

Natural Language Processing

Lecture

Jonas Rieger

rieger@statistik.tu-dortmund.de

WiSe 2023/24



Organization

Formalities

- Lecture + Tutorial (4+2 hours, 9 CP), Modules: BS 14, MS 6/7, MD E1/2, ME 7
- English, in person, Python (or R, but we suggest Python)
- Written exam (120 minutes = 60 minutes theory + 60 minutes programming)
 - Tue 04.02.2025 11:00–13:00 (~~HGI HS6~~ → changed to OH14 E23 and Chemie HS 3) — view: Tue 11.02.2025 11:00–12:00 (CDI foyer)
 - Wed 26.02.2025 9:00–11:00 (HGII HS5) — view: Tue 04.03.2025 11:00–12:00 (CDI foyer)

Lecture

- Tue 16:15–17:45 (CT ZE HS15) — first lecture: 08.10.2024
- Wed 16:15–17:45 (Chemie HS3) — last lecture: 30.01.2025

Tutorial/Exercise

- Tue 12-14 or Wed 10-12 (each CDI 121) — first/last: 15.10.2024/30.01.2025

Contents

- Text data handling (e.g., encoding) and visualizations
- **Preprocessing:** tokenization, stopwords, stemming, lemmatization, n-grams, Regex, tf-idf, Zipfs law, filtering
- Part-of-speech (POS) tagging and named entity recognition (NER)
- Sentiment analysis
- (Static) embeddings (**word2vec**, fastText, GLoVE, ...)
- (Probabilistic) topic models (pLSA, **LDA**, CTM, STM, ...)
- Neural and transformer-based topic models (e.g., **BERTopic**)
- Transformer-based (pretrained) language models
(e.g., BERT, **(Chat)GPT**, ...)
 - Fine-tuning, Parameter-Efficient Fine-Tuning (PEFT) & Low-Rank Adaptation (LoRA), few-shot learning, near-domain training, transfer learning, ...

Links

- Moodle-room with password "AttentionIsAllYouNeed":
<https://moodle.tu-dortmund.de/course/view.php?id=47532>
- Moodle Link can also be found in the LSF

Literature and material (as preparation)

- Machine Learning for Text, DOI:10.1007/978-3-319-73531-3
- Text Mining with R, <https://www.tidytextmining.com/>
- R packages: see <https://www.tidytextmining.com/preface.html>
- Python libraries: NLTK, Gensim, spaCy, CoreNLP, TextBlob, Scikit-learn, torch, transformers, ...
- Online class (StanfordNLP): <https://web.stanford.edu/class/cs224n/>
- Illustrations and explanations of transformers:
 - Jay Alammar: <https://jalammar.github.io/>
 - Marten Grootendorst: <https://newsletter.maartengrootendorst.com/>
- Book from Marten Grootendorst and Jay Alammar: <https://www.llm-book.com/>
- ACL Anthology: <https://aclanthology.org/>

1 Introduction

2 Fundamentals

3 Text Data as Features

4 Topic Models

5 Neural Networks

6 Embedding Methods

7 Language Models

8 Training Methods

9 Advanced Training Methods

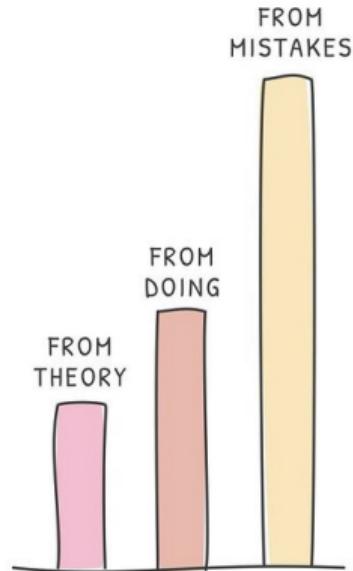
10 Evaluation of NLP Systems

11 Recent Architectures

Approach

- try to understand
- (ask questions)
- try out
- do mistakes
- (ask questions)
- do it right
- see the system doing weird things
- understand *ideas behind*
- get a good grade

HOW MUCH YOU LEARN



Buzz words

- big data
- unstructured data
- artificial intelligence (AI)
- machine learning (ML)
- deep learning

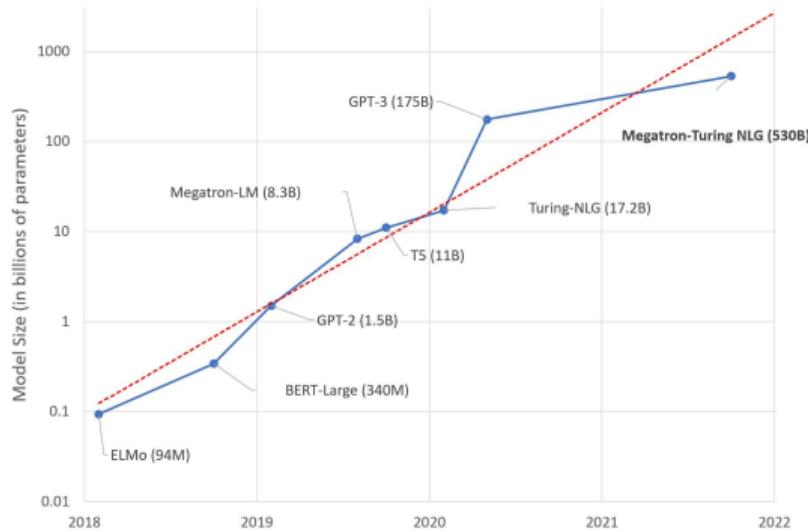
Example systems using text as data

- DeepL / GoogleTranslator / ...
- Siri / Alexa / ...
- labeled data needed
- structured commands (not really unstructured understanding by AI)
- you all know ChatGPT ...

Numbers (likely no longer accurate)

- (more than) 2.5 billion GB data created every day
- 80% unstructured, increasing?
- examples:
 - 750 million daily newspapers
 - 156 million emails every minute
 - 95 million photos/videos shared on Instagram every day

Language models



SOTA models are getting bigger and bigger. GPT-4 is said to be based on 8 models each having 220 billion parameters → 1.76 trillion parameters

Glossary |

natural language processing (NLP) A field of AI focused on enabling computers to understand, interpret, and produce human languages.

state-of-the-art (SOTA) The highest level of performance achieved in a particular task.

named entity recognition (NER) A subtask of NLP that classifies named entities into predefined categories.

part-of-speech (POS) tagging The process of marking each word in a text as corresponding to a particular part of speech.

question answering (QA) An NLP task where the system must generate a human-like answer to a given question.

machine translation (MT) The application of computers to translating texts from one natural language to another.

This glossary was created with the assistance of ChatGPT.

Glossary II

supervised machine learning (SL) A machine learning paradigm where the model is trained on a labeled dataset.

unsupervised machine learning Machine learning with datasets that don't have labeled responses.

bag-of-words (BOW) A representation of text that describes the presence of words within the document.

document-term matrix (DTM) A matrix describing the frequency of terms in a collection of documents.

neural network (NN) A set of algorithms, modeled loosely after the human brain, designed to recognize patterns in data.

multi-layer perceptron (MLP) A type of neural network consisting of at least three layers of nodes in a directed graph.

This glossary was created with the assistance of ChatGPT.

Glossary III

recurrent neural network (RNN) A type of neural network that processes sequences of data with temporal dependencies.

feed forward (FF) neural network A type of neural network in which the connections between the nodes do not form a cycle. It moves in only one direction, from input to output.

language model (LM) A type of model that is trained to predict the probability of a sequence of words in a sentence or document.

neural language model (NLM) A language model that uses neural network architectures to predict the probability of a sequence of words.

large language model (LLM) An extensive neural network language model with a large number of parameters, capable of understanding and generating human-like text based on the input it receives.

This glossary was created with the assistance of ChatGPT.

Glossary IV

multi-head attention (MHA) A mechanism in neural networks that allows the network to focus on different parts of the input, similar to the way humans pay attention to specific parts of the input when understanding languages, reading, etc.

pre-trained language model (PLM) A language model trained on a large dataset and fine-tuned for specific tasks.

fine-tuning (FT) Adjusting the parameters of a pre-trained model for a specific task.

zero-shot learning A machine learning scenario where the model is asked to solve a task it has not seen during training.

few-shot (FS) learning (FSL) A machine learning scenario where the model is trained to learn information with a very small amount of data.

This glossary was created with the assistance of ChatGPT.

Glossary V

parameter-efficient fine-tuning (PEFT) A strategy for fine-tuning pre-trained models in a way that requires relatively few updates to the model parameters, making it computationally cheaper.

in-context learning (ICL) A type of learning where a model uses the context of a new task to quickly adapt its knowledge from previous tasks to perform well on the new task.

pattern-verbalizer pair (PVP) A technique in few-shot learning where a pattern and a verbalization are created to convert a task into a form that a pre-trained language model can understand and respond to correctly.

This glossary was created with the assistance of ChatGPT.

Prompt: Please add short explanations of each concept to the following glossary written in LaTeX. Please add a short acknowledgement that I created the glossary with your help in the bottom of each slide. Please split the glossary in a number of slides so that it will be decently displayed. <LaTeX-Code containing the items without explanations>
I had to re-order the items manually and the splitting does not work very well

PETapter

Leveraging PET-style classification heads for modular few-shot parameter-efficient fine-tuning

- Paper: <https://openreview.net/forum?id=Dq8VpnQ4M6>
- Talk: find the slides in moodle

2

Fundamentals

- Data
- Regular Expressions
- Preprocessing
- Part-of-Speech Tagging
- Named Entity Recognition
- From Text to Numbers
- Evaluating Language Models

Input, string, character, word

Wisdom... comes not from age, but from education and learning.

— Anton Chekhov

- the complete quote is the input
- this may be stored as one string "Wisdom... comes ..."
- or as (tokenized) vector of strings "Wisdom", "comes", ...
- where each token is an instance of a word
- W, i, s, d, o, m, ., , ... are characters

Text encoding

- characters are grouped into a character set,
where each character is associated with a number
- characters are stored as bytes
- encoding brings you the key to unlock the code:
- the mapping between the bytes and the characters in the character set
- popular encodings (sometimes under other labels):
 - ASCII
 - ANSI
 - Latin-1
 - UTF-8
 - UTF-16
- whenever possible: stick to UTF-8 (covers all likely characters)

Resource: Why Encoding?

<https://www.w3.org/International/questions/qa-what-is-encoding.en.html>

File format

- do not mix up encoding with file format
- csv (comma separated value) ok
- xml (extensible markup language) good
- json (javascript object notation) best (and better to read)
- pdf (portable document format) labor-intensive
 - depends on whether originally an image or document
- doc(x) should be possible somehow
- txt totally fine, but unstructured
- images have to be converted to pdf
- always try to get your hands on json, xml or csv

JSON vs. XML

The screenshot shows two code editors side-by-side. The left editor is titled "personal.json" and contains the following JSON code:

```

1  "schemas": "personal-schema.json",
2  "personnel": [
3    "person": [
4      {
5        "id": "Big.Boss",
6        "name": {
7          "family": "Boss",
8          "given": "Big"
9        },
10       "email": "chief@oxygentools.com",
11       "link": {
12         "subordinates": [
13           "one.worker",
14           "two.worker",
15           "three.worker",
16           "four.worker",
17           "five.worker"
18         ]
19       }
20     },
21     {
22       "id": "one.worker",
23       "name": {
24         "family": "Worker",
25         "given": "One"
26       },
27       "email": "one@oxygentools.com",
28       "link": {"manager": "Big.Boss"}
29     },
30     {
31       "id": "two.worker",
32       "name": {

```

The right editor is titled "personal.xml" and contains the following XML code:

```

<personnel>
  <person>
    <id>Big.Boss</id>
    <name>
      <family>Boss</family>
      <given>Big</given>
    </name>
    <email>chief@oxygentools.com</email>
    <link>
      <subordinates>one.worker two.worker</subordinates>
    </link>
  </person>
  <person>
    <id>one.worker</id>
    <name>
      <family>Worker</family>
      <given>One</given>
    </name>
    <email>one@oxygentools.com</email>
    <link>
      <manager>Big.Boss</manager>
    </link>
  </person>
  <person>
    <id>two.worker</id>
    <name>
      <family>Worker</family>
      <given>Two</given>
    </name>
    <email>two@oxygentools.com</email>
    <link>

```

<https://www.oxygenxml.com/>

(formerly) twitter



Barack Obama ✅ @BarackObama · 13. Okt.

Congrats to @JHUAPL on confirming we successfully moved an asteroid! The historic DART mission proved humanity could stop a threatening asteroid if needed — and shows what's possible when we come together and do big things.



Alexandria Ocasio-Cortez ✅ @AOC · 16. Okt.

All in a day's work:

- ✓ Reopening @nysci after we supported \$12 million in FEMA recovery 🌎
- ✓ Announcing our project for the 1st STEM & space-science preschool in the US, right in working-class Corona! 🚀
- ✓ Block party after our \$225k community investment in @QCHnyc 🎉



Elon Musk ✅ @elonmusk · 13. Okt.

20,000!!



Jean-Claude Juncker ✅ @JunckerEU · 17. Okt. 2019

Where there is a will, there is a #deal - we have one! It's a fair and balanced agreement for the EU and the UK and it is testament to our commitment to find solutions. I recommend that #EUCA endorses this deal.



Miguel Berger ✅ @MiguelBergerAA · 20. Nov. 2021

Productive exchange with 🇩🇪 Deputy PM & FM H.E. Fouad Hussein @iraqimofa about 🇩🇪 & 🇪🇬 elections and migration, appreciate that more than 430 Iraqis were already able to return safely from Belarus. @GermanyBahrain @GermanyOnMENA @almaniadiplomatic @GermanyInIraq

see <https://developer.twitter.com> for API



Arnold ✅ @Schwarzenegger · 12 Std.
Happy Halloween 💃 @TheSlyStallone



X API access

Find the right access for you

Free	Basic	Pro	Enterprise
<p>For write-only use cases and testing the X API</p> <ul style="list-style-type: none">• Rate-limited access to v2 posting and media upload endpoints• 1,500 posts per month - posting limit at the user level• 1,500 posts per month - posting limit at the app level• 1 app ID• Login with X• Access to Ads API• Free	<p>For hobbyists or prototypes</p> <ul style="list-style-type: none">• Rate-limited access to suite of v2 endpoints• 3,000 posts per month - posting limit at the user level• 50,000 posts per month - posting limit at the app level• 10,000 posts per month - read limit at the app level• 2 app IDs• Login with X• Access to Ads API• \$100 per month	<p>For startups scaling their business</p> <ul style="list-style-type: none">• Rate-limited access to suite of v2 endpoints, including search and filtered stream• 288,000 posts per month - posting limit at the user level• 300,000 posts per month - posting limit at the app level• 1,000,000 posts per month - read limit at the app level• 3 app IDs• Login with X• Access to Ads API• \$5,000 per month	<p>For businesses and scaled commercial projects</p> <ul style="list-style-type: none">• Commercial-level access that meets your and your customer's specific needs• Managed services by a dedicated account team• Complete streams: replay, engagement metrics, backfill, and more features• Access to Ads API• Monthly subscription tiers
Get started	Subscribe now	Subscribe now	Apply now

Newspaper, party manifestos, ...

```
<?xml version="1.0" encoding="UTF-8"?>
<records sourcoid="6742" sourcename="The New York Times" timestamp="2016-05-24T00:01:02.097">
  <record>
    <record_id>3SC6-U0C0-0024-J4S4</record_id>
    <source_id>https://www.nexis.com/docview/getDocForCuiReq?csi=280434&oc=00240&perm=1</source_id>
    <publishDate>19940105T000000Z</publishDate>
    <series>The New York Times</series>
    <series2>Section C; Page 1; Column 1; Living Desk</series2>
    <title>Off the Menu</title>
    <leadtext>Off the Menu, a new weekly column charting restaurant news and dining trends, begins this Sunday. By Tim Golden</leadtext>
    <fulltext>It's been a year since the New York Times' Sunday magazine section was redesigned, and the paper's editors have decided to add another column to the mix. The new column, "Off the Menu," will be written by Tim Golden, who has been covering the restaurant scene for the Times since 1991. The column will focus on the latest trends in the restaurant industry, from new menu items to changes in service and atmosphere. It will also feature reviews of new restaurants and food-related events. The column will begin this Sunday, January 5, at 10:00 AM. To read more about the column and its author, visit the Times website at www.nytimes.com.</fulltext>
  </record>
  <record>
    <record_id>3SC6-U0C0-0024-J4S5</record_id>
    <source_id>https://www.nexis.com/docview/getDocForCuiReq?csi=280434&oc=00240&perm=1</source_id>
    <publishDate>19940105T000000Z</publishDate>
    <series>The New York Times</series>
    <series2>Section A; Page 1; Column 4; Foreign Desk</series2>
    <title>Mexican Rebels Are Retreating; Issues Are Not</title>
    <author>By TIM GOLDEN, A Special to The New York Times</author>
    <leadtext>The peasant guerrilla army that burst into Mexican life with the New Year retreat support us. Some are against us. But people see the advance that our army has made." He said I</leadtext>
    <fulltext>But while the rebels' offensive appeared to be waning, it seemed clear that Mexico's support us. Some are against us. But people see the advance that our army has made." He said I</fulltext>
  </record>
</records>
```

- newspaper articles: xml from databases, e.g., LexisNexis
 - (newer) party manifestos at least accessible as pdf
 - (newer) speeches often accessible as json or xml
 - older documents often as pdf scans

Websites

A screenshot of a Chinese news website from October 19, 2022, featuring a red banner at the top with the text "党的二十大特别报道" (Special Report on the 20th National Congress of the CPC). Below the banner, a large headline reads "党的二十大报告在广东省水利厅党员干部、厅直系统全体党员中引发热烈反响 以新安全格局保障广东经济社会的高质量发展" (The report of the 20th National Congress of the CPC has sparked a strong response among the leading cadres and all party members of the Guangdong Provincial Water Resources Department and the affiliated units, ensuring the high-quality development of the province's economy and society through a new security pattern). The page includes several smaller articles and images related to the congress.

Scraping

```
<!DOCTYPE html>
<html class="detail-html" lang="en"> [event] [scroll]
  <head> [ ] </head>
  <body>
    <div class="wrapper ">
      <div class="main main-article"> [overflow]
        <div class="main-in clearfix">
          > <div class="fl main-l"> [ ] </div>
          <div class="fr main-r clearfix" style="background-color: rgb(255, 255, 255);> [event] [scroll]
            <div id="list" class="main-600 fl"> [overflow]
              <!--文章列表-->
              <div class="article">
                <h2 class="leading_title">党的二十大报告在广东省水利厅党员干部、厅直系统全体党员中引发热烈反响</h2>
                <h1>聚力实施水利高质量发展蓝图，开创发展新天地</h1>
                <h2></h2>
                <div class="info">
                  <span id="">来源：南方都市报</span>
                  [Leerraum]
                  <span id="">2022年10月19日</span>
                  <span id=""></span>
                  [Leerraum]
                  <span>版次：GA07</span>
                  [Leerraum]
                  <span>作者：陈燕</span>
                </div>
                <div class="text">
                  
                  <p class="pub_image_info"> [ ] </p>
                  > <p> [ ] </p>
                  <p>    统筹：向雪巍 李斌 采写：南都记者 陈燕 通讯员 粤公宣</p>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
```

Difficulties: homonyms

Homonym	Meaning 1	Meaning 2
address	to speak to	location
apple	company	fruit
die	to die	singular of dice
jam	traffic jam	jelly
light	opposite of heavy	brightness
mean	average	not nice
right	opposite of left	correct
rock	stone	genre

Difficulties: negation, sarcasm, irony

- This movie is not that bad.
- This movie never entertained me.
- The least interesting movie.
- This movie is the (newest) shit.
- I guess he is the next Einstein.

Difficulties: negation, sarcasm, irony

- This movie is not that bad.
- This movie never entertained me.
- The least interesting movie.
- This movie is the (newest) shit.
- I guess he is the next Einstein. (sarcasm?)

- not such a big problem in news media
- more of a problem in social media texts

Difficulties: background knowledge

The trophy does not fit into the suitcase.

- ① It is too big.
- ② It is too small.

Computer has difficulties to understand the coreference of *it*.

Difficulties: emojis

- especially when working with social media data
- how to handle emojis?

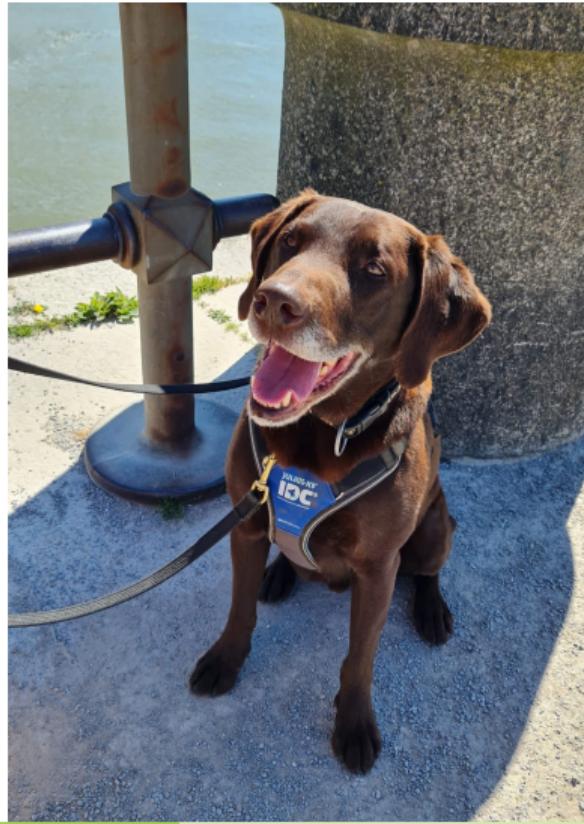
Difficulties: emojis

- especially when working with social media data
- how to handle emojis?
- depends on research question
- sometimes ok to remove
- often better to replace
- in R, e.g.,
 - textclean package (`replace_emoji`)
 - lexicon package (`hash_emojis`)

Regular expressions (Regex)

How to search in texts for:

- dog
- dogs
- Dog
- Dogs



Regex: disjunctions (OR link)

Digits/letters inside square brackets as options:

Pattern	Matches
dog	dog
Dog	Dog
[dD]og	dog, Dog
[0123456789]	any digit

Ranges:

Pattern	Matches
[A-Z]	an (single!) upper case letter
[a-z]	a (single!) lower case letter
[0-9]	any (single!) digit

Regex: negation in disjunctions

Negation:

- \wedge caret,
- caret refers to negation iff (if; and only if) first in square brackets

Pattern	Matches	Example
[^A-Z]	not an upper case letter	ABC_is a short form for alphabet.
[^Aa]	neither A nor a	ABC is a short form for alphabet.
[^A-Za-z]	not a letter	ABC_is a short form for alphabet.
[^A-Z a-z]	not a letter nor space	ABC is a short form for alphabet_
x^2	the pattern x^2	The square value of x is <u>x^2</u> .

Regex: disjunctions using the vertical |

There are a lot of synonyms for dog

- man's best friend
- doggo
- puppy
- ...

Pattern	Matches
dog man's best friend	both (complete) expressions
[dD]og [pP]uppy	dog, Dog, puppy, Puppy
a b c	= [abc] = [a-c]
four 4	four, 4

Regex: match groups using () multiple times using {}

It is possible to backreference specifying groups with ().

Pattern	Matches
(ABC)=\1	ABC=ABC
([A-Z])=\1	A=A, B=B, ...
(ABC abc)=\1	ABC=ABC, abc=abc
(ABC){3}	ABCABCABC
(ABC){3}=\1	ABCABCABC=ABC
(ABC){2,3}	ABCABC, ABCABCABC
(ABC){2,}	ABCABC, ABCABCABC, ...

- more complicated nestings possible
- try some regexes using <https://regex101.com/>

Regex: special characters ?*+.

Pattern	Matches	Example
colou?r	optional previous character	color, colour
oo*h!	0 or more of previous character	oh!, ooh!, oooh!, ...
o+h!	1 or more of previous character	oh!, ooh!, oooh!, ...
oh.	any character except newline	oho!, oha!, oh1! oh !, ...

Regex: special characters ?*+.

Pattern	Matches	Example
colou?r	optional previous character	color, colour
oo*h!	0 or more of previous character	oh!, ooh!, oooh!, ...
o+h!	1 or more of previous character	oh!, ooh!, oooh!, ...
oh.	any character except newline	oho!, oha!, oh1! oh !, ...
		dog, Dog, doggo, Doggo

- regexes are typically greedy
- use ? to make them lazy

Regex: special characters ?*+.

Pattern	Matches	Example
colou?r	optional previous character	color, colour
oo*h!	0 or more of previous character	oh!, ooh!, oooh!, ...
o+h!	1 or more of previous character	oh!, ooh!, oooh!, ...
oh.	any character except newline	oho!, oha!, oh1! oh !, ...
[dD]og(go)?	= [dD]og(go){0,1}	dog, Dog, doggo, Doggo

- regexes are typically greedy
- use ? to make them lazy

Regex: special characters ?*+.

Pattern	Matches	Example
colou?r	optional previous character	color, colour
oo*h!	0 or more of previous character	oh!, ooh!, oooh!, ...
o+h!	1 or more of previous character	oh!, ooh!, oooh!, ...
oh.	any character except newline	oho!, oha!, oh1! oh !, ...
[dD]og(go)?	= [dD]og(go){0,1}	dog, Dog, doggo, Doggo
		Ei!, Eiei!, Eieiei!, ...

- regexes are typically greedy
- use ? to make them lazy

Regex: special characters ?*+.

Pattern	Matches	Example
colou?r	optional previous character	color, colour
oo*h!	0 or more of previous character	oh!, ooh!, oooh!, ...
o+h!	1 or more of previous character	oh!, ooh!, oooh!, ...
oh.	any character except newline	oho!, oha!, oh1! oh !, ...
[dD]og(go)?	= [dD]og(go){0,1}	dog, Dog, doggo, Doggo
Ei(ei)*!	= Ei(ei){0,}!	Ei!, Eiei!, Eieiei!, ...

- regexes are typically greedy
- use ? to make them lazy

Regex: metacharacters

- `\d (\D)` matches (non-)digit characters
- `\w (\W)` matches (non-)word characters
- `\s (\S)` matches (non-)space characters

Pattern	Matches
<code>\d</code>	= [0-9]
<code>\w</code>	= [A-Za-z0-9_] (for ASCII)
<code>\s</code>	= [\n\r\t\f] (for ASCII)

Regex: anchors

- `^` matches start of line
- `$` matches end of line
- `\b` matches word boundary
- `\B` matches non-word boundary

Pattern	Input	Match
<code>^Fortuna</code>	Fortuna 1895 Düsseldorf	Fortuna
<code>^[0-9]+\$</code>	Fortuna 1895 Düsseldorf	
<code>\b\d+\b</code>	Fortuna 1895 Düsseldorf	1895
<code>\b\w+\$</code>	Fortuna 1895 Düsseldorf	<i>mostly:</i> sseldorf
<code>\b[\wäüöÄÜÖ]+\$</code>	Fortuna 1895 Düsseldorf	Düsseldorf

Know your encoding! and try to avoid non UTF-8 data :)

Regex: find all *the* in a text

Pattern	Issue
the	misses capitalization

Regex: find all *the* in a text

Pattern	Issue
the	misses capitalization
[tT]he	returns parts of, e.g., <i>other</i> , <i>theology</i>

Regex: find all *the* in a text

Pattern	Issue
the	misses capitalization
[tT]he	returns parts of, e.g., <i>other</i> , <i>theology</i>
[^a-zA-Z][tT]he[^a-zA-Z]	might be good

equivalents, similar approaches?

Regex: find all *the* in a text

Pattern	Issue
the	misses capitalization
[tT]he	returns parts of, e.g., <i>other</i> , <i>theology</i>
[^a-zA-Z][tT]he[^a-zA-Z]	might be good

equivalents, similar approaches? e.g., \b[tT]he\b

- matching *false positives* (type I error) affects precision
- not matching *false negatives* (type II error) affects recall
- know the difference between [^a-zA-Z][tT]he[^a-zA-Z] and \b[tT]he\b

Precision and recall

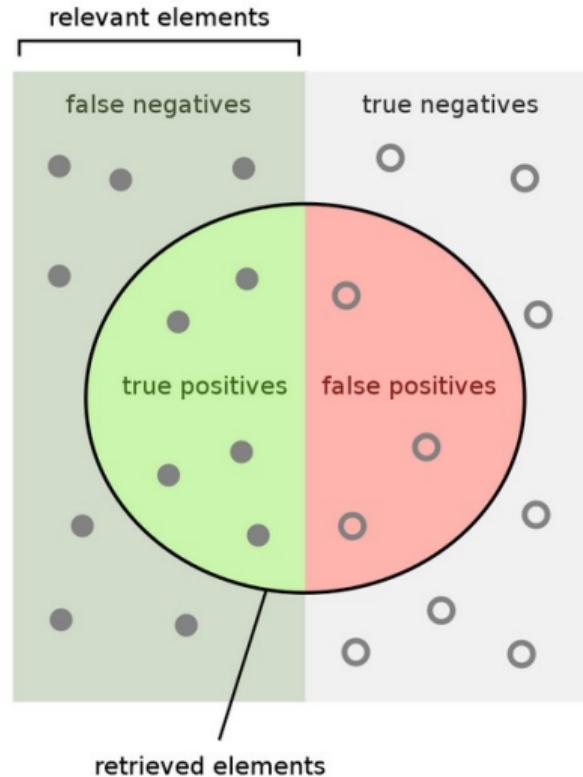
How many retrieved items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

How many relevant items are retrieved?

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

From Wikipedia, the free encyclopedia.
User: Walber.



Regex: quite powerful

- do not overdesign your methods
- word search is useful for first insight
- regexes make word search competitive in simple scenarios

Examples:

<https://trends.google.com/trends/>

<https://www.zeit.de/politik/deutschland/2019-09/bundestag-jubilaeum-70-jahre-parlament-reden-woerter-sprache-wandel> (German)

How to use ChatGPT as a teacher



<https://twitter.com/tunguz/status/1693310685009318260>

How to use ChatGPT as a teacher



<https://twitter.com/tunguz/status/1693310685009318260>

explain this regex: `(([a-zA-Z]-0-9)+\.\.)(a-zA-Z){2,})$`

Sure, I'd be happy to explain the regex pattern:

```
css Copy code
(([a-zA-Z]-0-9)+\.\.)(a-zA-Z){2,})$
```

However, it seems there might be a slight typo in the pattern since there is an extra closing parenthesis. Based on the structure, I'm going to assume you intended:

```
css Copy code
(([a-zA-Z]-0-9)+\.\.)(a-zA-Z){2,})$
```

Here's the breakdown:

1. `"("`: Start of a capturing group.
2. `"[a-zA-Z]-0-9)*"`: Match one or more alphanumeric characters or the hyphen (-). This will match strings like `'example-'`, `'abc'`, `'123'`, and so on.
3. `"\.\."`: Match a literal dot (period). The double backslashes are typically used in programming languages like Java where the backslash itself needs to be escaped. If you were using a language like JavaScript, you would only use one backslash (`\.`) to represent a literal dot.
4. `")"`: End of the capturing group that captures the subdomain or domain name followed by a dot, e.g., `'example.'`.
5. `"[a-zA-Z]{2,}"`: Match at least 2 or more alphabetical characters. This is typically used to match domain TLDs (like `'.com'`, `'.net'`, etc.).
6. `"$"`: End of the outer capturing group.
7. `"$"`: Assert the position at the end of a line. This ensures the pattern matches right up to the end of the input string.

Put together, this regular expression pattern matches strings that look like domain names, ensuring that the domain name ends with at least two alphabetical characters (like `'.com'` or `'.uk'`). For instance, it would match `'example.com'` or `'subdomain.example.net'`, but not `'example.ci'` because `'ci'` is not purely alphabetical.

Text normalization

standard preprocessing includes

- lowercase (maybe not for sentiment, machine translation)
- segmenting sentences
- tokenizing (segmenting words/units)
- punctuation removal
- number removal
- normalizing (stemming/lemmatizing) word representation
- denoising: stopword removal

Sentence splitting

an easy and straightforward way to segment sentences:

- regex: `[.:?!]\s`
- fails for
 - abbreviations: B. Obama, Dr. Müller
 - proper names: 1. FC Köln
 - ordinal numbers: 1. Bundesliga
 - (falsely) missing space: first sentence.second sentence.
- no (sufficient) straightforward solution

Tokenization

an easy and straightforward way to tokenize

- space characters between words: `\s` or `\w+`
(only for Arabic, Cyrillic, Greek, Latin, etc., based writing systems)
- tokens are of words between spaces

what happens with punctuation?

Punctuation

we can not just remove all forms of punctuation/symbols:

- Dr., M.Sc., Ph.D., AT&T
- B. Obama
- prices: \$45.55, €30
- dates: 01/02/06, 01.02.2006
- URLs: <https://www.tu-dortmund.de/>
- hashtags: #nlp
- email: rieger@statistik.tu-dortmund.de
- words that does not stand on its own: are in we're, don't

another problem:

- expressions like New York, rock 'n' roll (see also n-grams)
- hyphens: state-of-the-art
- sometimes: line breaks

Example sentence

they bought a swimming pool for their apartment in San Francisco and they

- how many tokens?
 - 13

Example sentence

they bought a swimming pool for their apartment in San Francisco and they

- how many tokens?
 - 13
 - 12

Example sentence

they bought a swimming pool for their apartment in San Francisco and they

- how many tokens?
 - 13
 - 12
 - 11

Example sentence

they bought a swimming pool for their apartment in San Francisco and they

- how many tokens?
 - 13
 - 12
 - 11
- how many types (=words)?
 - 12

Example sentence

they bought a swimming pool for their apartment in San Francisco and they

- how many tokens?
 - 13
 - 12
 - 11
- how many types (=words)?
 - 12
 - 11

Example sentence

they bought a swimming pool for their apartment in San Francisco and they

- how many tokens?
 - 13
 - 12
 - 11
- how many types (=words)?
 - 12
 - 11
 - 10

Example sentence

they bought a swimming pool for their apartment in San Francisco and they

- how many tokens?
 - 13
 - 12
 - 11
- how many types (=words)?
 - 12
 - 11
 - 10
 - 9

Tokenization using NLTK

Bird, Loper and Klein (2009), Natural Language Processing with Python. O'Reilly.

```
>>> text = 'That U.S.A. poster-print costs $12.40...'
>>> pattern = r'''(?x) # set flag to allow verbose regexps
... ([A-Z]\.)*          # abbreviations, e.g., U.S.A.
... | \w+(-\w+)*        # words with optional internal hyphens
... | \$?\d+(\.\d+)?%? # currency and percentages, e.g. $12.40, 82%
... | \.\\.\\.          # ellipsis
... | [][.,;`^():-_-'] # these are separate tokens; includes ], [
...
>>> nltk.regexp_tokenize(text, pattern)
['That', 'U.S.A.', 'poster-print', 'costs', '$12.40', '...']
```

- in R, quanteda offers great possibilities: <http://quanteda.io/>

N-grams

- different understandings of n-grams:
 - character based n-grams (original definition)
UNO = trigram (3-gram)
 - word based n-grams (useful for expressions)
The New York Times = tetragram (4-gram)
- character based easy to implement, but computationally expensive
- word based more complex:
 - dictionary
 - rule based: co-occurrence

Word similarity

- one (simple) option: character based n-grams
- jaccard, dice, overlap, cosine
- it holds $\text{Dice}(a, b) = \frac{2 \cdot \text{Jaccard}(a, b)}{\text{Jaccard}(a, b) + 1}$

$$\text{Jaccard}(a, b) = \frac{|a \cap b|}{|a \cup b|}$$

$$\text{Overlap}(a, b) = \frac{|a \cap b|}{\min\{|a|, |b|\}}$$

$$\text{Dice}(a, b) = 2 \frac{|a \cap b|}{|a| + |b|}$$

$$\text{Cosine}(a, b) = \frac{|a \cap b|}{\sqrt{|a|} \sqrt{|b|}}$$

Example: pairwise trigram similarity for work, worker, working

Levenshtein distance

$$\text{lev}(a, b) = \begin{cases} |a| & \text{if } |b| = 0 \\ |b| & \text{if } |a| = 0 \\ \text{lev}(\text{tail}(a), \text{tail}(b)) & \text{if } a[1] = b[1] \\ 1 + \min \begin{cases} \text{lev}(a, \text{tail}(b)) \\ \text{lev}(\text{tail}(a), b) \end{cases} & \text{else} \end{cases}$$

- minimal editing distance
- $\text{lev}(\text{work}, \text{worker}) = 2$, $\text{lev}(\text{work}, \text{working}) = 3$, $\text{lev}(\text{worker}, \text{working}) = 3$

Stemming

- morphemes: units that make up words
 - stem: core "meaning"
 - affixes: added to stems, e.g., for grammatical reasons
- there are a lot of inflections, often made by affixation
- singular/plural, gender, case, tense, ...
- sometimes inflections useful for analysis; often noise
- difficulties in matching those words
- one solution: just cut off the suffixes
- popular stemmer: porter¹
- remove -ed, -ing, -ion, sses -> ss, ational -> ate

¹M. F. Porter. "An algorithm for suffix stripping". In: Program 14.3 (1980), pp. 130–137.

Lemmatization

- alternative to stemming
- use morphological base form
- often dictionary based
- am, are, is -> be
- car, cars, car's, cars' -> car
- better -> good
- disgracefully -> NEG + grace
- meeting -> meet or meeting (unclear for dictionary based approach)
- He is reading stories -> He be read story

Lemmatization dictionaries: <https://github.com/michmech/lemmatization-lists>

Morphology

could become arbitrarily complex

e.g., the Turkish word: Uygarlastiramadiklarimizdanmissinizcasina

- (behaving) as if you are among those whom we could not civilize
- Uygar: civilized, las: become
- tir: cause, ama: not able
- dik: past, lar: plural
- imiz: p1pl, dan: abl
- mis: past, siniz: p2pl
- casina: as if

Lexical units

notation/terminology we may use in this lecture:

corpus	set of documents
document	(longer) piece of text
section/paragraph	logical section within a document
sentence	sequence of tokens in a section, (usually) ending with punctuation
phrase	part of sentence
token	an instance of a word, (usually) separated by word boundaries
word/type	unique representations of tokens
vocabulary	set of words/types for a corpus
stem	core of a word with inflection endings removed
lemma	logical base form of a word
POS-tag	part-of-speech interpretation of a token, e.g., verb, or noun

Stopwords

- idea: remove common words without semantic meaning
- most commonly applied preprocessing technique
- next section: statistically motivated approach
similar properties across documents
- easy approach: established stopword list
- (often) useful to denoise data (e.g., for topic models)
- (newer/complex) language models need information of stopwords

Order of preprocessing

- depends strongly on corpus and programming language
- my framework (often):
 - ① lowercase
 - ② punctuation
 - ③ numbers
 - ④ stopwords
 - ⑤ tokenize
 - ⑥ (lemmatize)
- steps must be meaningful together,
e.g., stopword removal and punctuation
- be conservative in stopword removal
- stemming easier (more popular?), but maybe loss of information

Pitfalls

- umlauts
- different encodings for files
- change of abbreviations,
e.g., parl vs. parliamentary
- change of denomination or expression,
e.g., minister of finance vs. finance minister

Excursion: tokenization in Chinese

Chinese, Japanese and Thai do not use spaces to separate words
for Chinese:

- words are compositions of characters named (han)zi
- on average: 2.4 characters equals one word
- how to tokenize?

Example

姚明进入总决赛

Yao Ming reaches the finals

3 words?
姚明 进入 总决赛
Yao Ming reaches finals

5 words?
姚 明 进入 总 决赛
Yao Ming reaches overall finals

7 words?
姚 明 进 入 总 决 赛
Yao Ming enter enter overall decision game

Chinese tokenization in practice

- treat each character (z_i) as token
- easy segmentation
- Thai and Japanese more complex,
e.g., neural sequence models, supervised ML

(A little bit) linguistics

There were 70 children there.

- idea: classify words into categories
- 8 important parts:
 - noun
 - verb
 - pronoun
 - preposition
 - adverb
 - conjunction
 - participle
 - article
- how/why to identify different classes?

(A little bit) linguistics

There were 70 children there.

There/PRO were/VERB 70/NUM children/NOUN there/ADV./PUNC

- idea: classify words into categories
- 8 important parts:
 - noun
 - verb
 - pronoun
 - preposition
 - adverb
 - conjunction
 - participle
 - article
- how/why to identify different classes?

Why part-of-speech (POS) tagging?

- can be useful for different NLP tasks
 - pronunciation in text-to-speech
 - sentiment: identify adjectives
 - machine translation: grammatical order differs between languages
- linguistic (language-analytic) research questions

Open vs. closed class

closed class

- words with grammatical function, e.g.,
 - a, an, the (articles)
 - she, he (pronouns)
 - on, under, over, near, by (prepositions)

open class

- usually content words, e.g.,
 - nouns
 - verbs
 - adjectives
 - adverbs
 - interjections

which class is more informative?

Tagset

	Tag	Description	Example
Open Class	ADJ	Adjective: noun modifiers describing properties	<i>red, young, awesome</i>
	ADV	Adverb: verb modifiers of time, place, manner	<i>very, slowly, home, yesterday</i>
	NOUN	words for persons, places, things, etc.	<i>algorithm, cat, mango, beauty</i>
	VERB	words for actions and processes	<i>draw, provide, go</i>
	PROPN	Proper noun: name of a person, organization, place, etc..	<i>Regina, IBM, Colorado</i>
Closed Class Words	INTJ	Interjection: exclamation, greeting, yes/no response, etc.	<i>oh, um, yes, hello</i>
	ADP	Adposition (Preposition/Postposition): marks a noun's spacial, temporal, or other relation	<i>in, on, by under</i>
	AUX	Auxiliary: helping verb marking tense, aspect, mood, etc.,	<i>can, may, should, are</i>
	CCONJ	Coordinating Conjunction: joins two phrases/clauses	<i>and, or, but</i>
	DET	Determiner: marks noun phrase properties	<i>a, an, the, this</i>
	NUM	Numeral	<i>one, two, first, second</i>
	PART	Particle: a preposition-like form used together with a verb	<i>up, down, on, off, in, out, at, by</i>
	PRON	Pronoun: a shorthand for referring to an entity or event	<i>she, who, I, others</i>
	SCONJ	Subordinating Conjunction: joins a main clause with a subordinate clause such as a sentential complement	<i>that, which</i>
Other	PUNCT	Punctuation	<i>, ; , ()</i>
	SYM	Symbols like \$ or emoji	<i>\$ %</i>
	X	Other	<i>asdf, qwfg</i>

Universal Dependencies v1: A Multilingual Treebank Collection (Nivre et al., LREC 2016)
<https://universaldependencies.org/>

How difficult is POS?

- ~ 15% words are ambiguous (-> 85% not)
- ambiguous words are more common
- ~ 60% tokens are ambiguous
- e.g., word *back*
 - please, take the back seat -> ADJ
 - a garden in the back -> NOUN
 - I will be back -> ADP or AUX
 - the senators will back the bill -> VERB
 - please give me my money back -> PART
 - I was 18 years old back then -> ADV
- book that flight vs. hand me that book

How to?

Janet will back the bill

- will: AUX/NOUN/VERB/PROPN ?
- bill: NOUN/VERB/PROPN ?
- a-priori (statistical) knowledge
 - *will* usually AUX
- neighboring words
 - after *the*, the next word is usually not a verb
- morphology
 - prefixes, e.g., *un-* usually mark ADJ: unable
 - suffixes, e.g., *-ly* usually mark ADJ: importantly
 - capitalization usually marks proper names (PROPN)

(Sota) performance and models

- no big improvements in the last decade
- tag accuracy around 97%
- use of labeled dataset
 - hidden Markov models (HMM)
 - conditional random fields (CRF)
 - large language models, as e.g., BERT
- human accuracy comparable
- baseline: 92%
 - tag every word with its most frequent tag in labeled data
 - tag unknown words as nouns

Named entity recognition (NER)

- more informative: open class words
- most informative: nouns and proper names (-> entities)
- often not single tokens, but compositions (harder than POS), e.g.,
 - TU Dortmund University -> ORG: organization
 - Cologne central station -> LOC: location
 - Olaf Scholz -> PER: person
 - Dortmund, Germany -> LOC/GPE: geo-political entity
- dates, times, prices could be also entities

Why NER?

- dense information
- sentiment towards an entity
- question answering (QA)
- information extraction
- information in texts is highly entity oriented

narrative extraction¹:

A	Someone
B	tells
C	someone
D	somewhere (that)
<hr/>	
E	someone
F	did something(s)
G	[to someone]
H	somewhere
I	at some time
J	for some reason

¹ Narrative Theory for Computational Narrative Understanding (Piper et al., EMNLP 2021)

<http://dx.doi.org/10.18653/v1/2021.emnlp-main.26>

Why NER is hard

- segmentation: POS token-wise, NER on compositions
- type ambiguity:
 - Washington/PER was born into slavery on a farm.
 - Washington/ORG went up 2 games to 1 in the four-game series.
 - Blair arrived in Washington/LOC for what may be his last state visit.
 - In June, Washington/GPE passed a primary seatbelt law.
- connect different representations:
 - more important for persons
 - e.g., Olaf Scholz, chancellor, Mr. Scholz, Olaf, he

Popular models/approaches

- use of labeled dataset
- e.g., HMM, CRF, BERT
- search queries on Wikipedia:
 - idea: important entities are on Wikipedia
 - n-grams of text as search queries
 - (restricted to at least one capitalized token)
 - best matching result as entity
 - meta data for entities
- related methods: coreference resolution

Vectorizing text

they bought a swimming pool for their apartment in San Francisco and they

a, and, apartment, bought, for, Francisco, in, pool, San, swimming, their, they, they
a, and, apartment, bought, for, in, San Francisco, swimming pool, their, they, they

- simpler models based on bag of words
- often punctuation and numbers discarded
- M number of documents, V size of vocabulary
- $N^{(m)}$ size of document m

$$D^{(m)} = \left(W_1^{(m)}, \dots, W_{N^{(m)}}^{(m)} \right), m = 1, \dots, M$$
$$W_n^{(m)} \in W = \{W_1, \dots, W_V\}, n = 1, \dots, N^{(m)}$$

Counting words

- term frequency (term equals type/word) $\text{tf}_{w,m}$
- document frequency df_w

$$\text{tf}_{w,m} = \sum_{n=1}^{N^{(m)}} \mathbb{1}(W_n^{(m)} = w)$$

$$\text{df}_w = \sum_{m=1}^M \mathbb{1}(\text{tf}_{w,m} > 0)$$

$$\text{tf}_w = \sum_{m=1}^M \text{tf}_{w,m}$$

Zipf's law

- in R: e.g., package `zipfR`
- power law distribution
- words are inversely proportional to its rank in the frequency table
- usually including stopwords, i.e., without stopword removal
- k rank, f_k frequency of word with rank k ,
- s exponent; often close to 1 in practice

$$p_k = \frac{1/k^s}{\sum_v 1/v^s}$$

- interpolated to the rank 1 frequency: $f_k = \frac{f_1}{k^s}$

CNN Covid news

- covid-19 related news in CNN liveblog
- January 22nd 2020 up until December 12th 2021
- 27,432 texts
- stopword removal
- lemmatization dictionary: <https://github.com/michmech/lemmatization-lists>
- at least two characters

6:55 a.m. MESZ, August 20, 2020

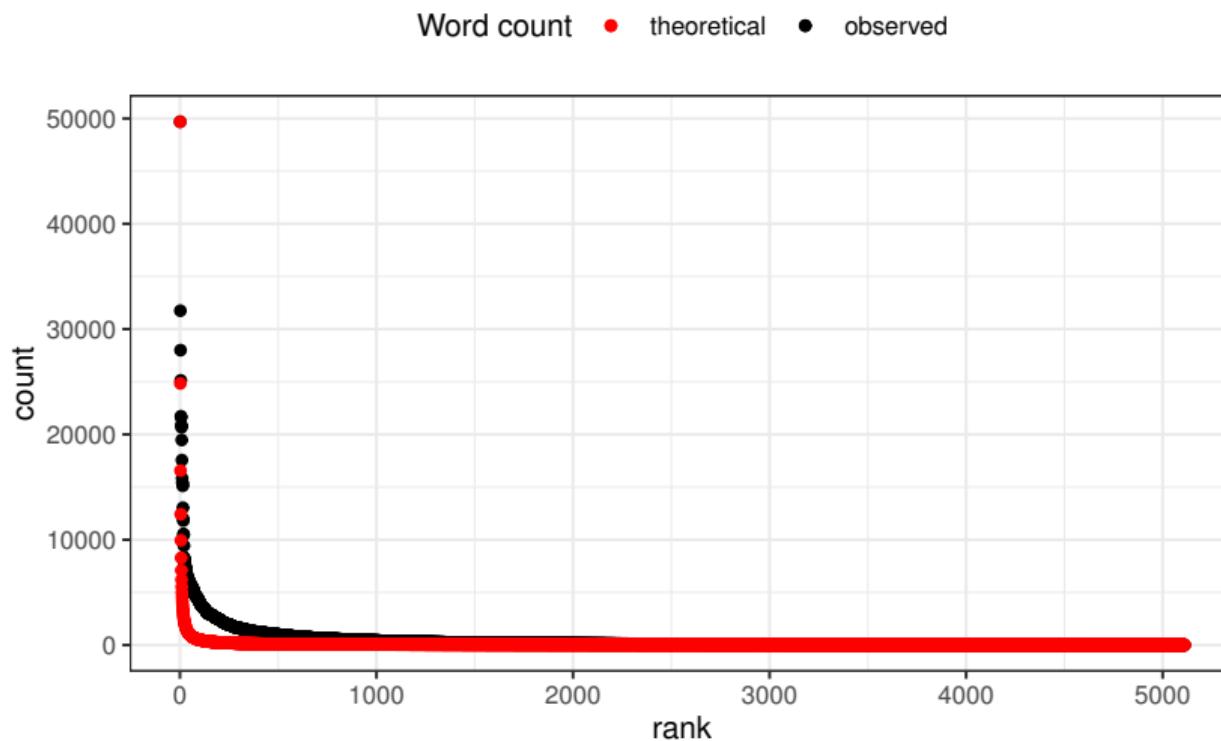
South Korea's latest church-linked coronavirus outbreak is turning into a battle over religious freedom

From CNN's Paula Hancocks in Seoul

A South Korean religious group at the center of a new coronavirus outbreak has been accused by the government of withholding key information and obstructing public health authorities in their fight against the pandemic.

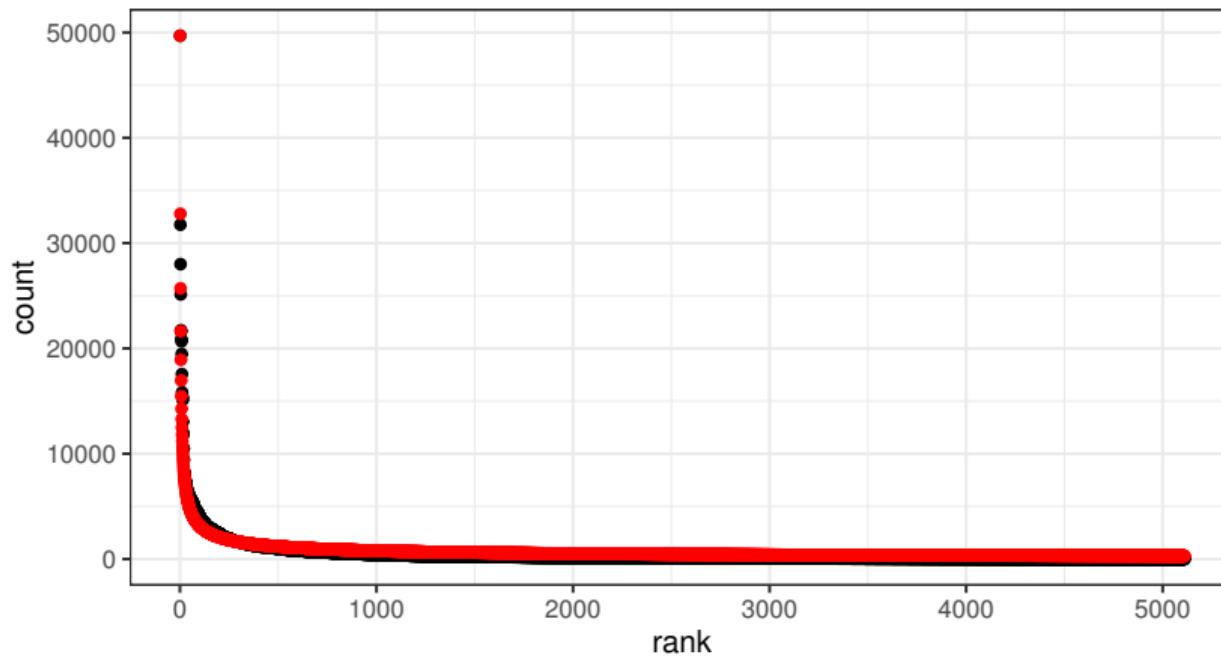
It's the latest clash between a religious group and the government of President Moon Jae-in, who is attempting to stem South Korea's coronavirus outbreak while fighting off accusations he is curbing religious freedom.

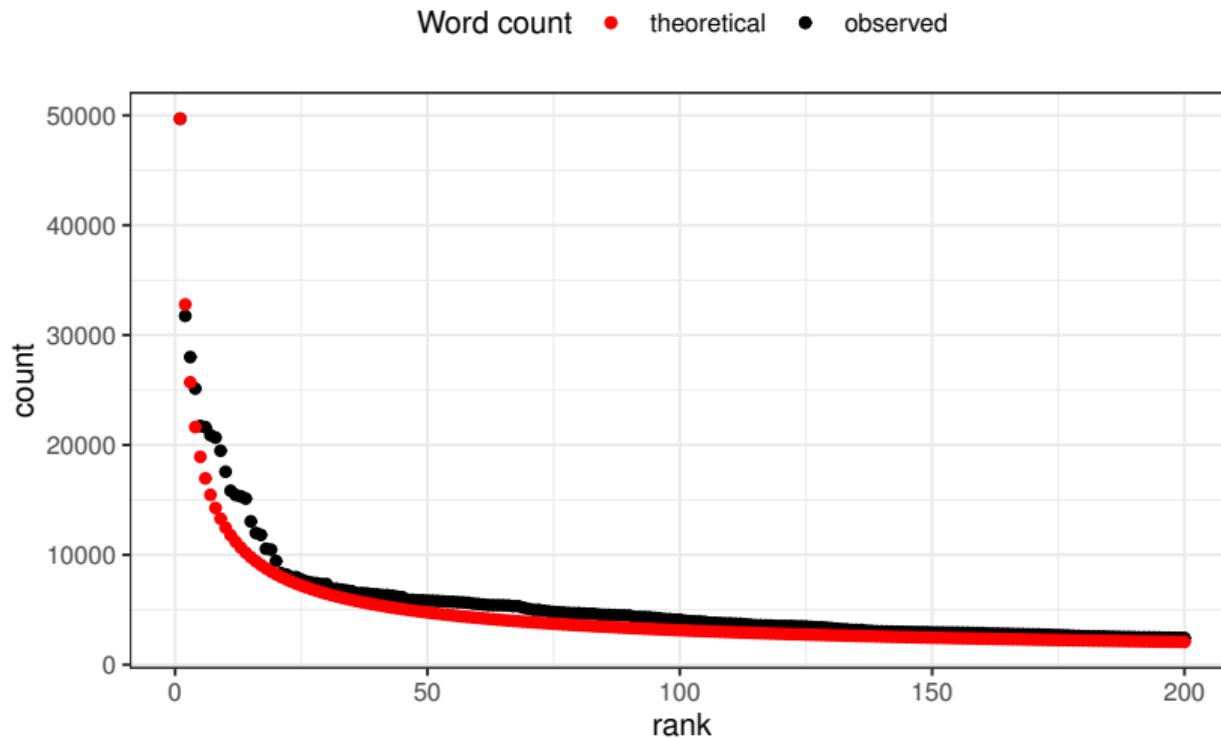
What happened: The new outbreak has been linked to the Sarang Jeil Church in Seoul, which has reported hundreds of

Zipf's law visualization; $s=1$ 

Zipf's law visualization; $s=0.6$

Word count ● theoretical ● observed



Zipf's law visualization; $s=0.6$ 

Zipf's law table

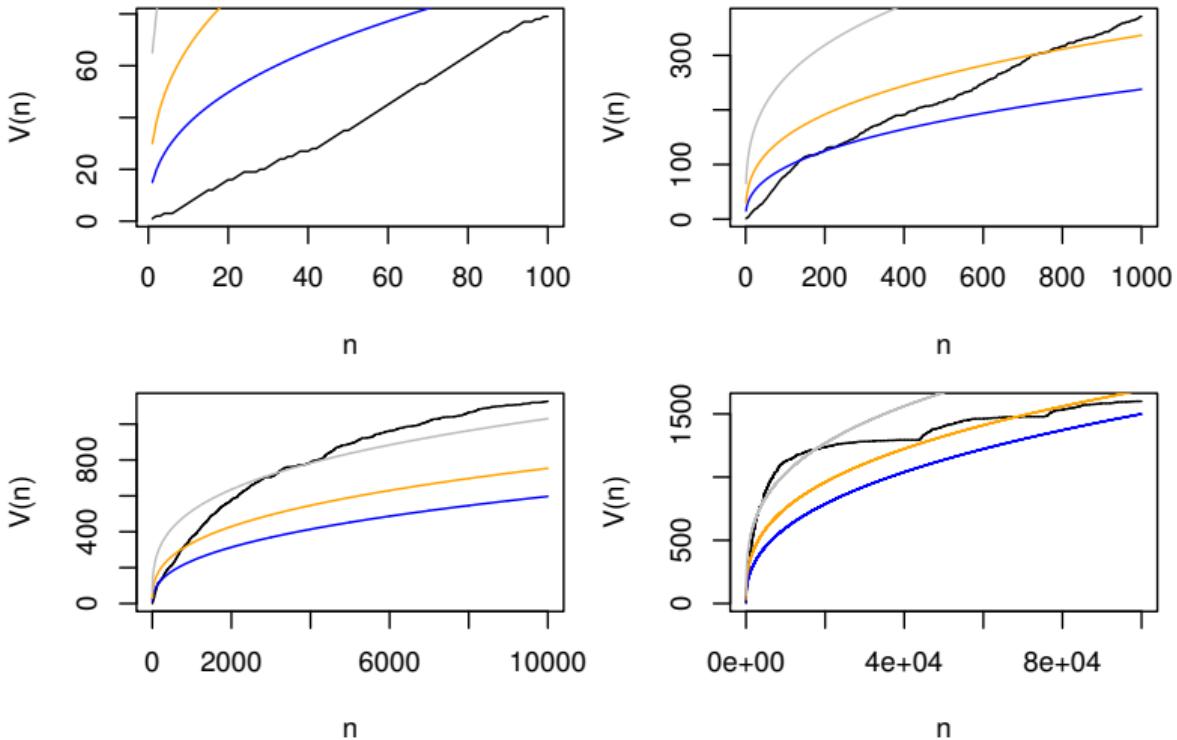
rank	word	tf_w	Zipf
1	say	49688	49688.00
2	case	31747	32781.85
3	coronavirus	27999	25702.70
4	will	25129	21627.96
5	covid	21725	18917.75
6	new	21618	16957.46
7	state	20893	15459.40
8	health	20673	14269.13
9	people	19473	13295.54
10	vaccine	17561	12481.06

Heap's law

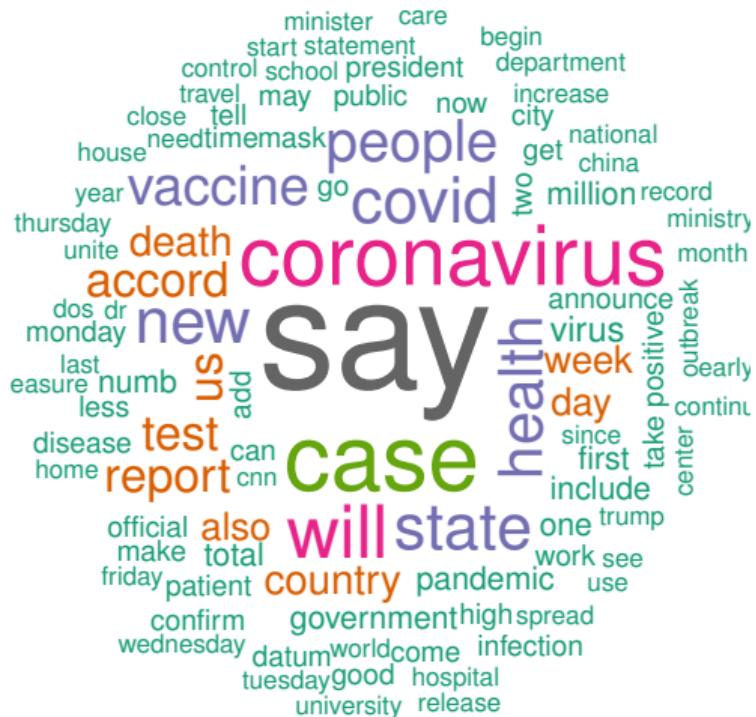
- related to Zipf's law (asymptotically equivalent)
- number of distinct words in a collection
- type-token relation
- n number of observed token
- $V(n)$ vocabulary size
- K, β parameters "to choose"/to fit, typically
 - K between 10 and 100
 - β between 0.4 and 0.6

$$V(n) = Kn^\beta$$

Visualization of Heap's law



Wordclouds



- R package wordcloud

(Fancy) wordclouds



- R package wordcloud2

Document-term matrix

	$tf_{w,m}$	a	and	apartment	bought	for	in	San Francisco	swimming pool	their	they
Doc 1	1	1		1	1	1	1	1	1	1	2
...											

$$dtm = \begin{pmatrix} tf_{1,1} & tf_{2,1} & \cdots & tf_{V,1} \\ tf_{1,2} & tf_{2,2} & \cdots & tf_{V,2} \\ \vdots & \vdots & \ddots & \vdots \\ tf_{1,M} & tf_{2,M} & \cdots & tf_{V,M} \end{pmatrix}$$

Term frequency-inverse document frequency

they bought a swimming pool for their apartment in San Francisco and they

- stopword list conservative
- still non-informative words in corpus
- weight words regarding appearance over documents
- tf-idf very popular for simple analyses

$$\text{idf}_w = \log \frac{M}{\text{df}_w} = -\log \frac{\text{df}_w}{M}$$

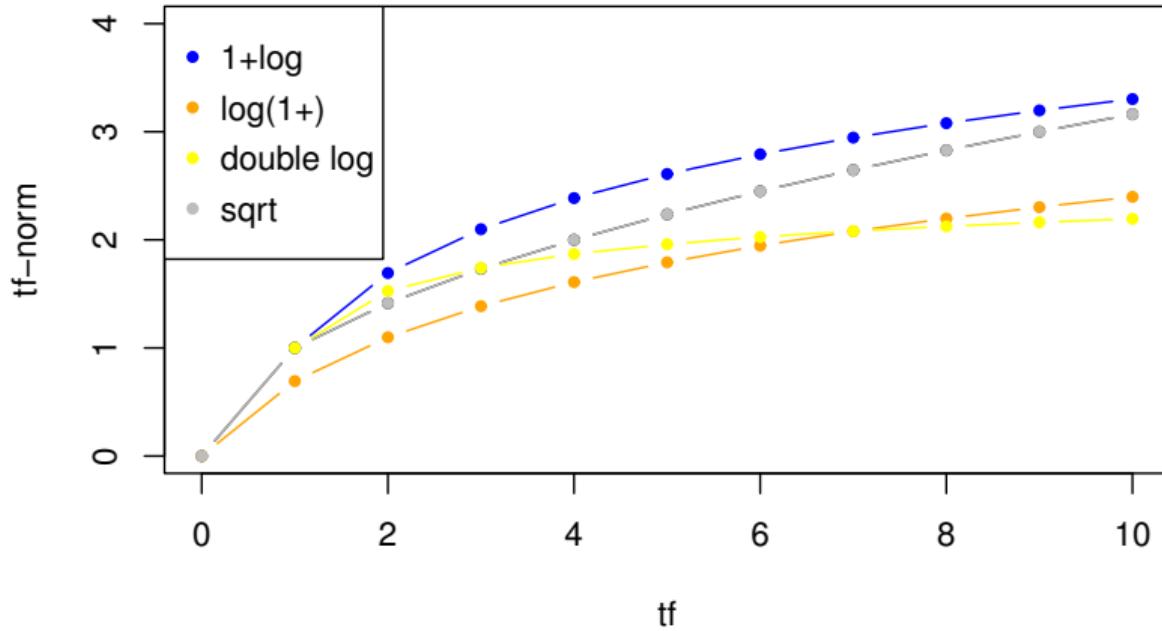
$$\text{tf-idf}_{w,m} = \text{tf}_{w,m} \text{idf}_w$$

- but why this way of normalization?

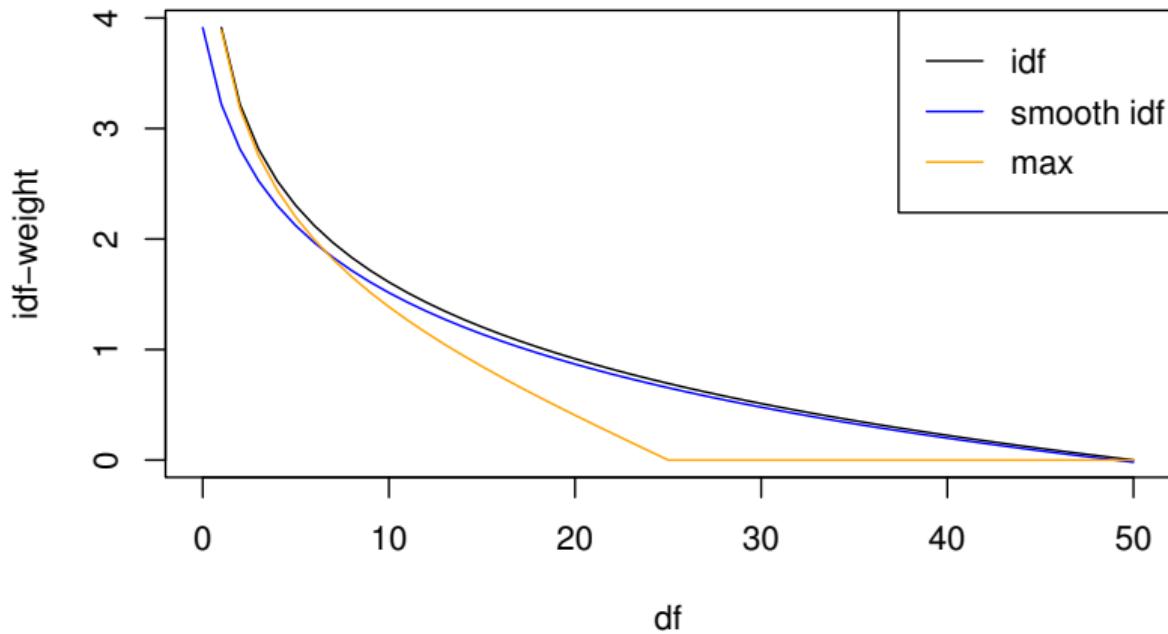
tf-idf variations

- you can think of several modifications of tf-idf
- $\text{tf}_{w,m}$: $1 + \log \text{tf}$, $\log(1 + \text{tf}_{w,m})$, $1 + \log(1 + \log \text{tf}_{w,m})$, $\sqrt{\text{tf}_{w,m}}$, ...
- idf_w : $\log \frac{M}{\text{df}_w + 1}$, $\max\{0, \log \frac{M - \text{df}_w}{\text{df}_w}\}$, ...
- you can also think of document normalization, e.g., $1/\|D^{(m)}\|$

tf variations



idf variations



Probabilistic language models

- assign probability to a phrase
- important to a lot of all NLP domains
- machine translation:
 $P(\text{large in size}) > P(\text{big in size})$
- spell correction:
 $P(\text{large in size}) > P(\text{large on size})$
- speech recognition:
 $P(\text{I saw a van}) \gg P(\text{eyes awe of an})$

Estimating $P(W)$

- conditional probabilities
- $P(B | A) = P(A, B)/P(A)$ $P(A, B) = P(A)P(B | A)$
- chain rule in general: $P(w_1, w_2, \dots, w_n) = P(w_1)P(w_2 | w_1) \dots P(w_n | w_1, \dots, w_{n-1})$
- count and divide?

Estimating $P(W)$

- conditional probabilities
- $P(B | A) = P(A, B)/P(A)$ $P(A, B) = P(A)P(B | A)$
- chain rule in general: $P(w_1, w_2, \dots, w_n) = P(w_1)P(w_2 | w_1) \dots P(w_n | w_1, \dots, w_{n-1})$
- count and divide?
- "too few" data
- simplifying to $P(w_i | w_1, \dots, w_{i-1}) \approx P(w_i | w_{i-k}, \dots, w_{i-1})$
with k "small", e.g., 1 or 2

My first language model

- ① I am Sam
- ② Sam I am
- ③ I do not like ham and eggs

- $P(\text{eggs} \mid \text{ham and}) = ?$
- $P(\text{do} \mid \text{I}) = 1/3$
- $P(\text{am} \mid \text{I}) = 2/3$

Evaluation

How good is our (first naive) language model?

- does our language model prefer actually observed sentences?
- training set: train parameters, see, e.g., 3 sentences before
- test set: unseen data to assess performance of model
- hundreds in total, but several established evaluation metrics, e.g.,
 - perplexity for language models
 - NPMI for language and topic models

Difficulties in evaluation

- common procedure:
 - train two different models A and B, e.g., binary classification
 - get predictions \hat{y}
 - calculate accuracy: $\frac{1}{|y|} \sum_y \mathbb{1}(y_i = \hat{y}_i)$
- not practical for language models
- extrinsic evaluation? gold standard, but time-consuming
- sometimes intrinsic evaluation, e.g., perplexity
 - bad approximation
 - only useful for first insights

Perplexity

- intuition: how well can our model predict the next word
- how good are unigrams for this?

Perplexity

- intuition: how well can our model predict the next word
- how good are unigrams for this?
 - terrible
 - global frequency
- the higher the probability the model assigns to the actual word, the better the model
- if a model assigns a high probability to the test set, it means that it is not surprised to see it: it's not **perplexed** by it

Perplexity definition

$$\text{Perplexity}(D^{(m)}) = \sqrt[N^{(m)}]{P(D^{(m)})^{-1}} = \sqrt[N^{(m)}]{\frac{1}{P(W_1^{(m)}, \dots, W_{N^{(m)}}^{(m)})}}$$

- perplexity is the inverse probability of the test set (normalized by the number of token)
- recap: $P(w_i \mid w_1, \dots, w_{i-1}) \approx P(w_i \mid w_{i-k}, \dots, w_{i-1})$
- perplexity (usually) computed on test data
- probabilities taken from (underlying) model
 - given the token(s) before (for n-gram models)
- **lower perplexity is better**

Pointwise mutual information (PMI)

$$\text{PMI}(x, y) = \log \frac{P(x, y)}{P(x)P(y)} = \log \frac{P(x | y)}{P(x)} = \log \frac{P(y | x)}{P(y)}$$

- $\text{PMI}(x, y) \in [-\infty, \min\{-\log P(x), -\log P(y)\}]$
- difference between joint occurrence and individual appearance
- $P(x | y)$ estimated, e.g., via sliding window or document-wise
- what if $P(x | y) = 0$?

Pointwise mutual information (PMI)

$$\text{PMI}(x, y) = \log \frac{P(x, y)}{P(x)P(y)} = \log \frac{P(x | y)}{P(x)} = \log \frac{P(y | x)}{P(y)}$$

- $\text{PMI}(x, y) \in [-\infty, \min\{-\log P(x), -\log P(y)\}]$
- difference between joint occurrence and individual appearance
- $P(x | y)$ estimated, e.g., via sliding window or document-wise
- what if $P(x | y) = 0$?
 - smoothing factor (recap tf-idf)
 - e.g., add 1
 - choice not trivial

Normalized PMI (NPMI)

- PMI not useful for most applications
- normalize to get a measure of likeliness of co-occurrence

$$\text{NPMI}(x, y) = \frac{\text{PMI}(x, y)}{-\log P(x, y)} \in [-1, 1]$$

- -1 : never occurring together, 1 : complete co-occurrence
- using NPMI, we can identify n-grams statistically
- NPMI serves as evaluation metric, e.g., for topic models
- is it useful for language model evaluation?

Evaluation

- several issues regarding evaluation
- we want leaderboards!
- contradicts the universality of the models
- leaderboards provide only one view of the *truth*¹

Model \neq truth

Grimmer and Stewart, 2013:

- ① all models are wrong, some are useful
- ② quantitative methods help people, but do not replace them
- ③ there is no universal best method for automated text analysis
- ④ validate, validate, validate

¹Utility is in the eye of the user: A critique of NLP leaderboards (Ethayarajh and Jurafsky, 2020)

3

Text Data as Features

- Text Data in Classification Tasks
- Sentiment Analysis
- Measures and Challenges in Supervised Text Classification
- Example: A Naive Bayes Classifier
- From Feature Extraction to Feature Selection
- Clustering
- Dimension Reduction: PCA, SVD, LSA

Is this spam?

Von Publishers@mail.wan-architecture.com ®, Clearing@mail.wan-architecture.com ®,

House@mail.wan-architecture.com ®, alex@mail.wan-architecture.com ®

An Recipients <Publishers@mail.wan-architecture.com> ®

29.10.2022, 00:20

Antwort an infopch@housemail.com ®

Betreff [SPAM] Payment Notice

PUBLISHERS CLEARING HOUSE INC

300 Jericho Quadrangle Ste 300 Jericho NY 11753

PAYMENT AND FINAL NOTICE.

This is your Publishers Clearing House payment notice. For more information, please read the attached pdf file.

Yours sincerely,

Todd Sloan

Publishers Clearing House

> ④ 1 Anhang: Publishers Clearing House.pdf 113 KB

↓ Speichern ▾

Movie reviews

- 50,000 (highly polar) movie reviews
- sentiment classification
- feature extraction + traditional machine learning
- dedicated text data methods
 - sentiment lexicon
 - (enhanced) language models

This movie was not very well directed. they almost totally disregarded the book.I guess they were tr...	negative
The buzz for this film has always been about the fabulous graphics that make Kevin Bacon disappear. ...	negative
A young girl becomes a war-time marine's pen-pal, and when he visits at war's end expecting someone ...	positive
This movie is a perfect adaptation of the English Flick Unfaithful. Ashmit plays the role of Richard...	negative
Robert Altman's downbeat, new-fangled western from Edmund Naughton's	negative

Article's topic label

Abstract

We propose a rolling version of the Latent Dirichlet Allocation, called RollingLDA. By a sequential approach, it enables the construction of LDA-based time series of topics that are consistent with previous states of LDA models. After an initial modeling, updates can be computed efficiently, allowing for real-time monitoring and detection of events or structural breaks. For this purpose, we propose suitable similarity measures for topics and provide simulation evidence of superiority over other commonly used approaches. The adequacy of the resulting method is illustrated by an application to an example corpus. In particular, we compute the similarity of sequentially obtained topic and word distributions over consecutive time periods. For a representative example corpus consisting of The New York Times articles from 1980 to 2020, we analyze the effect of several tuning parameter choices and we run the RollingLDA method on the full dataset of approximately 4 million articles to demonstrate its feasibility.

- assign one label to this abstract of a paper
- here: supervised -> choice of given labels
- unsupervised -> e.g., clustering, topic models

Ticket mail classification

Ticket Type	Scope
Incident	An unplanned interruption to the Service or reduction in the Quality of the Service. Failure of a Configuration Item that has not yet affected Service is also an Incident.
Problem	A cause of one or more Incidents. The cause is not usually known at the time a Problem Record is created, and the Problem Management Process is responsible for further investigation.
Feature Request	A request for new features currently not present in the Integration/Product.
CR	When a ticket is identified to be a Feature Request, the same is reassigned to the Development Team for feasibility analysis, evaluation and quoting.
CR	A request for a change to be made. Understanding change as the addition, modification or removal of anything that has impact on the Integration/Product.
Service Request	When a ticket is identified to be a CR, the same is reassigned to the Development Team for feasibility analysis, evaluation and quoting.
Service Request	A request for information, or question, or advice, or a Standard Change/Setup (E.g. add SKU). Everything not considered within the scope of the other types).

<https://support.techdynamics.com/support/solutions/articles/247924-tickets-types-and-priority-levels>

- important for customer service of big companies
- first stage of automated chat bot
- more enhanced: system knows what to answer
- (even partial) automation reduces human effort
- e.g., Generali (insurance company) receives more than one million claims via email per year
- several features possible, e.g., urgency

Text classification

- spam detection
- authorship identification
- assigning categories to
 - newspaper articles
 - research articles
 - ticket system claims
- sentiment analysis

Why sentiment?

- movies, books, hotels, products in general: positive vs. negative
- public sentiment: consumers confidence
- politics: what do people think about a candidate or issue
- economics: what do decision/policy makers think about economic situation
- could be used to predict market trends or election outcomes

Lexica

- Loughran-McDonald: English, financial focus, "negative", "positive", "litigious", "uncertainty", "constraining", or "superfluous", no weights, negations, amplifiers
- AFINN: English, often applied to Twitter data (but often works “worse” than VADER), $-5, -4, \dots, 5$
- VADER: specialized for social media, works well multi-purpose, contains negations, weights, amplifiers
- Opinion-Lexicon (also Liu-Lexicon), specialized for reviews, works well in other domains as well, no weights, negations, amplifiers
- WKWSCI-Lexicon, general purpose lexicon, works well/best in most of domains, contains weights, but no negations, amplifiers, contains POS tags
- TextBlob, badly documented...

Lexica (continued)

- NRC: English, "negative", "positive", "anger", "anticipation", "disgust", "fear", "joy", "sadness", "surprise", or "trust"
- Bing: English, "negative" or "positive"
- SentiWS: German, $[-1, 1]$
- ...
- some dictionaries with inflections, some without
- different sizes, e.g., SentiWS 1650 pos., 1800 neg.
- unsupervised method — but overall task might be supervised
- basically one challenge: match preprocessing

Problems with lexicon based methods

- (double) negations

I really (*don't*) like this movie
(*don't*) dismiss this movie

- add NOT_ to tokens between negation and following punctuation¹

I really don't NOT_like NOT_this NOT_movie
don't NOT_dismiss NOT_this NOT_movie

- new words

- find similar words that are known by the lexicon
- e.g., similarities between docs and lexicon halves², cf. embeddings

¹ Thumbs up? Sentiment Classification using Machine Learning Techniques (Pang et al., 2002)

<http://dx.doi.org/10.3115/1118693.1118704>

² Lex2Sent: A bagging approach to unsupervised sentiment analysis (Lange et al., 2024)

<https://aclanthology.org/2024.konvens-main.28/>

Supervised classification

- input:

- documents $D^{(m)}, m = 1, \dots, M_{\text{train}}$
- a fixed set of possible labels $C = \{c_1, \dots, c_J\}$
- given labels $c_m \in C, m = 1, \dots, M_{\text{train}}$
- documents $D^{(m)}, m = M_{\text{train}} + 1, \dots, M_{\text{train}} + M_{\text{predict}}$
- without labels c_m

- output:

- predicted labels $\hat{c}_m \in C, m = M_{\text{train}} + 1, \dots, M_{\text{train}} + M_{\text{predict}}$
- a general rule for incoming documents: $f : D \rightarrow c$

Evaluation

- Accuracy: $\text{Acc}(\hat{c}_m, c_m) = \frac{1}{M_{\text{train}}} \sum_{m=1}^{M_{\text{train}}} \mathbb{1}_{\{\hat{c}_m=c_m\}}(\hat{c}_m, c_m)$

Evaluation

- Accuracy: $\text{Acc}(\hat{c}_m, c_m) = \frac{1}{M_{\text{predict}}} \sum_{m=M_{\text{train}}+1}^{M_{\text{predict}}} \mathbb{1}_{\{\hat{c}_m=c_m\}}(\hat{c}_m, c_m)$
- True Positive: $\text{TP}_j(\hat{c}_m, c_m) = \sum_{m=M_{\text{train}}+1}^{M_{\text{predict}}} \mathbb{1}_{\{\hat{c}_m=c_m=j\}}(\hat{c}_m, c_m), \quad j = 1, \dots, J$

Evaluation

- Accuracy: $\text{Acc}(\hat{c}_m, c_m) = \frac{1}{M_{\text{predict}}} \sum_{m=M_{\text{train}}+1}^{M_{\text{predict}}} \mathbb{1}_{\{\hat{c}_m=c_m\}}(\hat{c}_m, c_m)$
- True Positive: $\text{TP}_j(\hat{c}_m, c_m) = \sum_{m=M_{\text{train}}+1}^{M_{\text{predict}}} \mathbb{1}_{\{\hat{c}_m=c_m=j\}}(\hat{c}_m, c_m), \quad j = 1, \dots, J$
- False Positive: $\text{FP}_j(\hat{c}_m, c_m) = \sum_{m=M_{\text{train}}+1}^{M_{\text{predict}}} \mathbb{1}_{\{\hat{c}_m=j, c_m \neq j\}}(\hat{c}_m, c_m)$
- False Negative: $\text{FN}_j(\hat{c}_m, c_m) = \sum_{m=M_{\text{train}}+1}^{M_{\text{predict}}} \mathbb{1}_{\{\hat{c}_m \neq j, c_m=j\}}(\hat{c}_m, c_m)$

Evaluation

- Accuracy: $\text{Acc}(\hat{c}_m, c_m) = \frac{1}{M_{\text{predict}}} \sum_{m=M_{\text{train}}+1}^{M_{\text{predict}}} \mathbb{1}_{\{\hat{c}_m=c_m\}}(\hat{c}_m, c_m)$
- True Positive: $\text{TP}_j(\hat{c}_m, c_m) = \sum_{m=M_{\text{train}}+1}^{M_{\text{predict}}} \mathbb{1}_{\{\hat{c}_m=c_m=j\}}(\hat{c}_m, c_m), \quad j = 1, \dots, J$
- False Positive: $\text{FP}_j(\hat{c}_m, c_m) = \sum_{m=M_{\text{train}}+1}^{M_{\text{predict}}} \mathbb{1}_{\{\hat{c}_m=j, c_m \neq j\}}(\hat{c}_m, c_m)$
- False Negative: $\text{FN}_j(\hat{c}_m, c_m) = \sum_{m=M_{\text{train}}+1}^{M_{\text{predict}}} \mathbb{1}_{\{\hat{c}_m \neq j, c_m=j\}}(\hat{c}_m, c_m)$
- Precision: $\text{Prec}_j(\hat{c}_m, c_m) = \frac{\text{TP}_j(\hat{c}_m, c_m)}{\text{TP}_j(\hat{c}_m, c_m) + \text{FP}_j(\hat{c}_m, c_m)}$
- Recall: $\text{Rec}_j(\hat{c}_m, c_m) = \frac{\text{TP}_j(\hat{c}_m, c_m)}{\text{TP}_j(\hat{c}_m, c_m) + \text{FN}_j(\hat{c}_m, c_m)}$

Evaluation

- Accuracy: $\text{Acc}(\hat{c}_m, c_m) = \frac{1}{M_{\text{predict}}} \sum_{m=M_{\text{train}}+1}^{M_{\text{predict}}} \mathbb{1}_{\{\hat{c}_m=c_m\}}(\hat{c}_m, c_m)$
- True Positive: $\text{TP}_j(\hat{c}_m, c_m) = \sum_{m=M_{\text{train}}+1}^{M_{\text{predict}}} \mathbb{1}_{\{\hat{c}_m=c_m=j\}}(\hat{c}_m, c_m), \quad j = 1, \dots, J$
- False Positive: $\text{FP}_j(\hat{c}_m, c_m) = \sum_{m=M_{\text{train}}+1}^{M_{\text{predict}}} \mathbb{1}_{\{\hat{c}_m=j, c_m \neq j\}}(\hat{c}_m, c_m)$
- False Negative: $\text{FN}_j(\hat{c}_m, c_m) = \sum_{m=M_{\text{train}}+1}^{M_{\text{predict}}} \mathbb{1}_{\{\hat{c}_m \neq j, c_m=j\}}(\hat{c}_m, c_m)$
- Precision: $\text{Prec}_j(\hat{c}_m, c_m) = \frac{\text{TP}_j(\hat{c}_m, c_m)}{\text{TP}_j(\hat{c}_m, c_m) + \text{FP}_j(\hat{c}_m, c_m)}$
- Recall: $\text{Rec}_j(\hat{c}_m, c_m) = \frac{\text{TP}_j(\hat{c}_m, c_m)}{\text{TP}_j(\hat{c}_m, c_m) + \text{FN}_j(\hat{c}_m, c_m)}$
- F_β score: $(F_\beta)_j(\hat{c}_m, c_m) = (1 + \beta^2) \frac{\text{Prec}_j(\hat{c}_m, c_m) \text{Rec}_j(\hat{c}_m, c_m)}{(\beta^2 \text{Prec}_j(\hat{c}_m, c_m)) + \text{Rec}_j(\hat{c}_m, c_m)}$

Evaluation

- Accuracy: $\text{Acc}(\hat{c}_m, c_m) = \frac{1}{M_{\text{predict}}} \sum_{m=M_{\text{train}}+1}^{M_{\text{predict}}} \mathbb{1}_{\{\hat{c}_m=c_m\}}(\hat{c}_m, c_m)$
- True Positive: $\text{TP}_j(\hat{c}_m, c_m) = \sum_{m=M_{\text{train}}+1}^{M_{\text{predict}}} \mathbb{1}_{\{\hat{c}_m=c_m=j\}}(\hat{c}_m, c_m), \quad j = 1, \dots, J$
- False Positive: $\text{FP}_j(\hat{c}_m, c_m) = \sum_{m=M_{\text{train}}+1}^{M_{\text{predict}}} \mathbb{1}_{\{\hat{c}_m=j, c_m \neq j\}}(\hat{c}_m, c_m)$
- False Negative: $\text{FN}_j(\hat{c}_m, c_m) = \sum_{m=M_{\text{train}}+1}^{M_{\text{predict}}} \mathbb{1}_{\{\hat{c}_m \neq j, c_m=j\}}(\hat{c}_m, c_m)$
- Precision: $\text{Prec}_j(\hat{c}_m, c_m) = \frac{\text{TP}_j(\hat{c}_m, c_m)}{\text{TP}_j(\hat{c}_m, c_m) + \text{FP}_j(\hat{c}_m, c_m)}$
- Recall: $\text{Rec}_j(\hat{c}_m, c_m) = \frac{\text{TP}_j(\hat{c}_m, c_m)}{\text{TP}_j(\hat{c}_m, c_m) + \text{FN}_j(\hat{c}_m, c_m)}$
- F_β score: $(F_\beta)_j(\hat{c}_m, c_m) = (1 + \beta^2) \frac{\text{Prec}_j(\hat{c}_m, c_m) \text{Rec}_j(\hat{c}_m, c_m)}{(\beta^2 \text{Prec}_j(\hat{c}_m, c_m)) + \text{Rec}_j(\hat{c}_m, c_m)}$
- F_1 score most popular

Macro- vs. micro average

- Precision (macro average): $\text{Prec}^{\text{macro}}(\hat{c}_m, c_m) = \frac{1}{J} \sum_{j=1}^J \text{Prec}_j(\hat{c}_m, c_m)$
- Precision (micro average): $\text{Prec}^{\text{micro}}(\hat{c}_m, c_m) = \frac{\sum_{j=1}^J \text{TP}_j(\hat{c}_m, c_m)}{\sum_{j=1}^J \text{TP}_j(\hat{c}_m, c_m) + \text{FP}_j(\hat{c}_m, c_m)}$

Macro- vs. micro average

- Precision (macro average): $\text{Prec}^{\text{macro}}(\hat{c}_m, c_m) = \frac{1}{J} \sum_{j=1}^J \text{Prec}_j(\hat{c}_m, c_m)$
- Precision (micro average): $\text{Prec}^{\text{micro}}(\hat{c}_m, c_m) = \frac{\sum_{j=1}^J \text{TP}_j(\hat{c}_m, c_m)}{\sum_{j=1}^J \text{TP}_j(\hat{c}_m, c_m) + \text{FP}_j(\hat{c}_m, c_m)}$
- Recall (macro average): $\text{Rec}^{\text{macro}}(\hat{c}_m, c_m) = \frac{1}{J} \sum_{j=1}^J \text{Rec}_j(\hat{c}_m, c_m)$
- Recall (micro average): $\text{Rec}^{\text{micro}}(\hat{c}_m, c_m) = \frac{\sum_{j=1}^J \text{TP}_j(\hat{c}_m, c_m)}{\sum_{j=1}^J \text{TP}_j(\hat{c}_m, c_m) + \text{FN}_j(\hat{c}_m, c_m)}$
- $\text{Acc}(\hat{c}_m, c_m) = \text{Rec}^{\text{micro}}(\hat{c}_m, c_m) = \text{Prec}^{\text{micro}}(\hat{c}_m, c_m) = F_1^{\text{micro}}(\hat{c}_m, c_m)$
- Macro F_1 score: $F_1^{\text{macro}}(\hat{c}_m, c_m) = \frac{1}{J} \sum_{j=1}^J (F_1)_j(\hat{c}_m, c_m)$

Macro- vs. micro average

- Precision (macro average): $\text{Prec}^{\text{macro}}(\hat{c}_m, c_m) = \frac{1}{J} \sum_{j=1}^J \text{Prec}_j(\hat{c}_m, c_m)$
- Precision (micro average): $\text{Prec}^{\text{micro}}(\hat{c}_m, c_m) = \frac{\sum_{j=1}^J \text{TP}_j(\hat{c}_m, c_m)}{\sum_{j=1}^J \text{TP}_j(\hat{c}_m, c_m) + \text{FP}_j(\hat{c}_m, c_m)}$
- Recall (macro average): $\text{Rec}^{\text{macro}}(\hat{c}_m, c_m) = \frac{1}{J} \sum_{j=1}^J \text{Rec}_j(\hat{c}_m, c_m)$
- Recall (micro average): $\text{Rec}^{\text{micro}}(\hat{c}_m, c_m) = \frac{\sum_{j=1}^J \text{TP}_j(\hat{c}_m, c_m)}{\sum_{j=1}^J \text{TP}_j(\hat{c}_m, c_m) + \text{FN}_j(\hat{c}_m, c_m)}$
- $\text{Acc}(\hat{c}_m, c_m) = \text{Rec}^{\text{micro}}(\hat{c}_m, c_m) = \text{Prec}^{\text{micro}}(\hat{c}_m, c_m) = F_1^{\text{micro}}(\hat{c}_m, c_m)$
- Macro F_1 score: $F_1^{\text{macro}}(\hat{c}_m, c_m) = \frac{1}{J} \sum_{j=1}^J (F_1)_j(\hat{c}_m, c_m)$
- weighted average: macro + weights (e.g., support of each label)
- when to use which?

“True” labels / gold standard

- how to get true labels?

“True” labels / gold standard

- how to get true labels?
- often no “real” true labels, e.g., sentiment
- manual coded (and validated) examples serve as gold standard
 - time consuming and (often) expensive
- non-observable “true” labels: platinum standard
- uncertainty due to
 - gold standard \neq platinum standard (error propagation)
 - often only small subsets (gold) labeled

Manual (hand-coded) rules

- human knowledge
 - blacklist for spam mail addresses
 - "dollars" AND "you have been selected" -> SPAM
- maintaining is important and expensive/time consuming
 - types of spam mails evolve
- established: complementary methods

Naive Bayes

- one of the simplest models
- more enhanced classifiers/machine learning algorithms work in a similar fashion
- bag of words
- relation to ("builds on") simple (unigram) language model
- cf. Machine Learning for Text, pp. 123–131

Bag of words

$D^{(m)}$: This movie
was not very well
directed. they al-
most totally disre-
garded the book.I
guess they were ...

w	$tf_{w,m}$
almost	1
book	1
directed	1
disregarded	1
guess	1
i	1
movie	1
not	1
the	1
they	2
this	1
totally	1
very	1
was	1
well	1
were	1
...	
$tf_m = (tf_{w,m})_{w=1,\dots,V}$	

Maximum a posteriori → Naive Bayes

$$\begin{aligned}c_m^{(MAP)} &= \arg \max_{c \in C} P(c \mid \text{tf}_m) \\&= \arg \max_{c \in C} \frac{P(\text{tf}_m \mid c)P(c)}{P(\text{tf}_m)} \\&= \arg \max_{c \in C} P(\text{tf}_m \mid c)P(c)\end{aligned}$$

$$c_m^{(NB)} = \arg \max_{c \in C} P(c) \prod_{w \in W} P(w \mid c)$$

Simplifications

- bag of words: position does not matter
- independence: word probabilities are independent for given class
- sum logs instead of multiplying (low) probabilities

$$\log(ab) = \log(a) + \log(b)$$

$$\Rightarrow c_m^{(NB)} = \arg \max_{c \in C} \left(\log P(c) + \sum_{w \in W} \log P(w | c) \right)$$

- Naive Bayes is a linear classifier: maximum of sum of weights

Estimation (problems)

- now: $|C| = J$
- "split" and "merge" all documents to class specific documents
- and count word frequencies per class $\text{tf}_{w|c_j}$
- what if $\text{tf}_{w|c_j} = 0$?

Estimation (problems)

- now: $|C| = J$
- "split" and "merge" all documents to class specific documents
- and count word frequencies per class $\text{tf}_{w|c_j}$
- what if $\text{tf}_{w|c_j} = 0$?
- $\hat{P}(w | c_j) = \frac{\text{tf}_{w|c_j}}{\sum_{w \in W} \text{tf}_{w|c_j}} = 0$
 - $\log 0$ undefined
 - $\prod_{w \in W} \hat{P}(w | c) = 0$
- imagine a document that contain for each possible class at least one word that does not appear in this class before

Laplace smoothing

- recap, e.g., tf-idf
- add, e.g., 1 to each count:

$$\hat{P}(w \mid c_j) = \frac{\text{tf}_{w|c_j} + 1}{V + \sum_{w \in W} \text{tf}_{w|c_j}}$$

- need not to be 1: α smoothing

$$\hat{P}(w \mid c_j) = \frac{\text{tf}_{w|c_j} + \alpha}{\alpha V + \sum_{w \in W} \text{tf}_{w|c_j}}$$

Practical implications

- unknown words (from training set) are discarded (from test set)
- stopwords are usually considered as features
- big corpus → large vocabulary
- aim: dimension reduction
- while preserving information
- find discriminatory words

Feature extraction from text

- from unstructured text to structured (tabular) data
- DTM (raw counts, tf-idf, other transformations)
- additional meta information
- number of tokens, number of sentences
 - mean number of tokens per sentence
- number of punctuation, exclamation marks, question marks, uppercase
 - relative
- ...
- results in a large amount of features

Feature selection

- find discriminatory features
- features that are non-randomly distributed over target variable
- how to identify?
- several scores to measure importance/impurity/information/dependence on somehow discretized features
 - Gini index
 - entropy
 - PMI
 - mutual information
 - information gain
 - χ^2 statistic

Binarize feature

- a lot of filters only for binary features
- factor/class variable:
 - class vs. not class
- continuous variables:
 - ① divide domain into subdomains
 - ② class vs. not class
- in the following: let f be a given feature

Gini index

- imbalance in the class distribution
- for a given binary/binarized feature
- estimate $P(c_j | f), j = 1, \dots, J$
- ensuring $\sum_{j=1}^J \hat{P}(c_j | f) = 1$

$$G(f) = 1 - \sum_{j=1}^J P(c_j | f)^2 \in [0, 1 - 1/J]$$

- 0 refers to *most informative* regarding classification
- what if: imbalanced class distribution?

Gini index

- imbalance in the class distribution
- for a given binary/binarized feature
- estimate $P(c_j | f), j = 1, \dots, J$
- ensuring $\sum_{j=1}^J \hat{P}(c_j | f) = 1$

$$G(f) = 1 - \sum_{j=1}^J P(c_j | f)^2 \in [0, 1 - 1/J]$$

- 0 refers to *most informative* regarding classification
- what if: imbalanced class distribution?
- not such a big problem (for feature selection) \Rightarrow does not affect order

$$G_{norm}(f) = 1 - \sum_{j=1}^J \left[\frac{P(c_j | f) / \# c_j}{\sum_{s=1}^J P(c_s | f) / \# c_s} \right]^2$$

Conditional entropy

- measure of certainty of differentiating class labels

$$\begin{aligned} E(f) = & - \sum_{j=1}^J \left[\frac{\#f}{M} P(c_j | f) \log P(c_j | f) \right. \\ & \left. + \frac{M - \#f}{M} P(c_j | \neg f) \log P(c_j | \neg f) \right] \end{aligned}$$

- $E(f) \in (0, \log J)$
- 0 refers to *most informative* regarding classification

PMI as filter method

- "correlation" between presence of feature and class labels
- does not (directly) include information about absence of feature

$$\text{PMI}_{\text{avg}}(f) = \sum_{j=1}^J \frac{\#c_j}{M} \log \frac{P(c_j | f)}{P(c_j)}$$

$$\text{PMI}_{\text{max}}(f) = \max_j \log \frac{P(c_j | f)}{P(c_j)}$$

- larger values are to be preferred

Mutual information (MI) and information gain (IG)

- both result in **same order as conditional entropy**

$$\begin{aligned} \text{MI}(f) = & \sum_{j=1}^J [P(c_j \wedge f) \log \frac{P(c_j | f)}{P(c_j)} \\ & + P(c_j \wedge \neg f) \log \frac{P(c_j | \neg f)}{P(c_j)}] \end{aligned}$$

- $\text{MI}(f) \in [0, \infty)$
- 0 refers to statistical independence of f and class distribution

$$\text{IG}(f) = - \sum_{j=1}^J \frac{\#c_j}{M} \log \frac{\#c_j}{M} - E(f)$$

- entropy in class distribution – conditional entropy
- $\text{IG}(f) = \text{MI}(f)$

χ^2 statistic

- contingency table
- difference between observed and expected frequency (under independence)
- possible to calculate for discretized features with I regions
(not necessarily binarized)
- $o_{ij} = \#f_i \wedge c_j, e_{ij} = \frac{\#f_i \# c_j}{M}$

$$\chi^2(f) = \sum_{i=1}^I \sum_{j=1}^J \frac{(o_{ij} - e_{ij})^2}{e_{ij}}$$

- Cramer's V to adjust for degrees of freedom

$$V(f) = \sqrt{\frac{\chi^2(f)}{M \min\{I, J\}}}$$

What else?

- filter methods useful for dimension reduction
- selection biased towards training set
 - be aware of overfitting
- do not just trust the(se) numbers
 - filter methods are univariate
 - maybe there is multivariate information
- alternatives:
 - wrapper methods
 - dimension reduction, as, e.g., principal component analysis
- assigning labels (unsupervised) - see section clustering and topics

What is clustering?

- finding groups (clusters) of observations
- given some characteristics
- \sim unsupervised classification
- no labeled data
 - scenarios with labeled data to assess performance
 - then: what if number clusters \neq number of classes?
- problem: what is a valid/good result?

Why clustering?

- group texts into (previously undefined) topics
- exploratory data analysis
- find representative examples for groups

Document similarity

- popular approach
- distance matrix (\sim similarity)
- maximizing inter-cluster distance
- minimizing intra-cluster distance
- several measures:

- e.g., cosine similarity:
$$\cos(D^{(m_1)}, D^{(m_2)}) = \frac{\sum_{w=1}^V \text{tf}_{w,m_1} \text{tf}_{w,m_2}}{\sqrt{\sum_{w=1}^V \text{tf}_{w,m_1}^2} \sqrt{\sum_{w=1}^V \text{tf}_{w,m_2}^2}}$$

Distance measures

$$\text{euclid}(D^{(m_1)}, D^{(m_2)}) = \sqrt{\sum_{w=1}^V (\text{tf}_{w,m_1} - \text{tf}_{w,m_2})^2}$$

$$\text{manhattan}(D^{(m_1)}, D^{(m_2)}) = \sum_{w=1}^V |\text{tf}_{w,m_1} - \text{tf}_{w,m_2}|$$

$$L_p(D^{(m_1)}, D^{(m_2)}) = \left(\sum_{w=1}^V |\text{tf}_{w,m_1} - \text{tf}_{w,m_2}|^p \right)^{1/p}$$

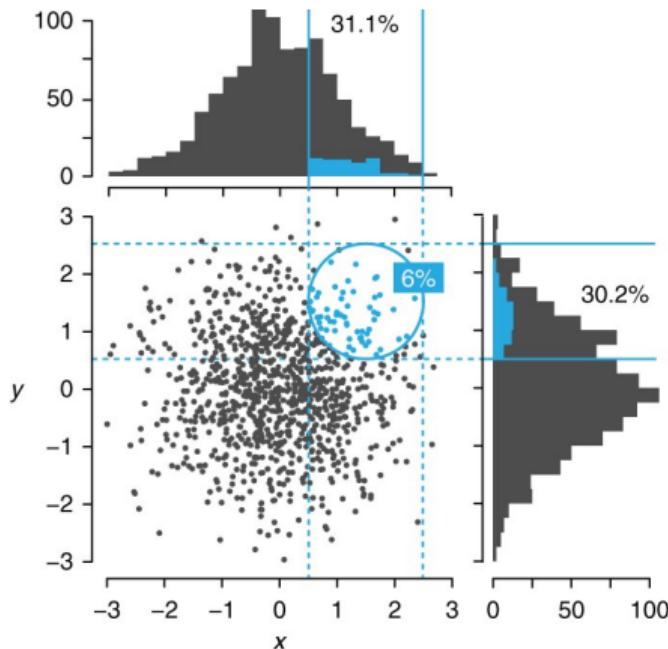
- most clustering methods work with distances

Curse of dimensionality

- here: dimension V
- d_{\max} maximum distance between two documents
- d_{\min} minimum distance between two documents

$$\frac{d_{\max} - d_{\min}}{d_{\min}} \xrightarrow[V \rightarrow \infty]{} 0$$

- implications:
 - differences in distances become small
 - ranks may remain meaningful
 - noise may cover signal if too much (meaningless) features
 - data get sparse



The curse(s) of dimensionality (Altman et. al, 2018). DOI: [10.1038/s41592-018-0019-x](https://doi.org/10.1038/s41592-018-0019-x)

k-means

- aim: divide data ($X = (x_m)_{m=1,\dots,M}$) into k subsets
 - $x_m = (x_{mp})_{p=1,\dots,P}$, P number of features
- each subset is represented by element-wise mean (μ_{c_j}) of observations
- minimizing least squares

$$\sum_{m=1}^M \sum_{p=1}^P (x_{mp} - \mu_{\hat{c}_m, p})^2$$

- squared deviations put more weight on large differences
- iterative estimation
- k user-chosen
- initialization important: <https://www.geeksforgeeks.org/ml-k-means-algorithm/>

k-medoids

- *k*-means uses squared euclidean distance
- for *k*-medoids we can choose a distance measure
- popular algorithm: partitioning around medoids (PAM)
 - build: find "best" initial set as representatives
 - swap: change each medoid to a "better" option regarding distance measure

Weighting of distances

- each dimension gets the same weight for k -means
- problem?

Weighting of distances

- each dimension gets the same weight for k -means
- problem?
- different scaling of the features
- need to normalize features!

Weighting of distances

- each dimension gets the same weight for k -means
- problem?
- different scaling of the features
- need to normalize features!

Finding the right k :

- typically scree plot / elbow plot

Hierarchical clustering

- another approach, typically agglomerative
 - ① every observation is a single cluster
 - ② merge the two most similar clusters
 - ③ repeat until only one cluster left
- result: dendrogram
- choose "interesting" subtrees as clusters
- modifications:
 - distance measure (for observations)
 - linkage measure (for clusters)

Linkages

- popular linkage measures to minimize:

$$\text{single: } \min_{a \in A, b \in B} d(a, b)$$

$$\text{complete: } \max_{a \in A, b \in B} d(a, b)$$

$$\text{average: } \frac{1}{|A||B|} \sum_{a \in A} \sum_{b \in B} d(a, b)$$

$$\text{Ward: } \frac{|A||B|}{|A \cup B|} (\mu_A - \mu_B)^2$$

- Ward only for squared euclidean distance

Find clusters from dendrogram

- various strategies:
 - extract clusters "by hand"
 - stop at a certain distance of clusters
 - stop if a certain number of clusters remain
- number of clusters k is not the only parameter for evaluation
- how well are the borders of segmentation?

Evaluation

- typically unsupervised (no labeled data)
- class sets: C_1, \dots, C_k
- silhouette:

- distance of x_m to own cluster

$$d_1(x_m) = \frac{1}{|C_{\hat{c}_m}| - 1} \sum_{y \in C_{\hat{c}_m}, y \neq x_m} d(y, x_m)$$

- distance of x_m to closest other cluster

$$d_2(x_m) = \min_{j \in \{1, \dots, k\}, j \neq \hat{c}_m} \frac{1}{|C_j|} \sum_{y \in C_j} d(y, x_m)$$

$$\text{silhouette}(x_m) = \frac{d_2(x_m) - d_1(x_m)}{\max\{d_2(x_m), d_1(x_m)\}} \in [-1, 1]$$

- if $|C_{\hat{c}_m}| > 1$, else 0
 - silhouette of whole clustering is $\frac{1}{M} \sum_m \text{silhouette}(x_m)$
 - curse of dimensionality leads to $\text{silhouette}(x_m) \rightarrow 0$

Supervised evaluation

- real world problems: no labeled data
- in research often important to assess quality of clustering
- use of labeled data for performance evaluation
- problems?

Supervised evaluation

- real world problems: no labeled data
- in research often important to assess quality of clustering
- use of labeled data for performance evaluation
- problems?
 - $k = J?$ not necessarily

Supervised evaluation

- real world problems: no labeled data
- in research often important to assess quality of clustering
- use of labeled data for performance evaluation
- problems?
 - $k = J?$ not necessarily
 - which class matches which cluster?
 - multiple classes in one cluster
 - multiple clusters for one class
- approaches:
 - best matching (hungarian/Kuhn's method)
 - compare all possible pairs

Evaluation measures

- precision (purity) and recall are useful to apply here
- F_1 measure as combination (cf., slide 111)

$$F_1 = \frac{2\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

- mutual information, entropy, ... can also be used

Principal component analysis (PCA)

- highly related to clustering: PCA as dimension reduction method
- $X \in \mathbb{R}^{M \times V}$ or $X \in \mathbb{R}^{M \times P}$
- normalize features such that $\sum_{m=1}^M x_{mp} = 0$
- idea: find transformation

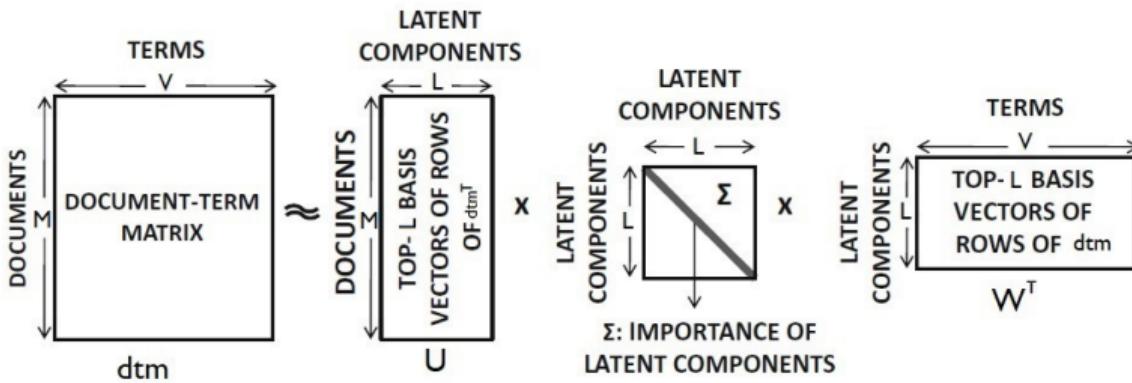
$$T = XW, \quad T \in \mathbb{R}^{M \times P}, \quad W \in \mathbb{R}^{P \times P}$$

- use truncated version $T_L \in \mathbb{R}^{M \times L}$, $W_L \in \mathbb{R}^{P \times L}$ for dimension reduction
- sometimes useful to use PCA before clustering, e.g., k -means

Singular value decomposition (SVD)

- associated to PCA via matrix factorization
- typically $X = \text{dtm} \in \mathbb{R}^{M \times V}$, maybe tf-idf weighted
- $X = U\Sigma W^T$, $U \in \mathbb{R}^{M \times M}$, $\Sigma \in \mathbb{R}_+^{M \times V}$, $W \in \mathbb{R}^{V \times V}$
- Σ rectangular diagonal matrix, singular values of X
- regarding PCA: $T = U\Sigma$
- truncated SVD: $X \approx U_L\Sigma_L W_L^T$, $U_L \in \mathbb{R}^{M \times L}$, $\Sigma_L \in \mathbb{R}_+^{L \times L}$, $W_L \in \mathbb{R}^{V \times L}$
- SVD applied to dtm is named latent semantic indexing/analysis (LSI/LSA)

Latent semantic analysis (LSA)



cf. Machine Learning for Text, pp. 37

- U, Σ, W all refers to the truncated versions here
- U and W are both non-deterministic
- hard to interpret
- overlapping vocabulary contradicts idea
- clusters in SVD representation can be interpreted as topics (cf. following slides)

Topic Models

- Probabilistic Topic Models
- Probabilistic Latent Semantic Analysis (pLSA)
- Latent Dirichlet Allocation (LDA)
- Perplexity and Coherence
- Extensions of LDA
- The Structural Topic Model (STM)
- Evaluation and Best Practice

Probabilistic topic models

- SVD = LSA is matrix factorization method
- now: probabilistic topic models¹
- notation: we switch from L (number of dimensions) in SVD/LSA to K (number of topics) in topic modeling
- basic idea: $D^{(m)}$ is mixture of topics $T_k, k = 1, \dots, K$

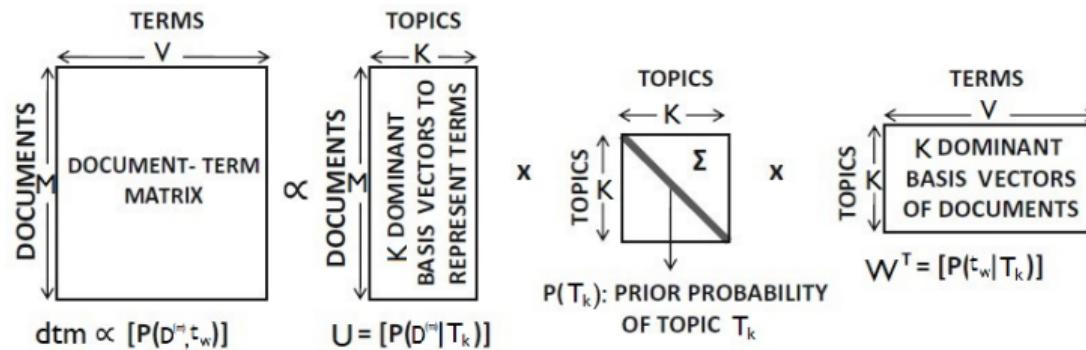
$$\sum_{k=1}^K P(D^{(m)} \in T_k) = \sum_{k=1}^K \theta_{m,k} = 1$$

- words are drawn from

$$P(W_n^{(m)}) = \sum_{k=1}^K P(W_n^{(m)} | D^{(m)} \in T_k) P(D^{(m)} \in T_k) = \sum_{k=1}^K \phi_{k,w} \theta_{m,k}$$

¹ Probabilistic topic models (Blei, 2012) <https://doi.org/10.1145/2133806.2133826>

Probabilistic LSA (pLSA)



cf. Machine Learning for Text, pp. 50

$$P(D^{(1)}, \dots, D^{(M)}) = \prod_{m=1}^M \prod_{n=1}^{N^{(m)}} \sum_{k=1}^K \phi_{k, W_n^{(m)}} \theta_{m,k}$$

$$P(w, d) = \sum_{k=1}^K P(w | k) P(k) P(d | k) = P(d) \sum_{k=1}^K P(w | k) P(k | d)$$

pLSA estimation

- estimation via expectation maximization (EM) algorithm:

- choose initial model parameters

(E-step) expect latent variables (e.g., topic assignments) from model parameters and the given data

(M-step) update model parameters to maximize the likelihood of observing the given data

- repeat

- E-step (estimate $T_n^{(m)}$):

$$P(T_n^{(m)} = T_k) \propto \phi_{k,W_n^{(m)}} \theta_{m,k}$$

- M-step (optimize ϕ and θ):

$$\theta_{m,k} \propto \sum_{w=1}^V \text{tf}_{w,m} P(T_n^{(m)} = T_k),$$

$$\phi_{k,w} \propto \sum_{m=1}^M \text{tf}_{w,m} P(T_n^{(m)} = T_k)$$

pLSA Example

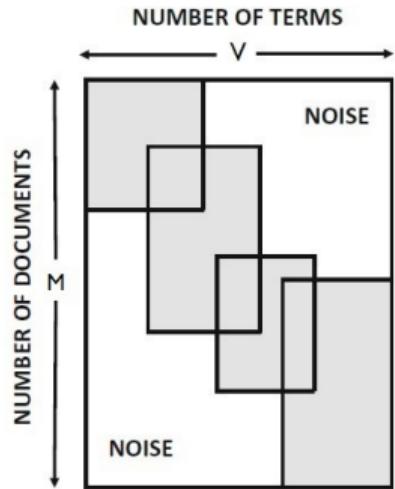
	LION	TIGER	CHEETAH	JAGUAR	PORSCHE	FERRARI	
CATS	\bar{X}_1	2	2	1	2	0	0
BOTH	\bar{X}_2	2	3	3	3	0	0
CARS	\bar{X}_3	1	1	1	1	0	0
	\bar{X}_4	2	2	2	3	1	1
	\bar{X}_5	0	0	0	1	1	1
	\bar{X}_6	0	0	0	2	1	2
	dtm						

$$\begin{array}{c}
 \text{dtm} \\
 \times \\
 \begin{matrix}
 \text{CATS} & \text{CARS} \\
 \hline
 \bar{X}_1 & \frac{2}{8} & 0 \\
 \bar{X}_2 & \frac{3}{8} & 0 \\
 \bar{X}_3 & \frac{1}{8} & 0 \\
 \bar{X}_4 & \frac{2}{8} & \frac{1}{4} \\
 \bar{X}_5 & 0 & \frac{1}{4} \\
 \bar{X}_6 & 0 & \frac{2}{4}
 \end{matrix}
 \end{array}
 \times
 \begin{matrix}
 \text{CATS} & \text{CARS} \\
 \hline
 \bar{X}_1 & \frac{8}{11} & 0 \\
 \bar{X}_2 & 0 & \frac{3}{11}
 \end{matrix}
 \times
 \begin{matrix}
 \text{LION} & \text{TIGER} & \text{CHEETAH} & \text{JAGUAR} & \text{PORSCHE} & \text{FERRARI} \\
 \hline
 \text{CATS} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & 0 \\
 \text{CARS} & 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3}
 \end{matrix}
 \times
 \mathbf{W}^T$$

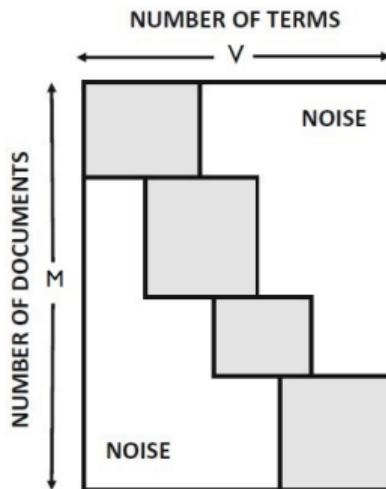
α Σ

cf. Machine Learning for Text, pp. 51

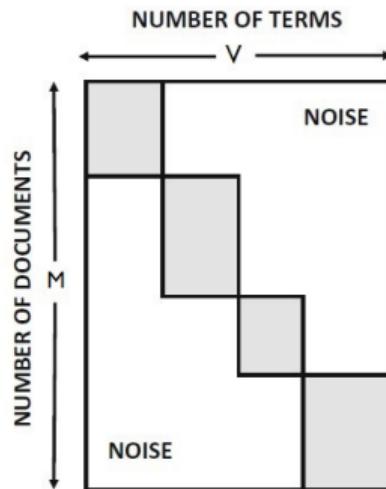
Mixed membership models



(a) Mixed membership and overlapping vocabulary
(e.g., PLSA, LDA)



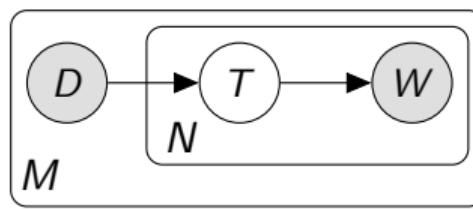
(b) Single membership and overlapping vocabulary
(e.g., k -means)



(c) Single membership and disjoint vocabulary
(e.g., co-clustering)

cf. Machine Learning for Text, pp. 80

Plate notation of pLSA



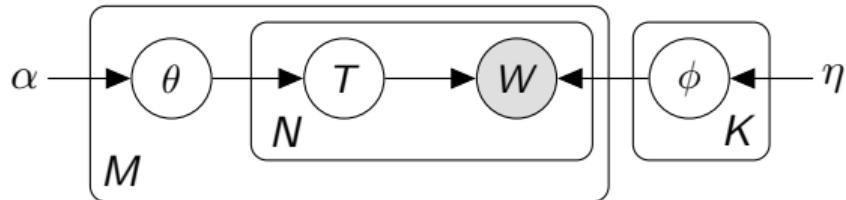
a observable variable



a latent variable

- plates are interpreted as loops or repetition of covered elements
- here:
 - M documents D
 - N token-topic-pairs (W, T) per document
 - arrows indicate dependency:
document affects (latent) topic, which affects (observed) word

Latent Dirichlet allocation (LDA)



$$W_n^{(m)} \mid T_n^{(m)}, \phi_k \sim \text{Discrete}(\phi_k) \quad \phi_k \sim \text{Dirichlet}(\eta)$$

$$T_n^{(m)} \mid \theta_m \sim \text{Discrete}(\theta_m) \quad \theta_m \sim \text{Dirichlet}(\alpha)$$

- α Dirichlet prior for topic distribution per document
- η Dirichlet prior for word distribution per topic
- mostly chosen as symmetric, because typically no a-priori information

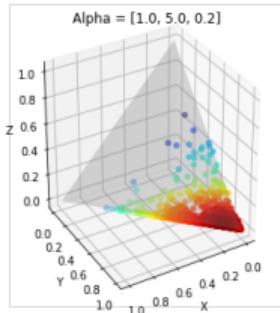
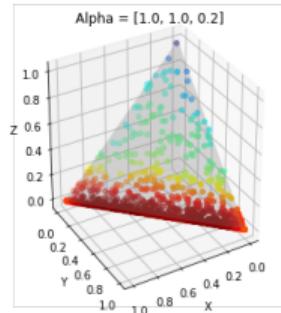
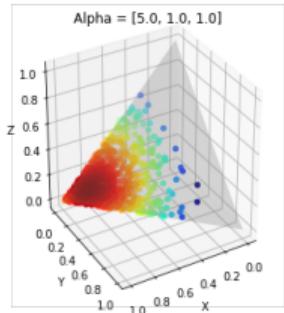
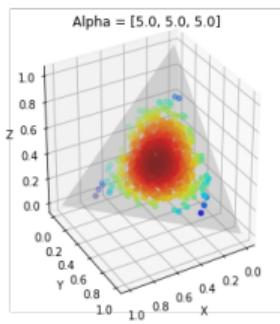
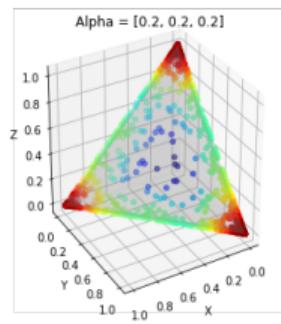
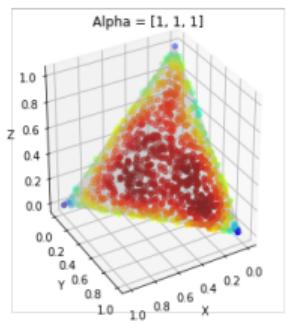
see also http://videolectures.net/mlss09uk_blei_tm/

Dirichlet priors

$$\text{Dirichlet}(x_1, \dots, x_K \mid \alpha) = \frac{\Gamma(\alpha K)}{\Gamma(\alpha)^K} \prod_{k=1}^K x_k^{\alpha-1}$$

- Γ gamma function
- Rethinking LDA: Why Priors Matter (Wallach et al., 2009)
 - promising to tune α
 - not necessary to tune η
 - how many topics per document do you expect?

Example Dirichlet distributions, $K = 3$



<https://towardsdatascience.com/light-on-math-machine-learning-intuitive-guide-to-latent-dirichlet-allocation-437c81220158>

Joint distribution

$$\begin{aligned} P(D, T) &= P(D \mid T)P(T) = \int \prod_{k=1}^K P(\phi_k \mid \eta) \prod_{m=1}^M \prod_{n=1}^{N^{(m)}} P(W_n^{(m)} \mid \phi_k) \, d\phi \\ &\quad \cdot \int \prod_{m=1}^M P(\theta_m \mid \alpha) \prod_{n=1}^{N^{(m)}} P(T_n^{(m)} \mid \theta_m) \, d\theta \\ &= \dots \\ &= \left(\frac{\Gamma(V\eta)}{(\Gamma(\eta))^V} \right)^K \cdot \prod_{k=1}^K \frac{\prod_{w=1}^V \Gamma(n_k^{(\bullet w)} + \eta)}{\Gamma(V\eta + n_k)} \\ &\quad \cdot \left(\frac{\Gamma(K\alpha)}{(\Gamma(\alpha))^K} \right)^M \cdot \prod_{m=1}^M \frac{\prod_{k=1}^K \Gamma(n_k^{(m \bullet)} + \alpha)}{\Gamma(K\alpha + N^{(m)})} \end{aligned}$$

Inference

- exact solution intractable
- various numerical approximations
- (collapsed) variational inference
- (collapsed) Gibbs sampling
- tensor decomposition

- here: only collapsed Gibbs sampling
- estimation does not have decisive influence, parameter choice (in conjunction with estimation process) more important¹

¹ On smoothing and inference for topic models (Asuncion et al., 2009)

<https://dl.acm.org/doi/10.5555/1795114.1795118>

Estimators (via Gibbs sampling)

$$\hat{\theta}_{m,k} = \frac{n_k^{(m\bullet)} + \alpha}{N^{(m)} + K\alpha}$$

$$\hat{\phi}_{k,w} = \frac{n_k^{(\bullet w)} + \eta}{n_k^{(\bullet\bullet)} + V\eta}$$

- $n_k^{(mw)}$ number of assignments of word w to topic k in text m
- • represents sum, e.g., $n_k^{(m\bullet)} = \sum_{w=1}^V n_k^{(mw)}$

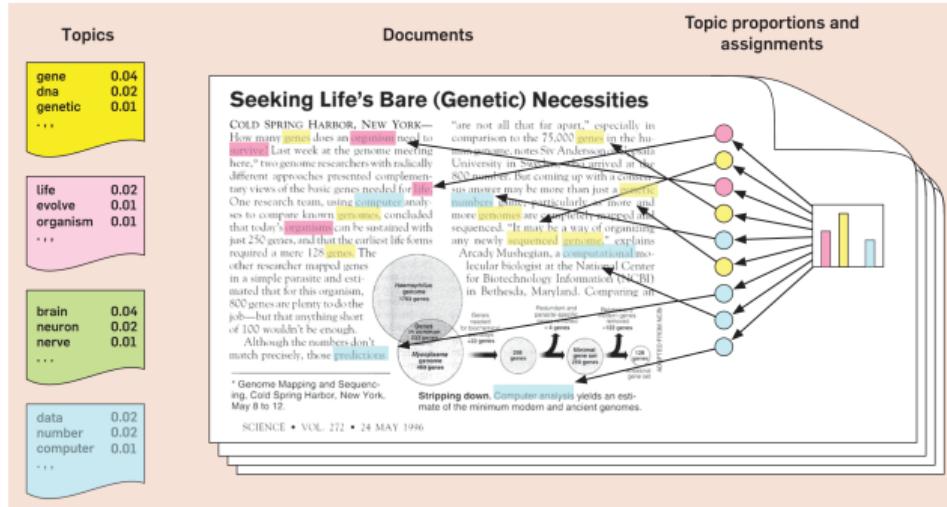
Collapsed Gibbs sampling

$$P(T_n^{(m)} = T_k \mid T_{-n}^{(m)}, d) \propto \frac{n_{-n,k}^{(\bullet w)} + \eta}{n_{-n,k}^{(\bullet\bullet)} + V\eta} \frac{n_{-n,k}^{(m\bullet)} + \alpha}{N_{-n}^{(m)} + K\alpha}$$

- $n_{-n,k}^{(mw)} = n_k^{(mw)}$ "minus" $T_n^{(m)}$

- ① choose priors α and η
- ② initialize topic assignments $T_n^{(m)}$ randomly
- ③ for every token in every document $W_n^{(m)}$:
 - ① remove $T_n^{(m)}$ from all counts
 - ② sample a "new" $T_n^{(m)}$ given the formula above
- ④ repeat

LDA result



(Blei, 2012)

- estimators $\hat{\theta}_{m,k}$ and $\hat{\phi}_{k,w}$
- soft cluster, mixed membership
- overlapping clusters of words by topic

Perplexity

$$\exp \left\{ -\frac{\sum_{m=1}^M \sum_{w=1}^V n_{\bullet}^{(mw)} \log \left(\sum_{k=1}^K \hat{\theta}_{m,k} \hat{\phi}_{k,w} \right)}{\sum_{m=1}^M N^{(m)}} \right\}$$

- in-sample: how good is the model's fit to the given data
- hold-out perplexity: how well can our model predict unseen documents

see also topicmodels: An R package for fitting topic models (Grün & Hornik, 2011)
<https://doi.org/10.18637/jss.v040.i13>

Coherence

Importance score:

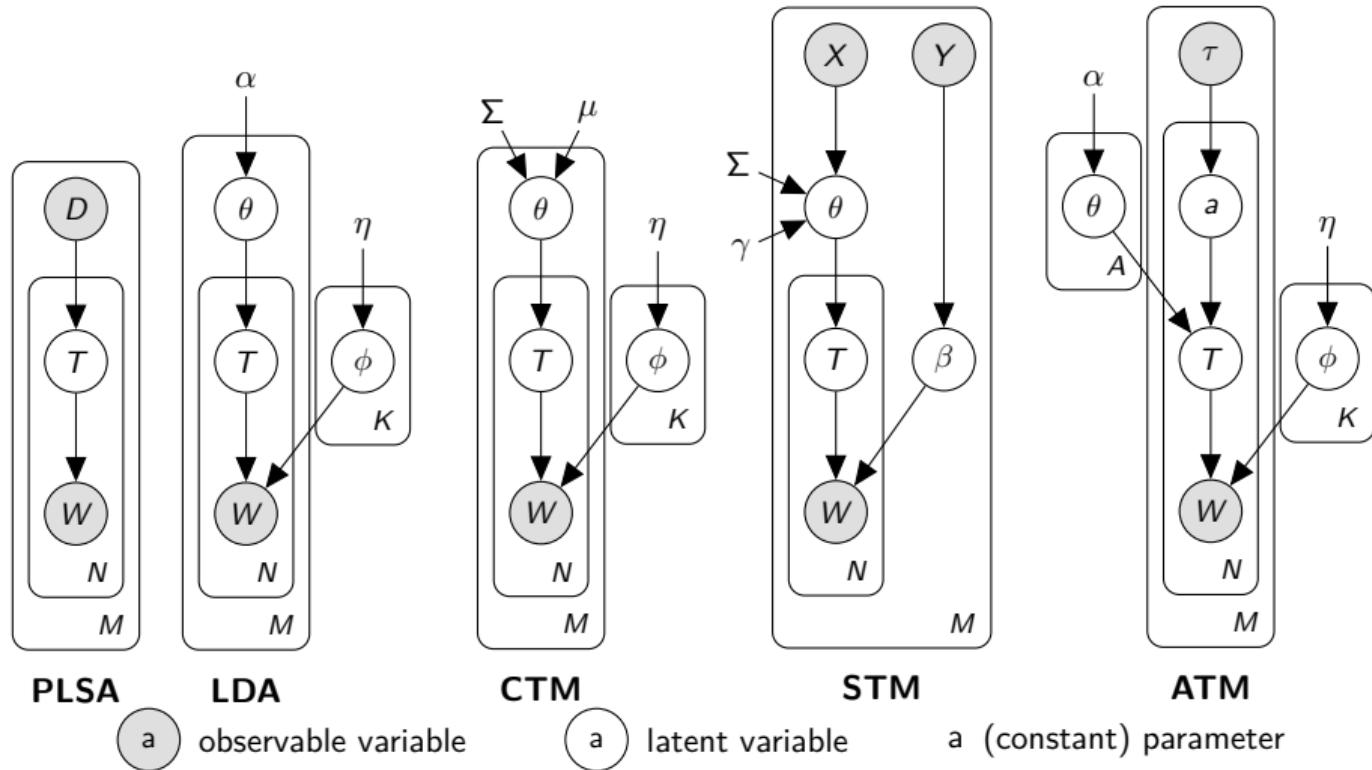
$$I(w, k) = \frac{n_k^{(\bullet w)}}{n_k^{(\bullet\bullet)}} \left[\log \left(\frac{n_k^{(\bullet w)}}{n_k^{(\bullet\bullet)}} + \varepsilon \right) - \frac{1}{K} \sum_{l=1}^K \log \left(\frac{n_l^{(\bullet w)}}{n_l^{(\bullet\bullet)}} + \varepsilon \right) \right]$$

- idea: how coherent is a single topic
- e.g., NPMI (see definition on slide 98)
- take top words (most representative) per topic and calculate their co-occurrence
- importance measure $I(w, k)$ to "filter" stopwords
- aggregation of values per topic for model evaluation

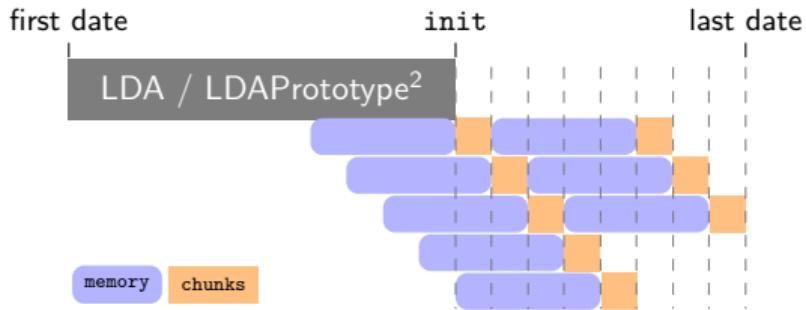
Extensions

- a lot of extensions/adaptions published
- hierarchical topic models
- correlated topic models (CTM)
- author-topic model (ATM)
- structural topic model (STM)
- time dynamic topic model (cDTM)
- online LDA

Schematic plate representation of topic models



RollingLDA¹



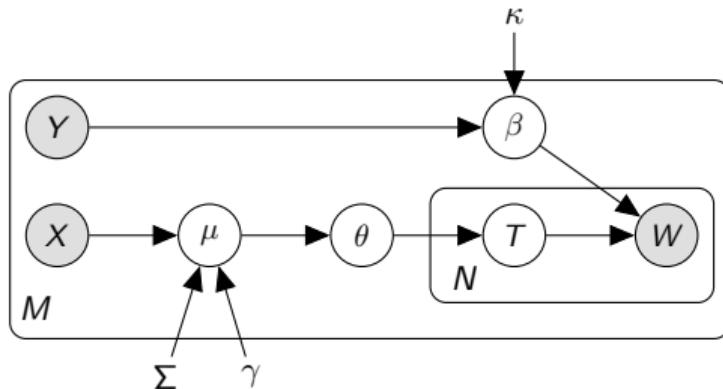
- discrete updates of LDA with new documents
- three additional parameters:
 - init: last date of initialization period
 - chunks: first date of each update period
 - memory: first date of each memory for the update periods

¹ RollingLDA: An Update Algorithm of Latent Dirichlet Allocation to Construct Consistent Time Series from Textual Data (Rieger et al., 2021) DOI: 10.18653/v1/2021.findings-emnlp.201, see also

<https://zpid.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=03addb74-2f86-4ed0-b877-af9c00a2959d> for an online talk on RollingLDA

² LDAPrototype: a model selection algorithm to improve reliability of latent Dirichlet allocation (Rieger et al., 2024) DOI: 10.7717/peerj-cs.2279

Structural Topic Model (STM)



- if no topical prevalence and no topical content covariates, STM becomes the correlated topic model (CTM)

<https://scholar.princeton.edu/files/bstewart/files/stmnips2013.pdf>

Generative model

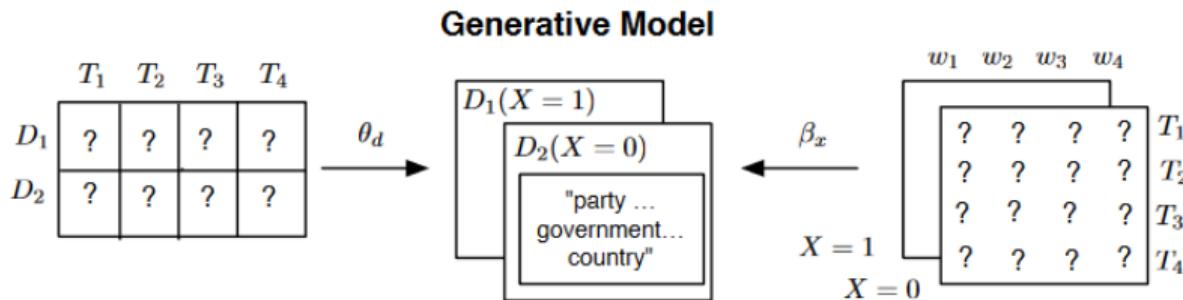
$$\begin{aligned} W_n^{(m)} \mid T_n^{(m)}, \beta_k^{(m)} &\sim \text{Discrete}(\beta_k^{(m)}) \\ T_n^{(m)} \mid \theta_m &\sim \text{Discrete}(\theta_m) \end{aligned}$$

where

$$\begin{aligned} \beta_k^{(m)} &\propto \exp(v + \kappa_k^{(t)} + \kappa_{y_m}^{(c)} + \kappa_{y_m, k}^{(i)}) \\ \kappa &\sim \text{Laplace}(0, \tau) \quad \tau \sim \text{Gamma}(s^\kappa, r^\kappa) \\ \theta_m &\sim \text{LogNormal}(\mu_m, \Sigma) \\ \mu_{m,k} = X_m \gamma_k &\quad \gamma_k \sim \mathcal{N}(0, \sigma_k^2) \quad \sigma_k^2 \sim \text{Gamma}(s^\gamma, r^\gamma) \end{aligned}$$

- v baseline word distribution
- y_m document-level content covariate

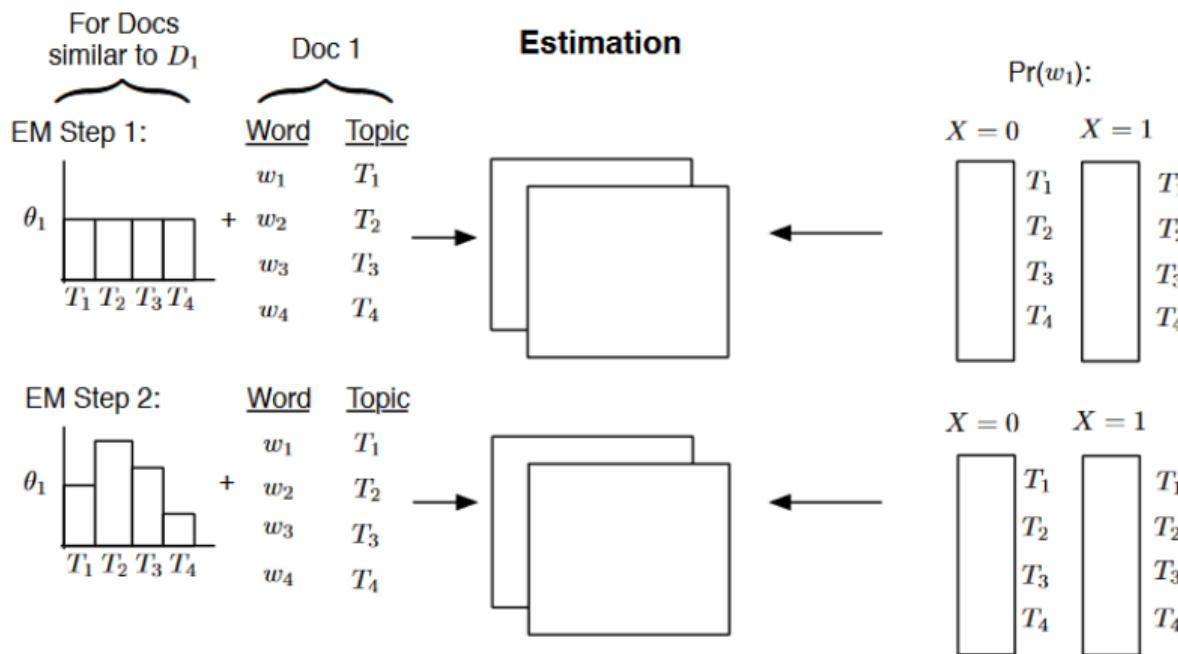
Generative model



- left: topical prevalence
- right: topical content

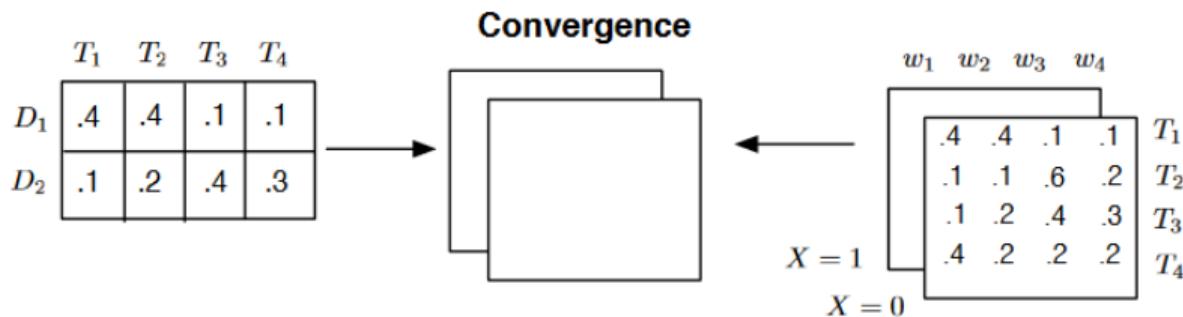
DOI: 10.18637/jss.v091.i02

EM algorithm



DOI: 10.18637/jss.v091.i02

Convergence



DOI: [10.18637/jss.v091.i02](https://doi.org/10.18637/jss.v091.i02)

Further resources

- <https://www.structuraltopicmodel.com/>
- <https://burtmonroe.github.io/TextAsDataCourse/Tutorials/IntroSTM.nb.html>
 - Introductory tutorial for STM
- <https://towardsdatascience.com/introduction-to-the-structural-topic-model-stm-34ec4bd5383>
 - Introduction to The Structural Topic Model (STM)
 - tds/medium story

Reading tea leaves¹

- perplexity measures fit (likelihood based)
- coherence based measures introduce the idea of clearly separated topics
 - contradicts the mixed-membership idea of LDA
- manual inspection popular (eye-balling)
- evaluation through downstream task straightforward

Reading tea leaves¹

- perplexity measures fit (likelihood based)
- coherence based measures introduce the idea of clearly separated topics
 - contradicts the mixed-membership idea of LDA
- manual inspection popular (eye-balling)
- evaluation through downstream task straightforward
- alternative: manual coding experiment¹
 - word & topic intrusion task
 - can a human identify a word/topic that does not belong to a topic/document?

¹ (Chang et al., 2009) <https://papers.nips.cc/paper/2009/hash/f92586a25bb3145facd64ab20fd554ff-Abstract.html>

Word/topic intrusion

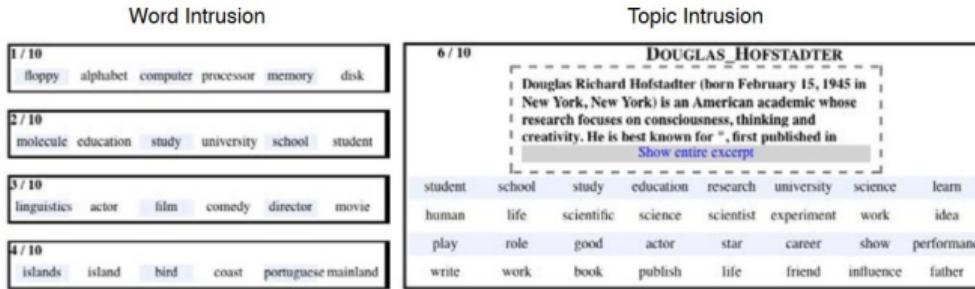


Figure 2: Screenshots of our two human tasks. In the word intrusion task (left), subjects are presented with a set of words and asked to select the word which does not belong with the others. In the *topic intrusion* task (right), users are given a document's title and the first few sentences of the document. The users must select which of the four groups of words does not belong.

(Chang et al., 2009)

Characteristics of (probabilistic) topic models/LDA

- clustering mostly hard, LDA soft
- special adaptations of clustering for text, e.g., priors
- question of how to evaluate (still) not solved

Characteristics of (probabilistic) topic models/LDA

- clustering mostly hard, LDA soft
- special adaptations of clustering for text, e.g., priors
- question of how to evaluate (still) not solved
- estimation via Gibbs is stochastic
- strongly depends on initialization
- in practice this leads to strongly different topics
 - there are ways to gain in reliability¹

Characteristics of (probabilistic) topic models/LDA

- clustering mostly hard, LDA soft
- special adaptations of clustering for text, e.g., priors
- question of how to evaluate (still) not solved
- estimation via Gibbs is stochastic
- strongly depends on initialization
- in practice this leads to strongly different topics
 - there are ways to gain in reliability¹
- still only bag-of-words (BOW) models
- popular in application (e.g., social sciences) studies

¹ Reliability evaluation and an update algorithm for the latent Dirichlet allocation (Rieger, 2022)

<https://doi.org/10.17877/DE290R-22949>

Which topic model to use and how¹

- “It’s fine to use LDA”
 - accessibility
 - performance of LDA (still) strong (cf., Harrando et al., 2021, Hoyle et al., 2022)
- topic model evaluation difficult
 - evaluate by hand (cf., Hoyle et al., 2021)
 - label the topics manually (LLMs are looking promising more recently!)
- resist the idea of *the right* number of topics! — try different numbers (and evaluate)
- LDA works better on longer documents (pages, news articles, etc.)
shorter documents (e.g., tweets) need more curation
- use Gibbs sampling
 - there is no (clear, published) evidence that it works better
 - it’s more like our all (researchers’) experience

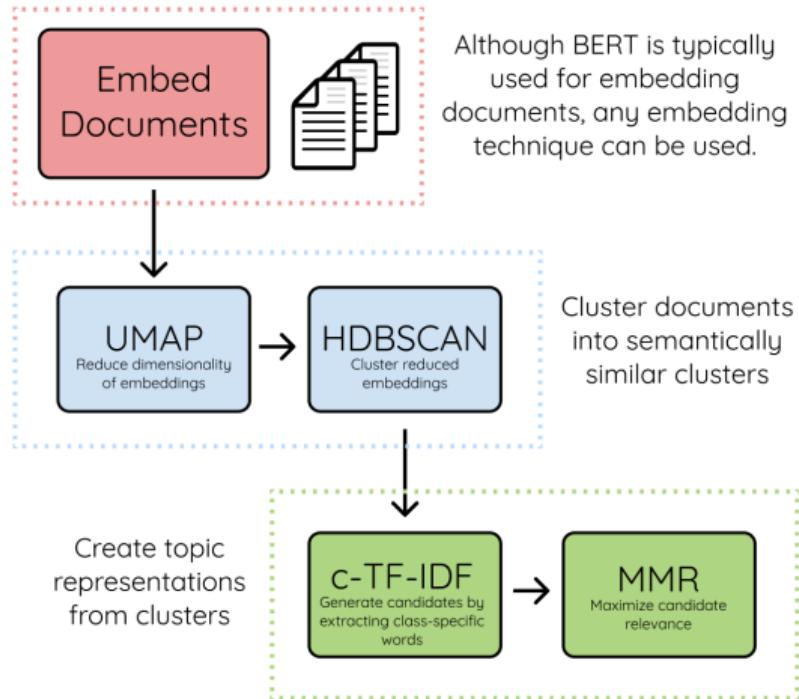
¹<https://maria-antoniak.github.io/2022/07/27/topic-modeling-for-the-people.html>

Preprocessing for topic modeling¹

- don't stem (cf., Schofield et al., 2016)
- be conservative in stopword removal (cf., Schofield et al., 2017)
- remove duplicates (cf., Schofield et al., 2017)
- curate your training data
- bias away your data from known structures
 - you don't want your topic model to reflect just known structures (e.g., authorship, genres)
 - you want to find unknown (new) themes
 - there is a dedicated preprocessing technique for this: Thompson et al. (2018)

¹<https://maria-antoniak.github.io/2022/07/27/topic-modeling-for-the-people.html>

BERTopic as a language model alternative to probabilistic topic modeling



(probabilistic) topic modeling

- soft clustering, mixed membership
- topic = words' probability distribution
- bag of words
- preprocessing needed

LM-based topic modeling

- (hard) clustering of documents
- topic = cluster representative
- (contextual) word embeddings
- typically no preprocessing

<https://github.com/MaartenGr/BERTopic/blob/master/docs/img/algorithm.png>

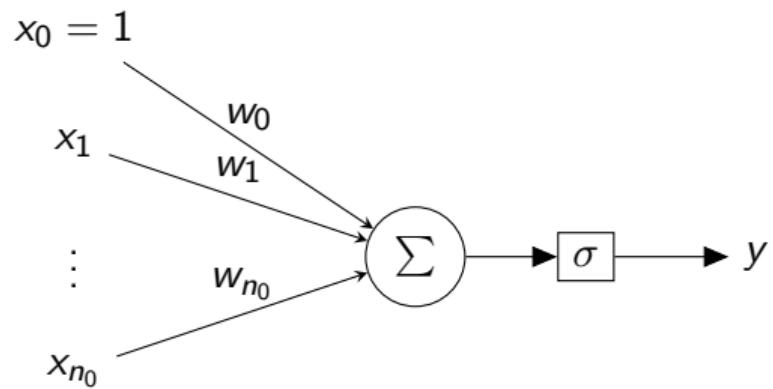
What We Can Do and Cannot Do with Topic Modeling¹

- do not overdesign, use proper/appropriate methods
- assess and report reliability and validity (especially when testing hypotheses)
- exploratory analysis method
- validate with respect to your research question at hand
- report all decisions (e.g., preprocessing)

¹ (Chen et al., 2023) DOI: [10.1080/19312458.2023.2167965](https://doi.org/10.1080/19312458.2023.2167965)

5 Neural Networks

Neural networks - one perceptron



- $z = \sum_{i=0}^{n_0} w_i x_i$
- $y = f(z)$
- f non-linear activation function, often sigmoid σ

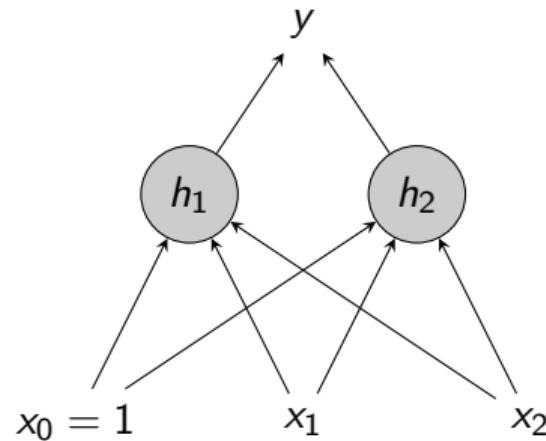
Non-linear activation function

- enables non-linearity in NNs
- sigmoid $\sigma(z) = 1/(1 + \exp -z)$
- $\tanh = (\exp(z) - \exp -z)/(\exp(z) + \exp -z)$
- rectified linear unit $\text{ReLU}(z) = \max\{0, z\}$
- ...
- softmax(z_i) = $\exp(z_i) / \sum_{j=1}^k \exp(z_j)$, $i = 1, \dots, k$

XOR problem

- why more than one layer networks?
- XOR = exclusive OR
- cannot be calculated by only one perceptron
- which activation function?

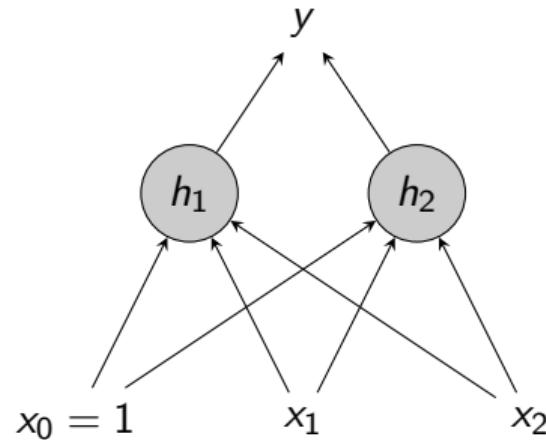
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0



XOR problem

- why more than one layer networks?
- XOR = exclusive OR
- cannot be calculated by only one perceptron
- which activation function? ReLU

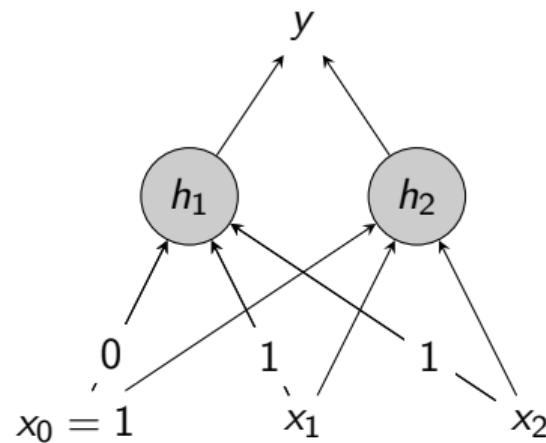
x_1	x_2	y	h_1	h_2
0	0	0	0	-1 → 0
0	1	1	1	0
1	0	1	1	0
1	1	0	2	1



XOR problem

- why more than one layer networks?
- XOR = exclusive OR
- cannot be calculated by only one perceptron
- which activation function? ReLU

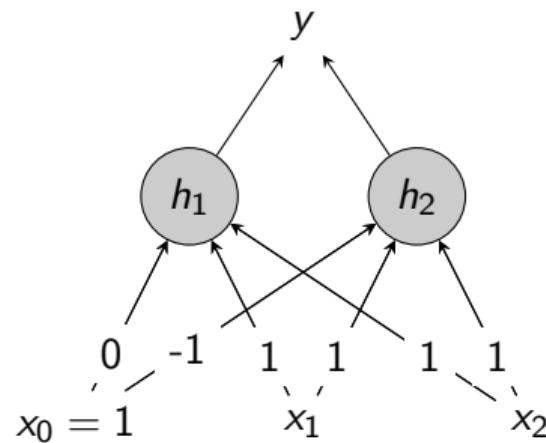
x_1	x_2	y	h_1	h_2	
0	0	0	0	-1	$\rightarrow 0$
0	1	1	1	0	
1	0	1	1	0	
1	1	0	2	1	



XOR problem

- why more than one layer networks?
- XOR = exclusive OR
- cannot be calculated by only one perceptron
- which activation function? ReLU

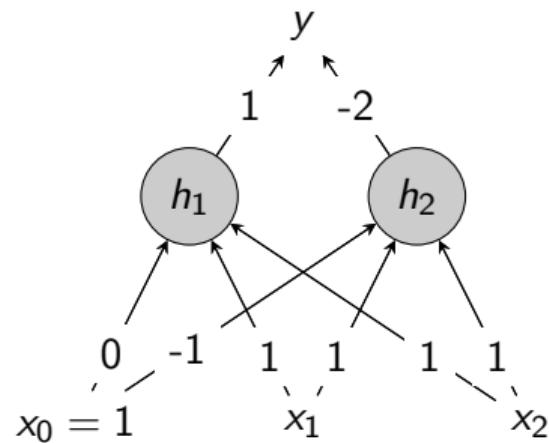
x_1	x_2	y	h_1	h_2	
0	0	0	0	-1	$\rightarrow 0$
0	1	1	1	0	
1	0	1	1	0	
1	1	0	2	1	



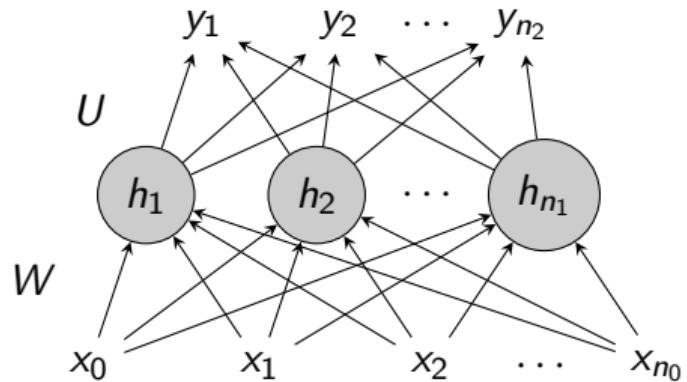
XOR problem

- why more than one layer networks?
- XOR = exclusive OR
- cannot be calculated by only one perceptron
- which activation function? ReLU

x_1	x_2	y	h_1	h_2	
0	0	0	0	-1	$\rightarrow 0$
0	1	1	1	0	
1	0	1	1	0	
1	1	0	2	1	

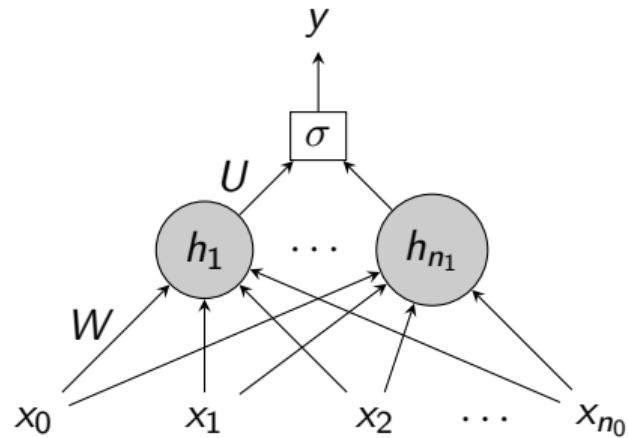
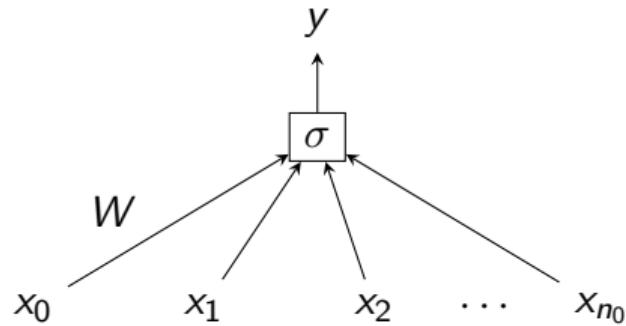


Feedforward (FF) neural networks



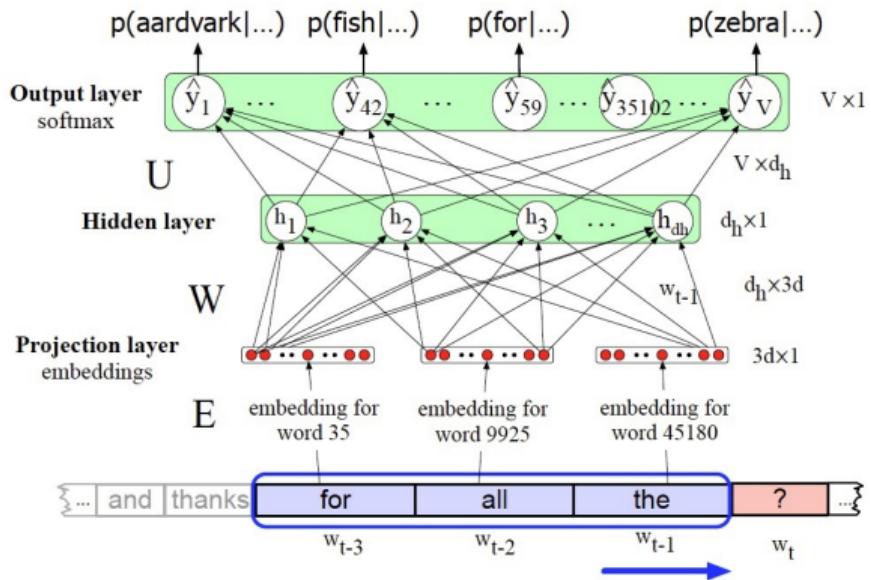
- for historic reasons also known as multi-layer perceptrons (MLP)
- diy: write CBOW, skip-gram, PV-DM and PV-DBOW as FF NN

Logistic regression as FF NN



- logistic regression is a NN without hidden layer (left figure)
- adding a hidden layer (right figure) maybe improve performance
- hidden layer allows for non-linear interactions between features
- power of deep learning, e.g., NN, is to learn feature interaction
- no need of complicated human-engineered features

Neural language model (NLM)



<https://web.stanford.edu/~jurafsky/slp3/>

- predict next word using a sliding window (fixed length)

NLM vs. n-gram LM

I have to make sure that the cat gets fed.
(not seen: dog gets fed)

- task: I forgot to make sure that the dog gets ...
- n-gram LM cannot predict "fed" (not seen)
- NLM finds similarity of embeddings of "cat" and "dog"
- NLM is able to generalize and predict "fed" after "dog" (not seen)

Training NNs

- for all training pairs (x, y)
 - forward step to find \hat{y}
 - compute loss $L(\hat{y}, y)$
 - update/optimize weights of output layer
 - assess the influence of (all) hidden nodes and update weights

Training NNs

- for all training pairs (x, y)
 - forward step to find \hat{y}
 - compute loss $L(\hat{y}, y)$
 - update/optimize weights of output layer
 - assess the influence of (all) hidden nodes and update weights
this is the “critical” part
- (let us have a short excursion to backpropagation; for more details, cf., deep learning lecture)

6

Embedding Methods

- Why Do We Need Embedding Methods?
- word2vec
- doc2vec
- Global Vectors for Word Representation (GloVe)
- fastText
- Lex2Sent
- Applications and Characteristics of Static Word Embeddings

Motivation

Olaf Scholz talks to the media in Berlin.
The chancellor addresses the journalists in the capital.

- no overlap of words - except stopwords
- (calculated) sentence similarity near 0
- meanings are very similar
- aim: build a model which ensures

Scholz ~ chancellor
talk ~ address
media ~ journalist
Berlin ~ capital

Idea

- find vector representation
- of lower dimensionality than vocabulary
- define words by their surrounding words
- short and dense instead of long and sparse vectors
- better ability to generalize (to similar but unseen words)
 - e.g., sentiment analysis
 - no need of exact same word in training and test
 - feature: (discrete) word identity → vector with (continuous) values
- easier to use in machine learning approaches

Synonyms and polysemy

- synonyms
 - clustered together in LSA, LDA into topics
 - get similar vectors for similar words
- polysemy
 - e.g., mouse (computer/animal)
 - treated as same word in topic models
 - also not covered by (global) embeddings
 - but can be modeled using contextual embeddings, cf. transformer

word2vec

- not counting co-occurrences
- prediction task: likeliness of co-occurrence
- fit a NN only as helper task → throw away the model itself
- take weights as the word embedding
- no need for human labels
- self-supervision: actually co-occurring words as gold standard

Skip-gram with negative sampling (SGNS)

approach:

- treat pairs of target word w and a neighboring context word c as positive examples.
- get negative examples by randomly sample other words from the vocabulary
- train logistic regression to distinguish between positive and negative
- learned weights = embeddings

SGNS training data

... lemon, a [tablespoon of **apricot** jam, a] pinch

- here: +/- 2 word window
- here: target word w : apricot
- here: $L = 4$ (positive) context words $c_i^{\text{pos}}, i = 1, 2, 3, 4$
- for each positive example k negative examples
- train classifier with positive and negative pairs
- result: embeddings (of size d) for each word in vocabulary

w	c^{pos}	$c^{\text{neg},1}$	$c^{\text{neg},2}$...	$c^{\text{neg},k}$
apricot	tablespoon	aardvark	seven	...	book
apricot	of	my	forever	...	example
apricot	jam	where	dear	...	on
apricot	a	fish	if	...	while

Probability estimation $P(+ \mid w, c)$

- sigmoid function (for one context word)

$$P(+ \mid w, c) = \sigma(cw) = \frac{1}{1 + \exp(-cw)}$$

$$\begin{aligned} P(- \mid w, c) &= 1 - P(+ \mid w, c) \\ &= \sigma(-cw) = \frac{1}{1 + \exp(cw)} \end{aligned}$$

- assume independence (to take lots of context words into consideration)

$$P(+ \mid w, c_1, \dots, c_L) = \prod_{i=1}^L \sigma(c_i w)$$

$$\log P(+ \mid w, c_1, \dots, c_L) = \sum_{i=1}^L \log \sigma(c_i w)$$

Skip-gram classifier idea

- estimate probability of w in context window
- by calculating similarities of embedding of w and context words

$$W = (W_{\text{in}}) = \begin{bmatrix} \text{aardvark} & & 1 \\ \text{apricot} & (w_{\text{in},i}^{\text{apricot}})_{i=1,\dots,d} & 2 \\ \dots & & \dots \\ \text{zebra} & & V \end{bmatrix}$$
$$C = (W_{\text{out}}) = \begin{bmatrix} \text{aardvark} & & 1 \\ \text{apricot} & (w_{\text{out},i}^{\text{apricot}})_{i=1,\dots,d} & 2 \\ \dots & & \dots \\ \text{zebra} & & V \end{bmatrix}$$

How to learn vectors

- initialize V d -dimensional embedding vectors randomly
- maximize similarity between target word and (positive) context words
- minimize similarity between target word and sampled (negative) words

$$\begin{aligned} L_{SGNS} &= -\log \left[P(+ \mid w, c^{\text{pos}}) \prod_{i=1}^k P(- \mid w, c^{\text{neg},i}) \right] \\ &= - \left[\log P(+ \mid w, c^{\text{pos}}) + \sum_{i=1}^k \log(1 - P(+ \mid w, c^{\text{neg},i})) \right] \\ &= - \left[\log \sigma(c^{\text{pos}} w) + \sum_{i=1}^k \log \sigma(-c^{\text{neg},i} w) \right] \end{aligned}$$

- stochastic gradient descent (learning rate: η)
- adjust word weights to make positive/negative pairs more/less likely

Learning step

- derivatives

$$\frac{\partial L_{SGNS}}{\partial c^{\text{pos}}} = (\sigma(c^{\text{pos}} w) - 1)w$$

$$\frac{\partial L_{SGNS}}{\partial c^{\text{neg},i}} = \sigma(c^{\text{neg},i} w)w$$

$$\frac{\partial L_{SGNS}}{\partial w} = (\sigma(c^{\text{pos}} w) - 1)c^{\text{pos}} + \sum_{i=1}^k \sigma(c^{\text{neg},i} w)c^{\text{neg},i}$$

- update steps

$$c_{t+1}^{\text{pos}} = c_t^{\text{pos}} - \eta(\sigma(c_t^{\text{pos}} w_t) - 1)w_t$$

$$c_{t+1}^{\text{neg},i} = c_t^{\text{neg},i} - \eta\sigma(c_t^{\text{neg},i} w_t)w_t$$

$$w_{t+1} = w_t - \eta \left[(\sigma(c^{\text{pos}} w) - 1)c^{\text{pos}} + \sum_{i=1}^k \sigma(c^{\text{neg},i} w)c^{\text{neg},i} \right]$$

Skip-gram as matrix factorization

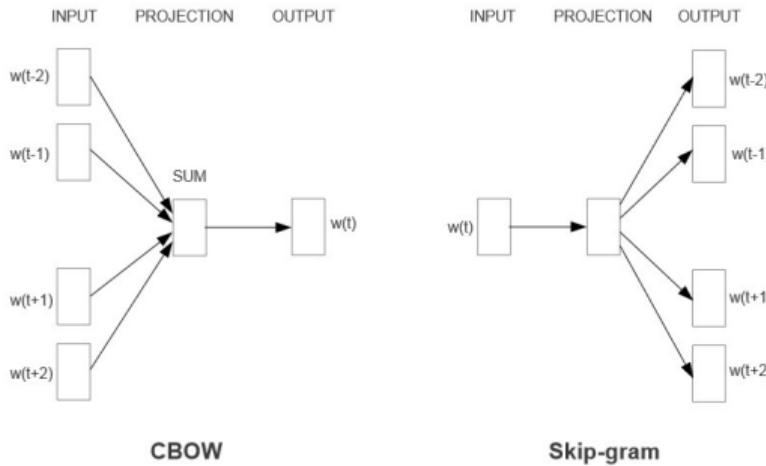
- skip-gram can be interpreted as matrix factorization of

$$W_{\text{in}} W_{\text{out}}^T = WC^T \approx X$$

- where X is the matrix of $\text{PMI}(x, y)$
- word2vec is (implicit) matrix factorization of $\text{PMI}^{0.75} - \log k$
- word2vec output are matrices
 - W target embeddings
 - C context embeddings
- common to use $w^{\text{apricot}} + c^{\text{apricot}}$ as word embedding for apricot

Skip-gram and CBOW as NNs

- idea: predict context from central word \rightarrow one hidden layer NN
- alternative: CBOW predicts central word from context \rightarrow one hidden layer NN



Efficient estimation of word representations in vector space (Mikolov et al., 2013)
<https://doi.org/10.48550/arXiv.1301.3781>

word2vec architecture & training algorithm

architecture

- skip-gram:
 - predicting context words
 - rare words are well represented
 - +/- 10 context window size
- CBOW:
 - predicting central word
 - faster
 - +/- 5 context window size

training algorithm

- negative sampling:
 - sampling of negative examples
 - needs multiple iterations for small amounts of training data
 - rare words are well represented
- hierarchical softmax:
 - Huffman tree
 - better representations for infrequent words
- it is possible to combine negative sampling and hierarchical softmax

see <https://code.google.com/archive/p/word2vec/> and <https://groups.google.com/g/word2vec-toolkit/c/WUWad9fL0jU/m/LdbWy1jQjUIJ>

Context window size

- small windows (+/- 2): syntactically similar words in same taxonomy
- e.g., Hogwarts nearest neighbors are other fictional schools:
 - Sunnydale
 - Evernight
 - Blandings
- large windows (+/- 5): related words in same semantic field
- e.g., Hogwarts nearest neighbors are words from Harry Potter world:
 - Dumbledore
 - half-blood
 - Malfoy
- authors recommend +/- 5 for CBOW and +/- 10 for skip-gram

see <https://code.google.com/archive/p/word2vec/>

How to "doc2vec"?

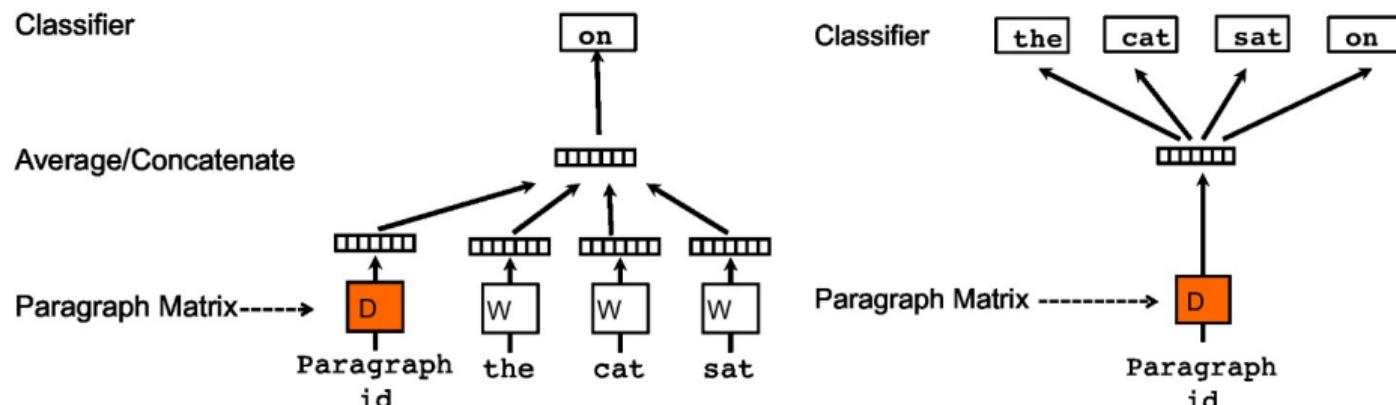
- naive approach?

How to "doc2vec"?

- naive approach? average word embeddings → does not work well

How to "doc2vec"?

- naive approach? average word embeddings → does not work well
- distributed memory/bag of words version of paragraph vector (PV-DM → left figure / PV-DBOW → right figure)
- find vector representation for paragraph/document



Distributed representations of sentences and documents (Le & Mikolov, 2014)
<https://doi.org/10.48550/arXiv.1405.4053>

Global vectors for word representation (GloVe)

- not
 - extracting embeddings from neural network
 - fake task
 - predicting context words (SGNS)
 - predicting target word (CBOW)
- but
 - optimize embeddings directly
 - co-occurrence matrix
 - $w_1 \cdot w_2 = \log \# \text{co-occurrence}(w_1, w_2)$
- <https://nlp.stanford.edu/projects/glove/>

GloVe: Global Vectors for Word Representation (Pennington et al., 2014)
<https://nlp.stanford.edu/pubs/glove.pdf>

fastText

- split words into n-grams
- train embeddings for n-grams and complete words
- sum n-gram and complete word embeddings → target embedding
- predict context words (not n-grams!) using target embedding (SGNS)

- understands suffixes and prefixes
- understands composed words
- works (often) also with unseen words
- <https://fasttext.cc/>

Enriching word vectors with subword information (Bojanowski et al., 2016)

<https://doi.org/10.48550/arXiv.1607.04606>

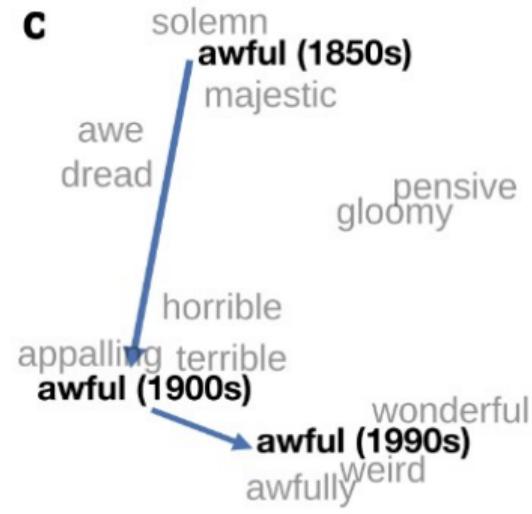
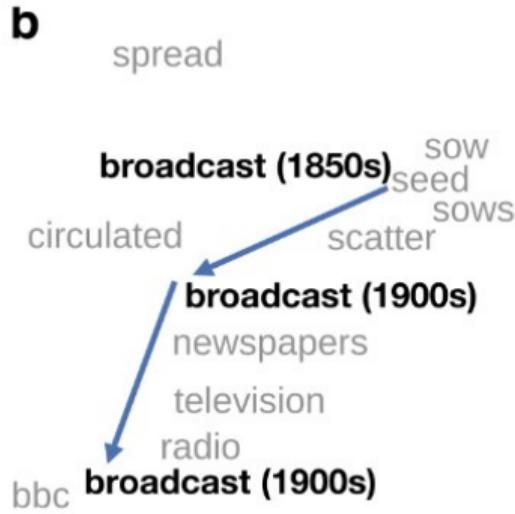
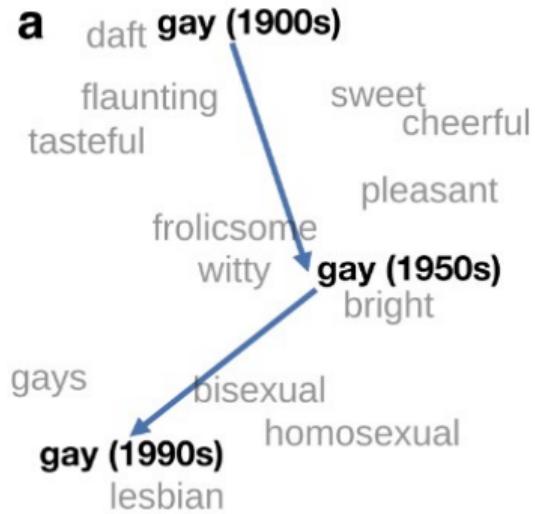
Lex2Sent

- unsupervised sentiment analysis/prediction (no need of labeled data)
- static embeddings, e.g., Doc2Vec, and sentiment dictionaries, e.g., WKWSCI
- split dictionary in two halves: positive, negative
- not bad -> negbad -> added as positive word
- $\text{diff}_d = \text{neg}_d - \text{pos}_d$ for each document d , where, e.g., neg_d is the cosine similarity of the document embedding to the negative lexicon half
- classification: $\text{diff}_d < 0$ -> text is predicted to be positive
- average diff_d over multiple resamplings of the same document to get rid of noisy/unreliable predictions
- turns out to better, more robust, than common dictionary approaches

Lex2Sent: A bagging approach to unsupervised sentiment analysis (Lange et al., 2024)

<https://aclanthology.org/2024.konvens-main.28/>

Diachronic embeddings



Diachronic word embeddings reveal statistical laws of semantic change (Hamilton et al., 2016)

<http://dx.doi.org/10.18653/v1/P16-1141>

Biases

- king relates to man as queen relates to woman ($\text{king} - \text{man} + \text{woman} = \text{queen}$)
- france – paris + tokyo =

Biases

- king relates to man as queen relates to woman ($\text{king} - \text{man} + \text{woman} = \text{queen}$)
- france – paris + tokyo = japan

Biases

- king relates to man as queen relates to woman ($\text{king} - \text{man} + \text{woman} = \text{queen}$)
- france – paris + tokyo = japan
- doctor – father + mother =

Biases

- king relates to man as queen relates to woman ($\text{king} - \text{man} + \text{woman} = \text{queen}$)
- france – paris + tokyo = japan
- doctor – father + mother = nurse

Biases

- king relates to man as queen relates to woman ($\text{king} - \text{man} + \text{woman} = \text{queen}$)
- france – paris + tokyo = japan
- doctor – father + mother = nurse
- programmer – man + woman =

Biases

- king relates to man as queen relates to woman ($\text{king} - \text{man} + \text{woman} = \text{queen}$)
- france – paris + tokyo = japan
- doctor – father + mother = nurse
- programmer – man + woman = homemaker¹
- not only retrospectively biased, unfair and sexist
- if used in hiring searches for, e.g., programmers amplifies gender bias¹
- but can also be used to study cultural biases²

¹ Man is to computer programmer as woman is to homemaker? Debiasing word embeddings (Bolukbasi et al., 2016)
<https://papers.nips.cc/paper/2016/hash/a486cd07e4ac3d270571622f4f316ec5-Abstract.html>

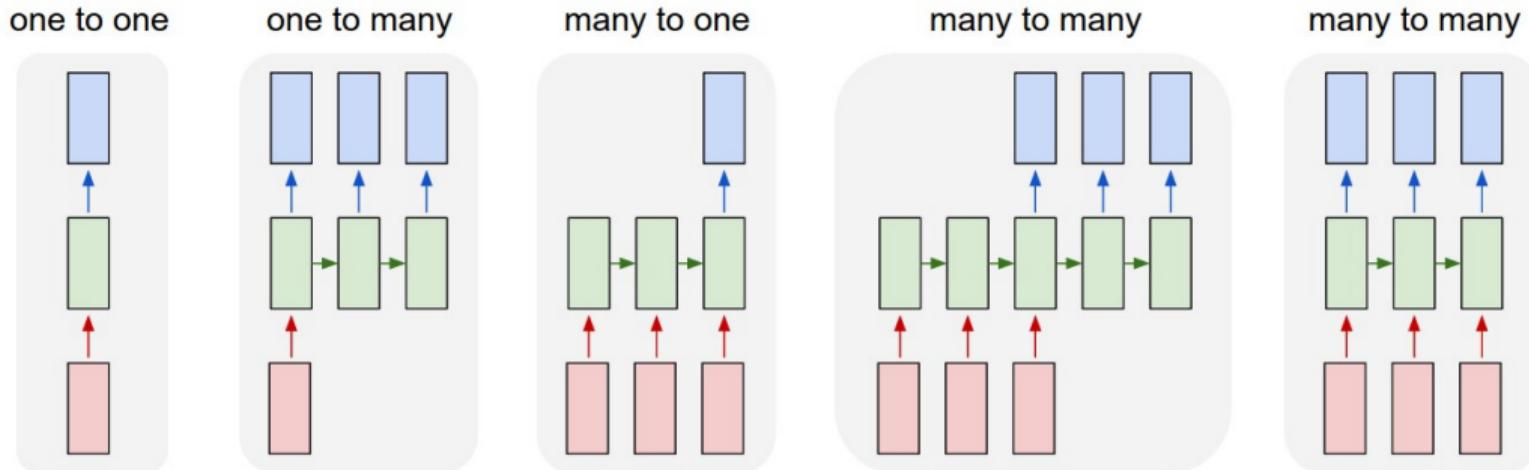
² Word embeddings quantify 100 years of gender and ethnic stereotypes (Garg et al., 2016)
<https://doi.org/10.1073/pnas.1720347115>

Language Models

- Sequence Modeling
 - Recurrent Neural Networks (RNN)
 - Long Short-Term Memory (LSTM)
 - Example: LSTMs in Machine Translation
 - Embeddings From Language Models (ELMo)
- Transformers
 - Elements of Transformers
 - Example: Step-By-Step Calculation
- Bidirectional Encoder Representations from Transformers (BERT)
 - Training BERT
 - Fine-Tuning BERT
- BERT-alike Models
 - Sentence-BERT
 - BERTopic
 - Decoding-enhanced BERT with Disentangled Attention (DeBERTa)
- Large Language Models

Sequence modeling

- e.g., sentiment classification for movie reviews
- word2vec still BOW
- sequence (order) (in most implementations) neglected

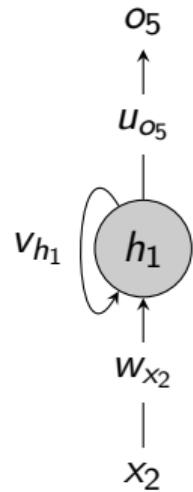


<http://karpathy.github.io/2015/05/21/rnn-effectiveness/>

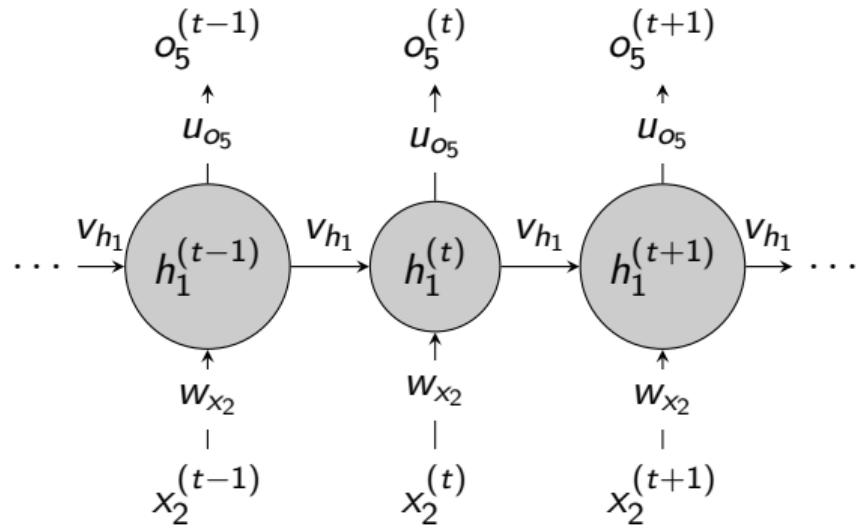
Recurrent neural network (RNN)

- idea: pass information from previous inputs into current calculation
- below: just a snippet of one stream of information, not a full NN!

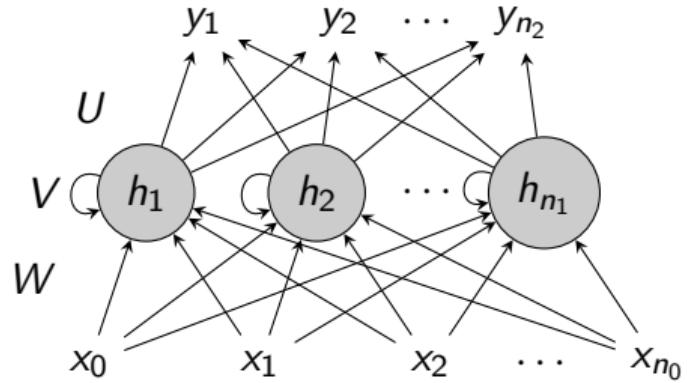
NN representation



unfolded representation

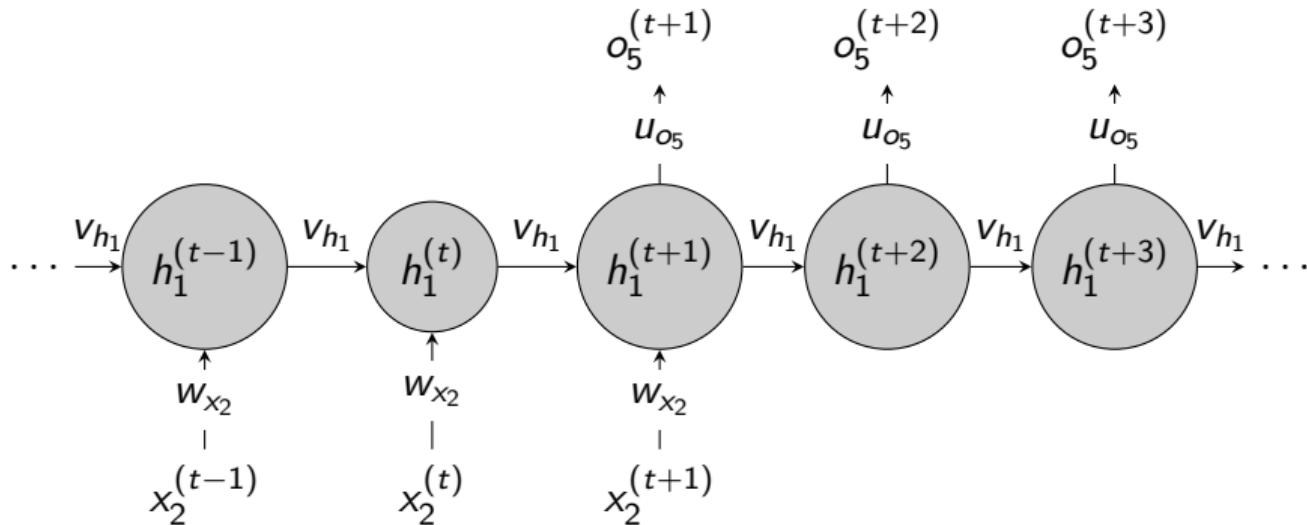


RNN in comparison to FF NN

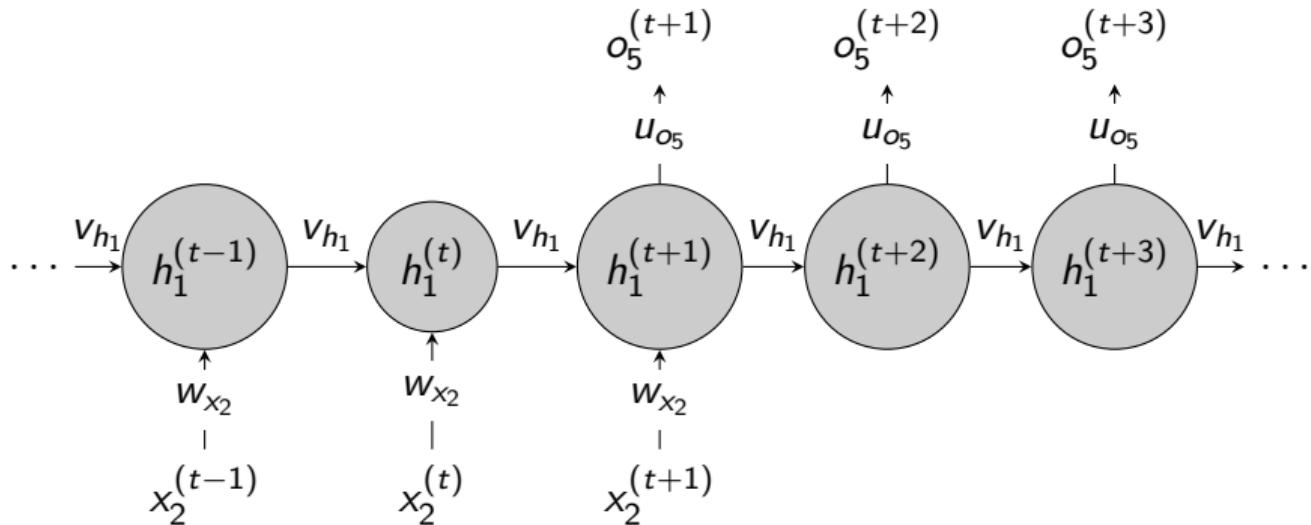


- traditional RNN structure; also known as
 - simple recurrent networks
 - Elman network
 - Jordan network
- more complex architectures possible

RNN architectures

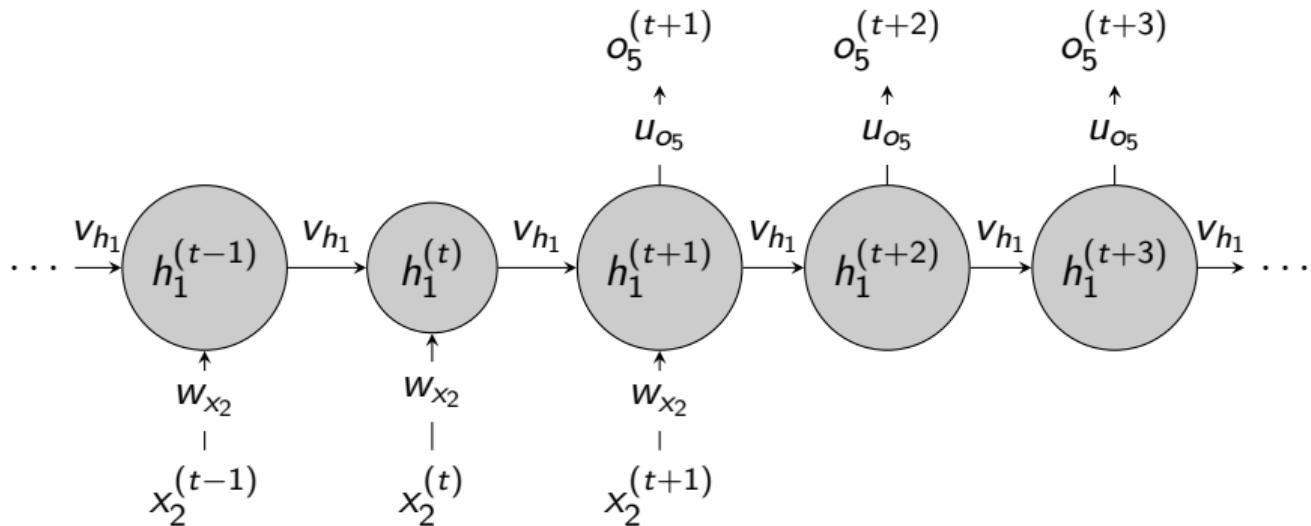


RNN architectures



- many to many, seq2seq
- advantage: short-term **memory**
- disadvantage:

RNN architectures

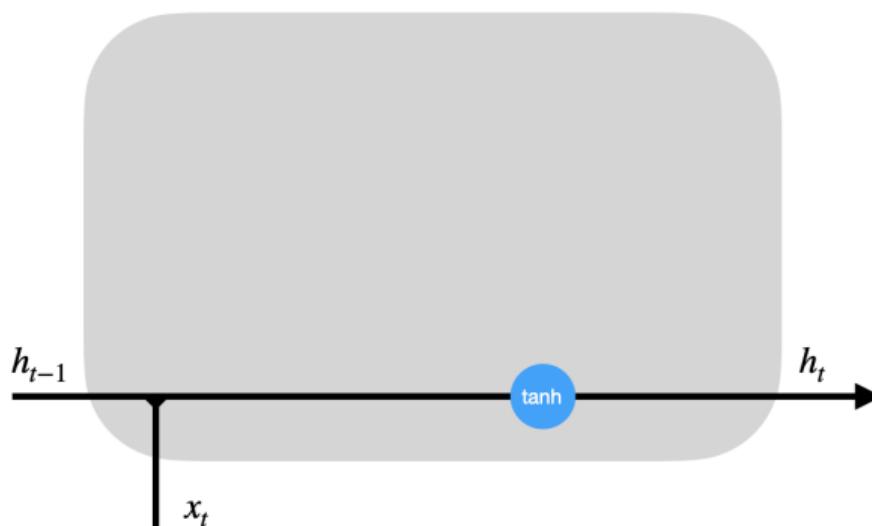


- many to many, seq2seq
- advantage: short-term **memory**
- disadvantage: **short-term** memory

One recurrent unit

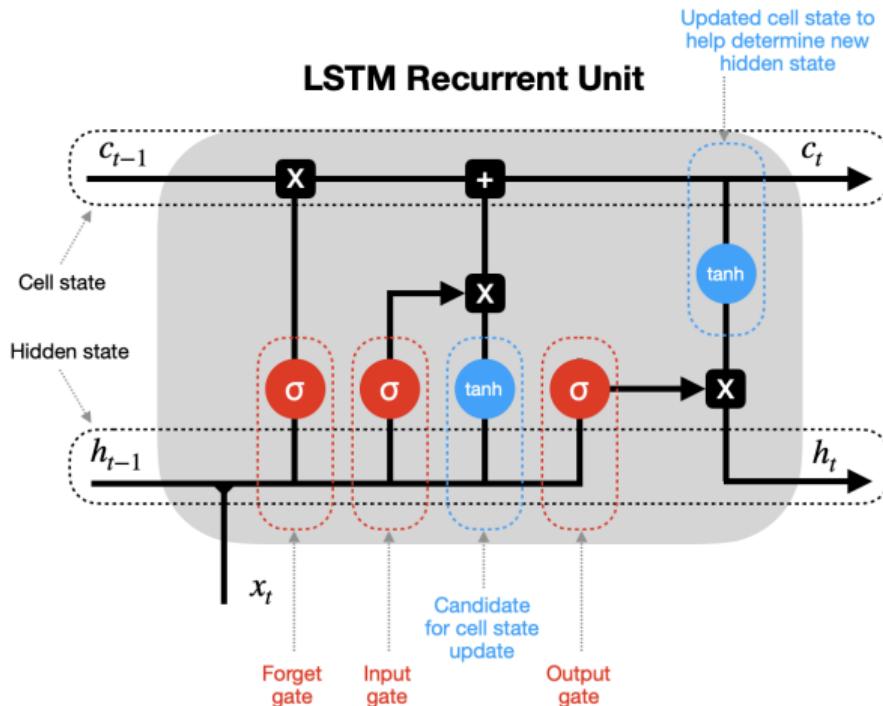
- simplify notation: $h_i^{(t)}, i = 1, \dots, n_1 \rightarrow h_t$
 - analogously we use x_t

Standard Recurrent Unit



<https://towardsdatascience.com/lstm-recurrent-neural-networks-how-to-teach-a-network-to-remember-the-past-55e54c2ff22e>

Long short-term memory (LSTM)



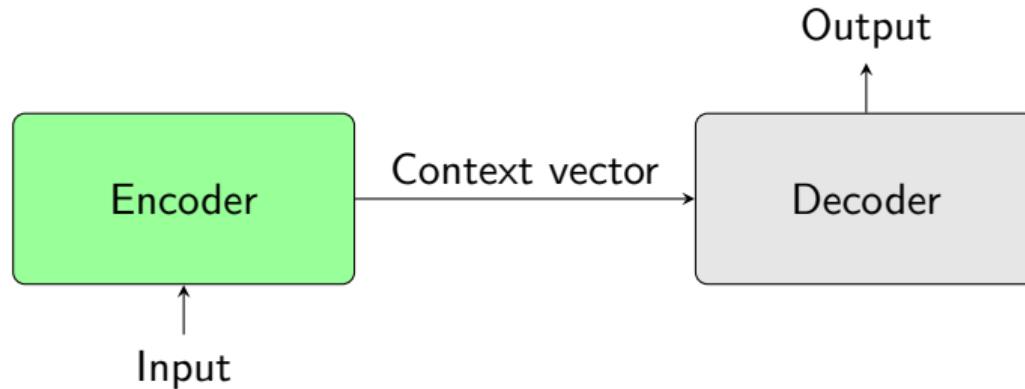
<https://towardsdatascience.com/lstm-recurrent-neural-networks-how-to-teach-a-network-to-remember-the-past-55e54c2ff22e>

LSTM in real world

- Siri
- Apple's auto-completion
- AlphaGo (software for the board game Go)
- GoogleTranslate was based on LSTMs, besides other systems
- no benefits in one to one scenarios
- (were) popular in seq2seq scenarios

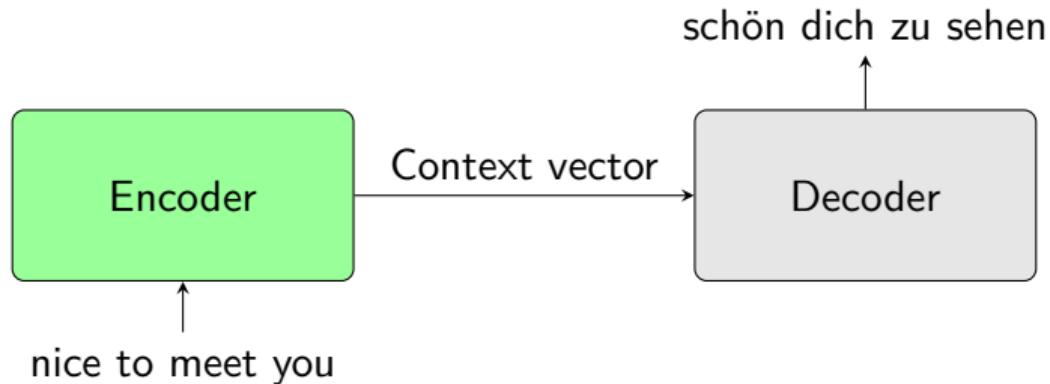
Encoder-decoder

- how to solve machine translation (MT) problem
- idea of encoder-decoder models:
 - ① encode on input sequence
 - ② decode to output sequence
- use, e.g., 2 LSTMs one after another
- concept used for (all) sota models: BERT, GPT-3, ...

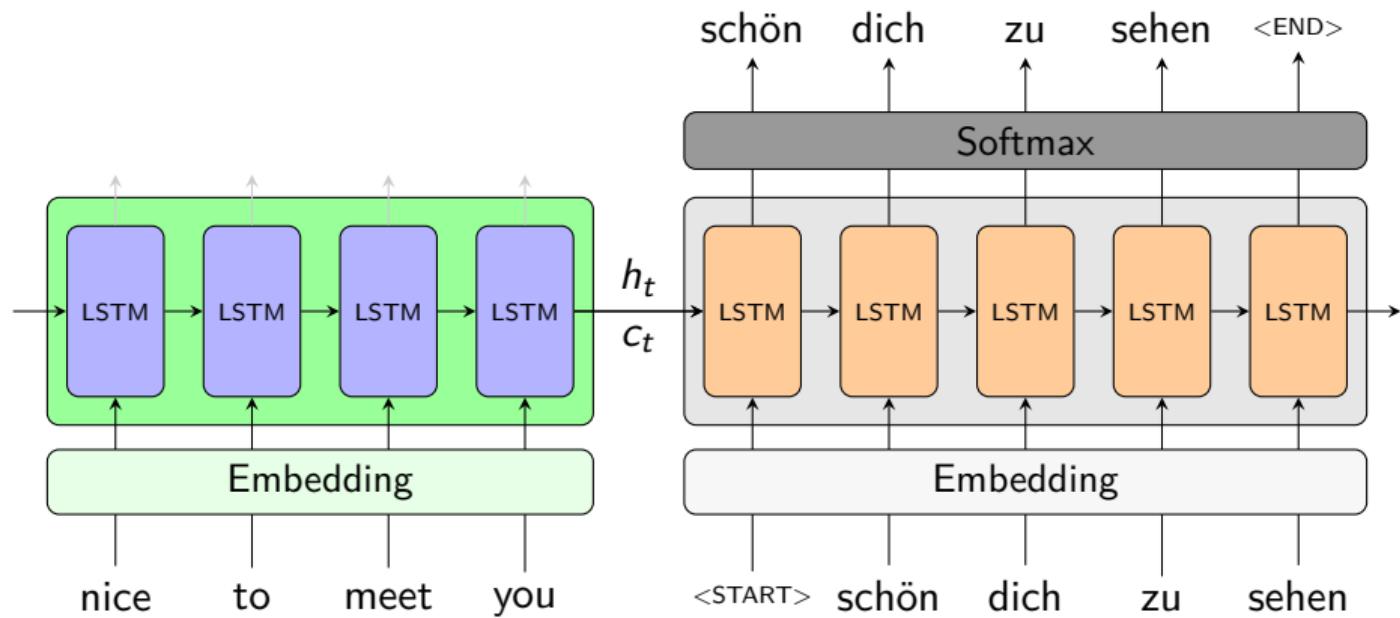


Encoder-decoder in MT

- input: nice to meet you
- task: translation to German
- (target) output: schön dich zu sehen
- encoder: collect information about input in context vector
- context vector: final state of encoder, encapsulates meaning of input
- decoder: predict output token by token using context vector



Example: encoder-decoder LSTM in MT

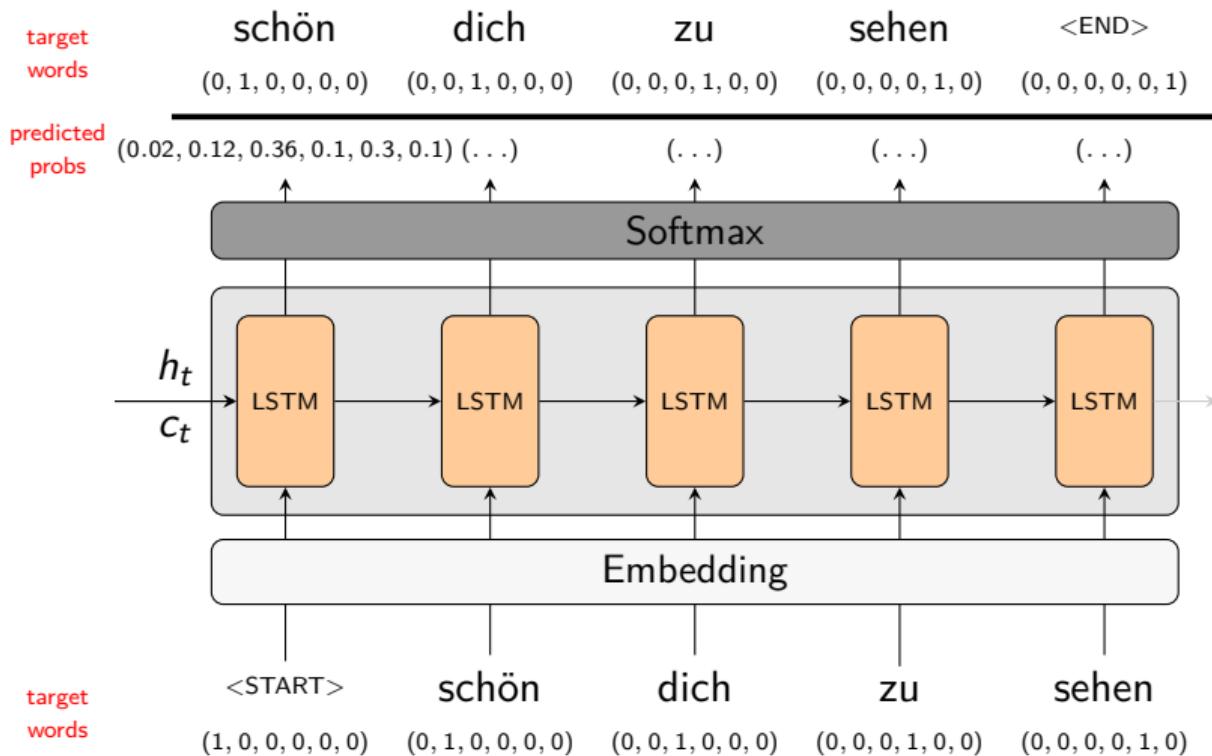


cf. <https://medium.com/analytics-vidhya/encoder-decoder-seq2seq-models-clearly-explained-c34186fbf49b>

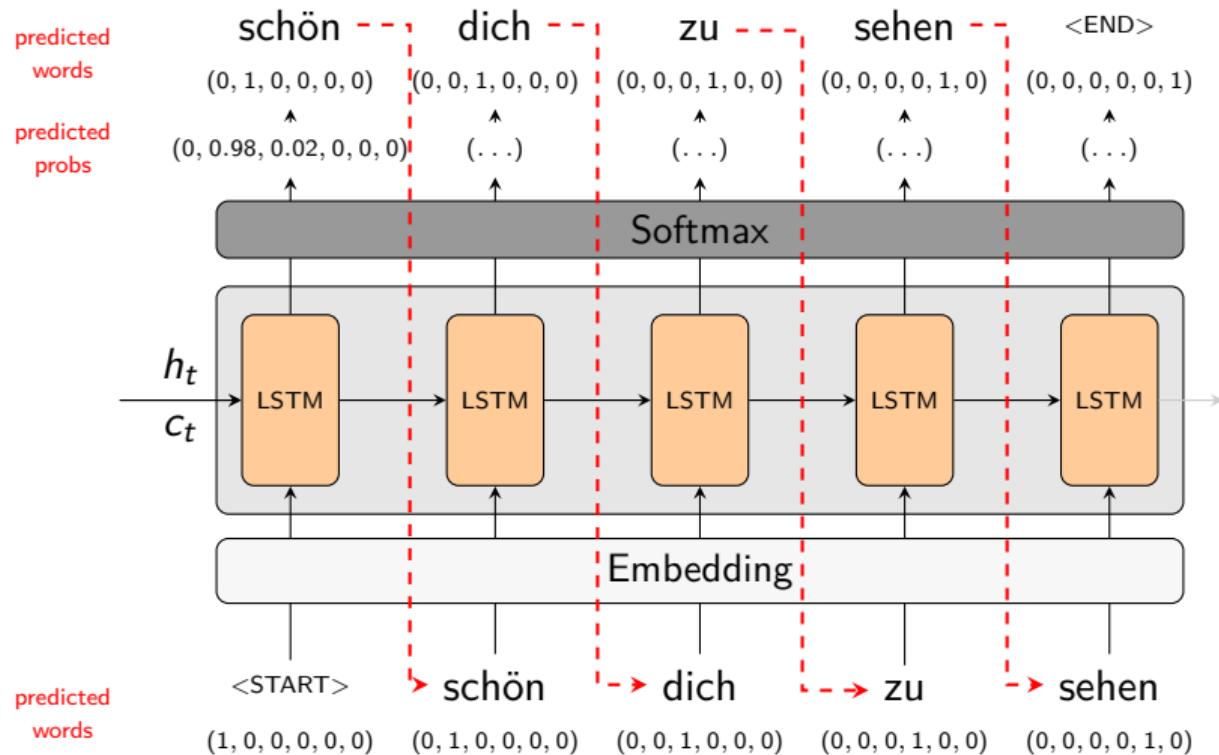
Training and testing phases

- vectorizing the tokens: one-hot-encoding → embeddings
- <START> and <END> also get one-hot-encoding & embeddings
- training & testing of encoder identical and straightforward
- instead, for decoder two different non-trivial approaches:
 - teacher forcing while training (cf. next slide):
use true labels as inputs, not the predicted output sequence
 - during the testing phase the predicted token are used as input for next token — as one-hot-encoded vector passed to the embedding layer

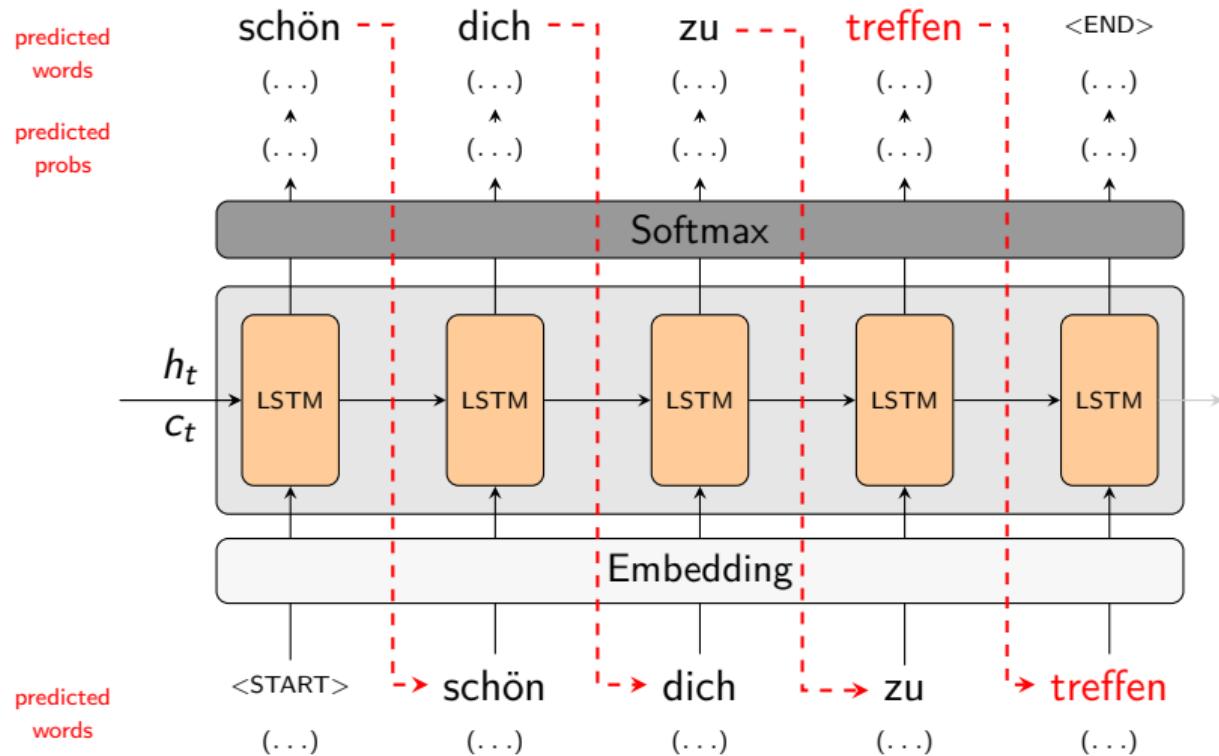
Decoder in the training phase



Decoder in the testing phase



Wrong predictions in testing phase



Embeddings from language models (ELMo)

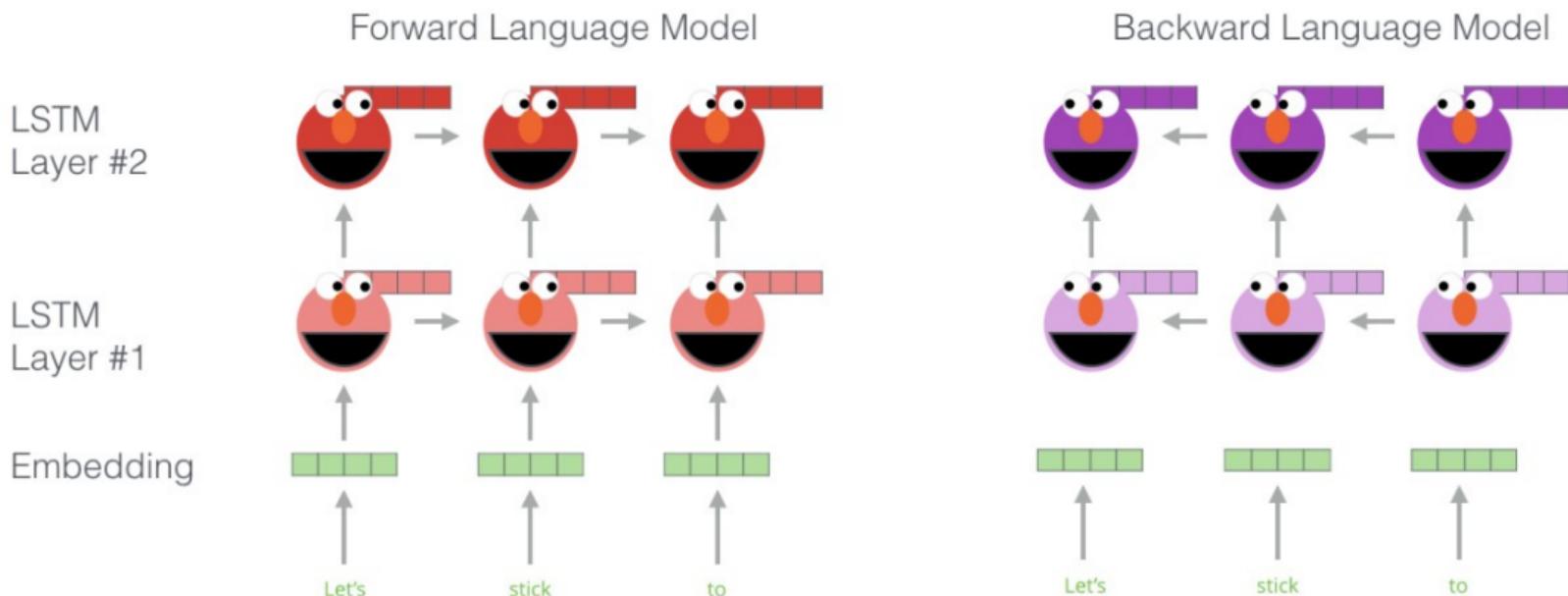
Let's stick to statistics.

The dog gets the stick.

- learn different embeddings for words in different contexts
- word2vec, fastText, GloVe would all result in the same embeddings for both appearances
- ELMo: based on LSTM layers
- idea: encode each word including its context
- first model using contextual embeddings
- bidirectional LSTM with LM objective (predict word)

ELMo — step 1

Embedding of “stick” in “Let’s stick to” - Step #1



<https://jalammar.github.io/illustrated-bert/>

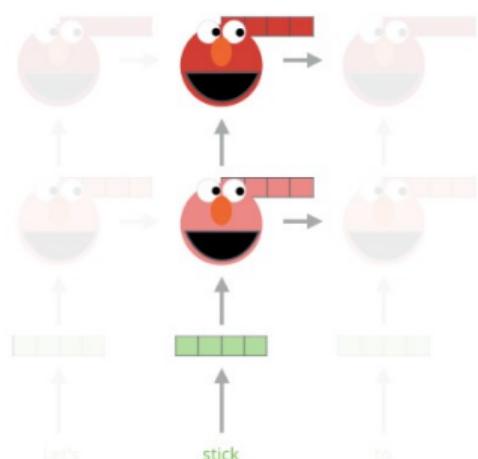
ELMo — step 2

Embedding of “stick” in “Let’s stick to” - Step #2

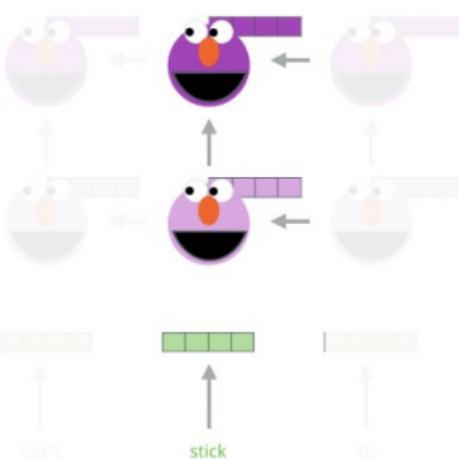
1- Concatenate hidden layers



Forward Language Model



Backward Language Model



2- Multiply each vector by a weight based on the task

$$\begin{array}{l} \text{---} \times s_2 \\ \text{---} \times s_1 \\ \text{---} \times s_0 \end{array}$$

3- Sum the (now weighted) vectors

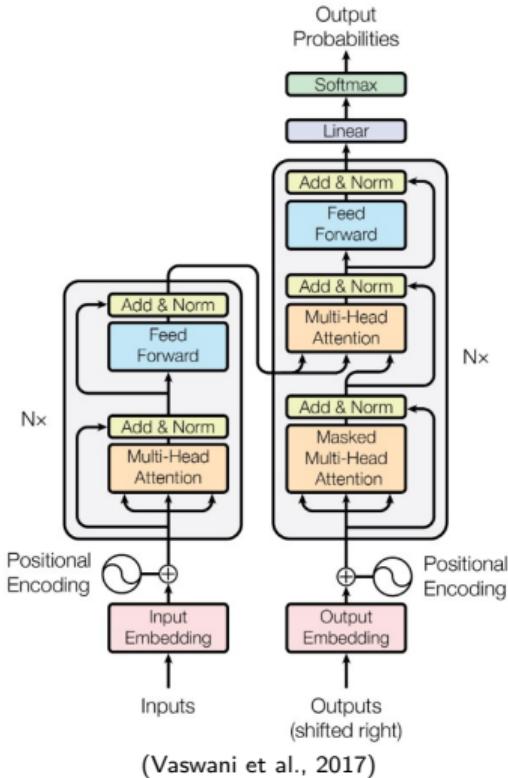


ELMo embedding of “stick” for this task in this context

<https://jalammar.github.io/illustrated-bert/>

Transformer idea

- RNN cannot be computed in parallel
- transformer enables parallel computation by using
 - attention
 - positional encoding
 - FF layers
- replacement of LSTMs
- better long-term memory



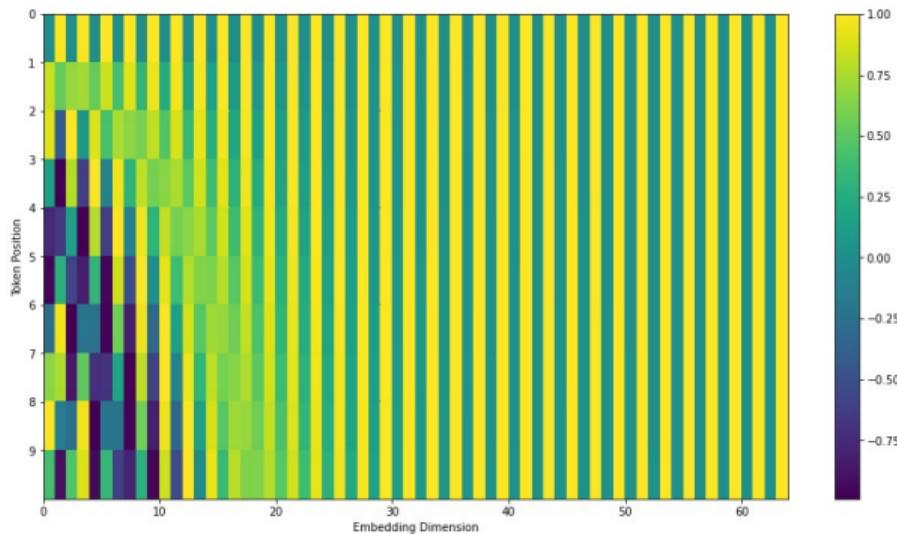
Positional encoding

$$\text{PE}(x \mid \text{pos}) = \begin{cases} \sin(\text{pos}/10000^{x/d}) & \text{if } x \text{ is even} \\ \cos(\text{pos}/10000^{(x-1)/d}) & \text{if } x \text{ is odd} \end{cases}$$

- enables parallel calculation
- is simply added to the word embedding
- pos is the token position $(0, \dots,)$
- x is the embedding dimension $(0, \dots, d)$

Positional encoding (visualization)

$$\text{PE}(x | \text{pos}) = \begin{cases} \sin(\text{pos}/10000^{x/d}) & \text{if } x \text{ is even} \\ \cos(\text{pos}/10000^{(x-1)/d}) & \text{if } x \text{ is odd} \end{cases}$$



<https://jalammar.github.io/illustrated-transformer/> — here $d = 64$

Attention is all you need¹

Self-attention

The trophy does not fit into the suitcase because it is too big/small. (cf. slide 29)

- question: what does "it" refers to in the sentence above
- human: if big → trophy, if small → suitcase
- not that easy for LM
- self-attention allows to look at other tokens' embeddings
- similar approach as in RNNs

¹Attention Is All You Need (Vaswani et al., 2017) <https://doi.org/10.48550/arXiv.1706.03762>

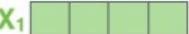
Self-attention elements

Input

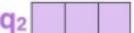
Thinking

Machines

Embedding

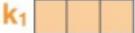
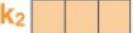
 x_1  x_2 

Queries

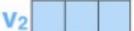
 q_1  q_2  WQ

- $x_i \in \mathbb{R}^{512}$
- $q_i, k_i, v_i \in \mathbb{R}^{64}$
- $W^Q \in \mathbb{R}^{512 \times 64}$
(W^K, W^V same)
- $V \neq \text{vocab size}$

Keys

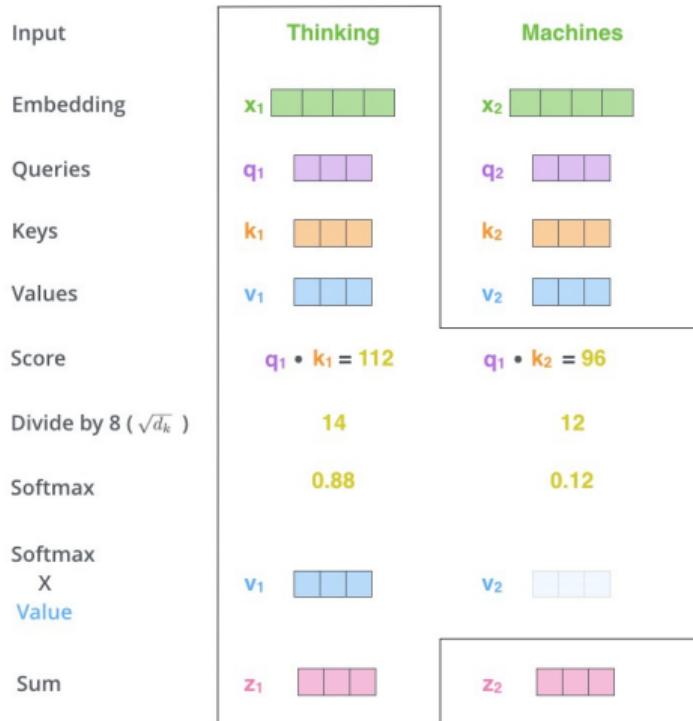
 k_1  k_2  WK

Values

 v_1  v_2  WV

<https://jalammar.github.io/illustrated-transformer/>

Self-attention calculation



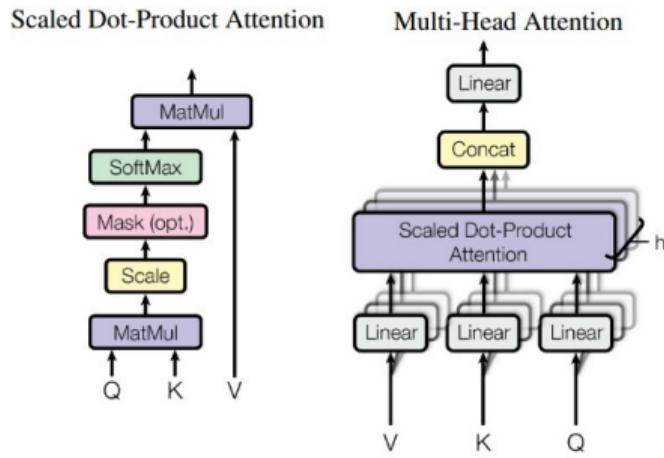
- ① get q_1, k_1, v_1 by multiplying x_1 with W^Q, W^K, W^V
- ② dot-product q_1 and k_i for all i in the same sequence
- ③ divide by 8
- ④ calculate softmax
- ⑤ weighted sum of v_i
- ⑥ z_1 resulting attention vector

<https://jalammar.github.io/illustrated-transformer/>

Self-attention matrix representation

- $Q = XW^Q$
- $K = XW^K$
- $V = XW^V$
- scaled dot-product attention
- $Z = \text{softmax}(QK^T / \sqrt{64})V$
- 64 because it is the respective dimension d_k
⇒ to prevent from extremely small gradients for the softmax function
- in general:
 - $d \times d_k$ dimension of W^Q and W^K
 - $d \times d_v$ dimension of W^V
 - $d_k = d_v = d/h = 64$ often selected ($d = 512$)
 - $h = 8$ number of (multi-head) attention layers (cf. following slides)

(Simple) self-attention and multi-head attention (MHA)



(Vaswani et al., 2017)

- idea: several (typically $h = 8$) parallel self-attention layers
- each layer returns an attention matrix $Z_i, i = 1, \dots, h$ (cf. previous slide)
- $Z = (Z_1, \dots, Z_h)W^0$
- $W^0 \in \mathbb{R}^{hd_v \times d} (= \mathbb{R}^{512 \times 512})$

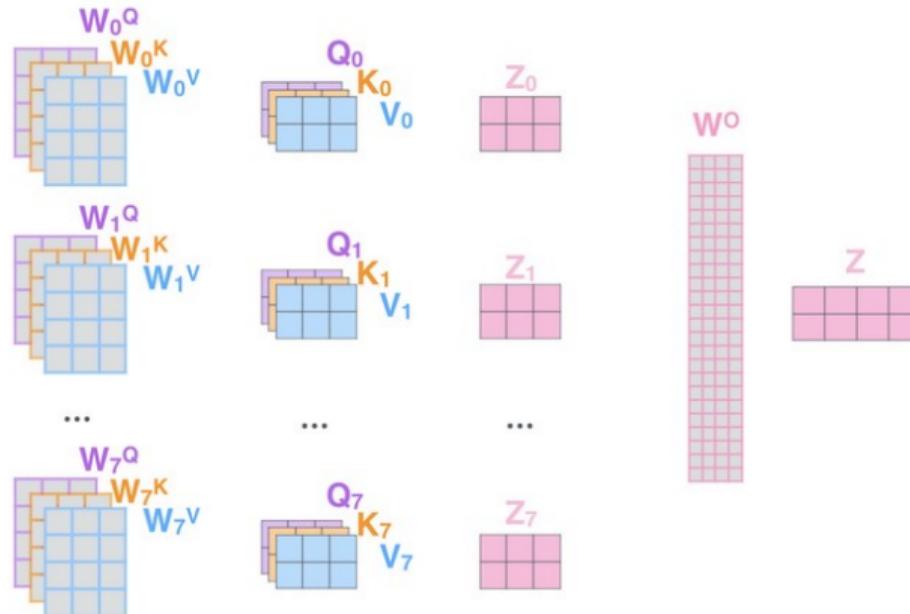
MHA in one figure

- 1) This is our input sentence* each word*
- 2) We embed
- 3) Split into 8 heads. We multiply X or R with weight matrices
- 4) Calculate attention using the resulting $Q/K/V$ matrices
- 5) Concatenate the resulting Z matrices, then multiply with weight matrix W^o to produce the output of the layer

Thinking
Machines



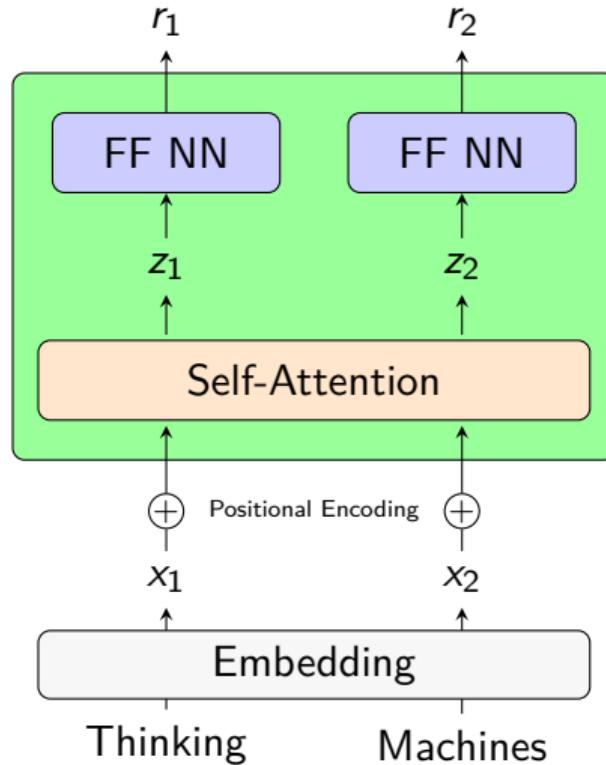
* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one



<https://jalammar.github.io/illustrated-transformer/>

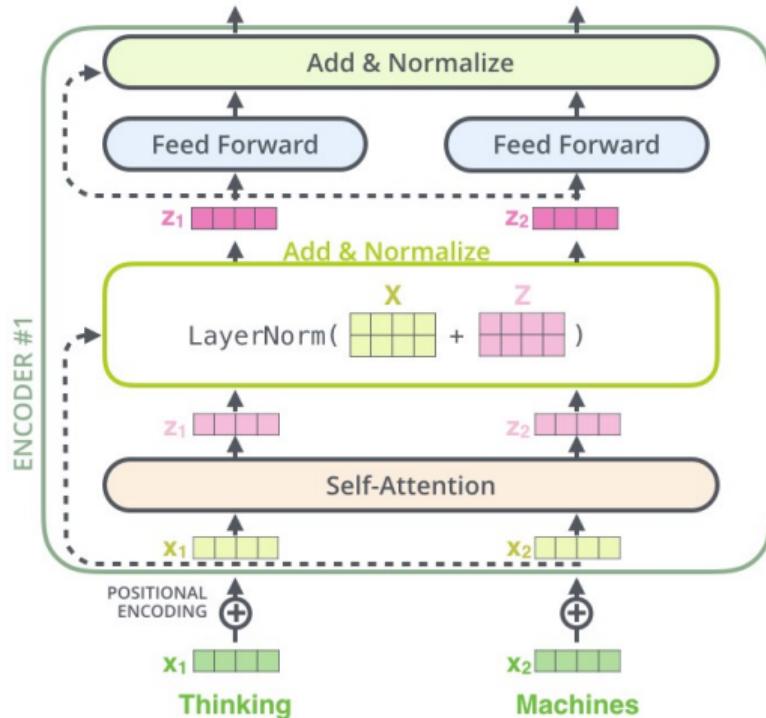
Now we're finally encoding

- on the right: first encoder layer
- all other layers get r vectors
- $\text{FFNN}(z) = \max(0, zW_1 + b_1)W_2 + b_2$
(two linear transformations + ReLU)
- typically:
 - $x_i \in \mathbb{R}^{512}$
 - $z_i \in \mathbb{R}^{512}$
 - $r_i \in \mathbb{R}^{512}$



Residuals and layer normalization

- skip/residual connection
- allow direct information flow
- (layer) normalization
- across the features
- to avoid covariate shift
- to avoid slow down training



<https://jalammar.github.io/illustrated-transformer/>

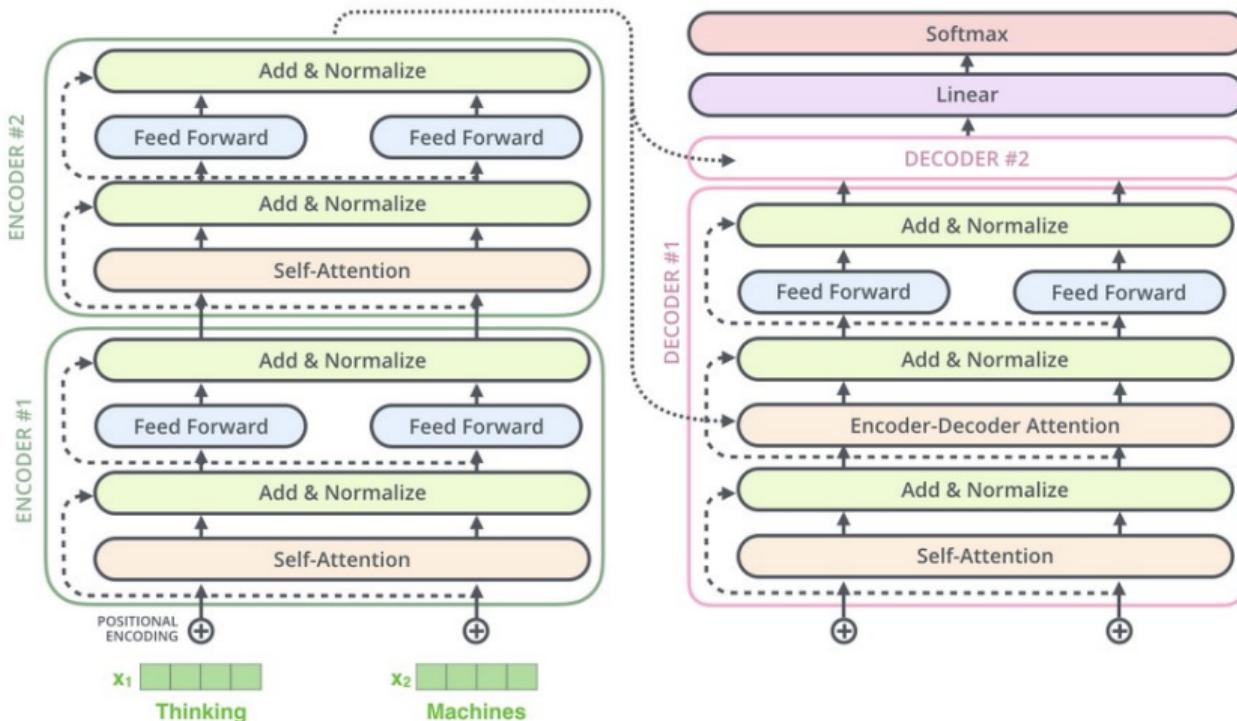
Decoder layers

- decoder is very similar to encoder layer
- masked MHA at the bottom
- padding mask: too long or too short sequences
 - pad 0 after shorter sequences
 - pad $\approx -\infty$ for pieces of longer sequences
- sequence mask:
 - set future positions to $\approx -\infty$
- input to decoder is shifted right
 - prevent from just learning the copy/paste task

Output layer

- after all decoder layers:
 - ① linear layer
 - ② softmax
- linear layer
 - fully connected NN (up-projection)
 - vocabulary size
 - results in logits
- softmax layer
 - transforms logits to pseudo-probabilities
 - then, arg max a typical prediction

A transformer architecture for $N = 2$



<https://jalammar.github.io/illustrated-transformer/>

cf. also <https://medium.com/@yulemoon/detailed-explanations-of-transformer-step-by-step-dc32d90b3a98>

Complexity comparison

Layer type	Complexity	Number of sequential operations
Attention	$O(n^2d)$	$O(1)$
RNN	$O(nd^2)$	$O(n)$
CNN	$O(knd^2)$	$O(1)$

- n is the sequence length
- d is the embedding size
- k is the kernel size
- often $n < d$
- more important: Attention better parallelizable due to less sequential operations

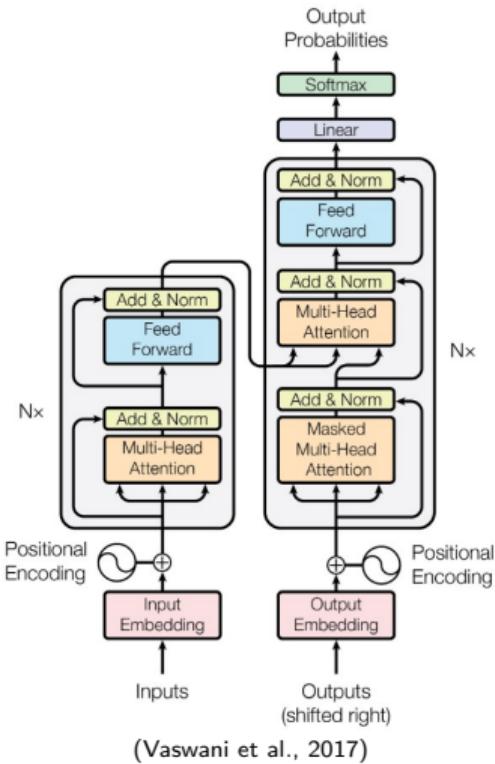
Step-by-step calculation of the transformer architecture¹ (encoder)

- ① data, vocabulary, vocabulary size
- ② encoding & static embedding
- ③ positional encoding
- ④ combining embedding and positional encoding
- ⑤ single-head attention (SHA) → multi-head attention (MHA)
 - ① calculating query, key, and value matrices
 - ② query and key matrix multiplication & scaling
 - ③ calculating softmax & multiplication with value matrix
 - ④ repeat, concatenate, linear transformation for final SHA/MHA output matrix
- ⑥ add & normalize
- ⑦ FFNN

¹inspired by <https://levelup.gitconnected.com/understanding-transformers-from-start-to-end-a-step-by-step-math-example-16d4e64e6eb1>

Step-by-step calculation of the transformer architecture (decoder)

- ⑧ masked MHA
- ⑨ decoder MHA
- ⑩ predicting tokens



Step 1 — data, vocabulary, vocabulary size

- assume a dataset containing (for simplicity: only) three sentences
 - ① I drink and I know things.
 - ② When you play the game of thrones, you win or you die.
 - ③ The true enemy won't wait out the storm, he brings the storm.
- tokenizing (for simplicity: without subword information) results in
 - ① I, drink, and, I, know, things
 - ② when, you, play, the, game, of, thrones, you, win, or, you, die
 - ③ the, true, enemy, won't, wait, out, the, storm, he, brings, the, storm
- the resulting set of vocabularies is given by

I, drink, and, know, things, when, you, play, the, game, of, thrones, win, or, die, true, enemy, won't, wait, out, storm, he, brings
- $V = 23$

Step 2 — encoding

encoding (just mixing up the vocabularies a little bit for demonstration):

1	2	3	4	5	6	7	8	9	10	11	12
I	drink	things	know	when	won't	play	out	true	storm	brings	game
13	14	15	16	17	18	19	20	21	22	23	
the	win	of	enemy	you	wait	thrones	and	or	die	he	

we select the second sentence as an example and input the first part into the encoder of the transformer, i.e. when you play game of thrones
5 17 7 12 15 19

Step 2 — static embedding

d	when	you	play	game	of	thrones
	5	17	7	12	15	19
1	0.79	0.38	0.01	0.12	0.88	0.60
2	0.60	-0.37	1.93	1.73	1.24	1.02
3	0.96	0.01	0.18	0.52	0.62	0.53
4	0.64	-0.21	0.31	-0.77	-0.36	0.51
5	0.97	0.90	0.56	0.06	0.49	0.93
6	0.20	-0.26	0.59	-0.63	-0.30	0.21

- the attention paper uses $d = 512$, we select $d = 6$ for demonstration
- the input embedding layer (somewhat a lookup table) is initialized randomly and updated during training
- the (current) static embedding for of is $(0.88, 1.25, 0.04, -0.03, 0.32, -0.28)$

Step 3 — positional encoding I

$$\text{PE}(x \mid \text{pos}) = \begin{cases} \sin(\text{pos}/10000^{x/d}) & \text{if } x \text{ is even} \\ \cos(\text{pos}/10000^{(x-1)/d}) & \text{if } x \text{ is odd} \end{cases}$$

- let us encode the position of the token of, which is the fifth token in our example sentence
- since programming languages usually start counting at 0, this refers to token position 4

pos	x	even/odd	formula	$\text{PE}(x \mid \text{pos})$
4	0	even	$\sin(4/10000^{0/6})$	-0.7568
4	1	odd	$\cos(4/10000^{0/6})$	-0.6536
4	2	even	$\sin(4/10000^{2/6})$	0.1846
4	3	odd	$\cos(4/10000^{2/6})$	0.9828
4	4	even	$\sin(4/10000^{4/6})$	0.0086
4	5	odd	$\cos(4/10000^{4/6})$	1.0000

- the positional encoding for of is given by the PE column

Step 3 — positional encoding II

- applying the same calculation to all input positions results in

d	when	you	play	game	of	thrones
	5	17	7	12	15	19
1	0.0000	0.8415	0.9093	0.1411	-0.7568	-0.9589
2	1.0000	0.5403	-0.4161	-0.9900	-0.6536	0.2837
3	0.0000	0.0464	0.0927	0.1388	0.1846	0.2300
4	1.0000	0.9989	0.9957	0.9903	0.9828	0.9732
5	0.0000	0.0022	0.0043	0.0065	0.0086	0.0108
6	1.0000	1.0000	1.0000	1.0000	1.0000	0.9999

Step 4 — combining embedding and positional encoding

d	when 5	you 17	play 7	game 12	of 15	thrones 19
1	0.79	0.38	0.01	0.12	0.88	0.60
2	0.60	-0.37	1.93	1.73	1.24	1.02
3	0.96	0.01	0.18	0.52	0.62	0.53
4	0.64	-0.21	0.31	-0.77	-0.36	0.51
5	0.97	0.90	0.56	0.06	0.49	0.93
6	0.20	-0.26	0.59	-0.63	-0.30	0.21

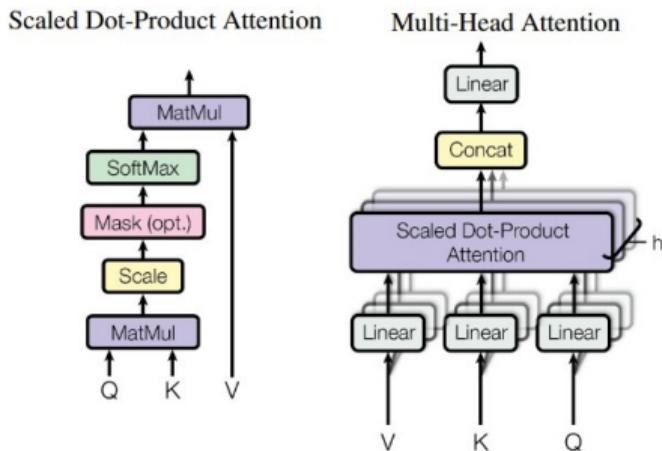
(input embedding) + (positional encoding)

d	when 5	you 17	play 7	game 12	of 15	thrones 19
1	0.0000	0.8415	0.9093	0.1411	-0.7568	-0.9589
2	1.0000	0.5403	-0.4161	-0.9900	-0.6536	0.2837
3	0.0000	0.0464	0.0927	0.1388	0.1846	0.2300
4	1.0000	0.9989	0.9957	0.9903	0.9828	0.9732
5	0.0000	0.0022	0.0043	0.0065	0.0086	0.0108
6	1.0000	1.0000	1.0000	1.0000	1.0000	0.9999

(input matrix for multi-head attention)

d	when 5	you 17	play 7	game 12	of 15	thrones 19
1	0.79	1.22	0.92	0.26	0.12	-0.36
2	1.60	0.17	1.51	0.74	0.59	1.30
3	0.96	0.06	0.27	0.66	0.80	0.76
4	1.64	0.79	1.31	0.22	0.62	1.48
5	0.97	0.90	0.56	0.07	0.50	0.94
6	1.20	0.74	1.59	0.37	0.70	1.21

Step 5 — single-head attention (SHA) → multi-head attention (MHA)



- ① calculating query, key, and value matrices
- ② query and key matrix multiplication & scaling
- ③ calculating softmax & multiplication with value matrix
- ④ repeat, concatenate & linear transformation for final SHA/MHA output matrix

Step 5.1 — calculating query, key, and value matrices

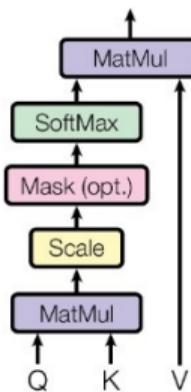
						(linear weights for query W^Q)		(query matrix Q)							
						0.52	0.45	0.91	0.69	3.88	3.80	4.08	3.42		
						0.05	0.85	0.37	0.83	2.55	1.86	2.77	1.78		
						0.49	0.10	0.56	0.61	=	3.39	3.60	3.49	2.72	
						0.71	0.64	0.40	0.14		1.02	1.18	1.24	1.30	
						0.76	0.27	0.92	0.67		1.90	1.56	1.88	1.53	
						0.85	0.56	0.57	0.07		3.04	2.90	2.73	2.22	
(input matrix for multi-head attention)						(linear weights for key W^K)		(key matrix K)							
when	0.79	1.60	0.96	1.64	0.97	1.20	0.74	0.57	0.21	0.73	3.71	4.04	4.15	3.41	
you	1.22	0.17	0.06	0.79	0.90	0.74	0.55	0.16	0.90	0.17	2.18	2.51	1.64	1.93	
play	0.92	1.51	0.27	1.31	0.56	1.59	0.25	0.74	0.80	0.98	=	3.28	3.11	3.65	3.01
game	0.26	0.74	0.66	0.22	0.07	0.37	0.80	0.73	0.20	0.31		1.07	1.13	1.64	1.35
of	0.12	0.59	0.80	0.62	0.50	0.70	0.37	0.96	0.42	0.08		1.49	1.97	2.14	1.81
thrones	-0.36	1.30	0.76	1.48	0.94	1.21	0.28	0.41	0.87	0.86		2.51	3.04	3.45	2.28
						(linear weights for value W^V)		(value matrix V)							
						0.62	0.07	0.70	0.95		3.63	4.58	4.21	4.76	
						0.20	0.97	0.61	0.35		2.22	1.83	2.17	3.25	
						0.57	0.80	0.61	0.50	=	3.12	3.77	3.41	4.33	
						0.67	0.35	0.98	0.54		1.09	1.65	1.32	1.38	
						0.47	0.83	0.34	0.94		1.72	2.34	1.80	2.21	
						0.60	0.69	0.13	0.98		2.63	3.98	2.93	3.36	

- we select $d_k = d_v = 4$ for simplicity

Step 5.2 — query and key matrix multiplication & scaling

(query matrix Q)				(transposed key matrix K^T)								(QK^T)			
3.88	3.80	4.08	3.42									58.34	31.29	49.73	19.75
2.55	1.86	2.77	1.78	\times	3.71	2.18	3.28	1.07	1.49	2.51	=	34.54	18.21	29.62	11.78
3.39	3.60	3.49	2.72		4.04	2.51	3.11	1.13	1.97	3.04		50.88	27.40	43.24	17.09
1.02	1.18	1.24	1.30		4.15	1.64	3.65	1.64	2.14	3.45		18.13	9.73	15.45	6.21
1.90	1.56	1.88	1.53		3.41	1.93	3.01	1.35	1.81	2.28		26.37	14.09	22.55	8.94
3.04	2.90	2.73	2.22									41.89	22.67	35.64	14.00
															20.10
															30.93

Scaled Dot-Product Attention



- calculating QK^T
- scaling with $\sqrt{d_k} = \sqrt{4} = 2$

29.17	15.64	24.86	9.88	14.10	21.58
17.27	9.11	14.81	5.89	8.30	12.84
25.44	13.70	21.62	8.54	12.27	18.85
9.06	4.87	7.72	3.10	4.42	6.70
13.19	7.04	11.28	4.47	6.35	9.74
20.95	11.34	17.82	7.00	10.05	15.46

Step 5.3 — calculating softmax

$$\text{softmax}(z_i) = \exp(z_i) / \sum_{j=1}^d \exp(z_j), \quad i = 1, \dots, d$$

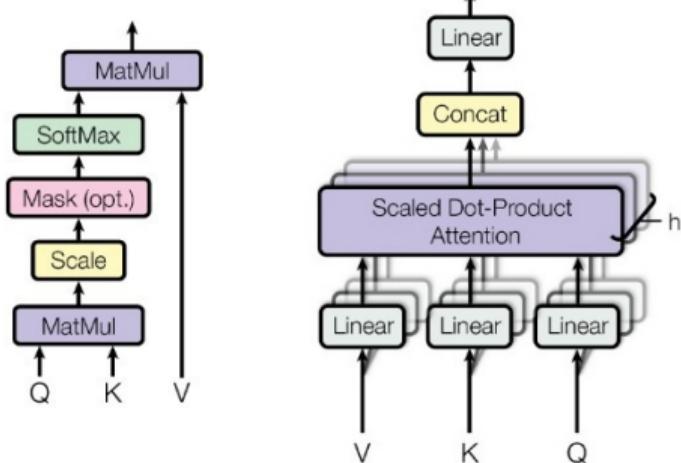
$$\begin{aligned}\text{softmax}(29.17) &= \exp(29.17) / (\exp(29.17) + \exp(15.64) + \exp(24.86) + \\ &\quad \exp(9.88) + \exp(14.10) + \exp(21.58)) = 0.9862\end{aligned}$$

$\text{softmax}(QK^T/2) =$	0.9862	0.0000	0.0133	0.0000	0.0000	0.0005	when
	0.9111	0.0003	0.0777	0.0000	0.0001	0.0108	you
	0.9772	0.0000	0.0214	0.0000	0.0000	0.0013	play
	0.7231	0.0108	0.1897	0.0019	0.0070	0.0676	game
	0.8450	0.0018	0.1251	0.0001	0.0009	0.0270	of
	0.9542	0.0001	0.0418	0.0000	0.0000	0.0040	thrones

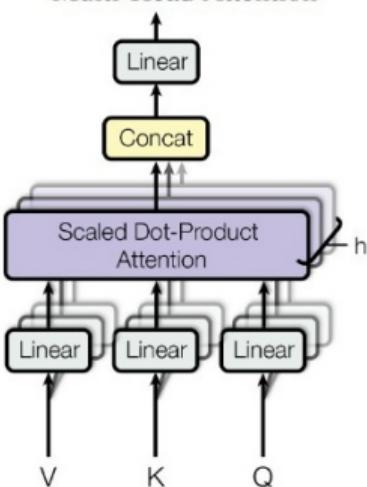
Step 5.3 — multiplication of softmax and value matrix

softmax($QK^T / 2$)						(value matrix V)				(Z)			
0.9862	0.0000	0.0133	0.0000	0.0000	0.0005	3.63	4.58	4.21	4.76	3.6227	4.5689	4.1987	4.7536
0.9111	0.0003	0.0777	0.0000	0.0001	0.0108	2.22	1.83	2.17	3.25	3.5790	4.5095	4.1332	4.7108
0.9772	0.0000	0.0214	0.0000	0.0000	0.0013	3.12	3.77	3.41	4.33	3.6174	4.5614	4.1908	4.7485
0.7231	0.0108	0.1897	0.0019	0.0070	0.0676	1.09	1.65	1.32	1.38	3.4326	4.3353	3.9277	4.5437
0.8450	0.0018	0.1251	0.0001	0.0009	0.0270	1.72	2.34	1.80	2.21	3.5343	4.4548	4.0688	4.6626
0.9542	0.0001	0.0418	0.0000	0.0000	0.0040	2.63	3.98	2.93	3.36	3.6049	4.5439	4.1717	4.7368

Scaled Dot-Product Attention



Multi-Head Attention



- calculating $Z = \text{softmax}(QK^T / \sqrt{d_k})V$
- this is the last step in the SHA setting
- in this example, we won't repeat this procedure h times to get MHA setting
 - if so: concatenate Z matrices

Step 5.4 — linear transformation for final SHA/MHA output matrix

$$\begin{array}{c}
 (Z) \\
 \begin{array}{cccc} 3.6227 & 4.5689 & 4.1987 & 4.7536 \end{array} \\
 \times \quad \begin{array}{ccccccccc} 0.80 & 0.34 & 0.45 & 0.54 & 0.07 & 0.53 & = & 12.33 & 10.87 & 8.07 & 8.71 & 7.12 & 9.16 \\ 0.85 & 0.74 & 0.78 & 0.50 & 0.75 & 0.55 & & 12.18 & 10.73 & 7.97 & 8.60 & 7.02 & 9.05 \\ 0.53 & 0.81 & 0.55 & 0.59 & 0.49 & 0.14 & & 12.32 & 10.85 & 8.06 & 8.70 & 7.10 & 9.14 \\ 0.70 & 0.60 & 0.12 & 0.42 & 0.29 & 0.87 & & 11.69 & 10.28 & 7.63 & 8.25 & 6.73 & 8.71 \\ 0.6049 & 4.5439 & 4.1717 & 4.7368 & & & & 12.03 & 10.59 & 7.86 & 8.49 & 6.93 & 8.95 \\ & & & & & & & 12.27 & 10.81 & 8.03 & 8.67 & 7.08 & 9.11 \end{array} \\
 \text{(linear weight matrix } W^0\text{)} \qquad \qquad \qquad \text{(output matrix of multi-head attention)}
 \end{array}$$

- once again we calculate a linear transformation, here: ZW^0
- dimensions of W^0 have to be set in order to get an output matrix that matches the dimensions of the input

Step 6 — add

- the input and output matrices are just added together

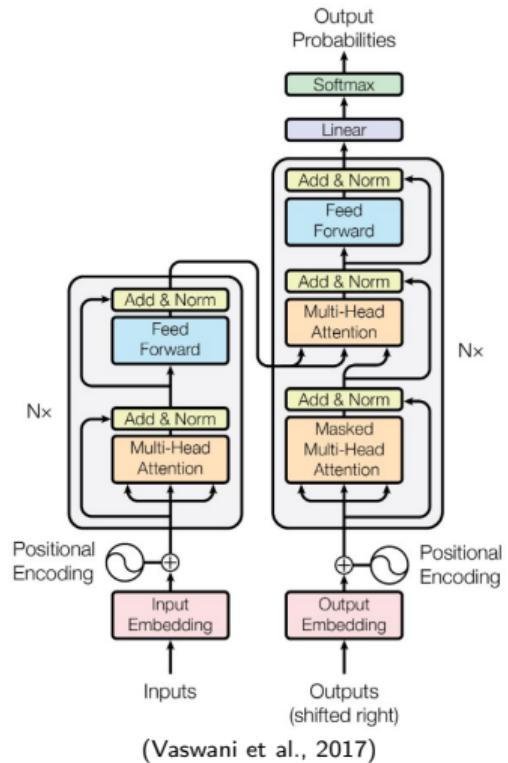
Step 6 — normalize

	(matrix to normalize)							(matrix after normalization)						
when	13.12	12.47	9.03	10.35	8.09	10.36	→	when	1.3176	0.9817	-0.7957	-0.1137	-1.2814	-0.1085
you	13.40	10.90	8.03	9.39	7.92	9.79		you	1.7078	0.4862	-0.9162	-0.2516	-0.9699	-0.0562
play	13.24	12.36	8.33	10.01	7.66	10.73		play	1.3026	0.9007	-0.9402	-0.1728	-1.2463	0.1561
game	11.95	11.02	8.29	8.47	6.80	9.08		game	1.4140	0.9236	-0.5159	-0.4209	-1.3015	-0.0993
of	12.15	11.18	8.66	9.11	7.43	9.65		of	1.4270	0.8628	-0.6030	-0.3412	-1.3184	-0.0271
thrones	11.91	12.11	8.79	10.15	8.02	10.32		thrones	1.0371	1.1596	-0.8738	-0.0408	-1.3454	0.0633

- row-wise (token-wise) we determine mean \bar{x} and standard deviation s_x
- we calculate a classical normalization $\frac{x - \bar{x}}{s_x + \epsilon}$
- we here select $\epsilon = 0.0001$

Step 7 — FFNN

- after add & norm, a classical FFNN is applied to the resulting matrix
- for the example, we assume this to be very simplistic
 - here: one linear layer + ReLU activation function $\max\{0, XW + b\}$
 - max is applied element-wise, b is added per row
 - realistic: multiple linear layers with activation functions



(Vaswani et al., 2017)

Step 7 — FFNN linear layer

(matrix after normalization)							(weight matrix W)						
when	1.3176	0.9817	-0.7957	-0.1137	-1.2814	-0.1085	0.50	0.05	0.97	0.22	0.56	0.02	
you	1.7078	0.4862	-0.9162	-0.2516	-0.9699	-0.0562	0.17	0.52	0.63	0.48	0.06	0.60	
play	1.3026	0.9007	-0.9402	-0.1728	-1.2463	0.1561	\times	0.53	0.87	0.47	0.10	0.31	0.79
game	1.4140	0.9236	-0.5159	-0.4209	-1.3015	-0.0993		0.83	0.58	0.38	0.09	0.64	0.25
of	1.4270	0.8628	-0.6030	-0.3412	-1.3184	-0.0271		0.81	0.85	0.74	0.35	0.31	0.53
thrones	1.0371	1.1596	-0.8738	-0.0408	-1.3454	0.0633		0.25	0.31	0.22	0.77	0.57	0.85
							(XW)						
when	-0.7555	-1.3047	0.5073	0.1392	0.0182	-0.8130	when	-0.7555	-1.3047	0.5073	0.1392	0.0182	-0.8130
you	-0.5575	-1.4466	0.7066	0.1121	0.2078	-1.0226	you	-0.5575	-1.4466	0.7066	0.1121	0.2078	-1.0226
play	-0.8078	-1.3957	0.4355	0.2933	0.0841	-0.7473	play	-0.8078	-1.3957	0.4355	0.2933	0.0841	-0.7473
game	-0.8378	-1.2790	0.5661	0.1330	-0.0421	-0.7045	game	-0.8378	-1.2790	0.5661	0.1330	-0.0421	-0.7045
of	-0.8173	-1.3315	0.5331	0.1548	0.0214	-0.7372	of	-0.8173	-1.3315	0.5331	0.1548	0.0214	-0.7372
thrones	-0.8552	-1.2530	0.3287	0.2716	-0.0276	-0.6433	thrones	-0.8552	-1.2530	0.3287	0.2716	-0.0276	-0.6433
(XW) + (bias vector b)							(XW + b)						
when	-0.3355	-1.1247	0.7573	0.5592	0.3682	-0.3630	when	-0.3355	-1.1247	0.7573	0.5592	0.3682	-0.3630
you	-0.1375	-1.2666	0.9566	0.5321	0.5578	-0.5726	you	-0.1375	-1.2666	0.9566	0.5321	0.5578	-0.5726
play	-0.3878	-1.2157	0.6855	0.7133	0.4341	-0.2973	play	-0.3878	-1.2157	0.6855	0.7133	0.4341	-0.2973
game	-0.4178	-1.0990	0.8161	0.5530	0.3079	-0.2545	game	-0.4178	-1.0990	0.8161	0.5530	0.3079	-0.2545
of	-0.3973	-1.1515	0.7831	0.5748	0.3714	-0.2872	of	-0.3973	-1.1515	0.7831	0.5748	0.3714	-0.2872
thrones	-0.4352	-1.0730	0.5787	0.6916	0.3224	-0.1933	thrones	-0.4352	-1.0730	0.5787	0.6916	0.3224	-0.1933
bias	0.4200	0.1800	0.2500	0.4200	0.3500	0.4500							

Step 7 — FFNN activation

$\max\{0, XW + b\} =$	when	0.0000	0.0000	0.7573	0.5592	0.3682	0.0000
	you	0.0000	0.0000	0.9566	0.5321	0.5578	0.0000
	play	0.0000	0.0000	0.6855	0.7133	0.4341	0.0000
	game	0.0000	0.0000	0.8161	0.5530	0.3079	0.0000
	of	0.0000	0.0000	0.7831	0.5748	0.3714	0.0000
	thrones	0.0000	0.0000	0.5787	0.6916	0.3224	0.0000

- after the FFNN there is an additional add & norm layer applied
- this completes the first encoder layer
- we won't calculate additional encoder layers here
- in practice there are N encoder layers, where the output of the add & norm layer (mentioned above) would serve as input for the second encoder layer

Decoder elements

- encoder input: when you play game of thrones
- decoder input: <start> you win or you die <end>
- most of the calculation in the decoder is the same
- three elements are new:
 - ⑧ decoder MHA
 - ⑨ masked MHA
 - ⑩ predicting tokens

Small detail: padding

- assume `n_sequence = 10` as sequence length of the model
- our input sentence is of size 6
- we have to pad the encoder input tokens 7, 8, 9, and 10 with 0s

d	when 5	you 17	play 7	game 12	of 15	thrones 19	PAD	PAD	PAD	PAD
1	0.79	0.38	0.01	0.12	0.88	0.60	0	0	0	0
2	0.60	-0.37	1.92	1.73	1.25	1.02	0	0	0	0
3	0.96	-0.15	-0.14	0.05	0.04	-0.12	0	0	0	0
4	0.64	-0.19	0.40	-0.58	-0.03	1.01	0	0	0	0
5	0.97	0.85	0.47	-0.07	0.32	0.71	0	0	0	0
6	0.20	-0.26	0.59	-0.62	-0.28	0.24	0	0	0	0

Padding in MHA

$$Z = \text{softmax} \left(\frac{QK^T + \text{MASK}}{\sqrt{d_k}} \right) V$$

- MASK is just a padding matrix in the encoder
- here $-\infty$ is the padding token, since it results in 0s after softmax
- in our example:

MASK =

0	0	0	0	0	0	$-\infty$	$-\infty$	$-\infty$	$-\infty$
0	0	0	0	0	0	$-\infty$	$-\infty$	$-\infty$	$-\infty$
0	0	0	0	0	0	$-\infty$	$-\infty$	$-\infty$	$-\infty$
0	0	0	0	0	0	$-\infty$	$-\infty$	$-\infty$	$-\infty$
0	0	0	0	0	0	$-\infty$	$-\infty$	$-\infty$	$-\infty$
$-\infty$									
$-\infty$									
$-\infty$									
$-\infty$									

Step 8 — masked MHA

$$Z = \text{softmax} \left(\frac{QK^T + \text{MASK}}{\sqrt{d_k}} \right) V$$

- masking in the decoder MHA works quite similar to padding
- we again use $-\infty$ as masking token
- we mask all future tokens, i.e.

$$\text{MASK} = \begin{matrix} 0 & -\infty \\ 0 & 0 & -\infty \\ 0 & 0 & 0 & -\infty \\ 0 & 0 & 0 & 0 & -\infty & -\infty & -\infty & -\infty & -\infty & -\infty \\ 0 & 0 & 0 & 0 & 0 & -\infty & -\infty & -\infty & -\infty & -\infty \\ 0 & 0 & 0 & 0 & 0 & 0 & -\infty & -\infty & -\infty & -\infty \\ -\infty & -\infty \\ -\infty & -\infty \\ -\infty & -\infty \\ -\infty & -\infty \end{matrix}$$

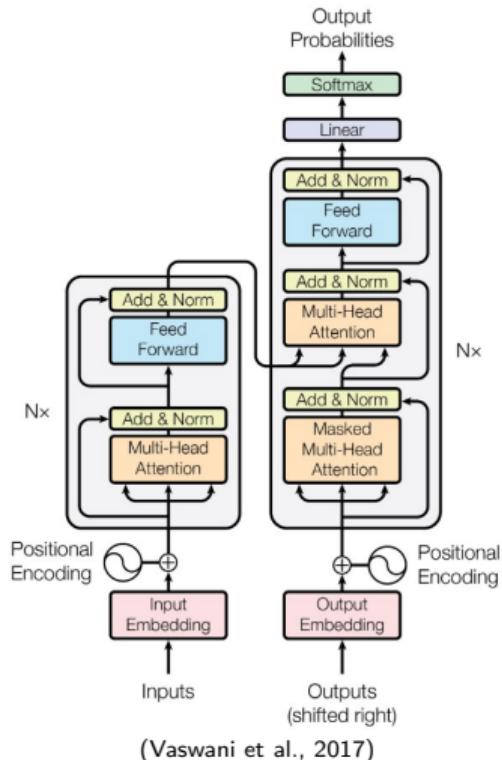
Input for the decoder

- <start> and <end>, as well as other special tokens (e.g., <sep>), are part of the vocabulary set
- the decoder input matrix looks like this:

d	<start>	you	win	or	you	die	<end>	PAD	PAD	PAD
	<start>	17	14	21	17	22	<end>			
1	0.51	0.38	0.91	0.12	0.38	0.60	0.11	0	0	0
2	0.83	-0.37	1.92	0.03	-0.37	1.22	0.12	0	0	0
3	0.22	-0.15	-0.14	0.05	-0.15	-0.12	-0.50	0	0	0
4	0.04	-0.55	0.20	-0.58	-0.55	-1.01	0.01	0	0	0
5	-0.11	0.85	0.77	-0.57	0.85	0.31	1.30	0	0	0
6	0.20	-1.80	0.59	-0.62	-1.80	0.24	0.012	0	0	0

Step 9 — decoder MHA

- $Z = \text{softmax}((QK^T + \text{MASK})/\sqrt{d_k})V$
 - Q is calculated from the output of the first add & norm layer from the decoder
 - $Q = X^D W^Q$
 - where X^D is the output from the add & norm layer
 - K, V are calculated using the output of the last encoder layer
 - $K = X^E W^K$
 - $V = X^E W^V$
 - where X^E is the output from the last encoder layer
- in the following add & norm layer, we calculate the normalization of $X^D + ZW^0$



(Vaswani et al., 2017)

Step 9 — decoder MHA MASK matrix

$$Z = \text{softmax} \left(\frac{QK^T + \text{MASK}}{\sqrt{d_k}} \right) V$$

- here we have a quite similar structure to encoder MHA
- Q comes from decoder with one more input token \rightarrow one row less to pad with $-\infty$

MASK =

0	0	0	0	0	0	$-\infty$	$-\infty$	$-\infty$	$-\infty$
0	0	0	0	0	0	$-\infty$	$-\infty$	$-\infty$	$-\infty$
0	0	0	0	0	0	$-\infty$	$-\infty$	$-\infty$	$-\infty$
0	0	0	0	0	0	$-\infty$	$-\infty$	$-\infty$	$-\infty$
0	0	0	0	0	0	$-\infty$	$-\infty$	$-\infty$	$-\infty$
0	0	0	0	0	0	$-\infty$	$-\infty$	$-\infty$	$-\infty$
0	0	0	0	0	0	$-\infty$	$-\infty$	$-\infty$	$-\infty$
$-\infty$									
$-\infty$									
$-\infty$									

Step 10 — predicting tokens

- we learn a final linear layer that projects the output of the last add & norm layer to logits for each token position
- output of last add & norm layer: $n_{\text{sequence}} \times d$ (here: 10×6)
- output of final linear layer: $n_{\text{sequence}} \times V$ (here: $10 \times 23 + \text{number of special tokens}$)
- we need a weight matrix W of size $d \times V$

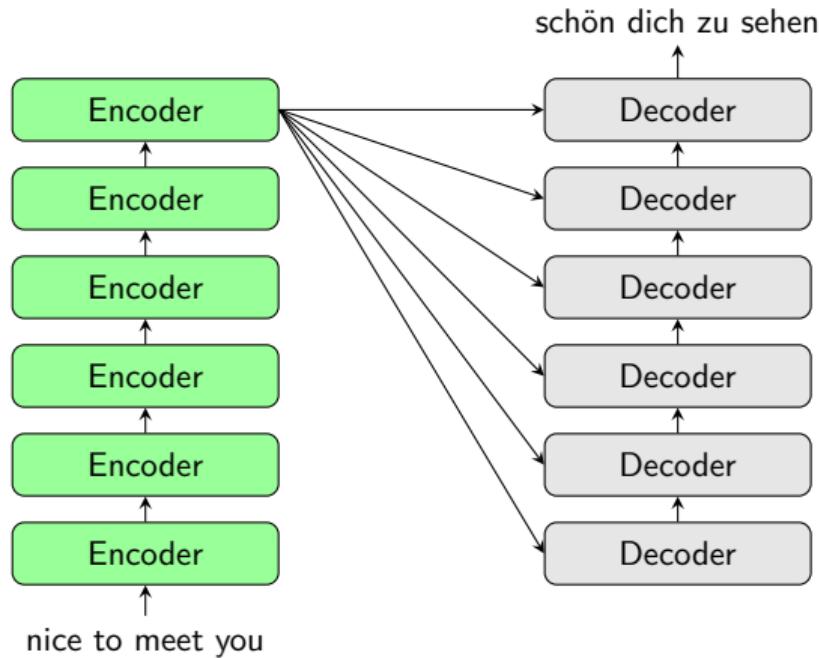
$$XW = L, \quad X \in \mathbb{R}^{n_{\text{seq}} \times d}, W \in \mathbb{R}^{d \times V}, L \in \mathbb{R}^{n_{\text{seq}} \times V}$$

- output of the final softmax layer is $\text{softmax}(L) \in [0, 1]^{n_{\text{seq}} \times V}$
- with $\sum_{i=1}^V \text{softmax}(L)_{i,j} = 1 \quad \forall j = 1, \dots, n_{\text{seq}}$
- these can be seen as pseudo-probabilities
- to get the encoding index for the prediction of token j simply apply $\arg \max_{i=1, \dots, V} L_{i,j}$

Further resources

see also <https://www.youtube.com/watch?v=EixI6t5oif0>

Breaking BERT¹ down^{2,3}

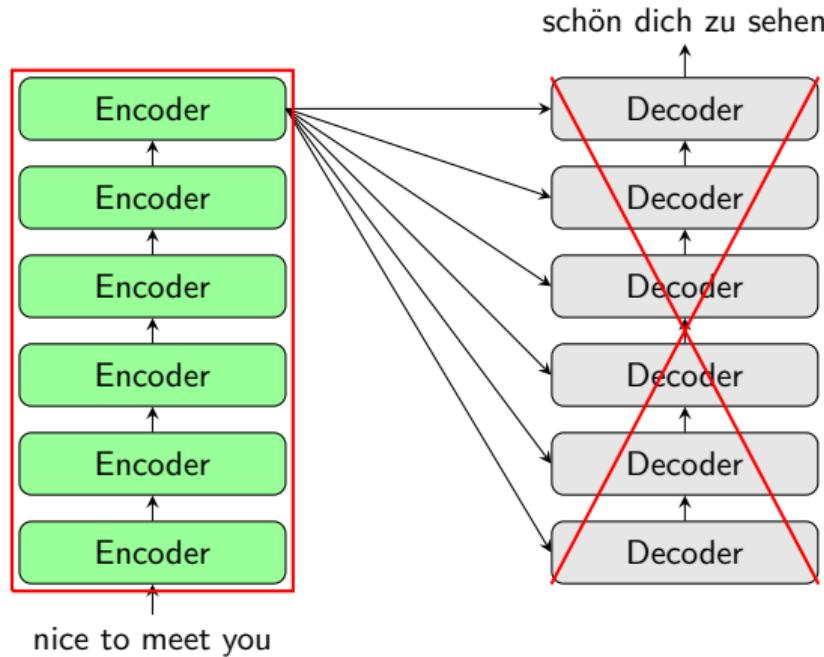


¹ Bidirectional Encoder Representations from Transformers

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Devlin et al., NAACL 2019) <http://dx.doi.org/10.18653/v1/N19-1423>

²<https://jalammar.github.io/illustrated-transformer/> ³<https://towardsdatascience.com/breaking-bert-down-430461f60efb>

Breaking BERT¹ down^{2,3}



¹ Bidirectional Encoder Representations from Transformers

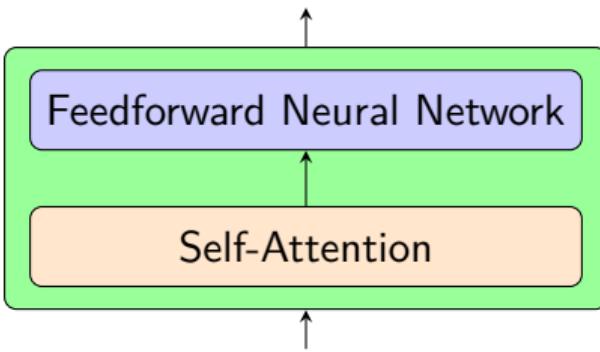
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Devlin et al., NAACL 2019) <http://dx.doi.org/10.18653/v1/N19-1423>

²<https://jalammar.github.io/illustrated-transformer/> ³<https://towardsdatascience.com/breaking-bert-down-430461f60efb>

Encoder vs. decoder

- originally BERT is an **encoder only**
 - bidirectional approach contradicts traditional LM
 - predicting masked tokens (inside a sentence)
 - output: embeddings
 - extractive tasks (classification)
- there are **decoder only** models as well, e.g., GPT
 - unidirectional
 - predicting all tokens (to the right of a sentence)
 - output: (sequence of) words
 - language generation
- **encoder-decoder** models useful, e.g., for MT

Encoder layer in BERT



- every encoder layer contains of
 - ① self-attention layer
 - ② FF layer
- input of first encoder: list of embeddings (typically $d = 512$)
- size of list: hyperparameter (\approx longest sequence in training data)
- input of all other encoders: output of previous encoder (same d)

BERT's innovation

- traditional LM: left-to-right
- ELMo: left-to-right and right-to-left
- BERT: bidirectional training
- using masked language modeling (see following slides)
- also uses a subword tokenizer (WordPiece), cf. fastText
- Google released two (pre-trained) versions:
 - pre-trained encoder
 - base: 12 layers, 768 output size, 12 MHA layers, 30 522 vocab
number of parameters: 110 million
 - large: 24 layers, 1024 output size, 16 MHA layers, 30 522 vocab
number of parameters: 340 million

How to train BERT

- pre-training was done on
 - BooksCorpus (800 million words)
 - English Wikipedia (2500 million words)
- took 4 days on 64 TPUs (tensor processing unit)
- TPUs are (even) faster in matrix operations than GPUs
- fine-tuning typically on single GPU

how to pre-train encoders?

Masked language modeling (MLM)

- BERT can see all the words in the sentences
- force BERT to learn embeddings without seeing the answer
- idea:
 - ① replace a fraction of words in the input with a special [MASK] token
 - ② predict these words

Masked language modeling (MLM)

- BERT can see all the words in the sentences
- force BERT to learn embeddings without seeing the answer
- idea:
 - ① replace a fraction of words in the input with a special [MASK] token
 - ② predict these words
- in BERT:
 - 15% of (sub)words are sampled to predict
 - 80%: replace with [MASK]
 - 10%: replace with random token
 - 10%: replace with self
 - why these values?
 - 100% [MASK] → only learning masked words' embeddings
 - 0% self → BERT would know:
predict non-masked → always wrong word
 - 0% random: analogously to 0% self

Next-sentence prediction

- pre-training includes a second objective
- given two sentences A and B
 - is B likely a sentence that follows A?
 - binary classification task
 - balanced, i.e., 50%/50% positive/negative examples
- idea: learn relationship between sentences
 - e.g., for question answering (QA) tasks

Next-sentence prediction

- pre-training includes a second objective
- given two sentences A and B
 - is B likely a sentence that follows A?
 - binary classification task
 - balanced, i.e., 50%/50% positive/negative examples
- idea: learn relationship between sentences
 - e.g., for question answering (QA) tasks
- technically (cf. next slide):
 - add [CLS] token at the beginning of first sentence
 - add [SEP] at the end of each sentence
 - add segment (sentence) embedding
 - predict [CLS] token
- turns out that next-sentence prediction is not that important for training

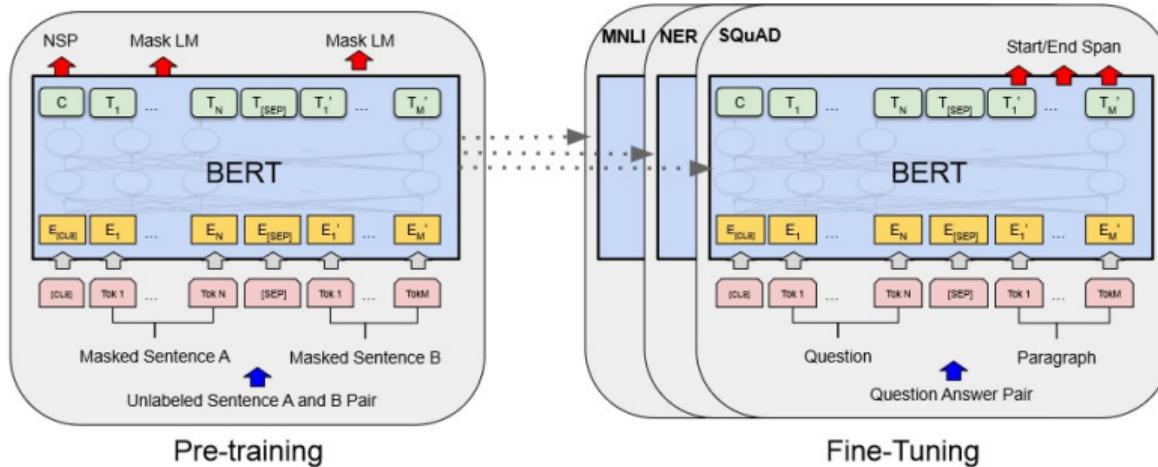
BERT input

Input	[CLS]	my	dog	is	cute	[SEP]	he	likes	play	# #ing	[SEP]
Token Embeddings	$E_{[CLS]}$	E_{my}	E_{dog}	E_{is}	E_{cute}	$E_{[SEP]}$	E_{he}	E_{likes}	E_{play}	$E_{# #ing}$	$E_{[SEP]}$
Segment Embeddings	E_A	E_A	E_A	E_A	E_A	E_A	E_B	E_B	E_B	E_B	E_B
Position Embeddings	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}

(Devlin et al., 2019)

- [CLS] useful for subsequent classification tasks
- [sentence A, sentence B] may be, e.g., [question, answer] for QA fine-tuning (cf. next slide)

Fine-tuning (FT) BERT

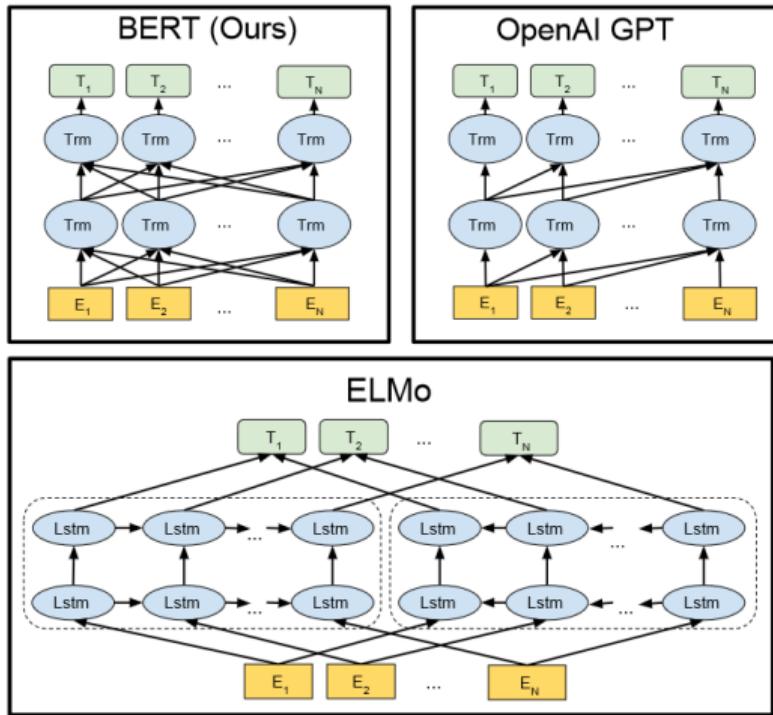


(Devlin et al., 2019)

- several different downstream NLP tasks, e.g., NER, sentence classification, QA
- output layer for token level tasks + [CLS] classification task
- fine-tuning all parameters

Differences in pre-training

- BERT: bidirectional
- GPT: left-to-right
- ELMo: left-to-right + right-to-left
- BERT & GPT: FT
- ELMo: feature-based



(Devlin et al., 2019)

Transformer-based models I

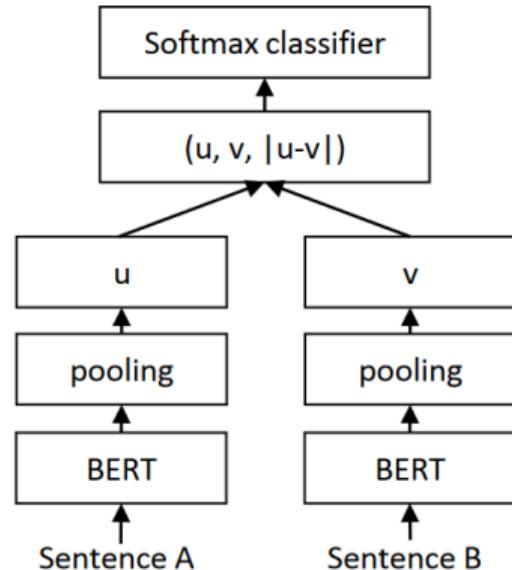
- a lot of modifications/improvements and similar architectures as BERT
- RoBERTa (English)
 - mainly just train BERT for longer and remove next sentence prediction
 - base: 12 layers, 768 output size, 50 265 vocab
 - large: 24 layers, 1024 output size, 50 265 vocab
- XLM-RoBERTa (multilingual)
 - trained on multilingual data
 - base: 12 layers, 768 output size, 250 002 vocab
 - large: 24 layers, 1024 output size, 250 002 vocab
- (m)DeBERTa (English, multilingual)
 - disentangled attention mechanism → 2 embedding vectors: content & position
 - base: 12 layers, 768 output size, 128 100 vocab (250 002 vocab)
 - large: 24 layers, 1024 output size, 128 100 vocab (250 002 vocab)

Transformer-based models II

- DistilBERT (English)
 - less parameters, compression technique, tries to mimic BERT (base)
 - 60% parameters of BERT base, 60% faster
 - preserving over 95% of performances (GLUE benchmark)
- SBERT
 - dedicated sentence embedding training objective
 - enhanced version of predicting the [CLS] token (see following slide)
- ALBERT, CamemBERT, ELECTRA, ...

Sentence-BERT (SBERT)

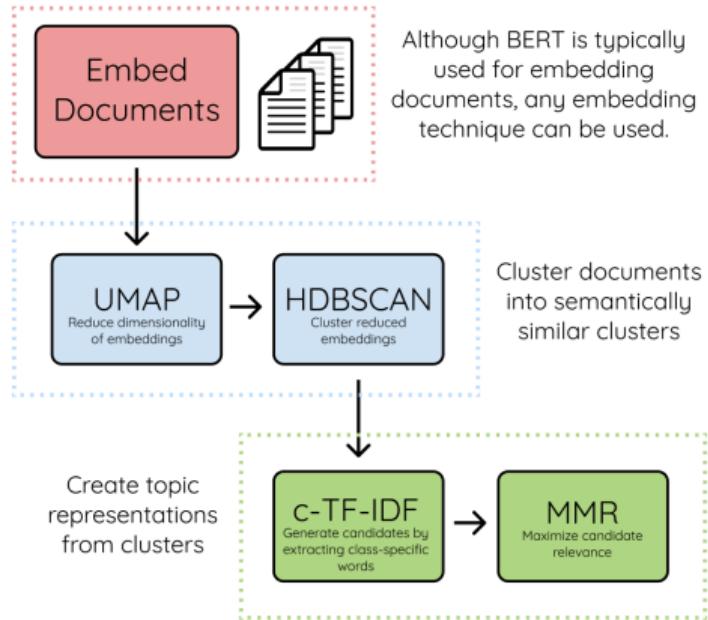
- aim: sentence embeddings
 - similar embeddings for similar sentences
- technique: siamese BERT-networks
- similarities between sentence-pairs
- classification objective: $\text{softmax}(W_t(u, v, |u - v|))$
- triplet objective: $\max(d(a, p) - d(a, n) + \epsilon, 0)$
 - a anchor sentence
 - p positive, n negative sentence
 - d euclidean distance, $\epsilon = 1$
- trained on STSb dataset
https://huggingface.co/datasets/stsb_multi_mt



<https://arxiv.org/abs/1908.10084>

Further resource: <https://www.pinecone.io/learn/sentence-embeddings/>

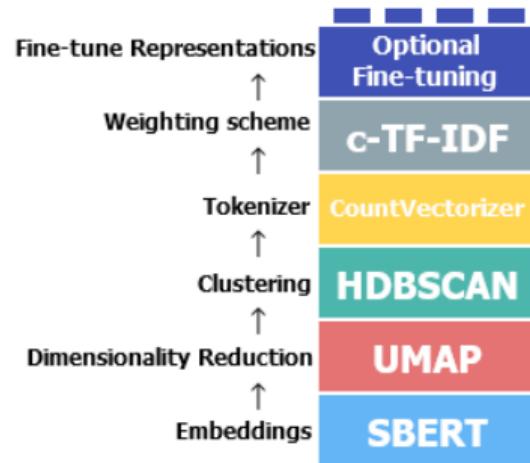
How BERTopic works



<https://github.com/MaartenGr/BERTopic/blob/master/docs/img/algorithm.png>

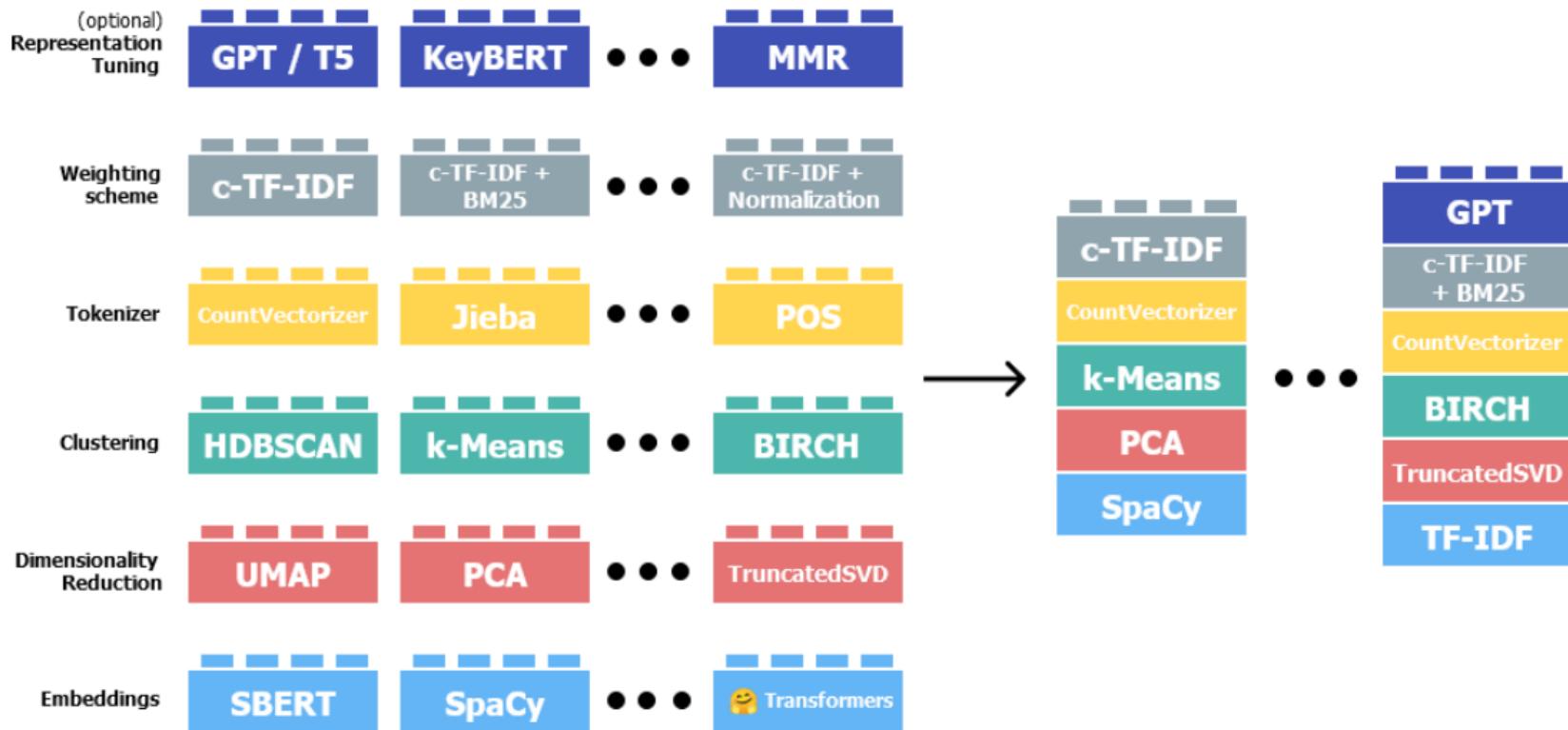
Default workflow of BERTopic

- SBERT as embedding technique
 - <https://github.com/UKPLab/sentence-transformers>
 - default: "all-MiniLM-L6-v2"
 - non-english: "paraphrase-multilingual-MiniLM-L12-v2"
- UMAP as dimension reduction technique
 - cf., PCA, truncated SVD
- HDBSCAN as clustering technique
 - cf., k-means
- CountVectorizer for tokenization
 - comparable to the presented approaches
- c-tf-idf as weighting technique
 - just a cluster-based tf-idf



<https://maartengr.github.io/BERTopic/algorithm/algorith.html>

Modularity of BERTopic



c-tf-idf

$$\text{c-tf-idf}_{w,c} = \frac{\text{tf}_{w,c}}{\sum_{v=1}^V \text{tf}_{v,c}} \log \left(1 + \frac{\frac{1}{C} \sum_{k=1}^C \sum_{v=1}^V \text{tf}_{v,k}}{\text{tf}_w} \right)$$

- instead of calculating idf based on documents, here based on clusters
- term-frequencies per cluster $\text{tf}_{w,c}$ are normalized
- nominator in logarithm: average number of words per class
- actually it should be named tf-icf: term-frequency – inverse-cluster-frequency
- it is also possible to use, e.g., the importance score from slide 164

Further resources on BERTopic

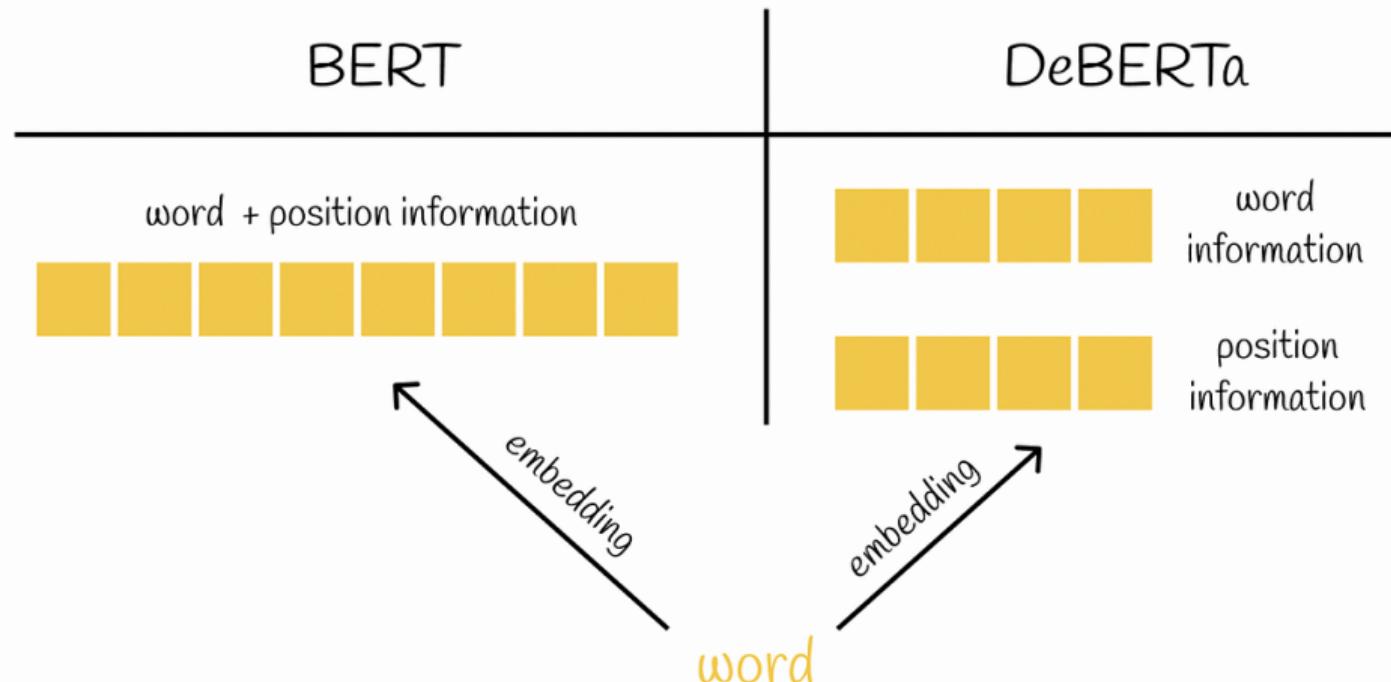
- <https://maartengr.github.io/BERTopic/>
- <https://youtu.be/uZxQz87l1b84>
 - Maarten Grootendorst (author) explains BERTopic
 - YouTube Video (53 min.)
- <https://youtu.be/fb7LENb9eag>
 - BERTopic explained
 - YouTube Video (45 min.)
- <https://python.plainenglish.io/topic-modeling-for-beginners-using-bertopic-and-python-aaf1b421afeb>
 - Topic Modeling For Beginners Using BERTTopic and Python
 - py/medium story (member-only)
- <https://towardsdatascience.com/let-us-extract-some-topics-from-text-data-part-iv-bertopic-46ddf3c91622>
 - Let us Extract some Topics from Text Data — Part IV: BERTopic
 - tds/medium story (member-only)
- <https://towardsdatascience.com/advanced-topic-modeling-with-bertopic-85fb8a90369e>
 - Advanced Topic Modeling with BERTopic
 - tds/medium story (member-only)

Decoding-enhanced BERT with disentangled attention (DeBERTa)

- in comparison to BERT and RoBERTa, three changes
 - ① disentangled attention,
i.e. each token is represented using two vectors that encode its content and position
 - ② enhanced mask decoder,
for information about absolute token position
 - ③ scale invariant fine-tuning,
for improving LLM training's generalizability

DOI: [10.48550/arXiv.2006.03654](https://doi.org/10.48550/arXiv.2006.03654)

Disentangled attention



<https://towardsdatascience.com/large-language-models-deberta-decoding-enhanced-bert-with-disentangled-attention-90016668db4b>

Disentangled attention

- a token at position i is represented using two vectors H_i and $P_{i|j}$
 - ① H_i representing the content
 - ② $P_{i|j}$ representing the relative position with the token at position j
- can be decomposed into four components:

$$\begin{aligned} A_{i,j} &= [H_i, P_{i|j}] \times [H_j, P_{j|i}]^T \\ &= H_i H_j^T + H_i P_{j|i}^T + P_{i|j} H_j^T + P_{i|j} P_{j|i}^T \end{aligned}$$

- the authors remove $P_{i|j} P_{j|i}^T$ for implementation since it does not provide additional information due to the “double” relative relation

Disentangled attention components

$$A_{ij} = \{ H_i \underbrace{\quad\quad\quad}_{\text{content } i} P_{ilj} \underbrace{\quad\quad\quad}_{\text{relative position of } i \text{ with } j} \} \times \{ H_j \underbrace{\quad\quad\quad}_{\text{content } j} P_{jli} \underbrace{\quad\quad\quad}_{\text{relative position of } j \text{ with } i} \}^T =$$

$$= \underbrace{H_i H_j^T}_{\text{content-to-content matrix}} x + \underbrace{H_i P_{jli}^T}_{\text{content-to-position matrix}} x + \underbrace{P_{ilj} H_j^T}_{\text{position-to-content matrix}} x + \underbrace{P_{ilj} P_{jli}^T}_{\text{position-to-position matrix}} x$$

<https://towardsdatascience.com/large-language-models-deberta-decoding-enhanced-bert-with-disentangled-attention-90016668db4b>

Disentangled attention calculation

- single-head attention example with $Q = XW^Q$, $K = XW^K$, $V = XW^V$
- denote: k maximum relative distance and $\delta(i, j) \in [0, 2k]$ the relative distance between tokens i and j with

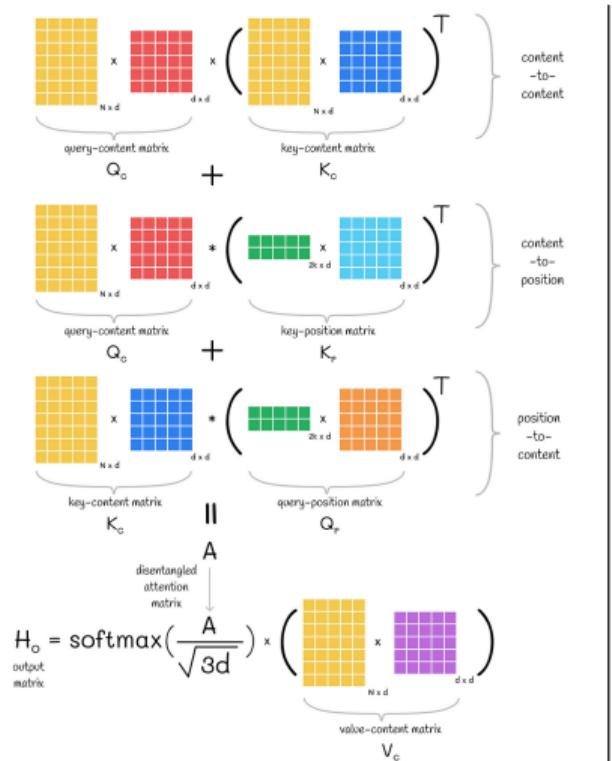
$$\delta(i, j) = \begin{cases} 0 & \text{for } i - j \leq -k \\ 2k - 1 & \text{for } i - j \geq k \\ i - j + k & \text{others} \end{cases}$$

- $P \in \mathbb{R}^{2k \times d}$ represents relative position embedding vectors (shared across all layers)
- $W^{Q,r}, W^{K,r} \in \mathbb{R}^{d \times d}$ are projection matrices — in addition to W^Q, W^K, W^V
- $Q^r = PW^{Q,r}$, $K^r = PW^{K,r}$

$$\tilde{A}_{i,j} = Q_i K_j^T + Q_i {K_{\delta(i,j)}^r}^T + K_j {Q_{\delta(j,i)}^r}^T \quad X_i \text{ indicates row } i \text{ of } X$$

$$Z = \text{softmax}(\tilde{A}/\sqrt{3d}) V$$

Disentangled attention visualization

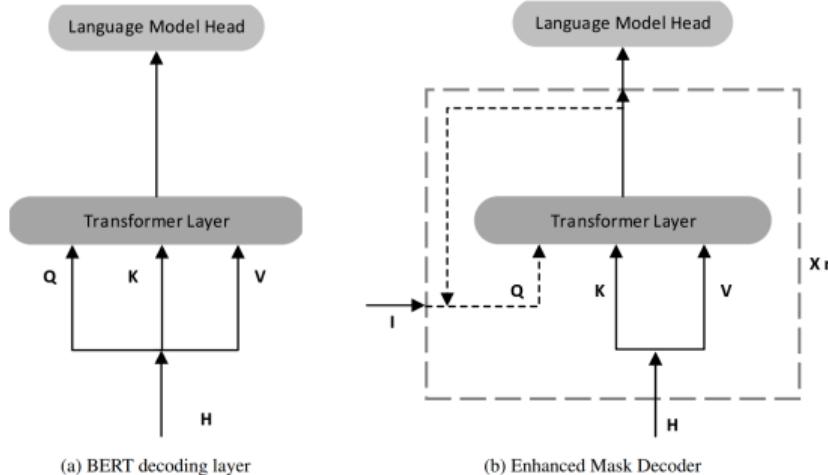


$H =$		input hidden vectors $N \times d$
$P =$		relative position embedding vectors $2k \times d$
$W_{qo} =$		projection query-content matrix $d \times d$
$W_{qr} =$		projection query-position matrix $d \times d$
$W_{kc} =$		projection key-content matrix $d \times d$
$W_{kp} =$		projection key-position matrix $d \times d$
$W_{qc} =$		projection value-content matrix $d \times d$

N – input sequence length
 d – hidden state dimension
 k – parameter controlling maximum relative distance

<https://towardsdatascience.com/large-language-models-deberta-decoding-enhanced-bert-with-disentangled-attention-90016668db4b>

Enhanced mask decoder (EMD)

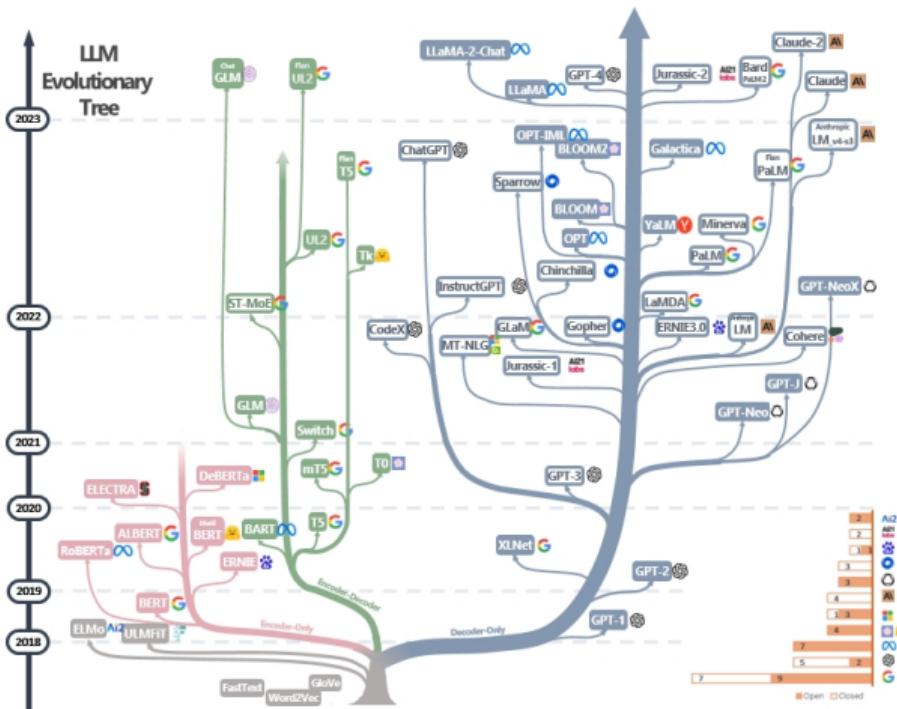


- EMD for absolute word positions
- in BERT: absolute positional encoding directly applied after input
- in DeBERTa:
 - content and (relative) position via disentangled attention
 - absolute position directly after all transformer layers, but before softmax layer
- for $n = 1$ and $I = H$ both versions are the same

Scale invariant fine-tuning (SiFT)

- SiFT introduces an additional adversarial training via perturbations
- two steps:
 - ① normalization of word embedding vectors
(inspired by layer normalization)
 - ② perturbation of normalized vectors

LLM evolutionary tree



<https://github.com/Mooler0410/LLMsPracticalGuide>

- it's not all ChatGPT
- but the LLM "market" is nowadays dominated by companies
- Part of the problem is that no university in the world today can afford to develop its own model like ChatGPT. [...] Not even us.¹
- the only remedy: stop focusing on ever larger and more powerful (and resource-hungry) models

¹translated: "Teil des Problems ist schon, dass es sich keine Universität der Welt heute leisten kann, ein eigenes Modell wie ChatGPT zu entwickeln. [...] Nicht einmal wir.", Prof. Fei-Fei Li (Stanford) in Der Spiegel Nr. 51, 16.12.2023.

Examples for pre-trained decoder language models

- GPT-3 (generative pretrained transformer)
 - successor of GPT and GPT-2 (12 layers, 768 output size, 50 257 vocab)
 - decoder only
 - 800GB storage for model parameters
- BLOOM (BigScience large open-science open-access multilingual language model)
 - decoder only
 - open access
 - developed as (free) alternative to GPT based models
- ChatGPT
 - at the moment (open) live study
 - no (formal) publication on it; 2023: but tech reports
 - open version an update of GPT-3 (GPT-3.5); 2023: payed access to GPT-4
- Gemini, PaLM, Claude, Falcon, LLaMA, Grok, OLMo, Llama 2, LLaMA 3.1, Mistral, Mixtral, T5, ...

Research priorities for universities (in my opinion; not exhaustive)

What?	Why?	How?
efficiency	saving resources (time, energy, money)	parameter-efficient fine-tuning (PEFT), few-shot learning, effective learning (e.g., Adam), short-cutting via early-exit strategies
robustness	(out-of-domain/near-domain) application in real world scenarios	regularization (e.g., Adam, dropout layer), (transfer learning)
flexibility	efficient near-domain/out-of-domain application in real world scenarios	transfer learning, PEFT
reutilization	democratization of research for faster progress	transfer learning, modularity (e.g., PEFT)

8

Training Methods

- Adam Optimizer
- Dropout Layers
- Fine-Tuning
- (Human-Labeled) Annotations
- Few- and Zero-Shot Learning

Adam optimizer¹

- adaptive moment estimation
- first-order gradient-based optimization of stochastic objective functions using lower-order moments
- fast convergence and robustness
- individual adaptive learning rates for different parameters (based on gradients' history)
- standard gradient descent:

$$\text{fixed learning rate } \eta \quad \text{with} \quad \theta_{t+1} = \theta_t - \eta \frac{\partial L}{\partial \theta}$$

- Adam ingredients:
 - ① momentum
 - ② root mean square propagation (RMSProp)

¹ Adam: A Method for Stochastic Optimization (Kingma and Ba, 2014) <https://doi.org/10.48550/arXiv.1412.6980>

Adam: momentum

- adding a fraction of previous gradient to current gradient

$$\theta_{t+1} = \theta_t - m_t \quad \text{with} \quad m_t = \beta_1 m_{t-1} + (1 - \beta_1) \frac{\partial L}{\partial \theta}$$

- β_1 is an additional hyperparameter
- results in a boost if gradients point in same direction
- results in a slow down for opposing gradients

Adam: RMSprop¹

- controlling overshooting by regulating step size

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t} + \epsilon} \frac{\partial L}{\partial \theta} \quad \text{with} \quad v_t = \beta_2 v_{t-1} + (1 - \beta_2) \left(\frac{\partial L}{\partial \theta} \odot \frac{\partial L}{\partial \theta} \right)$$

- β_2 is an additional hyperparameter
- \odot means pointwise multiplication
- weighted moving average of squared gradients (variance of gradients)
- results in a boost if variance of gradients is low
- results in a slow down if variance of gradients is high

¹https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

Adam updates

$$\begin{aligned}m_t &= \beta_1 m_{t-1} + (1 - \beta_1) \frac{\partial L}{\partial \theta} \\v_t &= \beta_2 v_{t-1} + (1 - \beta_2) \left(\frac{\partial L}{\partial \theta} \odot \frac{\partial L}{\partial \theta} \right) \\\theta_t &= \theta_{t-1} - \left(\alpha \frac{m_t}{\sqrt{v_t} + \epsilon} \right)\end{aligned}$$

- combination of momentum (with β_1) and RMSprop (with β_2) with learning rate α
- in final version an additional warmup procedure is implemented (see next slide)
- faster convergence, more robust, little memory requirement
- easy to implement → default optimizer in deep learning

Adam algorithm

Algorithm 1: Adam, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t .

Require: α : Stepsize
Require: $\beta_1, \beta_2 \in [0, 1]$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0 : Initial parameter vector
 $m_0 \leftarrow 0$ (Initialize 1st moment vector)
 $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
 $t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
 $t \leftarrow t + 1$
 $g_t \leftarrow \nabla_{\theta} f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
 $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
 $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
 $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
 $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
 $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)

(Kingma and Ba, 2014)

- Attention paper:
 $\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 10^{-9}$
 and adaptive learning rate
 $\alpha = d^{-0.5} \min(\text{step}^{-0.5},$
 $\text{step} \cdot \text{warmup_steps}^{-1.5})$
 with $\text{warmup_steps} = 4000$
- alternatives are, e.g.,
 - SGD with Momentum
 - Nesterov accelerated gradient
 - AdaGrad
 - RMSprop
 - AdaDelta
 - AMSGrad
 - NAdam

Dropout¹ idea

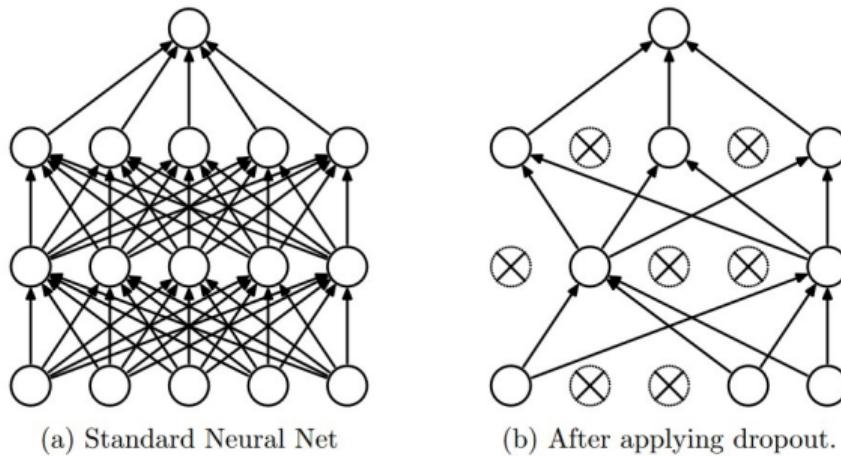


Figure 1: Dropout Neural Net Model. **Left:** A standard neural net with 2 hidden layers. **Right:** An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

- prevents overfitting by dropping neural units (nodes) by chance
- reduces complexity during training and increases robustness during inference
- simulates training a collection of 2^n networks (n nodes)

¹ Dropout: A Simple Way to Prevent Neural Networks from Overfitting (Srivastava et al., 2014)

<https://jmlr.org/papers/v15/srivastava14a.html>

Dropout

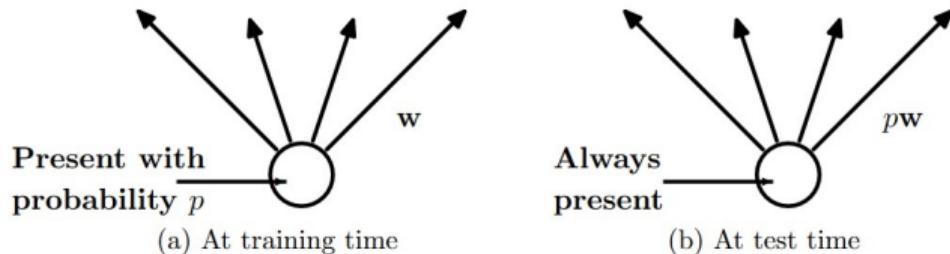


Figure 2: **Left:** A unit at training time that is present with probability p and is connected to units in the next layer with weights w . **Right:** At test time, the unit is always present and the weights are multiplied by p . The output at test time is same as the expected output at training time.

it is just that simple that the FFNN operation of a hidden unit i

$$w_i x + b_i \quad \text{becomes} \quad w_i \tilde{x} + b_i$$

with $\tilde{x} = r * x$ and $r_j \sim \text{Bernoulli}(p)$

- training: “sample” one network structure
- testing: use the combination/aggregation of all existing nodes
- note the detail on weights: w vs. pw
- Attention paper uses $p = 0.1$

PLMs, downstream tasks and fine-tuning

- you already know how transformers are (pre-)trained, cf. slide 229ff.
- for simplicity reasons, and since this is the common research scenario, we will focus on fine-tuning encoder-only models
- regarding fine-tuning a pre-trained model:
- let's come back to the very first example study (PETapter¹) and the respective Ukraine dataset
- (downstream) task: detect claims/arguments (and their stance) in text
- data: a filtered set of newspaper articles,
(hopefully) mostly about arms deliveries to Ukraine
- model class of choice: PLMs

¹<https://openreview.net/forum?id=Dq8VpnQ4M6>

Our downstream task

- PLMs are typically trained on next word prediction
- i.e., pre-training alone does not result in the ability to predict linguistic elements or stance
- to fine-tune a PLM, we need (at least a small) labeled dataset for supervised learning
- first design decision: how do you want to process data?
 - per article?

Our downstream task

- PLMs are typically trained on next word prediction
- i.e., pre-training alone does not result in the ability to predict linguistic elements or stance
- to fine-tune a PLM, we need (at least a small) labeled dataset for supervised learning
- first design decision: how do you want to process data?
 - per article? not useful (content-wise) and too long (technically)
 - per sentence?

Our downstream task

- PLMs are typically trained on next word prediction
- i.e., pre-training alone does not result in the ability to predict linguistic elements or stance
- to fine-tune a PLM, we need (at least a small) labeled dataset for supervised learning
- first design decision: how do you want to process data?
 - per article? not useful (content-wise) and too long (technically)
 - per sentence?
 - some content of sentences need context of previous/future sentence(s) to answer question whether it is positive/negative, claim/argument

Our downstream task

- PLMs are typically trained on next word prediction
- i.e., pre-training alone does not result in the ability to predict linguistic elements or stance
- to fine-tune a PLM, we need (at least a small) labeled dataset for supervised learning
- first design decision: how do you want to process data?
 - per article? not useful (content-wise) and too long (technically)
 - per sentence?
 - some content of sentences need context of previous/future sentence(s) to answer question whether it is positive/negative, claim/argument
 - we include context as kind of a rolling window (technical implementation later)
- so, we decided on the input; how does the output look like?

Our downstream task

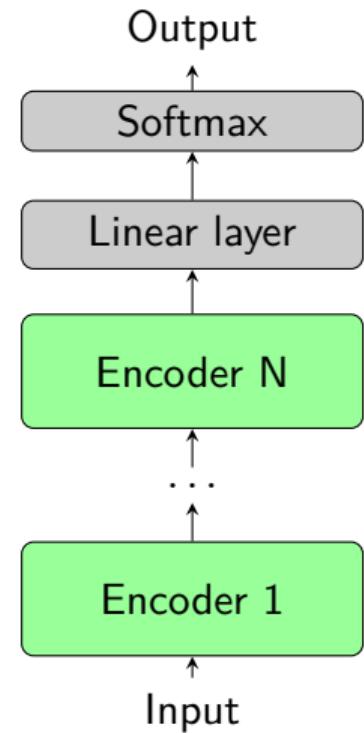
- PLMs are typically trained on next word prediction
- i.e., pre-training alone does not result in the ability to predict linguistic elements or stance
- to fine-tune a PLM, we need (at least a small) labeled dataset for supervised learning
- first design decision: how do you want to process data?
 - per article? not useful (content-wise) and too long (technically)
 - per sentence?
 - some content of sentences need context of previous/future sentence(s) to answer question whether it is positive/negative, claim/argument
 - we include context as kind of a rolling window (technical implementation later)
- so, we decided on the input; how does the output look like?
 - we have 4 relevant classes: claim for, claim against, argument for, argument against

Our downstream task

- PLMs are typically trained on next word prediction
- i.e., pre-training alone does not result in the ability to predict linguistic elements or stance
- to fine-tune a PLM, we need (at least a small) labeled dataset for supervised learning
- first design decision: how do you want to process data?
 - per article? not useful (content-wise) and too long (technically)
 - per sentence?
 - some content of sentences need context of previous/future sentence(s) to answer question whether it is positive/negative, claim/argument
 - we include context as kind of a rolling window (technical implementation later)
- so, we decided on the input; how does the output look like?
 - we have 4 relevant classes: claim for, claim against, argument for, argument against
 - and one additional class: not relevant

High level model architecture

- input: [target sentence] <sep> [context before] <sep> [context after]
- output ∈ {claim for, claim against, argument for, argument against, irrelevant}
- implications?
- linear layer: down-projection from $\mathbb{R}^{n_seq \times d}$ to \mathbb{R}^5
- naive approach: pooling the logits from $\mathbb{R}^{n_seq \times 5}$
- better: use of CLS token at the beginning of every input
 - this CLS token is useful in classification tasks, since it is said that it “contains” the meaning of the whole input
 - i.e., we just train linear weights $\mathbb{R}^{d \times 5}$ for the first row of the output matrix of the last encoder layer



Fine-tuning

- updating all pre-trained parameters as well as the additional final linear weights via backpropagation
- loss is calculated based on the comparison of true values and the output softmax probabilities
 - loss is optimized for this specific task (predicting the 5 classes)
 - all values of the final encoder output matrix, except the first row, are irrelevant for the loss
 - fine-tuned PLM gradually loses knowledge/ability about/for the pre-training task
 - catastrophic forgetting¹

¹Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem (McCloskey & Cohen, 1989)
DOI: 10.1016/S0079-7421(08)60536-8

True classes

- where do we get the true classes from?
- if we have the true classes, why is it necessary to learn a model to predict those classes?

True classes

- where do we get the true classes from?
- if we have the true classes, why is it necessary to learn a model to predict those classes?
- motivation for the task
 - we do not have the true classes
 - we want to reliably predict those
 - but: how do we know whether these are reliable?

True classes

- where do we get the true classes from?
- if we have the true classes, why is it necessary to learn a model to predict those classes?
- motivation for the task
 - we do not have the true classes
 - we want to reliably predict those
 - but: how do we know whether these are reliable?
- getting somewhat “true” classes by manual annotation

Annotations

- humans are presented with the sentences in a similar format to the one processed by the machine,
- i.e., they are provided with the context before, target sentence, context after
- they get a codebook as guideline on how to annotate sentences
 - can consist of a few sentences up to many pages (ours consists of 4 pages in the short version)
 - tailored for the dataset/application
 - time-consuming and costly to create
- however, no consistent annotations to be expected
- therefore examination of the quality of the annotations

Inter-coder reliability/agreement

- there are tons of possible solutions to this problem
- simple method:
 - the complete dataset is annotated multiple times by different coders
 - assessment of agreement via dedicated scores, e.g., Krippendorff's α
 - aggregation via majority vote
 - again, tons of possibilities to aggregate; but majority vote is still default
- common method:
 - a subset of the data is annotated by two or more coders
 - if the agreement is high, you go on with separated annotation sets
 - the inconsistencies in the subset that was annotated multiple times are resolved by one master coder

Inter-coder reliability/agreement

- there are tons of possible solutions to this problem
- simple method:
 - the complete dataset is annotated multiple times by different coders
 - assessment of agreement via dedicated scores, e.g., Krippendorff's α
 - aggregation via majority vote
 - again, tons of possibilities to aggregate; but majority vote is still default
- common method:
 - a subset of the data is annotated by two or more coders
 - if the agreement is high, you go on with separated annotation sets
 - the inconsistencies in the subset that was annotated multiple times are resolved by one master coder
- again: annotating data is time-consuming and costly

Few-shot learning

- few-shot learning means fine-tuning a PLM using a small amount of data
- why is few-shot learning beneficial?
 - real world scenario: only few annotated data
 - faster to learn (if no necessity to go for more iterations)
- just using small data is few-shot learning, but not promising
- in the following we have a look at some example few-shot methods:
 - pattern-exploiting training (PET) DOI: [10.18653/v1/2021.eacl-main.20](https://doi.org/10.18653/v1/2021.eacl-main.20)
 - SetFit DOI: [10.48550/arXiv.2209.11055](https://doi.org/10.48550/arXiv.2209.11055)
 - T-Few DOI: [10.48550/arXiv.2205.05638](https://doi.org/10.48550/arXiv.2205.05638)

Zero-shot prediction

- in comparison to few-shot learning, zero-shot predictions are generated without any fine-tuning
- one prominent example of zero-shot “learning” (ZSL) is in-context learning (ICL)
 - the input sequence itself contains information which is used to nudge the model
 - ICL is a mixture between ZSL and FSL
 - you provide examples to the system via input

The German GDP increases by 5%.
Paying taxes is painful.

The Wall Street was closed today.
The company made progress in optimizing its profit.

Positive
Negative

Neutral

The German GDP increases by 5%.
Manchester United won the Champions League.

OpenAI launched ChatGPT.
The company made progress in optimizing its profit.

Finance
Sports

Tech

Zero-shot prediction

- in comparison to few-shot learning, zero-shot predictions are generated without any fine-tuning
- one prominent example of zero-shot “learning” (ZSL) is in-context learning (ICL)
 - the input sequence itself contains information which is used to nudge the model
 - ICL is a mixture between ZSL and FSL
 - you provide examples to the system via input

The German GDP increases by 5%.
Paying taxes is painful.

The Wall Street was closed today.
The company made progress in optimizing its profit.

Positive
Negative

Neutral
Positive

The German GDP increases by 5%.
Manchester United won the Champions League.

OpenAI launched ChatGPT.
The company made progress in optimizing its profit.

Finance
Sports

Tech
Finance

9 Advanced Training Methods

- Pattern-Exploiting Training (PET)
- Iterative PET (iPET)
- Sentence Transformer Fine-Tuning (SetFit)
- Parameter-Efficient Fine-Tuning
 - Bottleneck Adapters
 - Compacter
 - Prefix Tuning
 - Low-Rank Adaptation (LoRA)
 - Infused Adapter by Inhibiting and Amplifying Inner Activations (IA)³
- T-FEW
- Quantization
 - Quantized LoRA (QLoRA)
- AdaSent
- PETapter
- Direct Preference Optimization (DPO)

Pattern-exploiting training (PET)

- cloze questions are common task types in elementary school
- PET leverages the language understanding of PLMs to fill gaps in cloze questions
 - gaps are represented by MASK tokens
 - MASK tokens are predicted by PET
 - predictions are mapped to class labels

PET: first example

- $x : \text{Best pizza ever!}, y : +1, y \in \{+1, -1\} = \{\text{positive, negative}\}$
- idea: add pattern, transform input x to
 Best pizza ever! It was MASK.
- PLMs predicts MASK token (probabilities)
- comparison to a set of verbalizers, here, e.g., $\{\text{great, bad}\}$
- selection of more probable verbalizer
- mapping from verbalizers to classes: great $\rightarrow +1$, bad $\rightarrow -1$
- calculation of loss based on classes

PET idea

- imagine these three statements
 - ① This was the best pizza I've ever had.
 - ② You can get better sushi for half the price.
 - ③ Pizza was average. Not worth the price.
- imagine statement 1 and 2 were labeled with l and l'
- what is the correct label for statement 3?

PET idea

- imagine these three statements
 - ① This was the best pizza I've ever had.
 - ② You can get better sushi for half the price.
 - ③ Pizza was average. Not worth the price.
- imagine statement 1 and 2 were labeled with l and l'
- what is the correct label for statement 3?
 - just from these two examples it's not possible to predict this reliably
- what if we add the task description: “Is the following statement about the price?”

PET idea

- imagine these three statements
 - ① This was the best pizza I've ever had.
 - ② You can get better sushi for half the price.
 - ③ Pizza was average. Not worth the price.
- imagine statement 1 and 2 were labeled with I and I'
- what is the correct label for statement 3?
 - just from these two examples it's not possible to predict this reliably
- what if we add the task description: “Is the following statement about the price?”
 - clearly I'
- what if we add the task description: “Is the following statement about pizza?”

PET idea

- imagine these three statements
 - ① This was the best pizza I've ever had.
 - ② You can get better sushi for half the price.
 - ③ Pizza was average. Not worth the price.
- imagine statement 1 and 2 were labeled with l and l'
- what is the correct label for statement 3?
 - just from these two examples it's not possible to predict this reliably
- what if we add the task description: "Is the following statement about the price?"
 - clearly l'
- what if we add the task description: "Is the following statement about pizza?"
 - clearly l
- few-shot classification tasks become much easier incorporating task descriptions

Pattern-verbalizer pair (PVP)

- how to include the task description?
- in the (first) example before
 - pattern: [x] It was [MASK].
 - verbalizer: great → +1, bad → -1
- one example PVP for our Ukraine arms deliveries downstream task
 - pattern: This sentence contains [MASK] [MASK] arms deliveries to Ukraine:
[target_sentence] [SEP] [context_before][SEP] [context_after]
 - verbalizer: argument against, argument for, claim against, claim for, nothing regarding
- multiple MASK tokens are possible
 - but not recommended (later: performance better for more naive PVPs)
 - number of MASK tokens must match the number of tokens of longest verbalizer

Defining PET: notation

- PLM
- $\text{MASK} \in V_{\text{PLM}}$
- \mathcal{L} label set for target classification task,
e.g., {argumentagainst, argumentfor, claimagainst, claimfor, irrelevant}
- pattern $P(x) \in V_{\text{PLM}}^{n^*(x)}$,
e.g., $P(x) = \text{This sentence contains [MASK] [MASK] arms deliveries to Ukraine: } [x]$
 - $n^*(x)$ depends on the input length: $n^*(x) = \min\{n_x + n_{\text{pattern}}, n_{\text{sequence}_{\text{PLM}}}\}$
- injective function $v : \mathcal{L} \rightarrow V^m$ mapping each of the labels to one of the vocabulary tokens,
e.g., $v(\text{irrelevant}) = \text{nothing}$ regarding (here, we have $m = 2$ MASK tokens)
- (P, v) are named PVPs

Defining PET: predicting MASKed tokens

- if $m = 1$:
- $\text{PLM}(w \mid P(x)) \in \mathbb{R}, w \in V, x \in V_{\text{PLM}}^{n^*(x)}$ is the output of the last linear layer of the PLM
- then

$$s(l \mid x) = \text{PLM}(v(l) \mid P(x)) \quad \text{score for label } l \in \mathcal{L}$$

$$q(l \mid x) = \frac{\exp(s(l \mid x))}{\sum_{l' \in \mathcal{L}} \exp(s(l' \mid x))} \quad \text{pseudo-probability for label } l \in \mathcal{L}$$

- if $m > 1$: $s(l \mid x) = \sum_{i=1}^m \text{PLM}(v(l)_i \mid P(x)), \quad v(l) = (v(l)_1, \dots, v(l)_m)$

Defining PET: loss calculation

- cross-entropy loss
 - single observation: $-\log(q(l^*(x) | x))$ with $l^*(x) \in \mathcal{L}$ being the true label of x
 - all training data: $L_{CE} = \sum -\log(q(l^*(x) | x))$
- catastrophic forgetting is an issue here
- to address this, the authors make use of an auxiliary task
 - typically it is not an issue to get hands on large unlabeled in-domain data
 - besides classification fine-tuning, a classical language model objective is learned
 - $P(x)$ is used but without predicting anything for the MASK token(s)
 - L_{MLM} is the loss of the mask language modeling task
- final loss is given by

$$L = (1 - \alpha)L_{CE} + \alpha L_{MLM}$$

- authors suggest $\alpha = 10^{-4}$

Combining PVPs in PET

- due to the nature of few-shot learning, small number of observations
- idea: fine-tune a PLM for several PVPs and combine
- combination: weighted or standard mean of the scores
- calculation of soft distribution

$$p(I | x) = \frac{\exp(s(I | x)/T)}{\sum_{I' \in \mathcal{L}} \exp(s(I' | x)/T)}$$

- temperature $T = 2$ for PET
- then: fine-tuning PLM not with hard labels $I^*(x)$, but using the soft labels and a standard classification head
 - important: here, cross-entropy “transforms” from $-\log(q(I^*(x) | x))$ to $-\sum_{I' \in \mathcal{L}} p(I' | x) \log(q(I' | x))$

Iterative PET (iPET)

- again: fine-tune a PLM for several PVPs
- then: enlarge dataset(s) with predicted labels for unlabeled data using ensemble of fine-tuned PLMs
- use each of the different (enlarged) datasets to further fine-tune the models
- repeat
- in the end, again, fine-tune PLM with classification head using soft labels
- see also following slide

(i)PET illustration

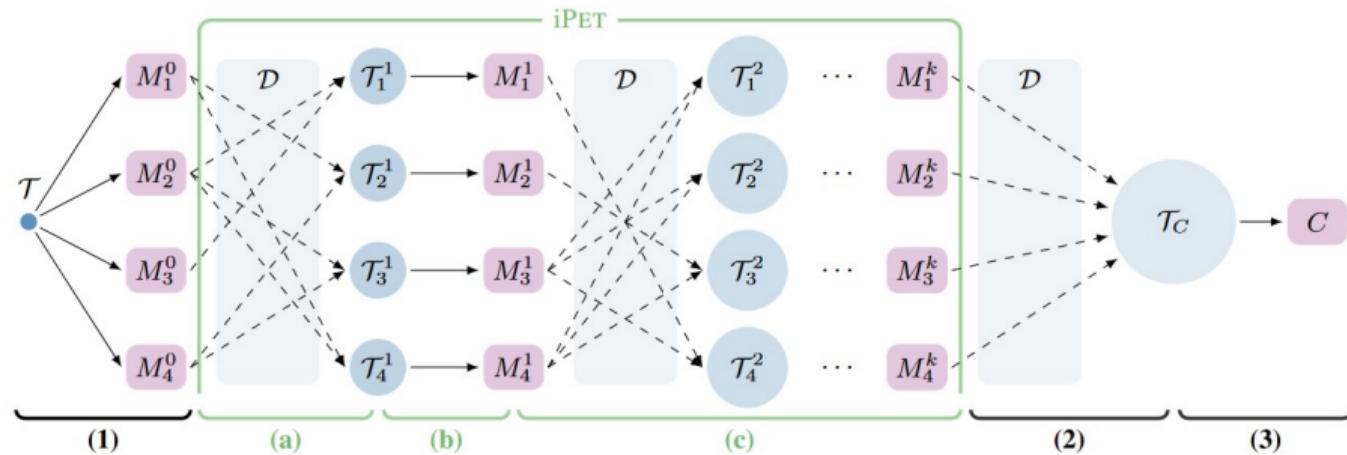


Figure 2: Schematic representation of PET (1-3) and iPET (a-c). **(1)** The initial training set is used to finetune an ensemble of PLMs. **(a)** For each model, a random subset of other models generates a new training set by labeling examples from \mathcal{D} . **(b)** A new set of PET models is trained using the larger, model-specific datasets. **(c)** The previous two steps are repeated k times, each time increasing the size of the generated training sets by a factor of d . **(2)** The final set of models is used to create a soft-labeled dataset \mathcal{T}_C . **(3)** A classifier C is trained on this dataset.

DOI: [10.18653/v1/2021.eacl-main.20](https://doi.org/10.18653/v1/2021.eacl-main.20)

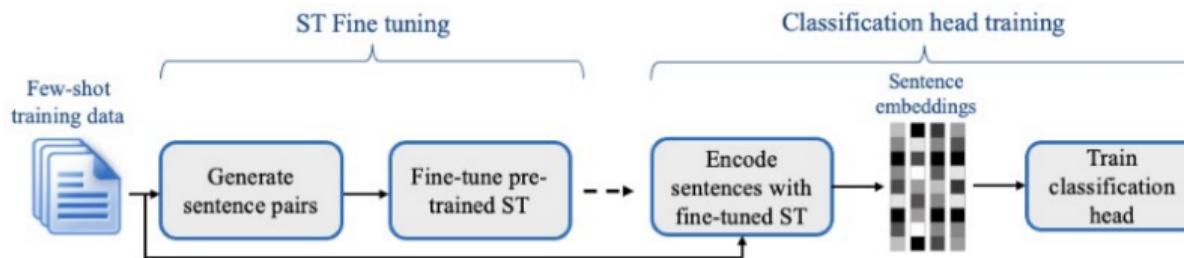
Summary of PET

- few-shot fine-tuning method
- full fine-tuning
- remarks:
 - number of MASK tokens must match the number of tokens of longest verbalizer
 - recommended to use just one MASK token per pattern
 - more naive patterns often work better
 - the format of the pattern is important DOI: [10.48550/arXiv.2111.13440](https://doi.org/10.48550/arXiv.2111.13440)
 - Q&A and prompt patterns work best
 - [x] Question: What is the topic of this article? Answer: [MASK]
 - [MASK] News: [x]
 - more naive patterns often work better DOI: [10.48550/arXiv.2312.16975](https://doi.org/10.48550/arXiv.2312.16975)

Sentence transformer fine-tuning (SetFit)

- few-shot method
- fine-tuning sentence transformer (ST) on small number of text pairs
- prompt-free, no need of verbalizer
- fast to train
- as performant as PET (and PEFT techniques)
- based on SBERT (siamese and triplet network structure)
- aim:
 - minimize distance between similar sentences
 - maximize distance between semantically distant sentences
- 2 step approach
 - ① fine-tuning ST using sentence pairs
 - ② classification head training

SetFit



- $(x_i, l * (x_i))$ represents one observation of the few-shot dataset
- generation/sampling of R positive triplets $(x_i, x_j, 1) : l^*(x_i) = l^*(x_j)$
- generation/sampling of R negative triplets $(x_i, x_j, 0) : l^*(x_i) \neq l^*(x_j)$
- authors suggest $R = 20$
- fine-tuning the ST using contrastive learning (cf., SBERT)
- the encoded embedding of the ST is then used for training a standard classification head
 - the authors use a logistic regression (linear layer + sigmoid)

Disadvantages of full fine-tuning

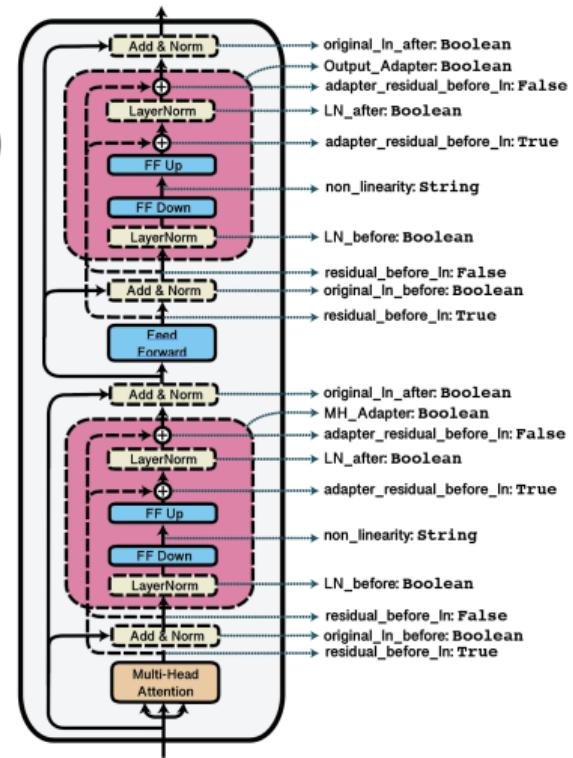
- all parameters are fine-tuned/trained/updated
 - requires a lot of disk space to save
 - requires a lot of cloud space to share/distribute
- takes a lot of time
 - + energy consumption
- requires large computing resources
 - + energy consumption
- takes a lot of time during inference (to predict)
 - + energy consumption

PEFT as efficient alternative to full fine-tuning¹

- add layers to the transformers architecture
 - freeze parameters from pre-trained language model (PLM)
 - train/update added layer
 - task-specific layers
 - modularity
-
- often as good as full fine-tuning
 - small in size (e.g., 3.3MB vs. 440MB)
 - faster to train

¹<https://adapterhub.ml/>

https://www.youtube.com/watch?v=_Z9qNT-g14U

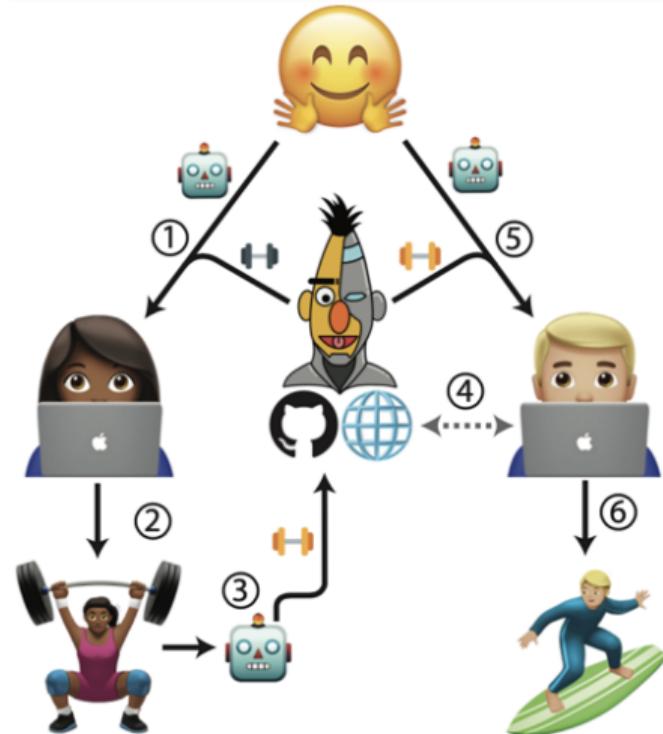


Adapterhub¹/Adapters² lifecycle

- ① get PLM from huggingface
- ② add adapters and train on downstream task
- ③ upload trained adapters to hub
- ④ download trained adapters from hub
- ⑤ get PLM from huggingface
- ⑥ do inference

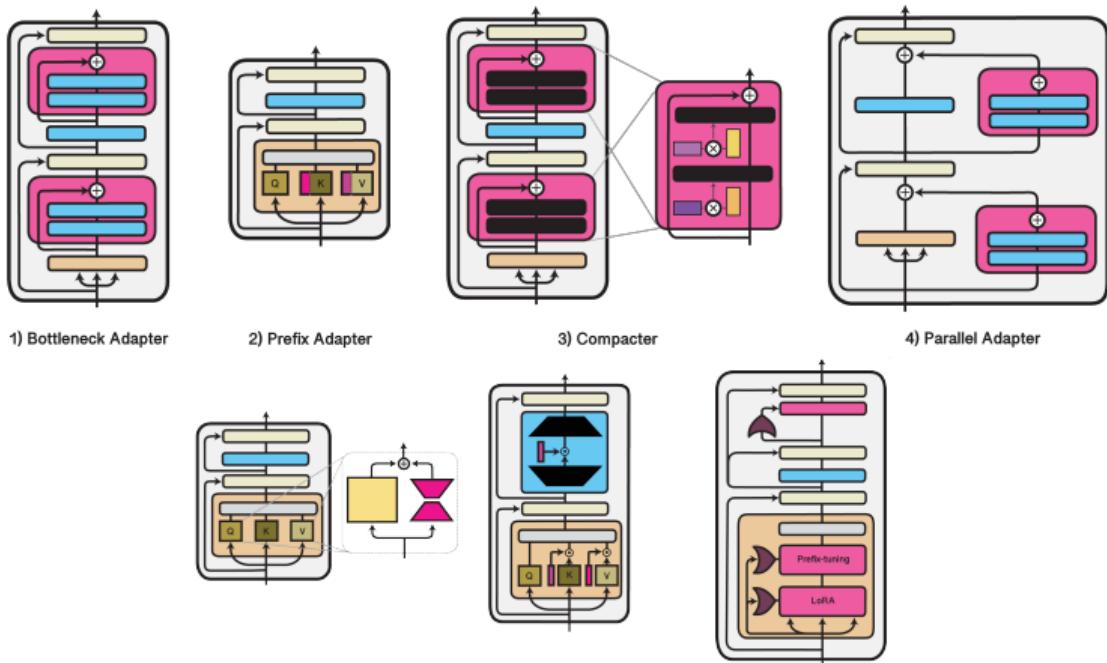
¹DOI: 10.18653/v1/2020.emnlp-demos.7

²DOI: 10.18653/v1/2023.emnlp-demo.13



Adapter architectures

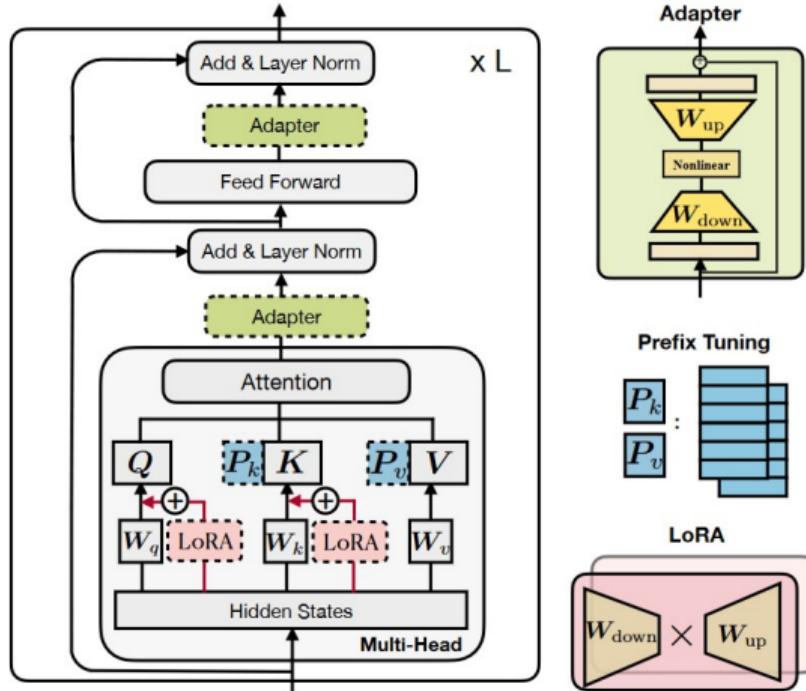
- first generation/idea of adapters was invented by Houlsby et al. (2019)¹
- more effective implementation by Pfeiffer et al. (2020)
- very dynamic research field
 - prefix tuning, LoRA, (IA)³, ...



¹<http://proceedings.mlr.press/v97/houlsby19a.html>

PEFT methods

- we will have a closer look at
 - adapters (and compacters)
 - prefix tuning
 - LoRA (and (IA)³)



<https://openreview.net/forum?id=0RDcd5Axok>

Elements of Transformer architecture (recap)

- SHA: $Z = \text{Attn}(Q, K, V) = \text{softmax}(QK^T / \sqrt{d_k})V$
 - $Q \in \mathbb{R}^{n \times d_k}, K \in \mathbb{R}^{m \times d_k}, V \in \mathbb{R}^{m \times d_v}$
 - $X \in \mathbb{R}^{n \times d}$
 - $W^Q \in \mathbb{R}^{d \times d_k}, W^K \in \mathbb{R}^{d \times d_k}, W^V \in \mathbb{R}^{d \times d_v}$
 - n sequence length of queries, m sequence length of key-values pairs
- MHA: $Z = (Z_1, \dots, Z_h)W^0$
 - $Z_i = \text{Attn}(xW_i^Q, CW_i^K, CW_i^V)$
 - $x \in \mathbb{R}^d, C \in \mathbb{R}^{m \times d}$
 - $W^0 \in \mathbb{R}^{d \times d}$ with (typically) $d_h = d/h$
 - $W_i^Q \in \mathbb{R}^{d \times d_h}, W_i^K \in \mathbb{R}^{d \times d_h}, W_i^V \in \mathbb{R}^{d \times d_h}$
- FFNN(x) = $\text{ReLU}(xW_1 + b_1)W_2 + b_2$
 - $W_1 \in \mathbb{R}^{d \times d_m}, W_2 \in \mathbb{R}^{d_m \times d}$
 - typically $d_m = 4d$

(Bottleneck) adapters

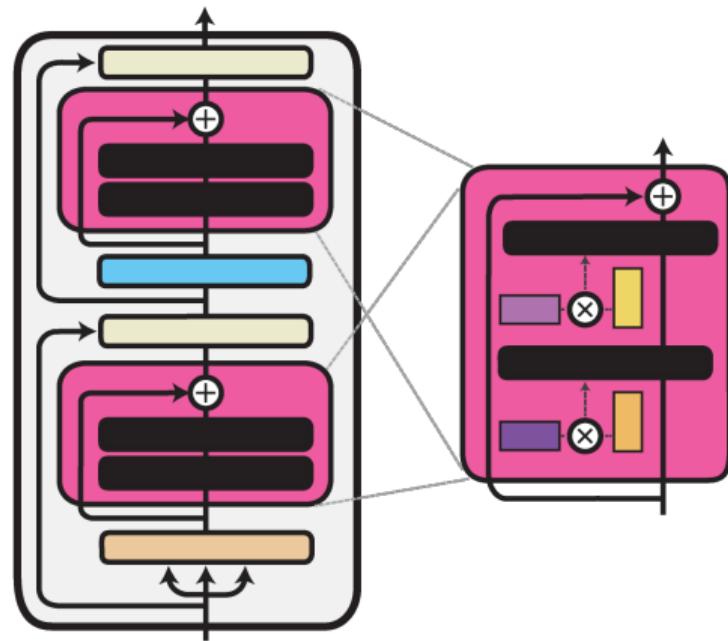
- $W_{\text{down}} \in \mathbb{R}^{d \times r}$, $W_{\text{up}} \in \mathbb{R}^{r \times d}$
- $r < d$ bottleneck dimension (hyperparameter)
- f nonlinear activation function
- (standard) residual connection (add & norm)

$$h \leftarrow h + f(hW_{\text{down}})W_{\text{up}}$$

- Houlsby et al. (2019): two (sequential) layers
 - ① after attention layer
 - ② after FFNN
- Pfeiffer et al. (2020): only after FFNN layer

Compacter

- parameterized hypercomplex multiplication (PHM) layers
- very similar to bottleneck adapters, but even more parameter efficient
- weight matrices are learned from Kronecker products
- Mahabadi et al. (2021):
<https://openreview.net/forum?id=bqGK5PyI6-N>



<https://adapterhub.ml/>

Prefix tuning

- adding / tunable prefix vectors to keys and values of MHA
- $P^K, P^V \in \mathbb{R}^{I \times d}$
- $P_i^K, P_i^V \in \mathbb{R}^{I \times d_h}$
- concatenation

$$Z_i = \text{Attn}(xW_i^Q, (P_i^K, CW_i^K), (P_i^V, CW_i^V))$$

- Li & Liang (2021): DOI: [10.18653/v1/2021.acl-long.353](https://doi.org/10.18653/v1/2021.acl-long.353)

Prefix tuning as an adapter

- prefix tuning can be rewritten to an adapter-like format
- $W_1 = W^Q P^{K^T} \in \mathbb{R}^{d_h \times I}$, $W_2 = P^V \in \mathbb{R}^{I \times d_h}$
 - if $I < d$, we have a down- and up-projection as for adapters
- $f = \text{softmax}$

$$h \leftarrow (1 - \lambda(x))h + \lambda(x)f(xW_1)W_2$$

- here: weighted addition; adapters typically unweighted
- here: parallel adaption; adapters typically sequential
- here: modification of attention heads; adapters can be inserted after attention or FFNN
- here: addition to each MHA head; adapters single-headed
 - full rank update with $I = d_h$; for adapters with $r = d$
 - for $I = r$ prefix tuning does not add more parameters than adapters

LoRA (low-rank adaptation)

- injecting additive low-rank matrices to mimic weight updates ΔW
- $W + \Delta W = W + W_{\text{down}} W_{\text{up}}$
- $W_{\text{down}} \in \mathbb{R}^{d \times r}$, $W_{\text{up}} \in \mathbb{R}^{r \times k}$, $k \neq r$
- $W \in \mathbb{R}^{d \times k}$ the pre-trained weight-matrix
- $s \geq 1$ scaling scalar (hyperparameter)

$$h \leftarrow h + s W_{\text{down}} W_{\text{up}}$$

- Hu et al. (2022): <https://openreview.net/forum?id=nZeVKeeFYf9>
- update is applied to projection matrices W^Q and W^V in MHA

LoRA advantages

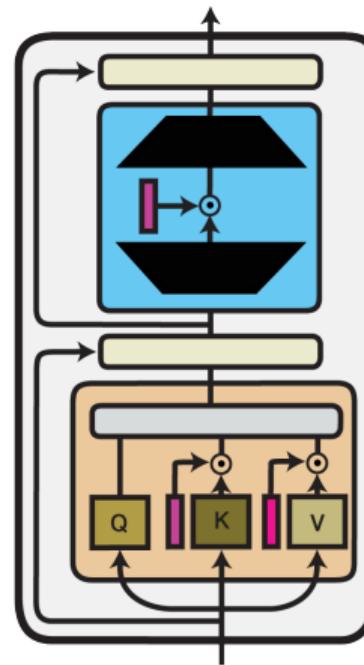
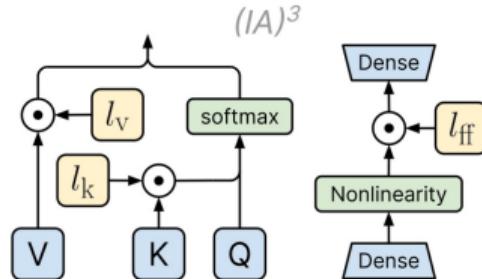
- there is no initial manipulation needed before training
 - we can choose $\Delta W = W_{\text{down}} W_{\text{up}} = 0^{d \times k}$
 - random Gaussian initialization for W_{up}
 - $W_{\text{down}} = 0^{d \times k}$
- there is no additional inference latency coming from the additional parameters
 - after training, we can save/store both
 - the manipulated weight matrix $W + \Delta W$
 - the updates W_{down} and W_{up}
 - little memory overhead
 - W can be recovered (for further training)

(IA)³ (infused adapter by inhibiting and amplifying inner activations)

- element-wise rescaling of inner model activations
- trainable vectors $l_k \in \mathbb{R}^{d_k}$, $l_v \in \mathbb{R}^{d_v}$, $l_{ff} \in \mathbb{R}^{d_m}$

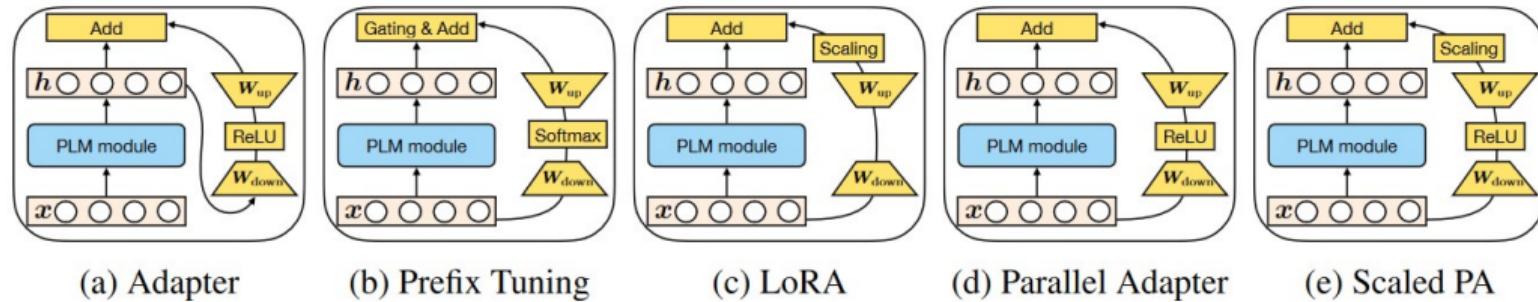
$$h \leftarrow l \odot hW$$

- in the original paper K , V and the FFNN are manipulated
- used by T-FEW
- Liu al. (2022): <https://openreview.net/forum?id=rBCvMG-JsPd>



<https://adapterhub.ml/>

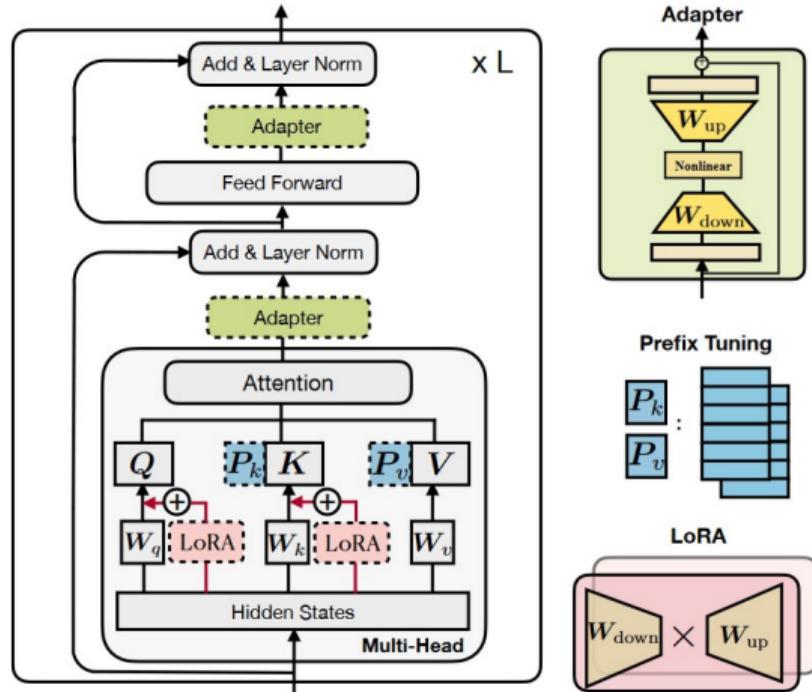
PEFT overview (selected)



method	format	modification	Δh	function
adapter	sequential	FFNN, Attn	$\text{ReLU}(hW_{\text{down}})W_{\text{up}}$	$h \leftarrow h + \Delta h$
parallel adapter	parallel	FFNN, Attn	$\text{ReLU}(xW_{\text{down}})W_{\text{up}}$	$h \leftarrow h + \Delta h$
multi-head parallel adapter	parallel	Q, K, V	$\text{ReLU}(xW_{\text{down}})W_{\text{up}}$	$h \leftarrow h + \Delta h$
scaled parallel adapter	parallel	FFNN, Attn	$\text{ReLU}(xW_{\text{down}})W_{\text{up}}$	$h \leftarrow h + s\Delta h$
LoRA	parallel	$W^Q, (W^K), W^V$	$xW_{\text{down}}W_{\text{up}}$	$h \leftarrow h + s\Delta h$
prefix tuning	parallel	K, V	$\text{softmax}(xW^Q P^{K^T})P^V$	$h \leftarrow (1 - \lambda)h + \lambda\Delta h$

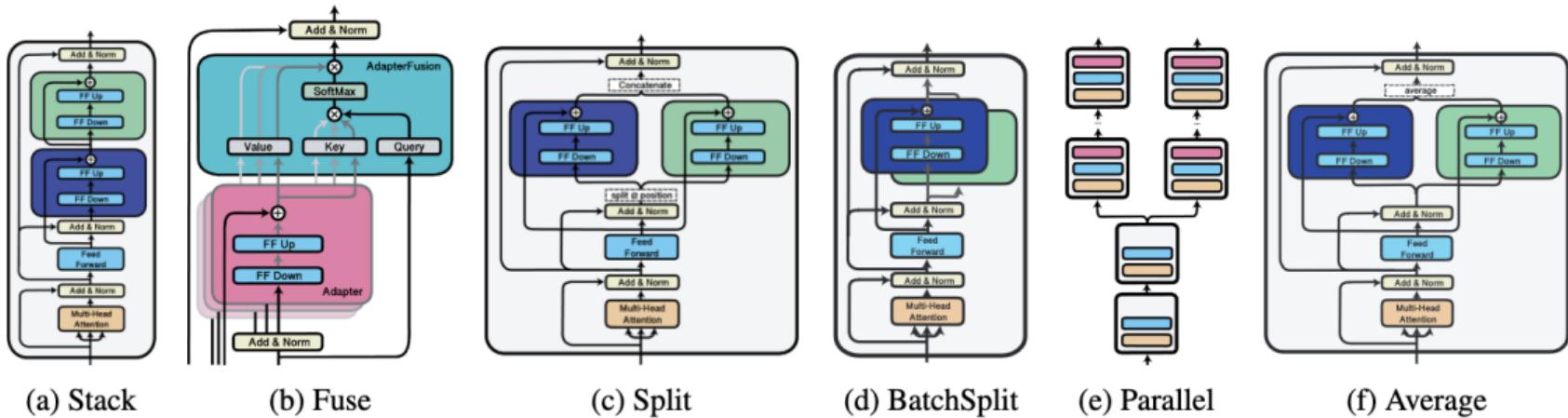
PEFT methods (recap)

- most PEFT methods can be somewhat interpreted as an adaptive structure with different design decisions
- most common (personal feeling):
 - bottleneck adapters
 - LoRA
- confusion in the illustration on the right:
 - LoRA is applied to W^Q , W^V by authors



<https://openreview.net/forum?id=0RDcd5Axok>

Adapter composition



- highly recommend Adapters library in Python
- e.g., LoRA + prefix tuning + adapter is referred to as mix-and-match (MAM) adapters
- UniPELT is one example for MAM DOI: [10.18653/v1/2022.acl-long.433](https://doi.org/10.18653/v1/2022.acl-long.433)

Modular deep learning

further reading

- <https://openreview.net/forum?id=0RDcd5Axok>
- DOI: 10.48550/arXiv.2312.12148
- <https://www.ruder.io/modular-deep-learning/>

T-FEW

- SOTA performance on RAFT benchmark
- recipe of different few-shot, PEFT and regularization ingredients
 - based on T0, a zero-shot variant of T5
 - summation of three losses
 - standard cross-entropy
 - unlikelihood loss
 - length normalization loss
 - (IA)³ as PEFT method
 - pre-training of (IA)³

- RAFT is a set of 11 few-shot datasets
- each consisting of 50 labeled examples
- 150–5,000 labeled examples for testing
- 2–77 classes

rank	model	overall accuracy
4	T-FEW	0.758
8	Human baseline	0.735
15	SetFit300	0.713
22	PET	0.696
25	SetFit	0.669

<https://raft.elicit.org/>

QLoRA¹ (quantized LoRA)

- regular FP16 fine-tuning of LLaMA (65B) requires > 780 GB GPU
- QLoRA enables fine-tuning of 65B models on a single 48GB GPU
 - preserving full 16-bit fine-tuning performance
- combination of 4-bit quantized PLMs and LoRA
- authors propose new data type NF4 (4-bit NormalFloat) optimal for \mathcal{N} weights
- usage of double quantization, i.e., quantizing quantization constants

¹<https://openreview.net/forum?id=OUIFPHEgJU>

FP64, FP32, minifloats (e.g., FP16, FP8, FP4)

- FP64 = double (double precision)
 - one bit for sign, 11 bits for exponent, 52 bits for fraction
 - machine epsilon: $2^{-52} = 2.22\text{e-}16 \rightarrow 16$ significant digits

$$(-1)^s \left(1 + \sum_{i=1}^{52} f_{52-i} 2^{-i} \right) 2^{e-1023}$$

- FP32 = float (single precision)
 - 1/8/23 bits for sign/exponent/fraction $\rightarrow 7$ significant digits
- minifloats
 - FP16 (half precision): 1/5/10 bits for sign/exponent/fraction $\rightarrow 4$ significant digits
 - BF16: 1/8/7 bits for sign/exponent/fraction $\rightarrow 3$ significant digits
 - FP8: different formats (DOI: 10.48550/arXiv.2209.05433), e.g., 1/4/3 or 1/5/2

cf. <https://docs.nvidia.com/deeplearning/performance/mixed-precision-training/index.html>

(Block-wise k-bit) quantization

- transformation: more informative → less informative
- e.g., FP32 \rightarrow Int8 $\in \{-127, -126, \dots, 127\}$ (no significant digits)

$$X^{\text{Int8}} = \text{round} \left(\frac{127}{\text{absmax}(X^{\text{FP32}})} X^{\text{FP32}} \right) = \text{round}(c^{\text{FP32}} X^{\text{FP32}})$$

- c quantization constant
- dequantization (inverse):

$$X^{\text{FP32}} = \text{dequant}(c^{\text{FP32}}, X^{\text{Int8}}) = \frac{X^{\text{Int8}}}{c^{\text{FP32}}}$$

(Block-wise k-bit) quantization

- transformation: more informative → less informative
- e.g., FP32 \rightarrow Int8 $\in \{-127, -126, \dots, 127\}$ (no significant digits)

$$X^{\text{Int8}} = \text{round} \left(\frac{127}{\text{absmax}(X^{\text{FP32}})} X^{\text{FP32}} \right) = \text{round}(c^{\text{FP32}} X^{\text{FP32}})$$

- c quantization constant
- dequantization (inverse):

$$X^{\text{FP32}} = \text{dequant}(c^{\text{FP32}}, X^{\text{Int8}}) = \frac{X^{\text{Int8}}}{c^{\text{FP32}}}$$

- what if: outliers?

(Block-wise k-bit) quantization

- transformation: more informative → less informative
- e.g., FP32 \rightarrow Int8 $\in \{-127, -126, \dots, 127\}$ (no significant digits)

$$X^{\text{Int8}} = \text{round} \left(\frac{127}{\text{absmax}(X^{\text{FP32}})} X^{\text{FP32}} \right) = \text{round}(c^{\text{FP32}} X^{\text{FP32}})$$

- c quantization constant
- dequantization (inverse):

$$X^{\text{FP32}} = \text{dequant}(c^{\text{FP32}}, X^{\text{Int8}}) = \frac{X^{\text{Int8}}}{c^{\text{FP32}}}$$

- what if: outliers?
 - sparsely populated bins
- what to do?

(Block-wise k-bit) quantization

- transformation: more informative → less informative
- e.g., FP32 \rightarrow Int8 $\in \{-127, -126, \dots, 127\}$ (no significant digits)

$$X^{\text{Int8}} = \text{round} \left(\frac{127}{\text{absmax}(X^{\text{FP32}})} X^{\text{FP32}} \right) = \text{round}(c^{\text{FP32}} X^{\text{FP32}})$$

- c quantization constant
- dequantization (inverse):

$$X^{\text{FP32}} = \text{dequant}(c^{\text{FP32}}, X^{\text{Int8}}) = \frac{X^{\text{Int8}}}{c^{\text{FP32}}}$$

- what if: outliers?
 - sparsely populated bins
- what to do?
 - chunk the input and apply n independent quantizations with constants c_i

NF4 (4-bit NormalFloat quantization)

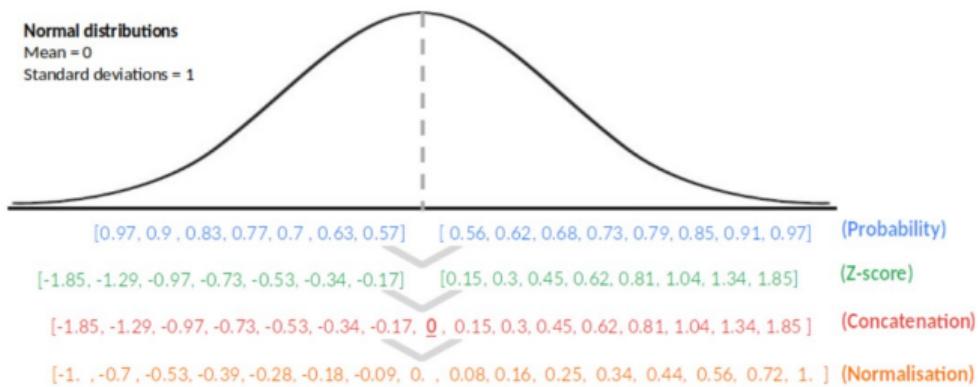
- quantile quantization via ecdf is expensive
 - hidden unit weights of NN are normally distributed → assume \mathcal{N}
 - transformation to one specific fixed distribution with range $[-1, 1]$
- ① estimate $2^k (= 16)$ quantiles straight from $\mathcal{N}(0, 1)$

$$q_i = \frac{1}{2} \left(Q_X \left(\frac{i}{2^k + 1} \right) + Q_X \left(\frac{i + 1}{2^k + 1} \right) \right)$$

- I was not able to reproduce the q_i from the following slide using the above formula from the paper. Please let me know, if you find the formula to create the probability values on the following slide.

- ② normalization to $[-1, 1]$
- ③ quantization as usual

NF4 generation



Steps for generating the NF4 data type values:

1. Generate 8 evenly spaced values from 0.56 to 0.97 (Set I).
2. Generate 7 evenly spaced values from 0.57 to 0.97 (Set II).
3. Calculate the z-score values for the probabilities generated in Step 1 and Step 2. For Set II, calculate the negative inverse of the z-scores.
4. Concatenate Set I, a zero value, and Set II together.
5. Normalize the values by dividing them by the absolute maximum value.

Figure 7: Steps required to construct the NF4 data type. Equidistant values of the probability density functions are converted to Z-scores through the quantile function. To ensure the usage of all 16 values the distribution is asymmetric. As such, one half of the normal distribution has 8 and the other 7 probability values. A zero is included to have a discrete zero point. All z-scores and zero are concatenated and then normalized into the range [-1, 1] to receive the final NF4 data type.

NF4 values

```
[ -1.0, -0.6961928009986877, -0.5250730514526367,  
-0.39491748809814453, -0.28444138169288635, -0.18477343022823334,  
-0.09105003625154495, 0.0, 0.07958029955625534,  
0.16093020141124725, 0.24611230194568634, 0.33791524171829224,  
0.44070982933044434, 0.5626170039176941, 0.7229568362236023, 1.0]
```

Double quantization (first quantization)

$$W^{\text{NF4}} = \frac{W^{\text{FP32}}}{c_2^{\text{FP32}}} \quad \text{with } c_2^{\text{FP32}} = \text{absmax}(W^{\text{FP32}})^{-1}$$

- e.g., $W^{\text{FP32}} \in \mathbb{R}^{b \times h}$
- with block size $bh/B_W = 64$ we have $c_2^{\text{FP32}} \in \mathbb{R}^{B_W}$ with $B_W = bh/64$
- each block i has its values in $[-1, 1]^{64}$ with $c_{2,i}^{\text{FP32}} = \text{absmax}((W^{\text{FP32}})_{(64i+1, \dots, 64(i+1))})^{-1}$
- c_2^{FP32} is 32-bit \rightarrow single quantization adds $32/64 = 0.5$ bits per parameter

Double quantization (second quantization)

$$c_2^{\text{FP8}} = \frac{c_2^{\text{FP32}} - \overline{c_2^{\text{FP32}}}}{c_1^{\text{FP32}}} \quad \text{with } c_1^{\text{FP32}} = \text{absmax}(c_2^{\text{FP32}} - \overline{c_2^{\text{FP32}}})^{-1}$$

- $c_2^{\text{FP32}} \in \mathbb{R}_+^{B_W}$
- with block size $B_W/B_c = 256$ we have $c_1^{\text{FP32}} \in \mathbb{R}^{B_c}$ with $B_c = B_W/256$
- each block i has its values in $\left[-\overline{c_2^{\text{FP32}}}, \approx \overline{c_2^{\text{FP32}}}\right]^{256}$
- quantizing the quantization constants (saves memory)
- c_2^{FP8} is 8-bit \rightarrow double quantization adds $8/64 + 32/(64 \cdot 256) = 0.127$ bits per parameter

QLoRA

- LoRA:

$$Y = XW + sXW_{\text{down}}W_{\text{up}}$$
$$X \in \mathbb{R}^{n \times d}, W \in \mathbb{R}^{d \times k}, W_{\text{down}} \in \mathbb{R}^{d \times r}, W_{\text{up}} \in \mathbb{R}^{r \times k}, s \in \mathbb{R}$$

- QLoRA:

$$Y^{\text{BF16}} = X^{\text{BF16}}W^{\text{BF16}} + X^{\text{BF16}}W_{\text{down}}^{\text{BF16}}W_{\text{up}}^{\text{BF16}}$$
$$W^{\text{BF16}} = \text{doubleDequant}(c_1^{\text{FP32}}, c_2^{\text{FP8}}, W^{\text{NF4}}) = \text{dequant}(\text{dequant}(c_1^{\text{FP32}}, c_2^{\text{FP8}}), W^{\text{NF4}})$$

- NF4 for W with $B_W = 64$ for higher precision
- FP8 for c_2 with $B_{c_2} = 256$ to save memory
- NF4 as storage data type, BF16 as computation data type
 - dequantize NF4 to BF16 for forward/backward pass and computation of weight gradients

QLoRA conclusions

- NF4 yields better performance than FP4 (1/2/1 or 1/3/0) or Int4
- 4-bit QLoRA using NF4 data types match 16-bit full fine-tuning performance
- 97.8% performance relative to ChatGPT (judged by GPT-4) using a single consumer GPU in a 12h fine-tuning
- 99.3% for professional GPU in a 24h fine-tuning
- 7B models can run on mobile phones
- authors claim that an iPhone 12 Plus can fine-tune 3 million tokens over night (charging) using QLoRA
- authors claim that this methods reduces environmental footprint by 72%

QLoRA conclusions

- NF4 yields better performance than FP4 (1/2/1 or 1/3/0) or Int4
- 4-bit QLoRA using NF4 data types match 16-bit full fine-tuning performance
- 97.8% performance relative to ChatGPT (judged by GPT-4) using a single consumer GPU in a 12h fine-tuning
- 99.3% for professional GPU in a 24h fine-tuning
- 7B models can run on mobile phones
- authors claim that an iPhone 12 Plus can fine-tune 3 million tokens over night (charging) using QLoRA
- authors claim that this methods reduces environmental footprint by 72%

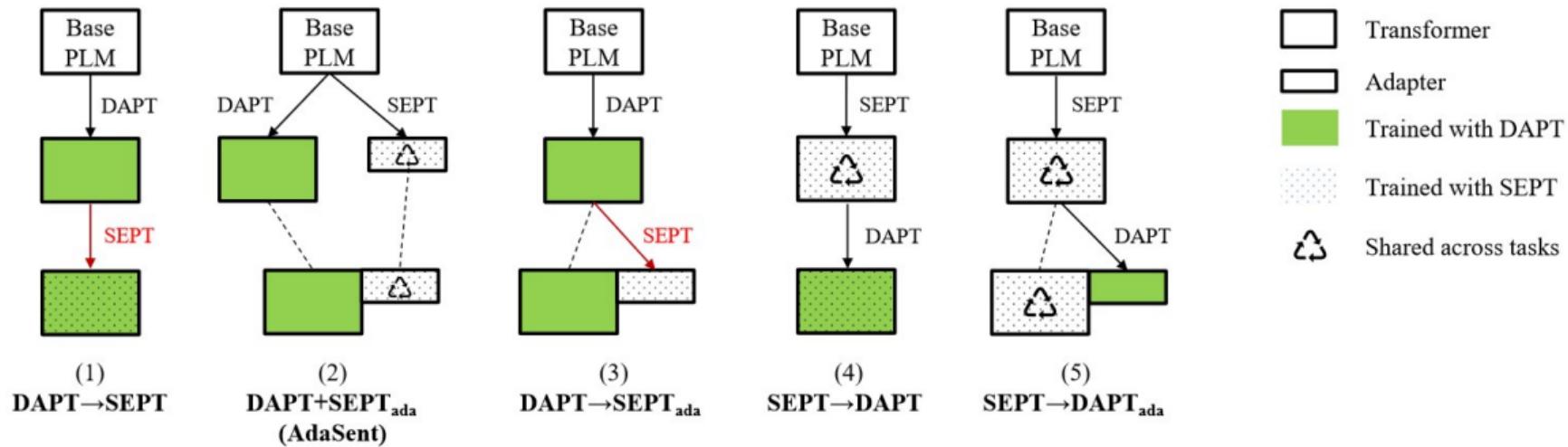
additional resources: <https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-quantization> for a visual introduction/guide to quantization (the whole page from Marteen Grootendorst, author of BERTopic, and also the one from Jay Alammar are recommended! check the literature and reference list on one of the first slides)

AdaSent¹

- domain-adaptive pre-training (DAPT) of PLM (i.e., not a ST)
- sentence embedding pre-training (SEPT) of PLM (as ST)
- no full fine-tuning, but leveraging adapters
- DAPT and SEPT can be executed independently
 - modularity saves resources
 - SEPT procedure is cross-domain
 - DEPT procedure “contains” domain knowledge
- applying, e.g., SetFit as downstream classification few-shot fine-tuning

¹DOI: 10.18653/v1/2023.emnlp-main.208

Variants of combining DAPT and SEPT



- AdaSent SEPT adapter can be reused with other DAPTed PLMs

AdaSent benchmark results |

Row No.	Model	AC	BANK	AMI	AMS	MI	MD	EMO	IMDB	TSE	TC	ARM	Avg.
<i>No SEPT</i>													
R1	Base	65.9	75.7	62.0	71.0	72.8	89.4	40.2	67.7	50.9	55.2	37.3	62.6
R2	DAPT	69.4	80.4	70.0	79.4	80.6	94.7	37.3	74.9	55.1	48.2	41.9	66.5
<i>Full SEPT</i>													
R3	SEPT (prev. SOTA)	76.1	77.0	66.8	73.3	78.4	90.6	52.2	84.8	63.2	63.6	44.2	70.0
R4	DAPT→SEPT	73.8	80.9	73.5	79.2	83.7	94.5	54.0	86.2	63.7	63.6	46.9	72.7
<i>SEPT on Adapter</i>													
R5	SEPT _{ada}	76.0	76.9	66.0	73.6	79.4	91.4	55.3	84.6	63.1	65.4	43.8	70.5
R6	DAPT→SEPT _{ada}	77.9	80.7	73.8	79.3	82.9	94.7	54.7	85.6	65.0	65.5	47.0	73.4
R7	AdaSent	77.9	80.6 [†]	73.7 [†]	80.5[†]	82.7 [†]	95.4[†]	54.1	86.7	65.2	63.0	48.1[†]	73.4
<i>DAPT on Adapter</i>													
R8	SEPT→DAPT _{ada}	72.8	79.5	69.8	78.0	81.0	93.7	48.3	84.4	59.2	59.4	44.9	70.1

Table 3: Classification accuracy on the MTEB classification tasks. Full SEPT means tuning all the PLM parameters in the sentence embedding pre-training. Best results on each dataset are in **bold**. [†] marks the cases where AdaSent outperforms SEPT (R5) with a statistical significance level of 0.05.

AdaSent benchmark results II

Row No.	Model	FPB	TFNS	TFNT	ADE	RCT	LED	Avg.
<i>No SEPT</i>								
R1	Base	49.2	51.1	57.7	60.7	49.6	64.2	55.4
R2	DAPT	50.3	56.3	64.8	67.8	57.8	66.7	60.6
<i>Full SEPT</i>								
R3	SEPT	63.0	65.0	62.2	62.3	61.5	65.6	63.3
R4	DAPT→SEPT	65.6	69.4	68.4	67.4	66.5	68.1	67.6
<i>SEPT on Adapter</i>								
R5	SEPT _{ada}	64.2	66.1	61.4	62.8	58.7	65.9	63.2
R6	DAPT→SEPT _{ada}	66.1	69.9	68.5	65.8	67.4	68.0	67.6
R7	AdaSent	66.4	69.8	68.6 [†]	67.8	67.5	67.8 [†]	68.0
<i>DAPT on Adapter</i>								
R8	SEPT→DAPT _{ada}	62.7	62.0	65.2	66.7	64.0	66.8	64.6

Table 4: Classification accuracy on the domain-specific datasets. Best results on each dataset are in **bold**. [†] marks the cases where AdaSent outperforms SEPT (R5) with a statistical significance level of 0.05.

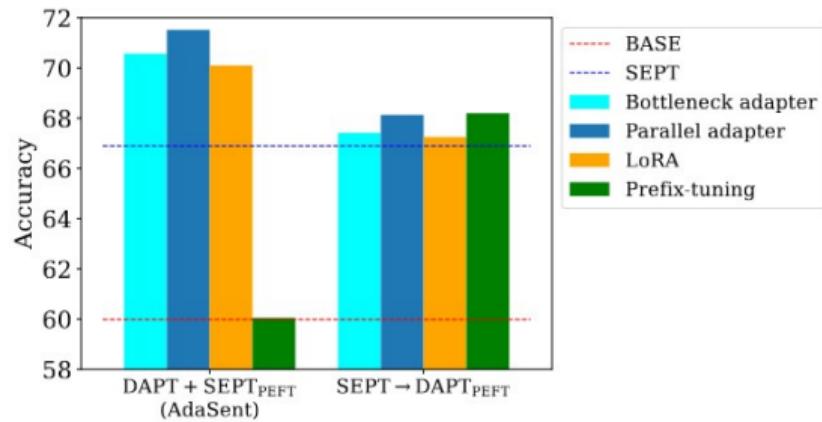
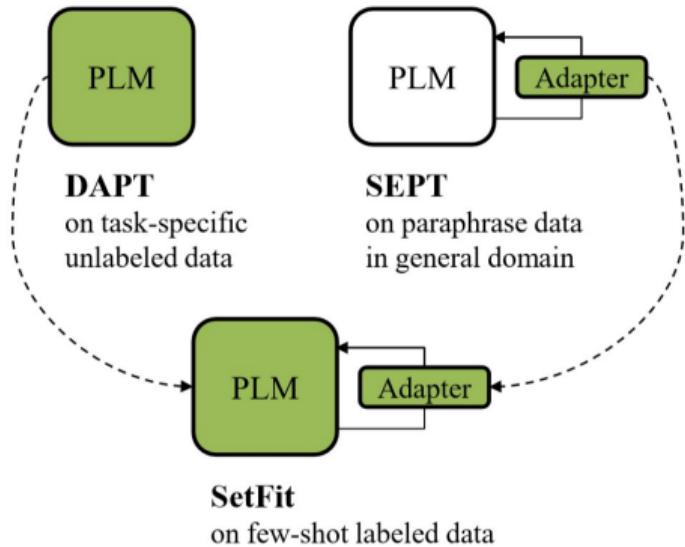


Figure 4: Averaged accuracy of different PEFT methods. SEPT_{PEFT} stands for SEPT on a PEFT module. More detailed results are available in Table 13.

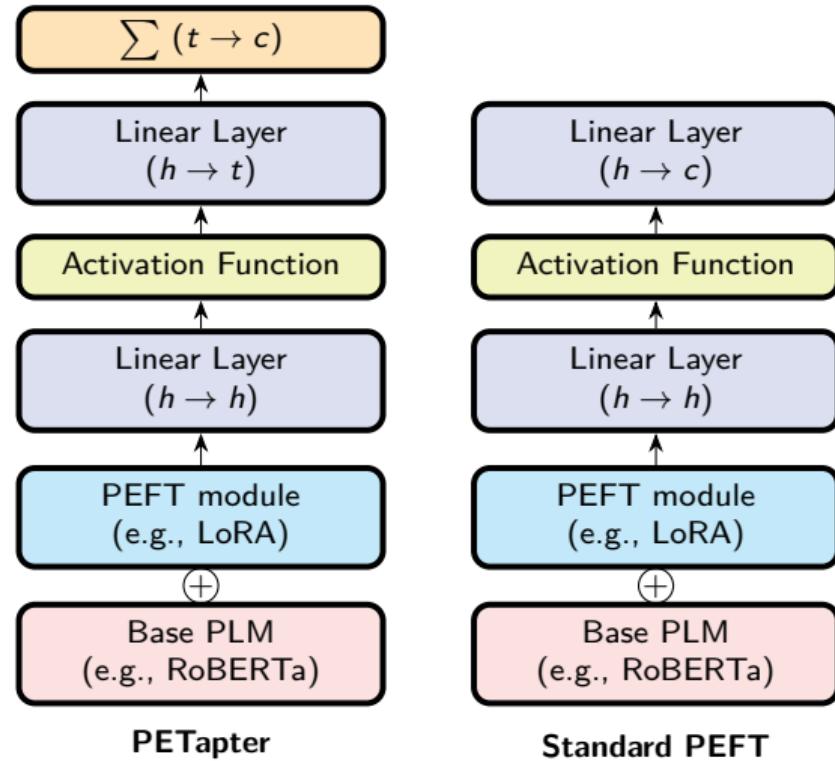
AdaSent conclusion

- DAPTeD PLM can be reused for specific domain
- SEPT adapter can be reused for all domains
- SetFit is applied as full fine-tuning
 - only updating the SEPT adapter is not sufficient



PETapter

- PEFT module with standard (linear) classification layer learn
 - one layer with $h \rightarrow h$ (hidden layer size)
 - one layer with $h \rightarrow c$ (number of classes)
 - and a non-linear act. function in between
- PETapter
 - modifies the final layer to $h \rightarrow t$ (t is the number of vocabulary tokens necessary to represent all verbalizers)
 - sums the likelihoods of the verbalizer tokens per class label
 - can be combined with classical PEFT modules



DOI: 10.48550/arXiv.2412.04975

Benchmark datasets: AG, Yahoo, Yelp

- all datasets in English
- **AG's News** consist of news articles of four distinct topics
 - World, Sports, Business, Sci/Tech
 - 120,000 training, 7600 test
- **Yahoo Questions** consist of discussions on platform Yahoo of ten distinct topics
 - 1,400,000 training, 60,000 test
- **Yelp Full** consist of restaurant reviews on Yelp on a five star scale
 - 650,000 training, 50,000 test

Benchmark datasets: AG, Yahoo, Yelp

- all datasets in English
- **AG's News** consist of news articles of four distinct topics
 - World, Sports, Business, Sci/Tech
 - 120,000 training, 7600 test
- **Yahoo Questions** consist of discussions on platform Yahoo of ten distinct topics
 - 1,400,000 training, 60,000 test
- **Yelp Full** consist of restaurant reviews on Yelp on a five star scale
 - 650,000 training, 50,000 test
- we always use the whole test dataset for evaluation
- we only use a few (maximum 100) observations for training
 - we created five different stratified datasets with $n = 10$
 - and five datasets with $n = 100$
 - for each of the three benchmark datasets

Benchmark datasets PVPs

AG's News

- prompt pattern: [MASK] News: [text]
- Q&A pattern: [text] [SEP] Question: What is the topic of this article? Answer: [MASK].
- verbalizers: World, Sports, Business, Tech (for Sci/Tech)

Yahoo Questions

- prompt pattern: [MASK] Question: [question_title] [question_content] [best_answer]
- Q&A pattern: [question_title] [question_content] [best_answer] [SEP] Question: What is the topic of this question? Answer: [MASK].
- verbalizers: Society, Science, Health, Education, Computer, Sports, Business, Entertainment, Relationship, Politics (for, e.g., Science & Mathematics)

Yelp Full

- prompt pattern: [text] [SEP] All in all, it was [MASK].
- Q&A pattern: [text] [SEP] Question: What does the customer think of this restaurant? Answer: [MASK].
- verbalizers: terrible, bad, okay, good, great (for 1 to 5 stars)

Our dataset: Ukraine arms deliveries

- 14,697 German news articles from 145 national newspapers
- “all” articles about arms deliveries to Ukraine ($\alpha = 90\%$, 79% precision, 86% recall)
- dataset was unlabeled
 - we manually labeled a subset
 - two-stage sampling (article, sentence) to prevent for overoptimistic evaluation
 - 766 train (from 294 articles)
 - 1000 test (from 369 articles)

DOI: 10.5771/2192-4007-2024-1-72

Expected label distribution using random selection

Label	Train				Test
	10	100	250	All	
argumentagainst	1.2	11.8	29.4	92	118
argumentfor	1.9	19.4	48.6	152	162
claimagainst	2.4	23.5	58.8	184	248
claimfor	4.6	45.2	113.2	354	456

- we created, again, stratified datasets with $n = 10, 100, 250$ observations from train
- five datasets for each n
- we use the whole test dataset for evaluation

Ukraine arms deliveries PVP

- how to include the task description?
 - pattern: This sentence contains [MASK] [MASK] arms deliveries to Ukraine:
[target_sentence] [SEP] [context_before][SEP] [context_after]
 - verbalizers: argument against, argument for, claim against, claim for, nothing regarding
- multiple MASK tokens are possible
- number of MASK tokens must match the number of tokens of longest verbalizer

Study design

- RoBERTa Base/Large as English-only PLMs
- XLM-RoBERTa Large as multilingual PLM
- we compare PET with standard PEFT (single linear layer) and our PETapter method
- LoRA, (IA)³, and Pfeiffer adapter as PEFT methods
- different PVPs, different number of shots n
- hyperparameters fixed
- each experiment is repeated 5 times
 - in addition to the 5 different stratified training samples
→ performance values from 25 models per training setting

Benchmark results

		Prompt Pattern				Q&A Pattern				Linear Layer		
		PETapter				PETapter						
		(IA) ³	LoRA	Pfeif.	PET	(IA) ³	LoRA	Pfeif.	PET	(IA) ³	LoRA	Pfeif.
RoBERTa Base	10 AG	0.668	0.681	0.683	0.804	0.596	0.672	0.680	0.800	0.297	0.293	0.316
		±.092	±.080	±.078	±.036	±.077	±.077	±.088	±.028	±.053	±.045	±.059
	10 Yahoo	0.236	0.292	0.271	0.545	0.274	0.295	0.285	0.515	0.105	0.116	0.123
		±.016	±.033	±.042	±.040	±.026	±.041	±.036	±.036	±.009	±.020	±.016
	10 Yelp	0.403	0.434	0.423	0.441	0.359	0.412	0.390	0.442	0.215	0.219	0.224
		±.037	±.035	±.038	±.017	±.037	±.042	±.047	±.025	±.017	±.023	±.027
	100 AG	0.856	0.861	0.860	0.875	0.863	0.861	0.862	0.871	0.837	0.862	0.864
		±.012	±.012	±.012	±.007	±.006	±.010	±.011	±.008	±.007	±.008	±.008
	100 Yahoo	0.622	0.642	0.642	0.666	0.622	0.637	0.633	0.659	0.418	0.636	0.633
		±.018	±.009	±.011	±.007	±.005	±.007	±.008	±.007	±.034	±.013	±.012
	100 Yelp	0.541	0.553	0.551	0.552	0.536	0.553	0.548	0.554	0.354	0.538	0.535
		±.010	±.018	±.016	±.014	±.014	±.015	±.014	±.015	±.022	±.018	±.020
RoBERTa Large	10 AG	0.641	0.714	0.702	0.842	0.611	0.746	0.738	0.836	0.305	0.373	0.443
		±.100	±.070	±.081	±.025	±.073	±.054	±.060	±.032	±.030	±.049	±.104
	10 Yahoo	0.242	0.331	0.290	0.574	0.323	0.365	0.346	0.550	0.124	0.150	0.169
		±.027	±.040	±.056	±.030	±.049	±.049	±.054	±.040	±.012	±.027	±.041
	10 Yelp	0.442	0.470	0.479	0.475	0.440	0.472	0.490	0.486	0.211	0.221	0.216
		±.040	±.041	±.035	±.026	±.049	±.049	±.046	±.041	±.010	±.012	±.014
	100 AG	0.868	0.873	0.875	0.877	0.876	0.870	0.873	0.874	0.833	0.875	0.875
		±.011	±.010	±.010	±.009	±.009	±.010	±.010	±.009	±.011	±.008	±.008
	100 Yahoo	0.654	0.662	0.661	0.680	0.655	0.654	0.656	0.675	0.364	0.648	0.647
		±.020	±.014	±.017	±.013	±.010	±.008	±.012	±.013	±.043	±.016	±.015
	100 Yelp	0.611	0.613	0.614	0.593	0.626	0.622	0.620	0.595	0.347	0.551	0.512
		±.011	±.014	±.010	±.014	±.008	±.013	±.013	±.016	±.019	±.019	±.043

Accuracy ± standard deviation

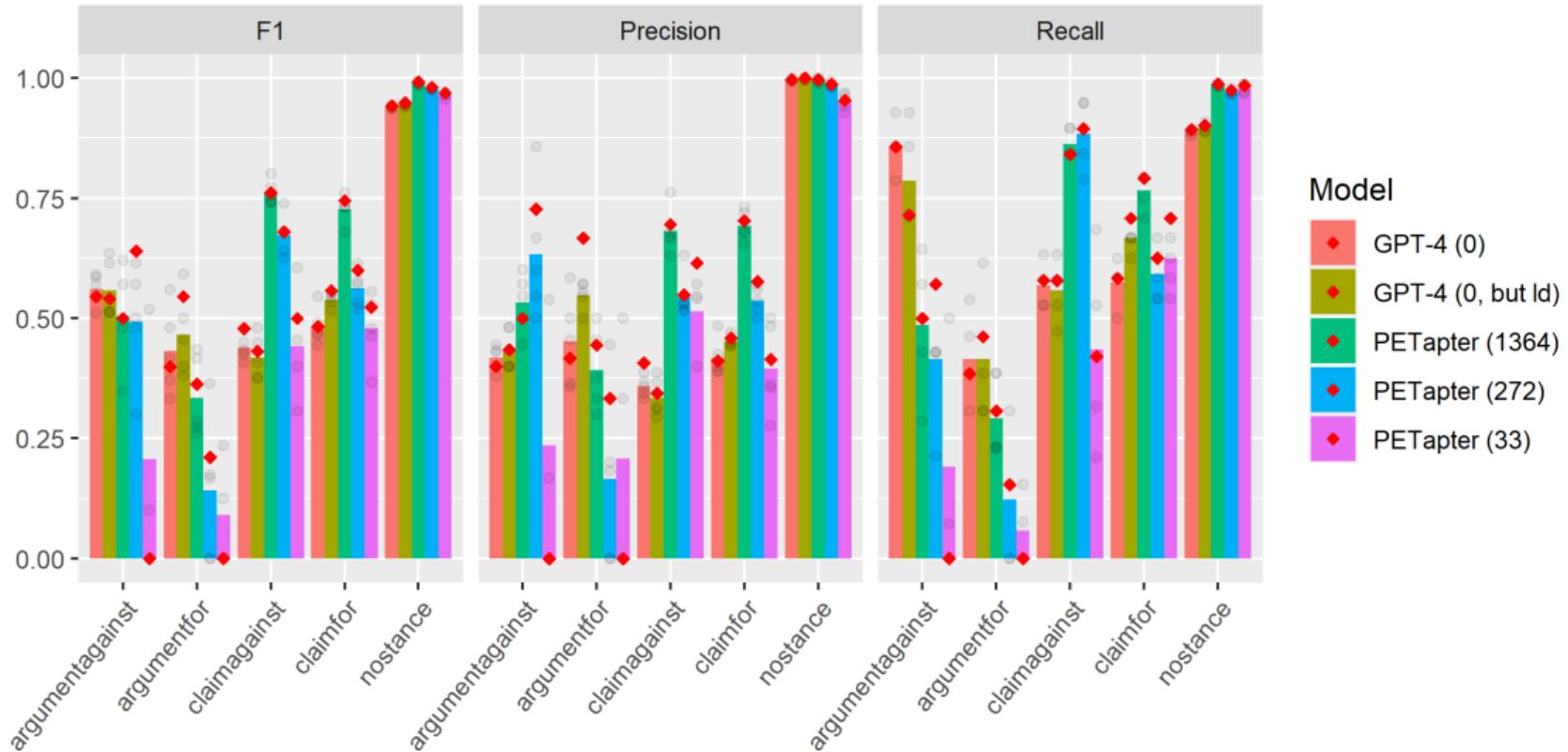
Ukraine results

Macro-F1	n	PETapter			PET	Linear Layer		
		(IA) ³	LoRA	Pfeiffer		(IA) ³	LoRA	Pfeiffer
	10	0.19 ± .023	0.27 ± .039	0.33 ± .027	0.40 ± .055	0.16 ± .000	0.16 ± .001	0.17 ± .021
	100	0.40 ± .029	0.58 ± .042	0.57 ± .035	0.59 ± .054	0.16 ± .000	0.20 ± .030	0.26 ± .035
	250	0.48 ± .036	0.67 ± .019	0.67 ± .018	0.67 ± .109	0.16 ± .003	0.37 ± .040	0.46 ± .082

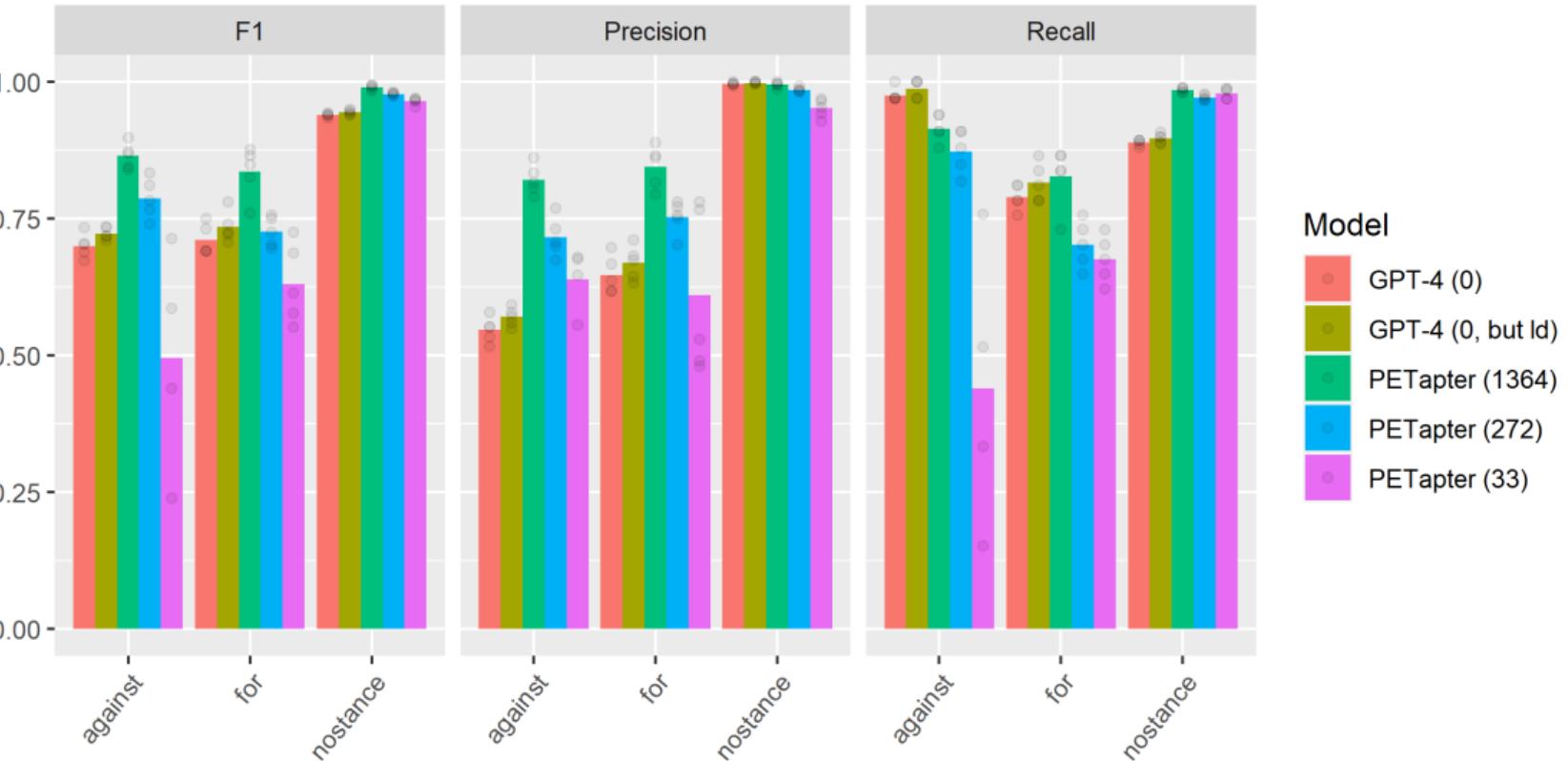
- PETapter already nearly as good as PET for $n = 100$
- PETapter as good as PET and more robust for $n = 250$
- right: Pfeiffer as PEFT method, $n = 250$
- more robust prediction with PETapter
- more accurate for larger classes

	Label	PETapter	PET	Lin. Layer
Precision	argumentagainst	0.54 ± .037	0.54 ± .122	0.39 ± .092
	argumentfor	0.64 ± .032	0.66 ± .145	0.47 ± .135
	claimagainst	0.68 ± .036	0.66 ± .143	0.49 ± .065
	claimfor	0.84 ± .022	0.83 ± .080	0.65 ± .064
Recall	argumentagainst	0.53 ± .064	0.57 ± .140	0.16 ± .098
	argumentfor	0.63 ± .053	0.61 ± .138	0.42 ± .147
	claimagainst	0.78 ± .028	0.75 ± .161	0.54 ± .121
	claimfor	0.78 ± .022	0.80 ± .051	0.74 ± .051
Macro-F1	argumentagainst	0.53 ± .045	0.55 ± .123	0.22 ± .100
	argumentfor	0.63 ± .024	0.63 ± .134	0.44 ± .138
	claimagainst	0.72 ± .028	0.70 ± .148	0.51 ± .084
	claimfor	0.81 ± .012	0.81 ± .040	0.69 ± .049

GPT comparison



GPT comparison (stances)



PVP results

- *Normal* and *Pattern* represents the PVPs presented before
- *No Pattern*:
[MASK] [MASK]: [target_sentence] [SEP] [context_before] [SEP] [context_after]
- for *Shuffle* we shuffled the verbalizers in the worst way possible
- for *Alpha* we just used *a* to *d* for four labels
- we used LoRA in combination with PETapter on the Ukraine dataset

Macro-F1	n	No Pattern			Pattern		
		Alpha	Normal	Shuffle	Alpha	Normal	Shuffle
	10	0.20 ±.025	0.22 ±.031	0.22 ±.033	0.23 ±.067	0.27 ±.039	0.27 ±.043
	100	0.40 ±.027	0.38 ±.035	0.37 ±.027	0.52 ±.051	0.58 ±.042	0.54 ±.048
	250	0.60 ±.025	0.59 ±.030	0.58 ±.032	0.65 ±.019	0.67 ±.019	0.66 ±.017

PETapter majority vote results

- mean (majority) macro-F1 scores (\pm standard deviation)
- LoRA with PETapter on Ukraine
- *Manual*: prompt pattern for benchmark datasets, combination of *Pattern* and *Normal* for Ukraine
- *Autom.*: *No Pattern* and *Alpha*

Data	<i>n</i>	PVP	Mean	Majority
AG	10	Manual	$0.71 \pm .080$	$0.71 \pm .071$
AG	100	Manual	$0.87 \pm .010$	$0.87 \pm .009$
Yahoo	10	Manual	$0.30 \pm .044$	$0.32 \pm .034$
Yahoo	100	Manual	$0.66 \pm .013$	$0.66 \pm .012$
Yelp	10	Manual	$0.45 \pm .055$	$0.45 \pm .053$
Yelp	100	Manual	$0.61 \pm .013$	$0.62 \pm .009$
Ukraine	10	Manual	$0.27 \pm .039$	$0.27 \pm .029$
Ukraine	100	Manual	$0.58 \pm .042$	$0.59 \pm .025$
Ukraine	250	Manual	$0.67 \pm .019$	$0.69 \pm .018$
Ukraine	10	Autom.	$0.20 \pm .025$	$0.20 \pm .021$
Ukraine	100	Autom.	$0.40 \pm .027$	$0.41 \pm .018$
Ukraine	250	Autom.	$0.60 \pm .025$	$0.62 \pm .025$

PETapter: computational resources

- scalable: PEFT modules in general need less VRAM
- easy to share: PETapter/PEFT modules only needs 16 MB disk space (instead of 2.2 GB for full fine-tuning)

Training times on Ukraine dataset

- PET processes 7 obs/s during training and 8 obs/s during testing
- PETapter processes 25 obs/s during training and 51 obs/s ((IA)³/LoRA) or 42 obs/s (Pfeiffer) during testing

Training times on benchmark for $n = 100$

RoBERTa	Architecture	AG	Yahoo	Yelp
Base	PETapter	0.33	0.33	0.32
Base	PET	0.38	0.39	0.39
Large	PETapter	0.65	0.64	0.65
Large	PET	1.00	1.00	1.00
Large	PET	6.1s	6.2s	6.2s

System: 48 GB NVIDIA RTX 6000 Ada, Intel Xeon W7-3445 20×2.6 GHz, 256 GB DDR5-4800 RAM

PETapter conclusions

PETapter

- combines advantages of few-shot learning (PET) and PEFT modules
- performs similarly well as PET
- seems to produce more robust predictions than PET
- is easy/fast to fit, save, combine, and share with less computational resources needed

Aligning LLMs with human preferences

- traditional: reinforcement learning with human feedback (RLHF)
 - ① pretrain and/or fine-tune base LLM
 - ② get pairs of outputs
 - ③ obtain human feedback (often in a binary scenario: chosen/rejected)
 - ④ LLM trained to human feedback
- alternative: direct preference optimization (DPO)
 - ① pretrain and/or fine-tune LLM
 - ② get pairs of outputs
 - ③ LLM trained to triplets of negative and positive examples

DOI: [10.48550/arXiv.2305.18290](https://doi.org/10.48550/arXiv.2305.18290)

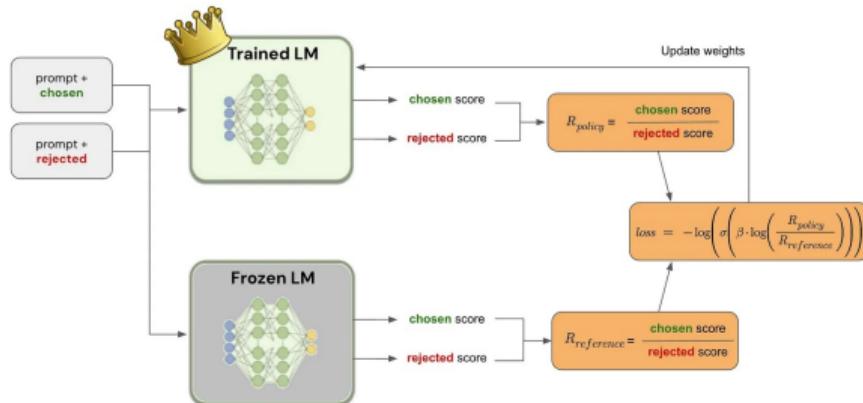
Preference data

- DPO is based on triplets of positive (or chosen) and negative (or rejected) examples/answers
- i.e. for each prompt there is a better and a worse option
- same data is used for RLFH
- in DPO, no reinforcement learning is applied, but direct training to the preference data

Direct preference optimization

- σ sigmoid function, β hyperparameter
- π_θ trained LLM's prediction, π_{ref} reference LLM's prediction
- loss function

$$L_{DPO}(\pi_\theta, \pi_{\text{ref}}) = -\log \sigma \left(\beta \frac{\pi_\theta(y_+ | x)}{\pi_{\text{ref}}(y_+ | x)} - \beta \frac{\pi_\theta(y_- | x)}{\pi_{\text{ref}}(y_- | x)} \right)$$



<https://medium.com/@joaoalages/direct-preference-optimization-dpo-622fc1f18707>

10 Evaluation of NLP Systems

- Topic Model Evaluation
- Language Model Evaluation

General notes on evaluating NLP systems

- aligning with human preferences (cf. slide 379) is (still) the gold standard for NLP systems' evaluation/comparison/benchmarking
- always (mostly) when automated measures are used, this is due to limited possibilities, e.g.,
 - human annotations are expensive, time consuming, difficult to get
 - there is some need for automation or simplification of the task due to inherent disagreement/uncertainty of possible gold standard labels

General notes on evaluating NLP systems

- aligning with human preferences (cf. slide 379) is (still) the gold standard for NLP systems' evaluation/comparison/benchmarking
- always (mostly) when automated measures are used, this is due to limited possibilities, e.g.,
 - human annotations are expensive, time consuming, difficult to get
 - there is some need for automation or simplification of the task due to inherent disagreement/uncertainty of possible gold standard labels
- automated measures have several flaws:
 - lexical matching (e.g., ROUGE) with only one ground truth often not suitable
 - NLP community developed more sophisticated measures (e.g., BERTscore) incorporating for this — but featuring new issues (check <https://aclanthology.org/2021.wmt-1.57/> for a discussion on machine translation scores)
 - for the evaluation of novel LLMs, the question remains whether the evaluation set was part of the respective training set (my personal feeling is that a lot of automated benchmarks nowadays are meaningless due to this issue)

How should topic models be evaluated

- “When developing topic models, a critical question that should be asked is: How well will this model work in an applied setting?”
- most automated topic model measures focus on topic-word sets, not topic-document sets
 - authors propose that doing both would be best
- authors evaluate coherence-based measures (the most commonly used measure in topic model community), e.g., NPMI
- they show that those can be unreliable
 - they do not result in the best model wrt human judgment
 - i.e. they do not correlate well with human perception
- “The comparative performance of coherence measures can vary significantly depending on the aggregate calculation method used and the way the data has been prepared.”

DOI: 10.18653/v1/2021.naacl-main.300

An example on how difficult it is to define a suitable measure

- the authors make 5 propositions to make interpretability measurable
 - ① If coherence scores are robust, they should correlate.
 - ② An interpretable topic is one that can be easily labeled.
 - ③ An interpretable topic has high agreement on labels.
 - ④ An interpretable topic is one where the document-collection is easily labeled.
 - ⑤ An interpretable topic word-set is descriptive of its topic document-collection.

An example on how difficult it is to define a suitable measure

- the authors make 5 propositions to make interpretability measurable
 - ① If coherence scores are robust, they should correlate.
 - ② An interpretable topic is one that can be easily labeled.
 - ③ An interpretable topic has high agreement on labels.
 - ④ An interpretable topic is one where the document-collection is easily labeled.
 - ⑤ An interpretable topic word-set is descriptive of its topic document-collection.
- the authors conduct experiments for all five facets of interpretability
- conclusion: topic model evaluation is an open problem

DOI: [10.18653/v1/2021.naacl-main.300](https://doi.org/10.18653/v1/2021.naacl-main.300)

“Is Automated Topic Model Evaluation Broken?”

- my personal feeling: “yes, it is - as it is the case for all automatic NLP evaluations”
- the authors identify a “validation and standardization gap”
 - validation: lack of correlation of coherence measures to human judgment
 - standardization: evaluating topic models on standardized benchmarks
- evaluation should not be method-driven, but needs-driven,
i.e. evaluation has to be formulated corpus-specific
 - contradicts the authors’ idea of standardization
 - there is no generic corpus
 - nor is there a generic topic model analysis (application)

<https://openreview.net/pdf?id=tjdHCnPqoo>

“Are Neural Topic Models Broken?”

- as a follow-up study, the authors investigate whether NTMs (e.g., compared to LDA) provide
 - ① stable results
 - ② results that correspond to human preferences
- authors show that standard LDA (with Gibbs sampling) performs best in their study
 - especially wrt stability/reproducibility of the results
 - but also wrt alignment with human judgments
- “the driving question for methodological development and evaluation should not be how to demonstrate an improvement in SOTA, it should be why the model is being created”

DOI: [10.18653/v1/2022.findings-emnlp.390](https://doi.org/10.18653/v1/2022.findings-emnlp.390)

LLMs can be used as helpers for topic modeling

- authors investigate usefulness of LLMs for
 - automatic evaluation of topic models
 - guidance for selection of number of topics
- authors find that
 - LLM judgment correlates strongly with human judgment (better than existing coherence-based metrics as, e.g., NPMI)
 - optimal number of topics (K) is vague, but LLMs can guide towards reasonable choice

DOI: 10.18653/v1/2023.emnlp-main.581

The right choice of a NTM excels LDA

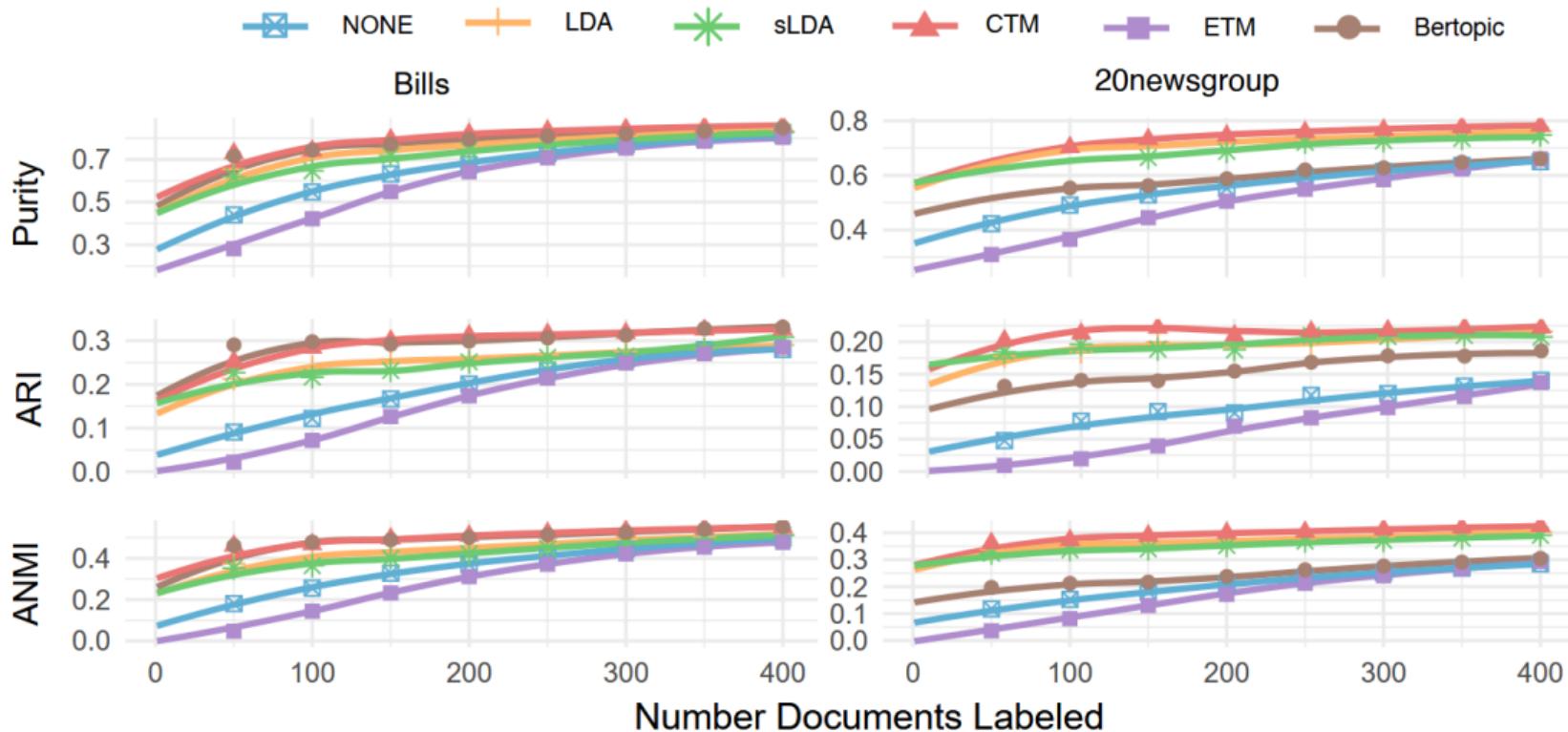
- validity of automated measures questionable
 - may overlook model's benefits in **real-world** applications
- interactive task-based evaluation of different topic models
- comparison of LDA, sLDA and BERTopic with contextualized topic model (CTM) and embedded topic model (ETM)
- NTMs result in consistently (and far) better coherence scores compared to LDA

Dataset	LDA	sLDA	CTM	ETM	BERTopic
Bills	0.07	0.09	0.09	0.13	0.15
Newsgroup	0.06	0.05	-0.09	0.09	0.10

Table 1: On average, NTMs have higher NPMI coherence than LDA, where BERTopic has the highest coherence, followed by ETM. However, the NTMs with higher coherence are not better than CTM and LDA under a task-based experiment (Figure 1).

<https://aclanthology.org/2024.eacl-long.51/>

BERTopic and ETM do not perform well



CTM excels LDA (by small margin)

- LDA dominates social science applications
 - ease of usage, low complexity of model, fast calculation
- here, the authors show that there might be a (slightly) better model than LDA if you put some effort in the model selection process
- in the two papers before, the authors were able to show that LDA performs
 - best
 - better than many NTMs
- topic model evaluation is an unsolved problem

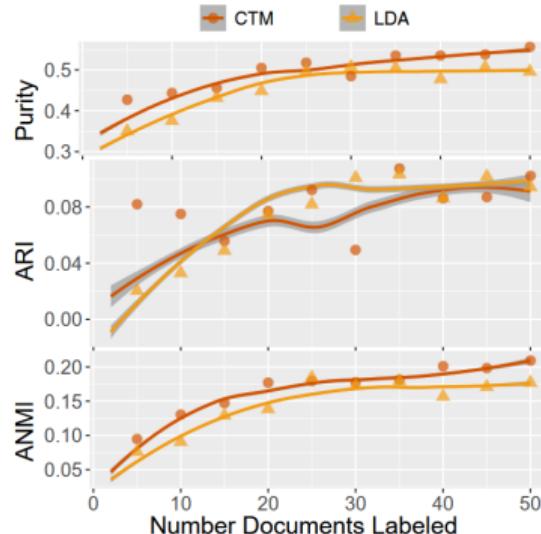


Figure 4: We run a followup pilot study with six social science experts (three in each group) on their internal social science dataset (800 documents). They are familiar with the topics in the dataset. Up to the 50th document labeled, CTM still generalizes well for expert datasets and expert users.

Language model evaluation is just as unsolved

- BLEU: +1–2 points (\approx SOTA) \rightarrow 50% *actually* better¹
- metrics (nearly always) substitutes for human perception²
- stance detection: even for humans difficult task³
- (classical) recommender foster building of filter bubbles^{4,5}

¹Tangled up in BLEU: Reevaluating the evaluation of automatic machine translation evaluation metrics (Mathur et al., 2020)

²Text as data: The promise and pitfalls of automatic content analysis methods for political texts (Grimmer and Stewart, 2013)

³(Mis)alignment between stance expressed in social media data and public opinion surveys (Joseph et al., 2021)

⁴News recommendation and filter bubble (Yao and Hauptmann, 2018)

⁵How to lose the information war (Jankowicz, 2020)

Tanya Goyal @tanyaagoyal · 13h
✨ New preprint ✨
Zero-shot GPT-3 does "better" at news summarization than any of our fine-tuned models. Humans like these summaries better. But all of our metrics think they're MUCH worse.

Work w/ @jessyjli, @gregd_nlp. Check it out here: arxiv.org/abs/2209.12356
[1/6]

CNN article: <https://www.cnn.com/2022/09/09/politics/judge-throws-out-trumps-rico-lawsuit-against-hillary-clinton-and-democrats/index.html>
Length-constrained prompt
Summarize the above article in 2 sentences.
GPT-3: Donald Trump's lawsuit against Hillary Clinton, the Democratic National Committee, and several other individuals and entities has been dismissed by a federal judge. The judge ruled that Trump's claims are unsupported by any legal authority and that many of his specific factual assertions are implausible or unsupported.

Keyword-constrained prompt
Summarize the above article briefly focusing on Alina Habba.
GPT-3: Alina Habba is an attorney for Donald Trump. She stated that Trump will "immediately move to appeal this decision." Habba also said that the judge's opinion is "rife with erroneous applications of the law" and that it "disregards the numerous independent governmental investigations which substantiate our claim that the defendants conspired to falsely implicate our client and undermine the 2016 Presidential election."

News Summarization and Evaluation in the Era of GPT-3 (Goyal et al. 2022)

Don't treat disagreement as noise!

- annotator disagreement: often there is no true label
- sometimes there can't even be a true label
- new research direction: make use of uncertainty/disagreement in annotations
- best practice for human annotations: DOI: [10.6028/NIST.TN.2287](https://doi.org/10.6028/NIST.TN.2287)
- annotator-specific classification heads: DOI: [10.18653/v1/2023.emnlp-main.126](https://doi.org/10.18653/v1/2023.emnlp-main.126)
- annotator representations: DOI: [10.18653/v1/2023.findings-emnlp.832](https://doi.org/10.18653/v1/2023.findings-emnlp.832)
- using annotators' demographics to predict annotation label: DOI: [10.18653/v1/2023.emnlp-main.415](https://doi.org/10.18653/v1/2023.emnlp-main.415)

“Rethinking Model Evaluation as Narrowing the Socio-Technical Gap”

- the authors question such general-purpose models
- as already mentioned in the topic model evaluation subsection, here the authors “urge the community to develop evaluation methods based on real-world socio-requirements and embrace diverse evaluation methods with an acknowledgment of trade-offs between realism to socio-requirements and pragmatic costs to conduct the evaluation”
- instead of measuring (and reporting) semi-informative measures (such as accuracy) for a vast amount of different tasks/datasets, it’s better to evaluate models task-specific with a multitude of performance measures representing a variety of quality facets of models’ outputs

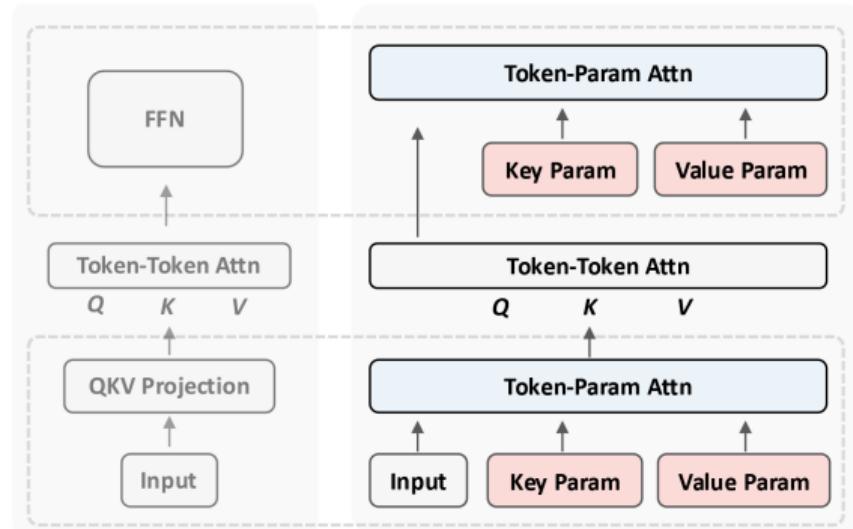
DOI: [10.48550/arXiv.2306.03100](https://doi.org/10.48550/arXiv.2306.03100)

11 Recent Architectures

- Tokenformer
- Mamba
- Hymba
- Kolmogorov-Arnold Networks

Tokenformer

- eliminates retraining need for increasing model size
- supports incremental increase of parameter's count
- leverages the attention mechanism
 - attention facilitates token-token interactions
 - replacing original projection matrices with token-parameter **attention** (pattention) layers
 - query: input
 - keys and values: model parameters



Transformer

Tokenformer

<https://openreview.net/pdf?id=oQ4igHyh3N>

Methodology

- idea: replacing all linear projections (attention, FFNN) with pattention layers
- two new sets of n learnable parameters $K_P \in \mathbb{R}^{n \times d_1}, V_P \in \mathbb{R}^{n \times d_2}$
- d_1 input, d_2 output dimension
- τ scaling factor (default: \sqrt{n})
- f non-linear activation function (default: GeLU)
- pattention layer (general form):

$$\tilde{A}_{i,j}(X, K_P) = f \left(\frac{\tau A_{i,j}}{\sqrt{\sum_{k=1}^n A_{i,k}^2}} \right) \quad \text{with} \quad A = XK_P^T$$

- output of pattention layer: $Z(X, K_P, V_P) = \tilde{A}(X, K_P)V_P$

Calculations

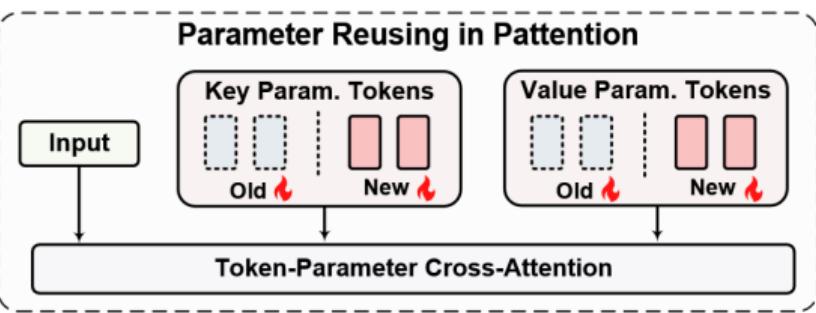
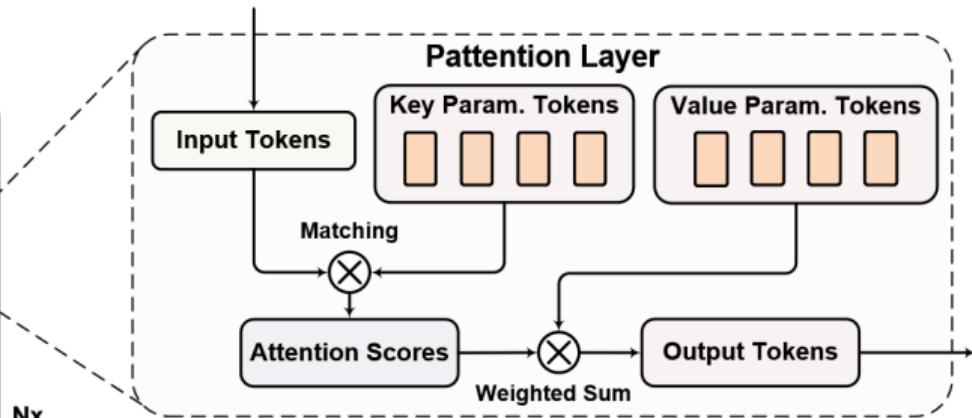
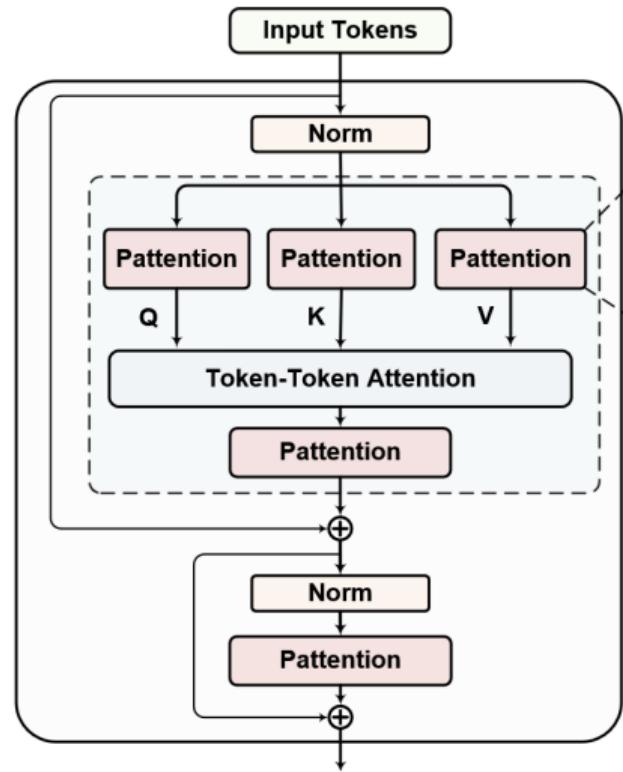
- query, key, value matrices calculation using pattention
 - $Q = Z(X, K_P^Q, V_P^Q)$, $K_P^Q, V_P^Q \in \mathbb{R}^{n_Q \times d}$
 - $K = Z(X, K_P^K, V_P^K)$, $K_P^K, V_P^K \in \mathbb{R}^{n_K \times d}$
 - $V = Z(X, K_P^V, V_P^V)$, $K_P^V, V_P^V \in \mathbb{R}^{n_V \times d}$
- $X_Z = \text{softmax}(QK^T / \sqrt{d})V$
- output of (former) attention layer modified using pattention
 - MHA = $Z(X_Z, K_P^0, V_P^0)$, $K_P^0, V_P^0 \in \mathbb{R}^{n_0 \times d}$
- output of (former) classical FFNN layer modified using pattention
 - FFNN = $Z(X_{\text{FFNN}}, K_P^{\text{FFNN}}, V_P^{\text{FFNN}})$, $K_P^{\text{FFNN}}, V_P^{\text{FFNN}} \in \mathbb{R}^{n_{\text{FFNN}} \times d}$
- overall architecture:
 - LN: layer normalization as usual
 - $X_{\text{inter}} = X_{\text{in}} + \text{MHA}(\text{LN}(X_{\text{in}}))$
 - $X_{\text{out}} = X_{\text{inter}} + \text{FFNN}(\text{LN}(X_{\text{inter}}))$

Incremental scaling

- denote $K_P = K_P^{\text{old}}, V_P = V_P^{\text{old}} \in \mathbb{R}^{n \times d}$
- appending new key-value parameter tokens $K_P^{\text{new}}, V_P^{\text{new}} \in \mathbb{R}^{m \times d}$
- $K_P^{\text{scale}} = [K_P^{\text{old}}, K_P^{\text{new}}] \in \mathbb{R}^{(m+n) \times d}$
- $V_P^{\text{scale}} = [V_P^{\text{old}}, V_P^{\text{new}}] \in \mathbb{R}^{(m+n) \times d}$
- initializing the new parameters with zero let the model resume with the former model state

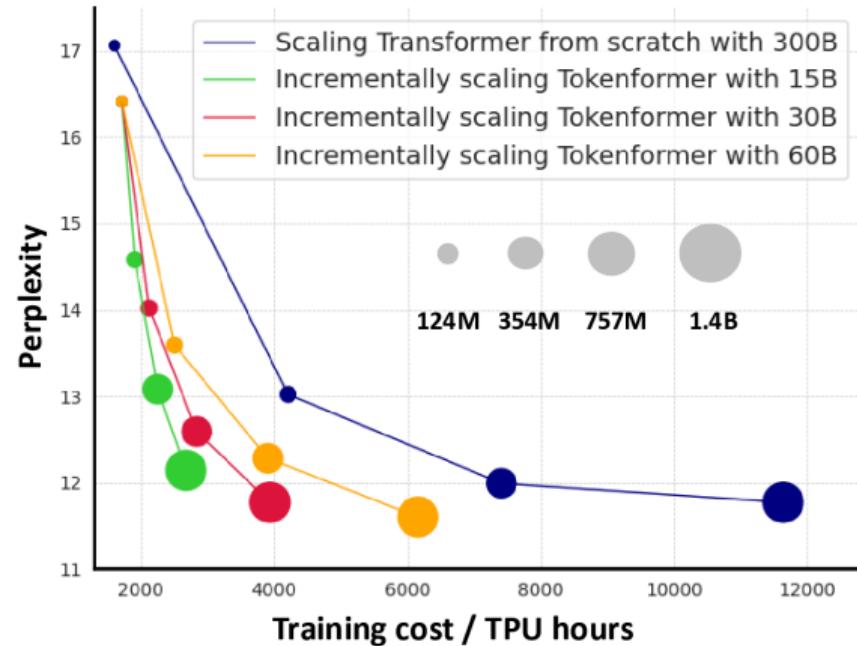
Schematic representation of tokenformer

TokenFormer



Training complexity

- tokenformer achieves performance (wrt perplexity) comparable to original transformer trained from scratch
- need of only 30B tokens training data budget compared to 300B for transformer
- computation time: 1/3 of transformer (transformer 1.4B with 300B tokens vs. tokenformer 1.4B with 30B tokens)



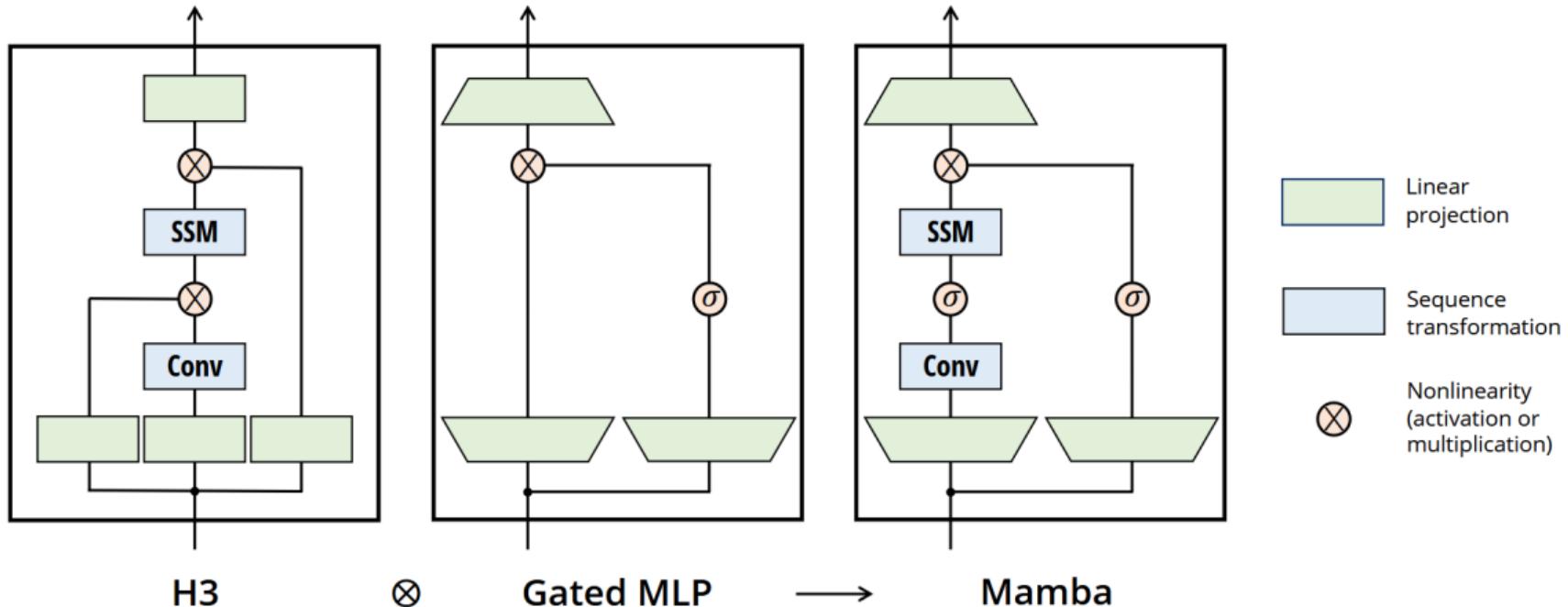
Tokenformer

- in fact, in the end, pattention layers are just additional FFNN layers

Original paper: <https://openreview.net/pdf?id=oQ4igHyh3N>

Short explanation video: <https://www.youtube.com/watch?v=4weeoIjWIXI>

Mamba

**H3** \otimes **Gated MLP** \longrightarrow **Mamba**

paper: <https://openreview.net/forum?id=tEYskw1VY2>

updated version on arXiv: DOI: 10.48550/arXiv.2312.00752

visual guide: <https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mamba-and-state>

talk: <https://www.youtube.com/watch?v=X-7rgesJaGM>

Issues with the transformer architecture

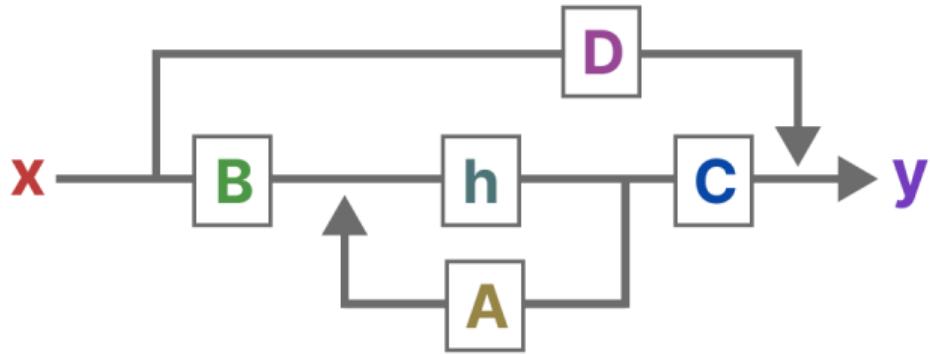
- the core of the transformer architecture is **attention**
- one central point of attention is that it is parallelizable
- how do we achieve parallelizability?
 - the calculated attention of, e.g., the third to the second token does not depend on the attention of the second to the first token
 - if (in the decoder) our model predicts a token, for the prediction of the following token all attention scores (the complete matrix) has to be recomputed
- in general, when generating the next token, we always need to recalculate the attention for the entire sequence
- generating N tokens needs roughly N^2 computations

The best of both worlds

- transformer architecture with attention:
 - fast to train due to parallelizability
 - slow to infer due to quadratic scaling in sequence length
- recurrent neural networks:
 - relatively fast to train using semi-parallelizability
 - short memory since hidden state is overwritten
 - fast for inference due to linear scaling in sequence length

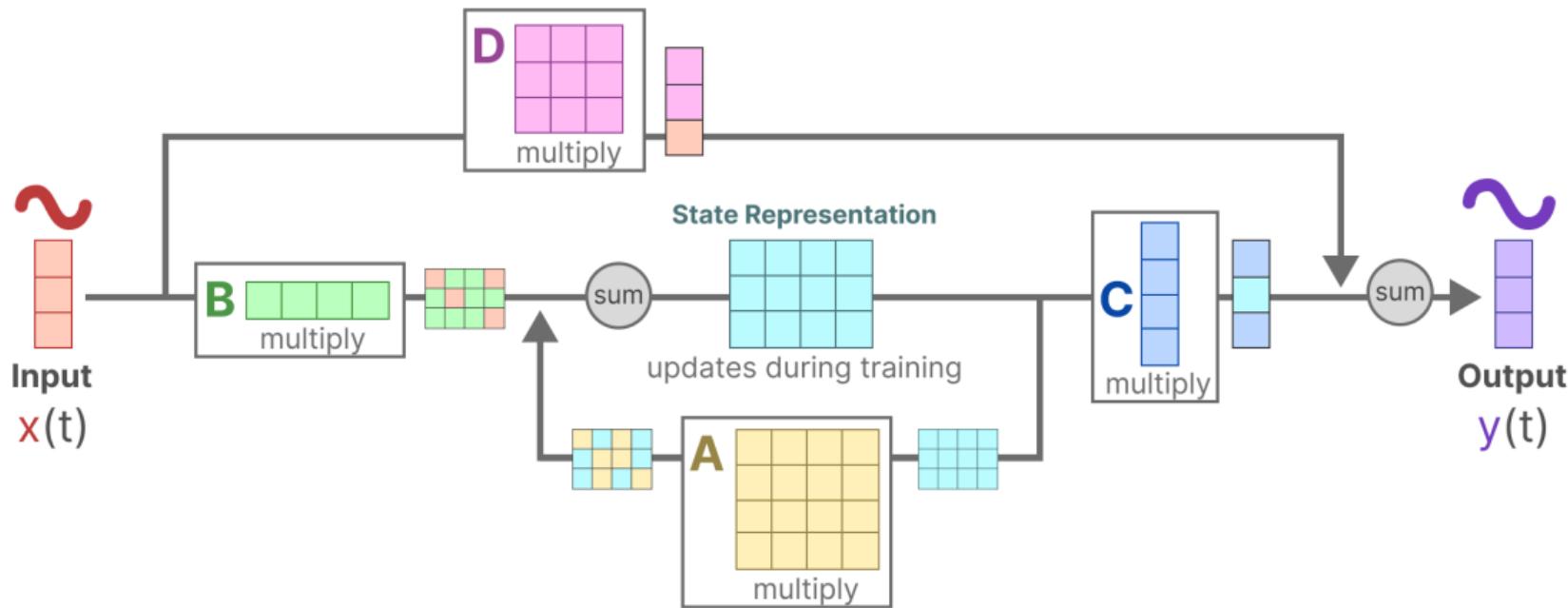
DOI: [10.48550/arXiv.2011.04006](https://doi.org/10.48550/arXiv.2011.04006)

State-space models (SSMs) to the rescue

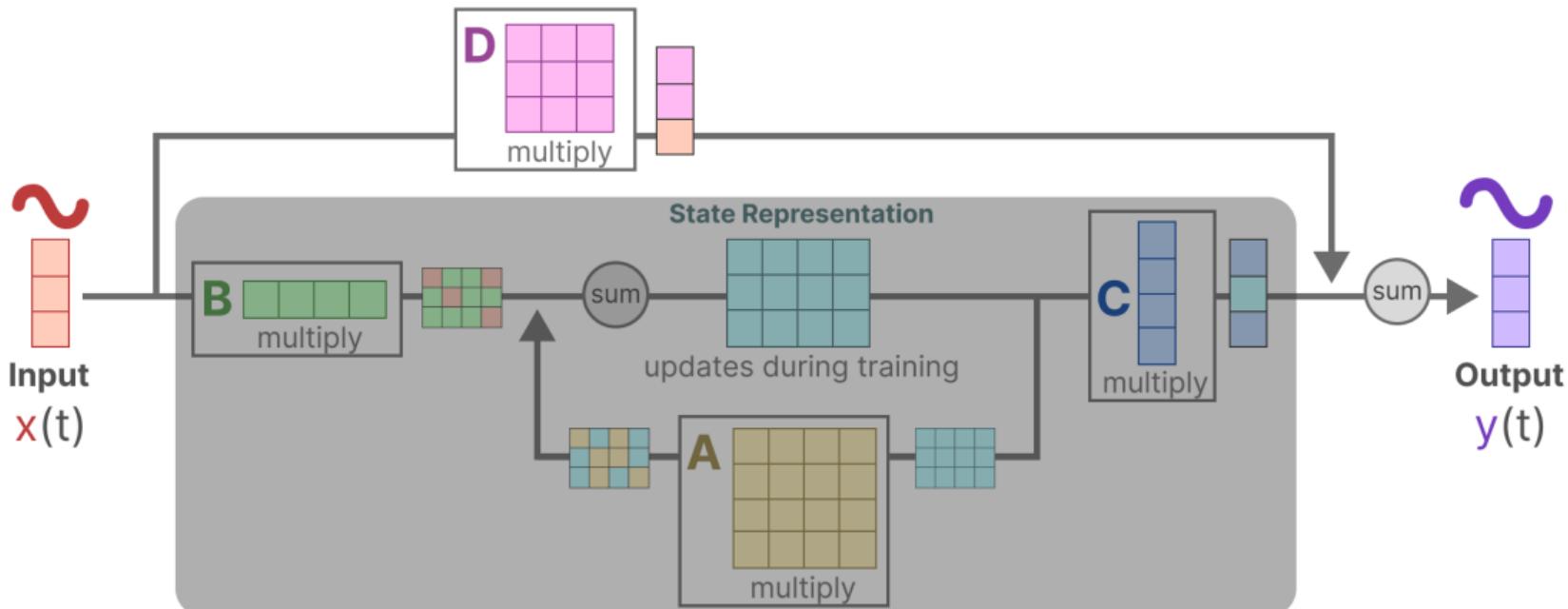


- state equation: $h'(t) = Ah(t) + Bx(t)$
- output equation: $y(t) = Ch(t) + Dx(t)$
- $x(t) \in \mathbb{R}$ input sequence
- $h(t) \in \mathbb{R}^n$ latent state representation
- $y(t) \in \mathbb{R}$ output sequence

SSM calculation

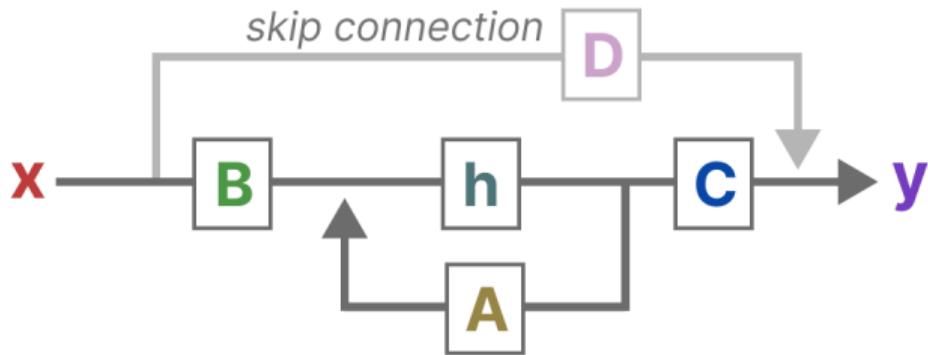


SSM calculation



State Space Model

SSM discretization



- transformation (zero-order hold) from “continuous” (Δ, A, B) to “discrete parameters” (\bar{A}, \bar{B}) to come from function-to-function to seq2seq
 - $\bar{A} = \exp(\Delta A)$
 - $\bar{B} = (\Delta A)^{-1}(\exp(\Delta A) - I)\Delta B$
- $h_t = \bar{A}h_{t-1} + \bar{B}x_t$
- $y_t = Ch_t$
- Δ (learnable parameter) is the step size, i.e., the resolution of the input

Representations of discretized SSM

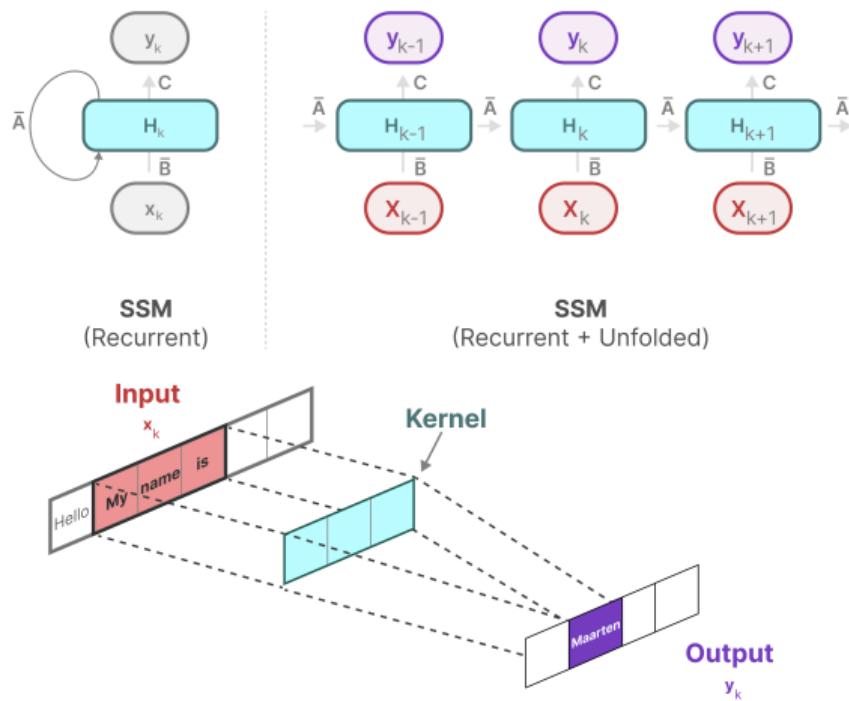
- recurrent representation (classical RNN)

- $h_0 = \bar{B}x_0, y_0 = Ch_0$
- $h_1 = \bar{A}h_0 + \bar{B}x_1, y_1 = Ch_1$
- $h_2 = \bar{A}h_1 + \bar{B}x_2, y_2 = Ch_2$

- convolutional representation (CNN)

- $\bar{K} = (C\bar{B}, C\bar{A}\bar{B}, \dots, C\bar{A}^k\bar{B}, \dots)$
- $y_0 = C\bar{B}x_0$
- $y_1 = C\bar{A}\bar{B}x_0 + C\bar{B}x_1$
- $y_2 = C\bar{A}^2\bar{B}x_0 + C\bar{A}\bar{B}x_1 + C\bar{B}x_2$

DOI: 10.48550/arXiv.2110.13985



(In fact) the best of both worlds

- choose representation depending on task
 - during training: CNN
 - during inference: RNN
- still some flaws: linear time invariant, i.e. A, B, C are fixed/shared
- static representation / not content-aware
 - i.e. all tokens “attend” the others in the same manner
- this results in very static computations, esp. in comparison to transformer-based models
 - attention: every token attends the others individually

Structured state spaces for sequences (S4)

- very important: how to find/train A
- SSMs were not able to cover long range dependencies very well
- in comparison to transformers, SSMs had very small context size
- in particular, SSMs suffered from the vanishing/exploding gradients problem
- theory on high-order polynomial projection operators (HiPPO) matrix allows the SSM to memorize the history of the input
- attend the newer signals over the older
- $\text{HiPPO} = A \in \mathbb{R}^{n \times n}$ is much better than random initialization

$$A_{ij} = \begin{cases} \sqrt{2i+1}\sqrt{2j+1} & \text{if } i > j \\ i+1 & \text{if } i = j \\ 0 & \text{if } i < j \end{cases}$$

DOI: 10.48550/arXiv.2111.00396

annotated S4: <https://srush.github.io/annotated-s4/>

Advancements of Mamba

- still some open issues why SSMs are not competitive to transformer models in NLP
 - SSMs (e.g., S4) are not able to focus on or to ignore inputs
 - as presented, they are linear time invariant (LTI)
 - SSMs cannot perform content-aware reasoning
- Mamba introduces two main contributions:
 - a selective scan algorithm to filter (ir)relevant information
 - a hardware-aware algorithm for efficient storage of parameters (during training)
 - this results in the selective SSM or S6 model
- Mamba can be used in a similar way as self-attention blocks

Trade-off between compression and memory

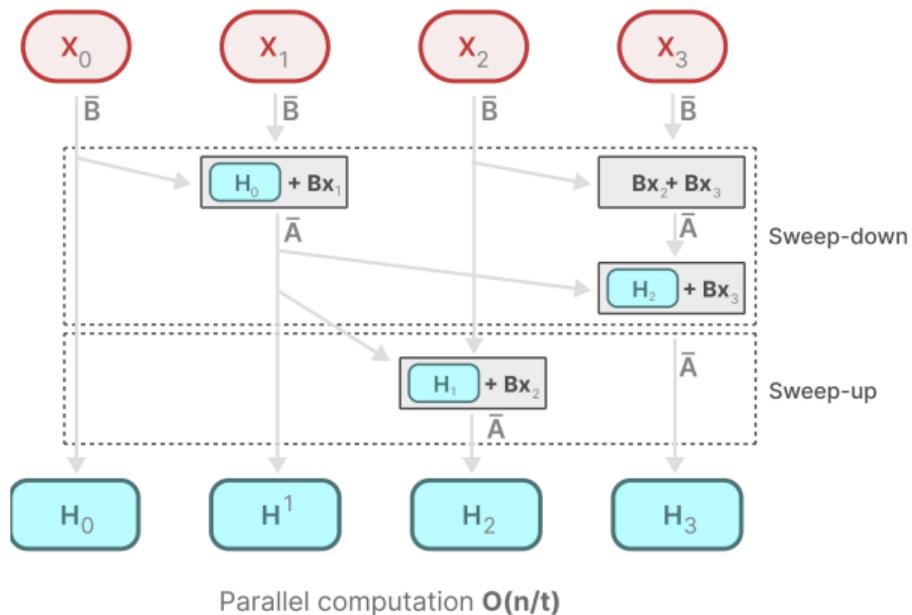
- once again, it's about the best of both worlds
- i.e. trade-off between
 - major compression (less powerful, more efficient) — RNN/S4
 - no compression (more powerful, less efficient) — transformer
- Mamba aims for compress data selectively
- which leads to loosing the ability of the CNN representation with its parallelizability (since it assumes a static/fixed kernel)

Parameters of Mamba

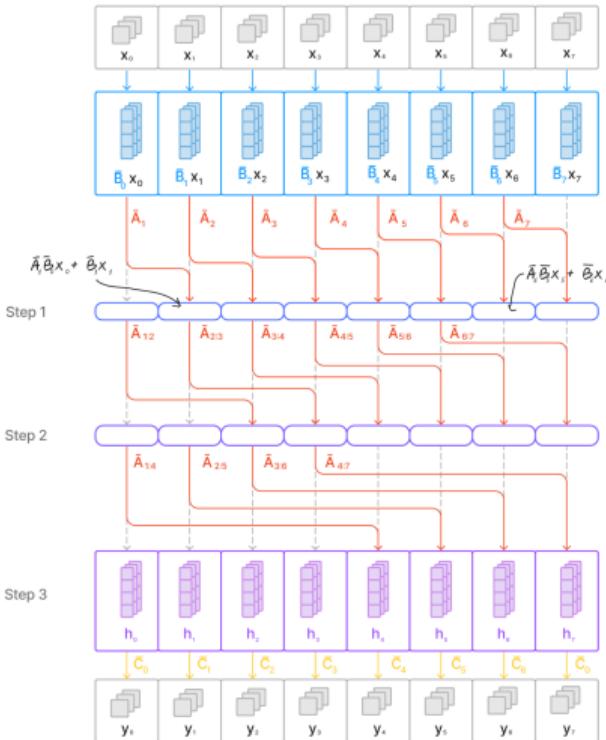
- in S4, the matrices $A, B, C \in \mathbb{R}^{n \times d}$ are static with fixed dimensions n (hidden size) and d (input size $\hat{=}$ embedding size); input and output $X, Y \in \mathbb{R}^{n_{\text{seq}} \times d}$
- A : weights for the evolution of the (hidden) state
- B : weights for the input, adding it to the state
- C : weights for the state, adding it to the output
- Mamba (SSM + selection) makes B, C, Δ input-dependent,
i.e. it results in different B, C matrices for different input tokens (\rightarrow content-awareness)
- A stays static (for computational reasons, cf. parallel scan)
- smaller step size Δ results in ignoring some words, larger step size in focusing on the content (similar to BOW)

Selective scan algorithm

- we still preserved the recurrent representation (only lost the convolutional one)
- the inherent recurrence is sometimes also known as scan operation
- Mamba makes parallel calculation possible through the parallel scan algorithm
 - assumption: order of operations does not matter
 - i.e. we can somehow “divide and conquer”

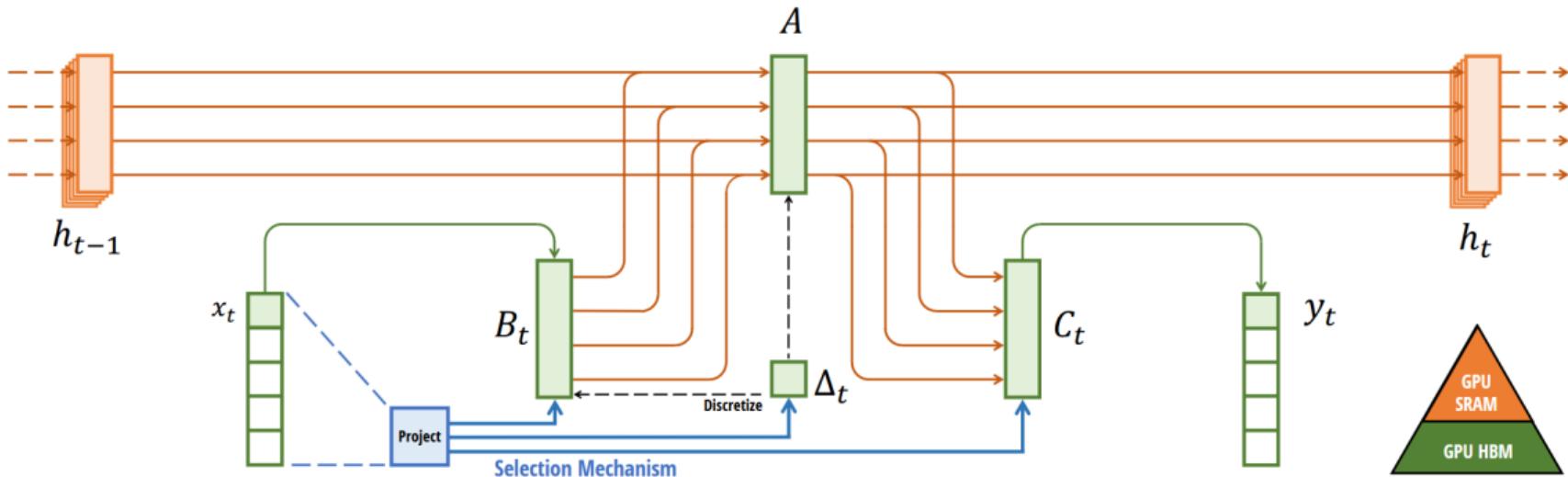


Selective scan algorithm example calculation



<https://bocachancla.blog/posts/mamba/>

Hardware-aware state expansion in selective SSM (S6) model

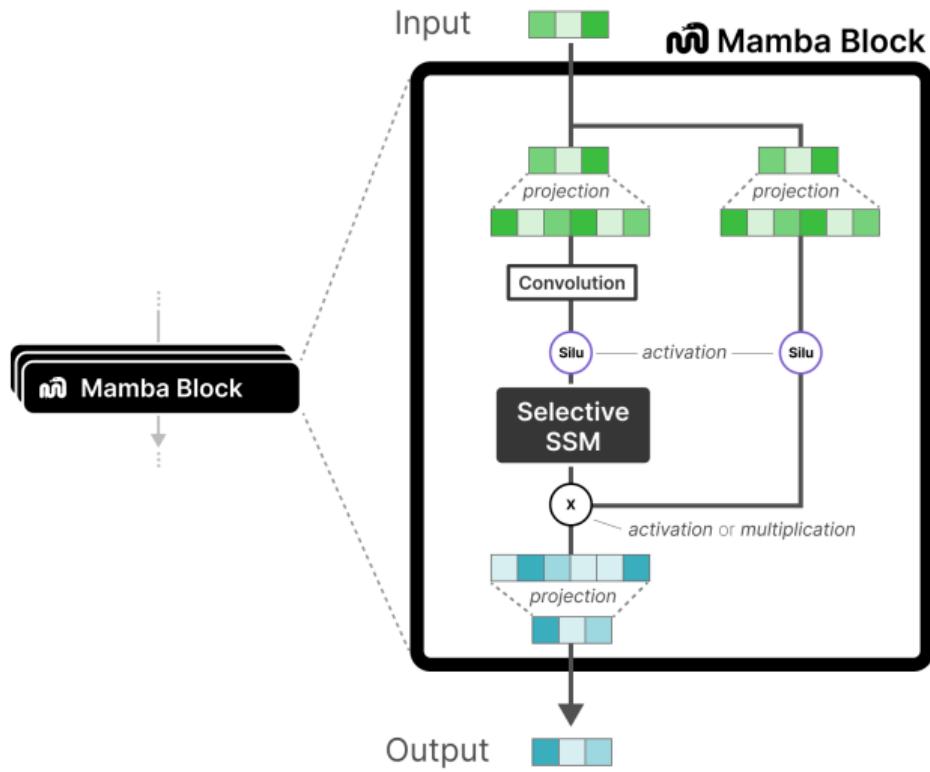


- making use of different GPU memory types
 - static RAM (SRAM) that is fast, but expensive/small
 - dynamic RAM (DRAM) that is slightly less efficient
- kernel fusion (hidden state calc.) to prevent from copying data from DRAM to SRAM
- recomputation (during backward pass) faster than storing parameters

Mamba block

key components of Mamba:

- recurrent SSM through discretization
- HiPPO initialization on matrix A to capture long-range dependencies
- parallel/selective scan algorithm to selectively compress information
- hardware-aware algorithm to speed up computation



Hymba: A hybrid-head architecture for small LMs

- transformer dominant choice for LMs due to strong performance
- Mamba offers efficient hardware optimization
 - but still struggles with longer memory
- Hymba (by nvidia) introduces
 - enhanced efficiency (over transformer) and improved performance (over Mamba)
 - by means of several design modifications
- in addition, the paper on Hymba reveals
 - over 50% of attention computation can be replaced by cheaper SSM computation
 - most global attention can be replaced by local attention by SSM heads
 - cache of K and V is highly correlated across heads and layers
 - can be shared across heads (group query attention) and layers (KV cache sharing)

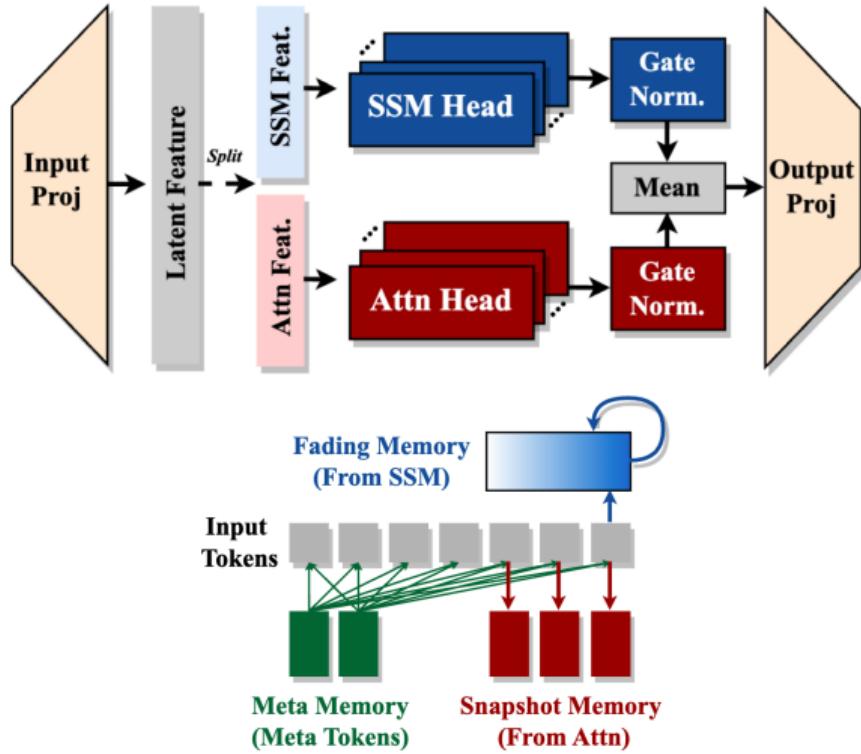
original paper: <https://openreview.net/forum?id=A1ztozypga>

technical note from the authors (from nvidia): <https://developer.nvidia.com/blog/hymba-hybrid-head-architecture-boosts-small-language-model-performance/>

short explanation video: <https://www.youtube.com/watch?v=9EdYwMMABnA>

Hybrid-head module

- fusing (parallel) attention and SSM heads
- $Y = W_0(\beta_1 \text{LN}(Y_Z) \beta_2 \text{LN}(Y_{SSM}))$
- $Y_Z = \text{softmax}(QK^T / \sqrt{d}) W^V \tilde{X}$
- $Y_{SSM} = W^G \tilde{X} \odot \alpha(A, B, C, \Delta) W^{SSM} \tilde{X}$
- \tilde{X} is the original input matrix X prepended with meta tokens



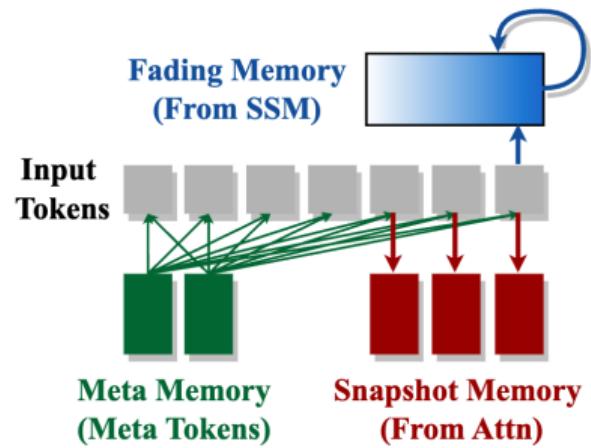
Formal definition of hybrid-head module

$$\alpha(A, B, C, \Delta)_{ij} = C_i \left(\prod_{k=j+1}^i \exp(A\Delta_k) \right) B_j \Delta_j$$

- A, B, C, Δ are SSM parameters
- α implements/makes use of all transition parameters A between the respective tokens
- $Y_{SSM} = W^G \tilde{X} \odot \alpha(A, B, C, \Delta) W^{SSM} \tilde{X}$
- $Y = W_0(\beta_1 \text{LN}(Y_Z) \beta_2 \text{LN}(Y_{SSM}))$

Integration of meta tokens

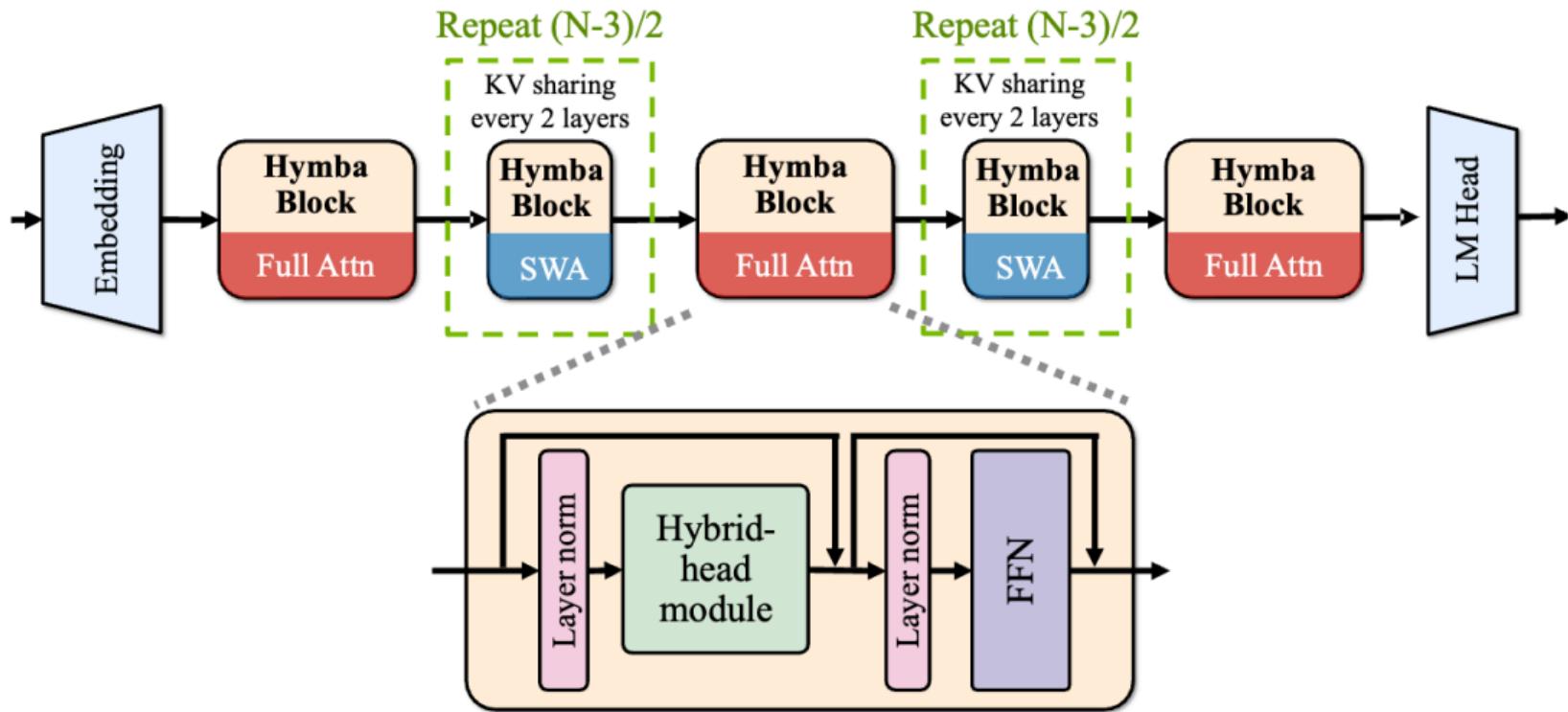
- $\tilde{X} = [R, X]$ is the original input matrix X prepended with meta tokens R
- the authors add 128 pretrained embeddings prepended to inputs
- acting as learned cache initialization to enhance focus on relevant information
- dual purpose:
 - mitigating attention drain by acting as backstop tokens
 - encapsulating compressed world knowledge



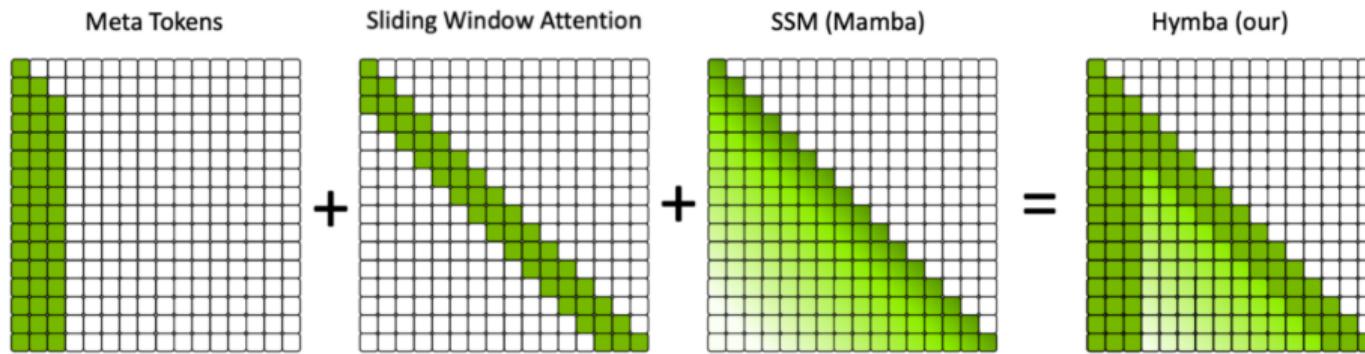
Crucial elements of Hymba

- hybrid architecture between Mamba and attention
 - Mamba great at summarization and closer focuses on the current token
 - attention more precise and includes whole memory
- sliding window attention (SWA): three “full” attention layers, SWA used in 90% of layers
- cross-layer KV cache sharing between every two consecutive attention layers (in addition to KV sharing between heads)
- 128 meta-tokens helping to store some general knowledge
- attention heads improve task performance (while increasing KV cache requirements)
- Hymba mitigates: hybrid-head module to parallel combine local and global attention
- the advancements improve throughput by three times and reduce cache by almost four times without sacrificing performance (at least for “small” LMs)

Architecture



Attention matrix



- four types of elements in the attention matrix
 - meta (first 128), beginning-of-sequence (one column after meta), self-attention (diagonal), cross-attention (off-diagonal)
- in transformers more concentrated on beginning-of-sequence and self-attention
- in Hymba complementary elements of meta, attention and SSM
 - hybrid-head setting leads to more balanced distribution of attention scores (and better performance)

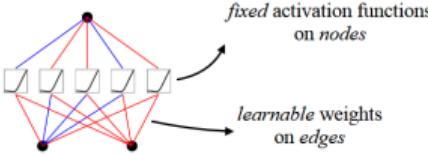
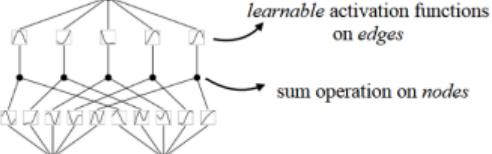
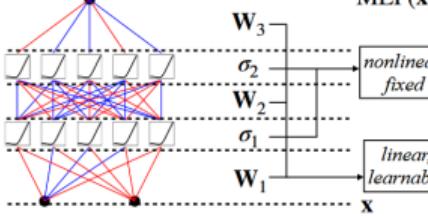
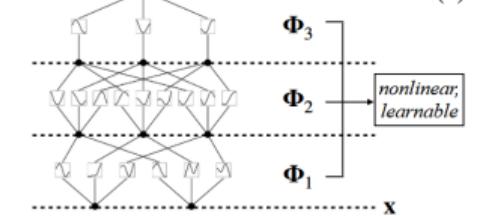
Roadmap of design decisions

Configuration	Commonsense Reasoning (%)	Recall (%)	Throughput (token/sec)	Cache Size (MB)	Design Reason
Ablations on 300M model size and 100B training tokens					
Transformer (LLaMA)	44.08	39.98	721.1	829.4	Accurate recall while inefficient
State Space Models (Mamba)	42.98	19.23	4720.8	1.9	Efficient while inaccurate recall
A. + Attention heads (sequential)	44.07	45.16	776.3	311.9	Enhance recall capabilities
B. + Multi-head structure (parallel)	45.19	49.90	876.7	295.7	Better balance of two modules
C. + Local / global attention	44.56	48.79	2399.7	78.1	Boost compute/cache efficiency
D. + KV cache sharing	45.16	48.04	2756.5	76.3	Cache efficiency
E. + Meta tokens	45.59	51.79	2695.8	76.9	Learned memory initialization
Scaling to 1.5B model size and 1.3T training tokens					
F. + Size / data	60.45	67.61	666.1	124.7	Further boost task performance

Conclusions

- Hymba is as performant as “small” LLMs
- Hymba implements some architecture decisions for different reasons:
 - hybrid-head: over 50% of attention computation can be replaced by cheaper SSM computation
 - SWA: most global attention can be replaced by local attention by SSM heads
 - KV sharing: cache of K and V is highly correlated across heads and layers
→ can be shared across heads (group query attention) and layers (KV cache sharing)

Teaser: Kolmogorov-Arnold networks (KAN)

Model	Multi-Layer Perceptron (MLP)	Kolmogorov-Arnold Network (KAN)
Theorem	Universal Approximation Theorem	Kolmogorov-Arnold Representation Theorem
Formula (Shallow)	$f(\mathbf{x}) \approx \sum_{i=1}^{N(c)} a_i \sigma(\mathbf{w}_i \cdot \mathbf{x} + b_i)$	$f(\mathbf{x}) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^n \phi_{q,p}(x_p) \right)$
Model (Shallow)	(a) 	(b) 
Formula (Deep)	$\text{MLP}(\mathbf{x}) = (\mathbf{W}_3 \circ \sigma_2 \circ \mathbf{W}_2 \circ \sigma_1 \circ \mathbf{W}_1)(\mathbf{x})$	$\text{KAN}(\mathbf{x}) = (\Phi_3 \circ \Phi_2 \circ \Phi_1)(\mathbf{x})$
Model (Deep)	(c) 	(d) 

<https://openreview.net/forum?id=0zo7qJ5vZi>