacked. For now, the technology is still in its
12. i _ f a n _ _, so we shall have to wait and see what the future holds.

Language modelling

- An (often probabilistic) language model = a function that assigns a probability over a piece of text so that ‘natural’ pieces have a larger probability

P(”the the the the the”) – pretty small

P(“the cat in the hat”) – pretty big

P(“the cat in the hat”) > *P*(“the hat in the cat”)

P(“the cat in the hat”) ?? *P*(“the hat on the cat”)

Language models and representations

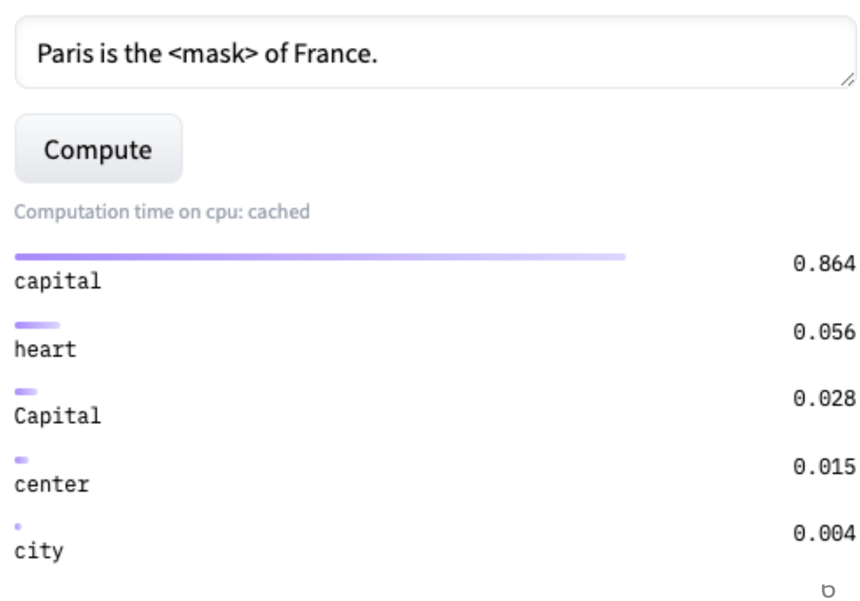
- *Model* = a “simplified”, abstract representation of something, often in a computational form
  - e.g. tossing a coin
  - e.g. probabilistic model of whether forecast
- How do we represent a piece of text (e.g. word, sentence, paragraph, document, corpus)?
  [use ‘document’ as an umbrella term here]

Language models and representations

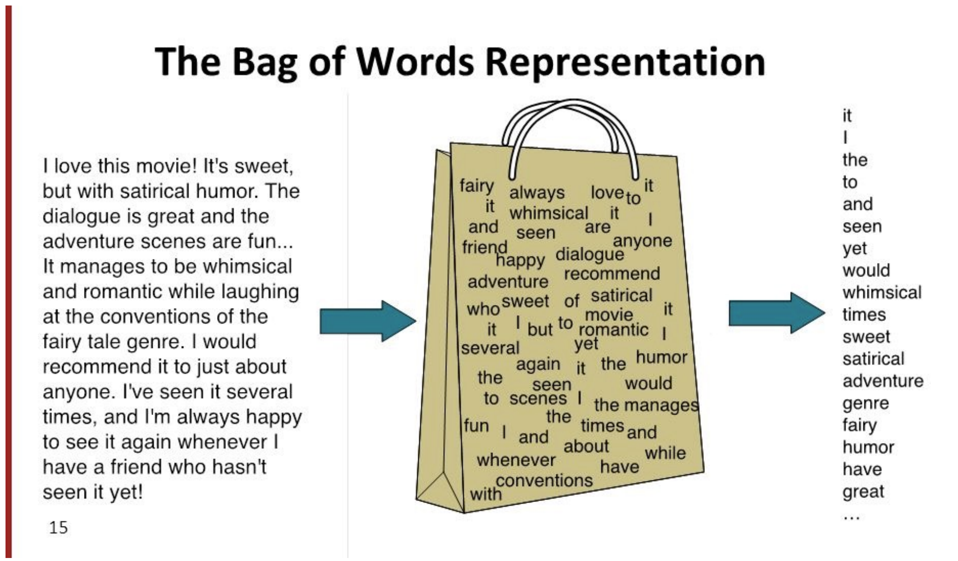
https://huggingface.co/roberta-base

https://huggingface.co/bert-base-cased

https://huggingface.co/bert-base-chinese

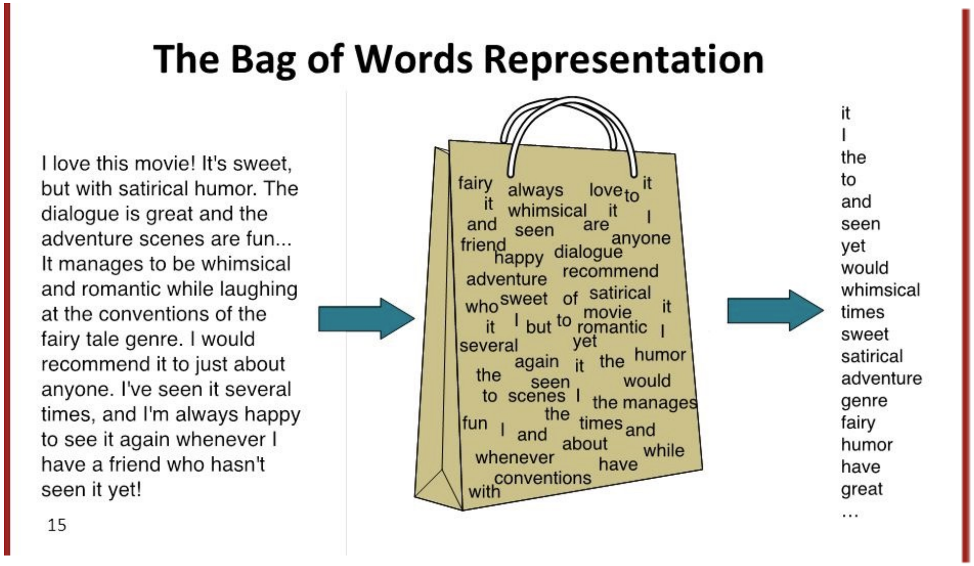


BoW representation



- Simplest model: “bag of words” (BoW)
  - Reduce each document into a bag of words
- Bag is a set with repetitions, but it could be used with binary indicators too.

BoW representation



- Note on terminology:
  - Bag of terms
  - Bag of tokens
  - Later, we will talk about bag of stems, etc.

BoW representation

- Questions:
  - Is the meaning lost without order?
    John is quicker than Mary. vs *Mary is quicker than John.*
  - What about negations?
  - Are all words equally important?
  - Is the meaning lost without context? Ambiguities?
  - Would it work for all languages?
  - What about (important) combinations of words?

- BoW is however efficient
  - Used in many NLP systems

BoW representation

- **Use all words? Or skip some?**
  - How about so-called **stop-words** (the, and, of, it, …)?
  - Most frequent ones?
- **Do we want to “rank” them?**
  - Give some weight to more important ones
  - How do you decide on importance/relevance?

- Let’s see some linguistic hypotheses and theories

Frequency of words

- Some words are very frequent
  - e.g. "the", "of", "to"
- Many words are less frequent
  - e.g. "bazinga"
  - ~50% terms appears once
- Zipf's Law
  - frequency of any word in a given collection is inversely proportional to its rank in the frequency table.

George Zipf
1902-1950

MANCHESTER
1824
The University of Manchester

Zipf's law

Wikipedia abstracts

→ 3.5M En abstracts

$r \times P_r \cong const$ →

$r \times freq_r \cong const$

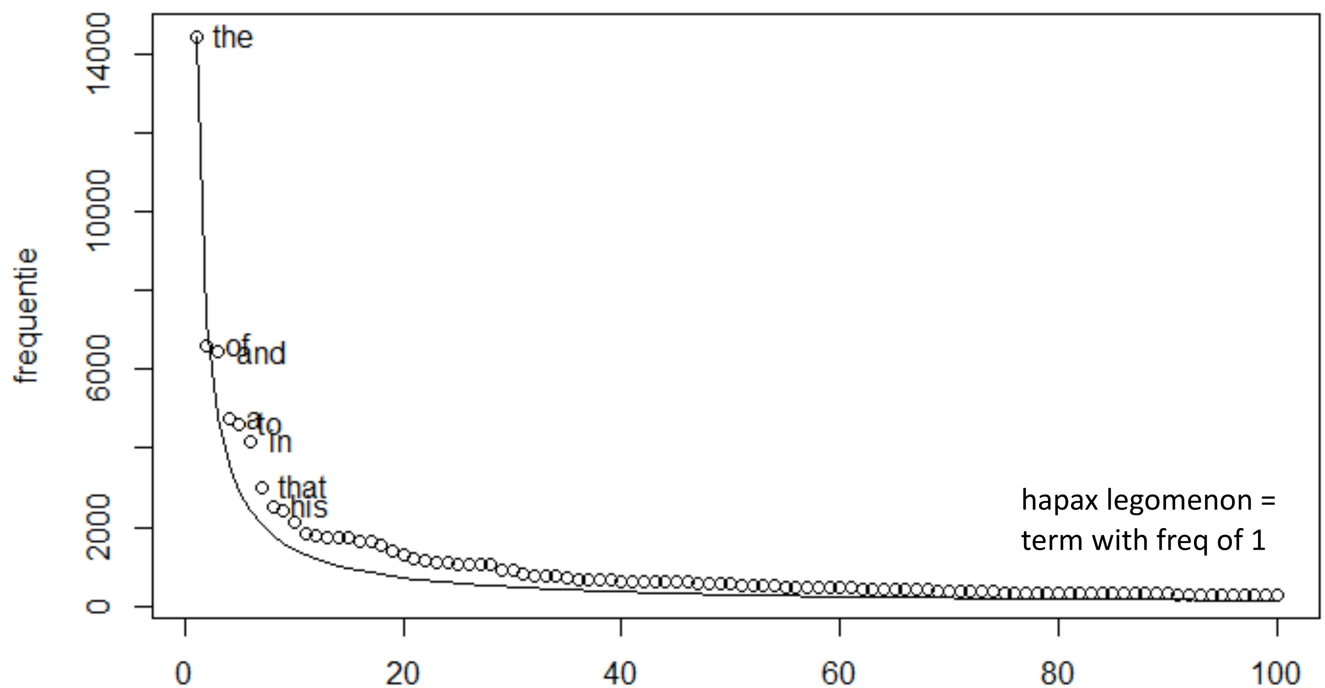
Term	Rank	Frequency	r x freq
the	1	5,134,790	5,134,790
of	2	3,102,474	6,204,948
in	3	2,607,875	7,823,625
a	4	2,492,328	9,969,312
is	5	2,181,502	10,907,510
and	6	1,962,326	11,773,956
was	7	1,159,088	8,113,616
to	8	1,088,396	8,707,168
by	9	766,656	6,899,904
an	10	566,970	5,669,700
it	11	557,492	6,132,412
for	13	493,374	5,970,456
as	14	480,277	6,413,862
on	15	471,544	6,723,878
from	16	412,785	7,073,160

Think about rare words:

- spelling errors
- names
- emails
- codes

The University of Manchester

Zipf's law

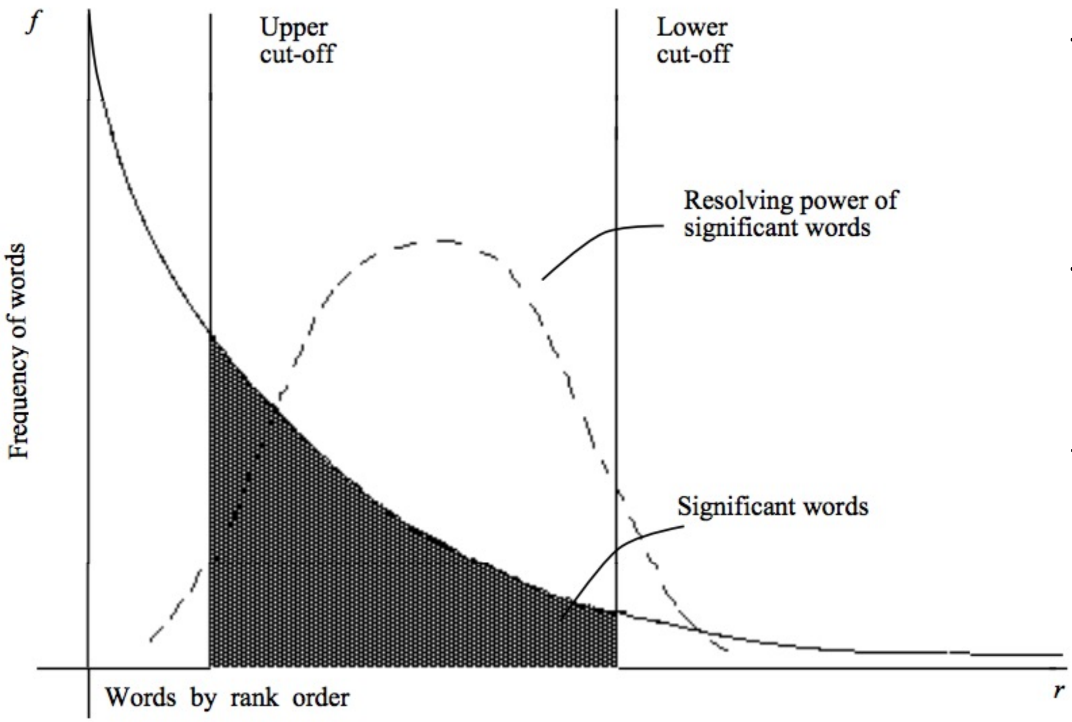


hapax legomenon =
term with freq of 1

$$r \times Pr \cong const$$

r = rank of term according to frequency
Pr = probability of appearance of term

Luhn's hypothesis



The words exceeding the upper cut-off are considered to be common.

Those below the lower cut-off rare, and therefore not contributing significantly to the content of the article.

Hans Peter Luhn
1896-1964

http://www.dcs.gla.ac.uk/Keith/pdf/Chapter2.pdf

LUHN, H.P., 'The automatic creation of literature abstracts', IBM Journal of Research and Development, 2, 159-165 (1958).

MANCHESTER 1824
The University of Manchester

Removing stop words

- Highly frequently occurring words have low distinguishing power for representing documents
  - Top 30 words account for ~30% of mentions
  - Closed class function words (the, and, of, it, …)

- Therefore, such words are often filtered oiut
- **Stop-word lists**
  - E.g. https://gist.github.com/sebleier/554280
  - Different lists for different applications, domains, etc.

[But not every application would benefit from removing stop words]

Vector representation

- Vector representation is a way to implement the BoW model.

- How to represent documents as vectors?
- Vocabulary *V* = set of terms left after pre-processing (tokenization, stop-word removal, case folding, etc.)
- Each document is represented as a |*V*|-dimensional vector:

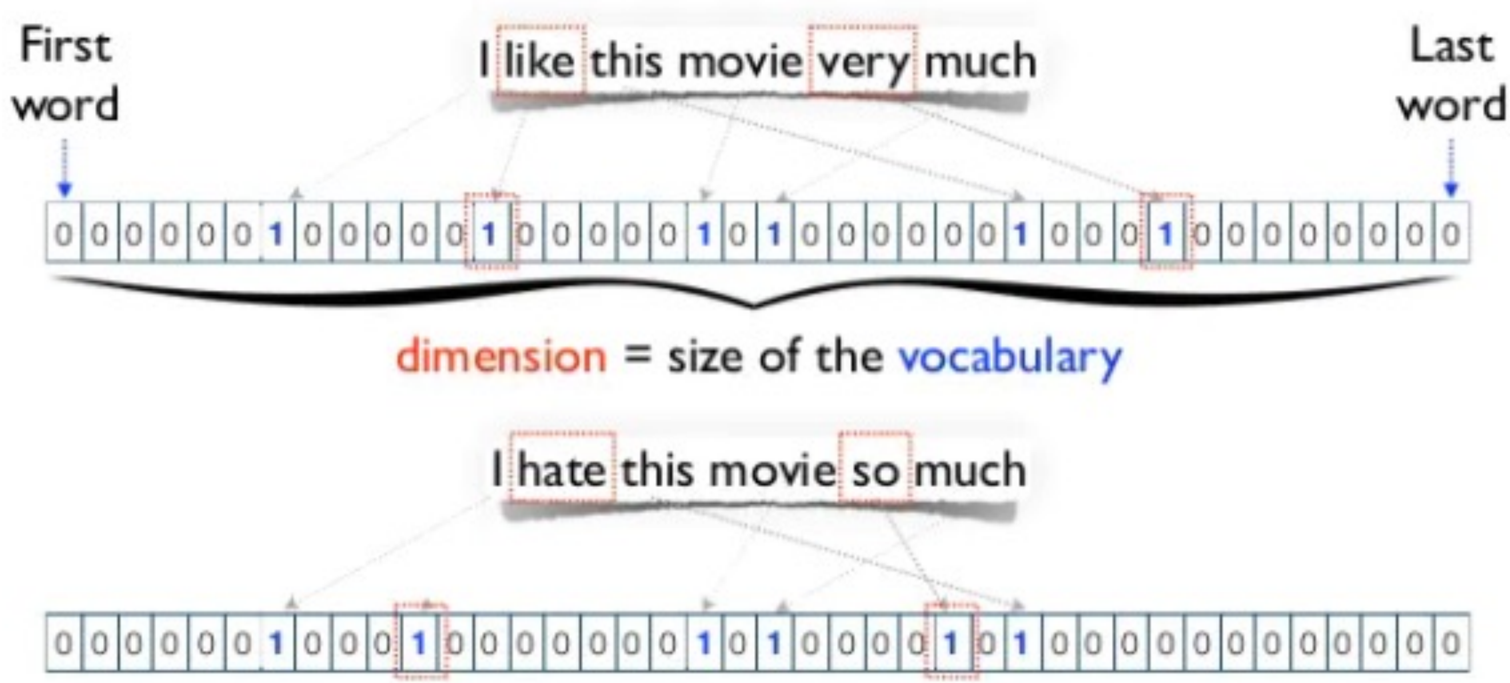
$$d = [w_1 , w_2 , . . . , w_{|V|}]$$

w_i is a <u>weight</u> of term *i* in the vocabulary

- Very high-dimensional representation
  - Millions of dimensions
  - These are very **sparse** vectors - most entries are zero

Weights

- Represent terms as an **incidence** (0, 1)



The University of Manchester

Weights

- Would a **frequency** vector be better?
  - Intuitively: if a term occurs many times in a document, is that word more important?
  - But:
    - Almost all documents will have (many) determiners?
    - Rare terms are more discriminative than frequent terms
    - Documents have different lengths: will clearly affect the counts

- Raw term frequency is not what we want:
  - E.g. term that appears 10 times is more important than a term with 1 occurrence *but not 10 times!*

Importance does not increase proportionally with term frequency

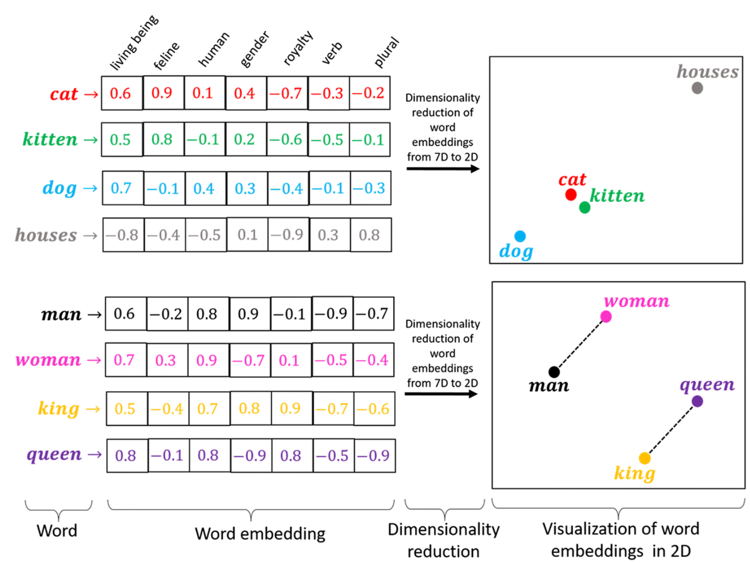
Weights

- We will consider different weights when we look into Information Retrieval next week
  - E.g. *inverse document frequency* (in how many documents this word appears)

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

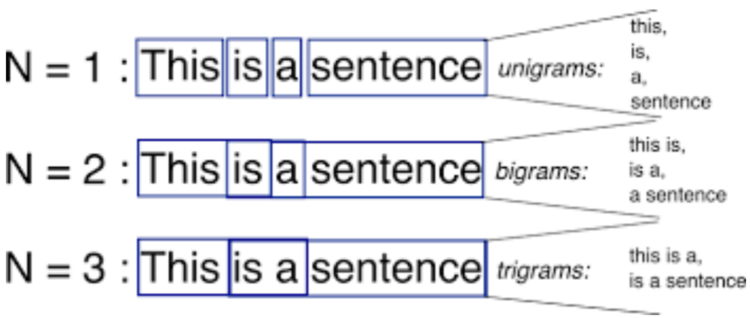
$tf_{i,j}$ = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

- Note again: these are very sparse models. Can we get more **dense** models to represent words and thus documents?
  - We will look later at *word embeddings*.



Combinations of words?

- Why only single words? Maybe a bag of n-grams?
  - Can n-grams represent some context?
- Reminder **n-grams**
  - a contiguous sequence of n *words* from a given sequence of text
  - 1-gram = “unigram”; “bigram” = 2-gram; “trigram” = 3-gram, etc.
  - Frequency of longer n-grams is often quite low

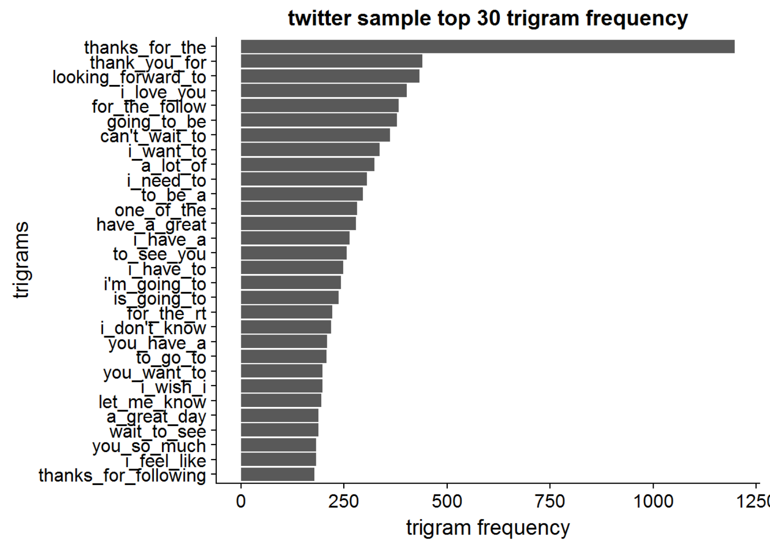


- VSM representation with n-grams?

Combinations of words

3–4-grams word

	Word	Count			Word	Count	
1	one of the	303,265	•••	11	to be a	100,285	•••
2	as well as	291,495	•••	12	a number of	98,337	•••
3	part of the	151,966	•••	13	is one of	90,869	•••
4	I do n't	148,894	•••	14	of the most	90,028	•••
5	a lot of	145,674	•••	15	out of the	87,383	•••
6	the United States	144,719	•••	16	the use of	82,268	•••
7	be able to	132,872	•••	17	you want to	81,888	•••
8	in order to	128,631	•••	18	there is a	80,923	•••
9	some of the	119,133	•••	19	the number of	77,974	•••
10	the end of	118,539	•••	20	end of the	77,110	•••



Trigrams

Frequency	Token
42030	THE U. S.
27260	IN NINETEEN EIGHTY
24165	CENTS A SHARE
18233	NINETEEN EIGHTY SIX
16786	NINETEEN EIGHTY SEVEN
15316	FIVE MILLION DOLLARS
14943	MILLION DOLLARS OR
14517	MILLION DOLLARS IN
12327	IN NEW YORK
11981	A YEAR EARLIER

Probabilistic language models

- Other ways to model text data?
- There is a lot about uncertainty and probability in how we use languages
- For example, is word ‘running’ more likely to follow word ‘rabbit’ than word ‘flying’

  I saw a rabbit <u>running</u> yesterday.

  I saw a rabbit <u>flying</u> yesterday.

- **How do we know that?**

Probabilistic language models

$$p(\text{"I saw a rabbit running yesterday"}) = ?$$

$>$ $$p(\text{"I saw a rabbit flying yesterday"}) = ?$$

$$p(\text{“I} \rightarrow \text{saw} \rightarrow \text{a} \rightarrow \text{girl} \rightarrow \text{running} \rightarrow \text{yesterday})$$

$>$ $$p(\text{“a} \rightarrow \textit{girl} \rightarrow \text{saw} \rightarrow \text{I} \rightarrow \text{running} \rightarrow \text{yesterday})$$

A **language model** (LM) measures the probability of natural language utterances, giving higher scores to those that are more common, more grammatical, more “natural”.

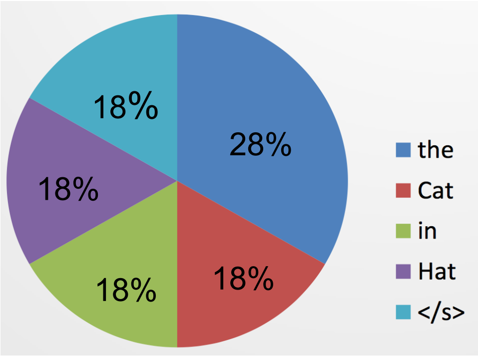
MANCHESTER
1824
The University of Manchester

Unigram Language Model

Consider the sequence **S = w_1 w_2 … w_n** *as a sequence of independent unigrams (single tokens)*

$$P(S) = P(w_1) * P(w_2) * \dots * P(w_n) = \prod_i P(w_i)$$

$$p(\text{"the cat in the hat </s>"})$$
$$= p(\text{"the"})\,p(\text{"cat"})\,p(\text{"in"})\,p(\text{"the"})\,p(\text{"hat"})\,p(\text{"</s>"})$$
$$= 0.28 \times 0.18 \times 0.18 \times 0.28 \times 0.18 \times 0.18$$



$$p(\text{"the hat in the cat </s>"}) = ?$$

Unigram Language Models

- Note: we assumed here that words appeared **independently (and identically distributed, *iid*)** of each other
  - That can't be true!
  - But it works pretty well (for some applications)

- How do we find these probabilities?
  - Use **corpus frequencies** as relative counts – count how many times they appear in a real-world dataset.
- Parameters of unigram LMs = word probabilities estimated from a corpus
  - How many?

Unigram Language Models

- How to deal with unseen (**out-of-vocabulary, OOV**) words that may appear in a sentence we want to estimate the probability for?
  - E.g. misspellings or new words
- Go with *p(OOV) = 0?* Harsh?
- Or somehow to estimate these probabilities
  - Same for all OOV words?
  - E.g. **add-one smoothing**
  - Other smoothings – later.

$$P(w) = \frac{\#w + 1}{\sum_{w' \in D}(\#w' + 1)}$$

Unigram Language Models

- Key issue: complete loss of order information with the unigram model

$$p(\text{"the the the the the </s>"}) > p(\text{"the cat in the hat </s>"})$$

$$p(\text{"cat the hat in the </s>"}) = p(\text{"the cat in the hat </s>"})$$

- How to add some sequencing into modelling?
- Chain rule

Chain Rule

- Chain rule in probability theory:
  - Calculate the *joint distribution* of a set of random variables using only conditional probabilities.

e.g. S = the cat in the hat

$$p(\mathrm{S}) = p(\text{the}) \cdot p(\text{cat}|\text{the}) \cdot p(\text{in}|\text{the cat}) \cdot p(\text{the}|\text{the cat in}) \cdot p(\text{hat}|\text{the cat in the})$$

So, for utterance: $w_1\ w_2\ w_3\ \ldots\ w_L$

$$\boxed{p(w_1, w_2, w_3, \ldots, w_L)}$$

$$= p(w_1)\,p(w_2|w_1)\,p(w_3|w_1,w_2)\,p(w_4|w_1,w_2,w_3)\cdots p(w_L|w_1,w_2,\ldots,w_{L-1})$$

$$= \prod_{k=1}^{L} p(w_k|w_1,w_2,\ldots,w_{k-1})$$

Chain Rule

- Approximation: take (only) a fixed number of words before the current one

$$p(w_{1:L}) = \prod_{k=1}^{L} p(w_k | w_{k-N+1:k-1})$$

- Unigram model (N=0)

$$p(w_{1:L}) = \prod_{k=1}^{L} p(w_k | w_{k-N+1:k-1}) = \prod_{k=1}^{L} p(w_k)$$

- Bigram model (N=1)

$$p(w_{1:L}) = \prod_{k=1}^{L} p(w_k | w_{k-N+1:k-1}) = \prod_{k=1}^{L} p(w_k | w_{k-1})$$

MANCHESTER 1824
The University of Manchester

Bigram Model

$$p\left(w_{1:L}\right)=\prod_{k=1}^{L} p\left(w_k \mid w_{k-N+1:k-1}\right)=\prod_{k=1}^{L} p\left(w_k \mid w_{k-1}\right)$$

$$p(\text{"the cat in the hat </s>"})$$
$$= p(\text{"the"})\,p(\text{"cat"}\mid\text{"the"})\,p(\text{"in"}\mid\text{"cat"})\,p(\text{"the"}\mid\text{"in"})\,p(\text{"hat"}\mid\text{"the"})\,p(\text{"</s>"}\mid\text{"hat"})$$

Bi-gram model corresponds to a Markov chain. It assumes that a state depends only on its previous state not on the other events that occurred before it.

MANCHESTER
1824
The University of Manchester

Trigram Model

w_1 $p(w_1)$

w_1 w_2 $p(w_2|w_1)$

w_1 w_2 w_3 $p(w_3|w_2,w_1)$

~~w_1~~ w_2 w_3 w_4 $p(w_4|w_3,w_2)$

~~w_1 w_2~~ w_3 w_4 w_5 $p(w_5|w_4,w_3)$

~~w_1 w_2 w_3~~ w_4 w_5 w_6 $p(w_6|w_5,w_4)$

~~w_1 w_2 w_3 w_4~~ w_5 w_6 w_7 $p(w_7|w_6,w_5)$

~~w_1 w_2 w_3 w_4 w_5~~ w_6 w_7 w_8 $p(w_8|w_7,w_6)$

$\prod$

N-gram Models

- N-gram model assumes

$$p\left(w_k \middle| w_{1:k-1}\right) = p\left(w_k \middle| w_{\boxed{k-N+1:k-1}}\right)$$

We only care about **N-1** previous "states" (words) of w_k

For example: If we focus on the 10th word in a sequence:

$$\ldots \quad w_6 \quad w_7 \quad w_8 \quad w_9 \quad w_{10} \quad w_{11} \quad w_{12} \quad w_{13} \quad \ldots$$

N=4: $p\left(w_{10} \middle| w_{1:9}\right) = p\left(w_{10} \middle| w_{7:9}\right)$

N=3: $p\left(w_{10} \middle| w_{1:9}\right) = p\left(w_{10} \middle| w_{8:9}\right)$ → trigram model

N=2: $p\left(w_{10} \middle| w_{1:9}\right) = p\left(w_{10} \middle| w_{9}\right)$ → bigram model

N=1: $p\left(w_{10} \middle| w_{1:9}\right) = p\left(w_{10}\right)$ → unigram model

The University of Manchester

N-gram Probabilities

$$\text{N-gram model: } p\left(w_{1:L}\right) = \prod_{k=1}^{L} \boxed{p\left(w_k \middle| w_{k-N+1:k-1}\right)}$$

- How to estimate the *N-gram conditional probability* using a **training corpus**
  - Based on word counts.
  - Using a statistical model, e.g., hidden Markov model and Gaussian mixture model.
  - Using a machine learning model, e.g., neural network.

Count-based Estimation

- Estimate N-gram probability based on counts.

For the bigram case:

$$p\left(w_k \middle| w_{k-1}\right) = \frac{count\left("w_{k-1}w_k"\right)}{\sum_w count\left("w_{k-1}w"\right)} = \frac{count\left("w_{k-1}w_k"\right)}{count\left("w_{k-1}"\right)}$$

denoted by

$$\frac{C\left(w_{k-1:k}\right)}{C\left(w_{k-1}\right)}$$

Generalise to N-gram case

$$p\left(w_k \middle| w_{k-N+1:k-1}\right) = \frac{C\left(w_{k-N+1:k}\right)}{C\left(w_{k-N+1:k-1}\right)}$$

Count-based Estimation

- Example:

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

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

P(I | <s>) = 2/3 = 0.67
P(Sam | <s>) = 1/3 = 0.33
P(am | I) = 2/3 = 0.67

P(</s> | Sam) =
P(Sam | am) =
P(do | I) =

P(not | do) =
P(eggs | green) =

Count-based Estimation

- Example:

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

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

P(I | <s>) = 2/3 = 0.67
P(Sam | <s>) = 1/3 = 0.33
P(am | I) = 2/3 = 0.67

P(</s> | Sam) = 1/2 = 0.5
P(Sam | am) = 1/2 = 0.5
P(do | I) = 1/3 = 0.33

P(not | do) = 1/1 = 1
P(eggs | green) = 1/1 = 1

Count-based Estimation

- Get raw bigram frequencies from a larger corpus:

count(*want chinese*) = 6; *count*(*to spend*) = 211

	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

From 9220 sentences
(but can/should get billion-word corpus)

MANCHESTER 1824
The University of Manchester

Count-based Estimation

- Noramalise with unigram counts:

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

N-gram Probabilities

$$\text{N-gram model: } p\left(w_{1:L}\right) = \prod_{k=1}^{L} p\left(w_k \middle| w_{k-N+1:k-1}\right)$$

- Example:
  p("I want to eat Chinese food") = *p*(*I*)**p(want | I) * . . .*

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Count-based Estimation

- Examples – similar process for N-grams

$$p\left(\text{"want"}\middle|\text{"I"}\right) = \frac{C\left(\text{"I want"}\right)}{C\left(\text{"I"}\right)} \text{ (bi-gram examle)}$$

$$p\left(\text{"machine"}\middle|\text{"support vector"}\right) = \frac{C\left(\text{"support vector machine"}\right)}{C\left(\text{"support vector"}\right)} \text{ (tri-gram example)}$$

Homework and further reading

- Do your homework (estimated time ~ 2 hours)
  - Practice tokenisation, **sentence segmentation**, n-gram extraction, n-gram modelling
- Ch3 in: Jurafsky, Martin: *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition*
  - https://web.stanford.edu/~jurafsky/slp3/
- Ch3 *in:* Bird, Klein, Loper: *Natural Language Processing with Python: analyzing text with the Natural Language Toolkit*
  - http://www.nltk.org/book/