Natural Language Processing

Prof. Dr. Jannik Strötgen
jannik.stroetgen@h-ka.de

Summer 2024

Hochschule Karlsruhe
University of
Applied Sciences

Fakultät für
Informatik und
Wirtschaftsinformatik

Reminder

- Please join the ILIAS course:
  https://ilias.h-ka.de/goto.php?target=crs_853687&client_id=HSKA

Password:
NLP2024NLP

Thank you!

Organisation

Exercise Sheets and Presentations

- **ex00** (python tutorial) covered in last week‘s tutorial
- **ex01** (will be) released via ILIAS and (briefly) covered in this week’s tutorial
- **Solutions of ex01** will be presented in next week’s tutorial (April 3, 2024)
- We aim at releasing exercise sheets weekly, i.e., you have 1 week to work on them

Presentations during the tutorial sessions:

Date	Exercise sheet	Presenters	Presenters' solution
April 3, 2024	ex_01	Simon, Oliver, Lea	

Preliminary Agenda

Date	Topic
19.03.2024	Organisation & motivation
26.03.2024	Introduction to NLP
02.04.2024	*no lecture (Easter)*
09.04.2024	Pre-Processing and Part-of-Speech Tagging
16.04.2024	Parsing
23.04.2024	Named Entity Recognition and Disambiguation
30.04.2024	Similarity and Search
07.05.2024	Language Models: Static Word Embeddings

Date	Topic
14.05.2024	Contextual Embeddings
21.05.2024	*no lecture (Whitsun break)*
28.05.2024	Text Mining and Sentiment Analysis
04.06.2024	Information Extraction & QA
11.06.2024	Applications exploiting NLP
18.06.2024	NLP with LLMs
25.06.2024	My Research Topics
02.07.2024	Recap, exam preparation

[1] Foundations &
[2] Pre-processing

Outline

[1] **Foundations**

- **Again: Why is natural language processing (NLP) difficult?**
- Evolution of NLP methods
- Crash course: Elements of language
- Morphology

[2] Pre-processing

- Text segmentation
  - Sentence splitting
  - Tokenization
- Token normalization
  - Stemming
  - Lemmatization

[1] Foundations
Again: Why is NLP difficult?

Let‘s Brainstorm

Let’s think about the input...

- What is language?
- What is text?
- What does it contain?
- How is it represented?

... and the output:

- What do we want to achieve?
- What tasks do we have to solve?
- What output do we need to generate?

Example: Low-Resource NLP

Hedderich et al. (2021): A survey on recent approaches for natural language processing in low-resource scenarios. NAACL'21.

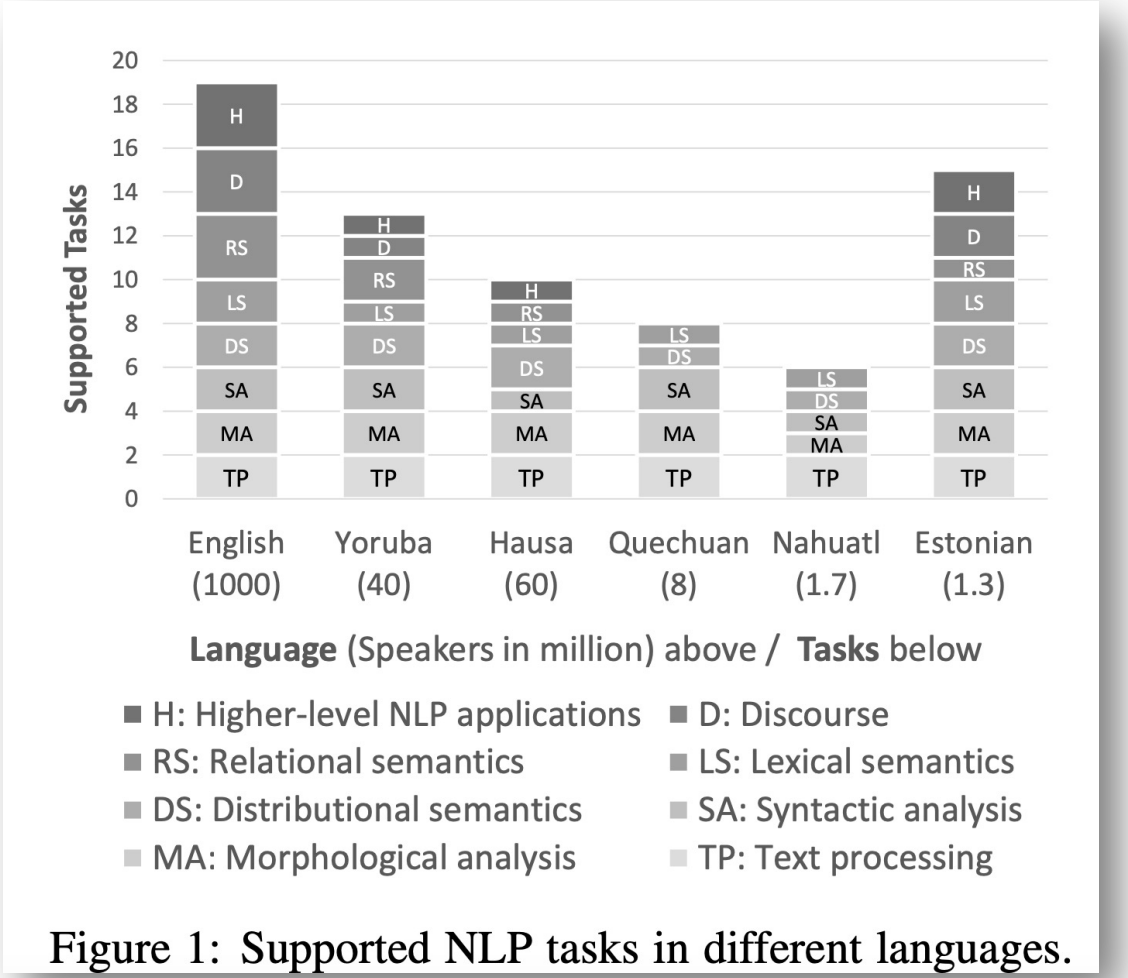
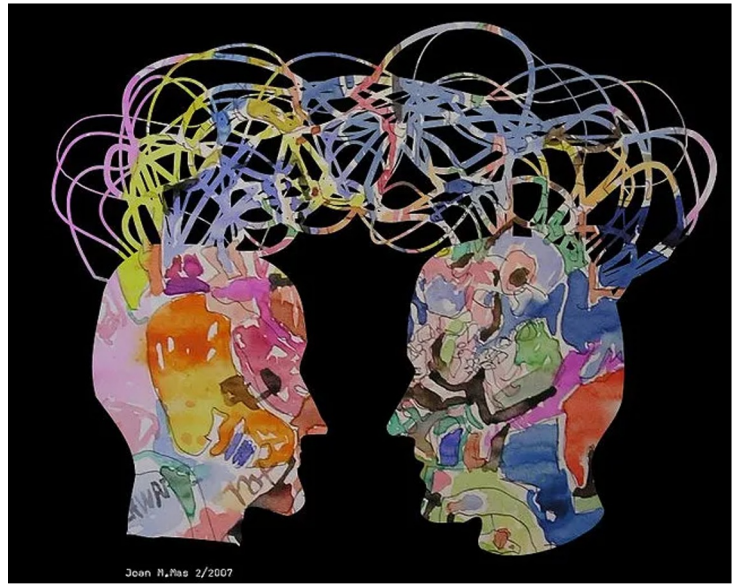


Figure 1: Supported NLP tasks in different languages.

Social Communication

Function of language:

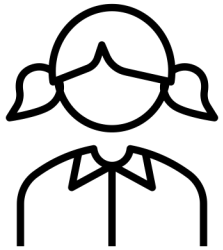
- Language use is a social ability
- Language facilitates **communication** between people
- Communication is the **transfer of information**
- Language presupposes **knowledge about the world**
- Language only reflects the **surface of meaning**
- There is evidence that parts of the human brain developed to facilitate language understanding

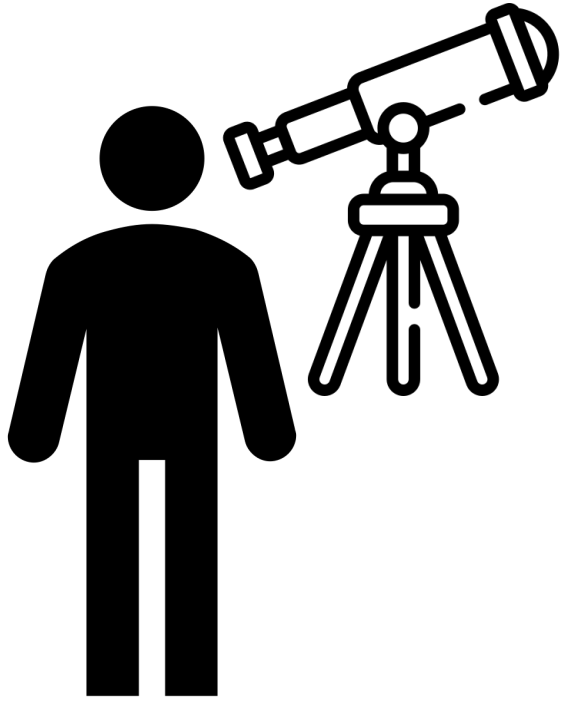


"Watercolour blobs from a paint trial sheet, masked with one of my drawings —collected in the book 1000 Heads". Joan M. Ma/Flickr/Some Rights Reserved

But: computers have no brain and do not socialize!

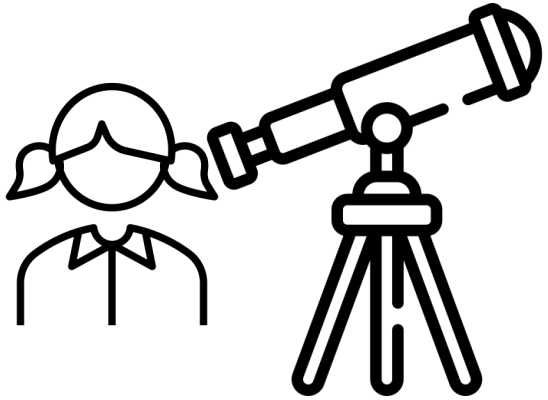
Ambiguity: Syntactic Ambiguity

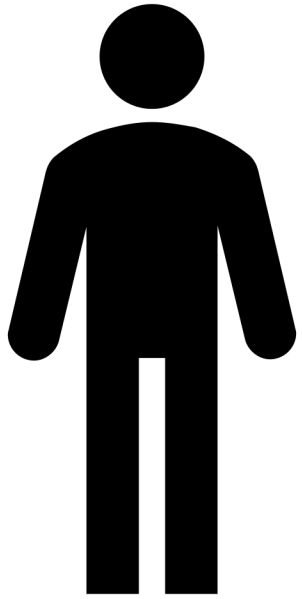




The girl saw the man with the telescope.

Ambiguity: Syntactic Ambiguity





The girl saw the man with the telescope.

Ambiguity: Anaphora Resolution

We gave the monkeys bananas because **they** were hungry.

vs.

We gave the monkeys bananas because **they** were over-ripe.

Ambiguity: Interpretation & World Knowledge

Every American has a president.

vs.

Every American has a mother.

Some Ambiguity Examples

speech recognition

- it's hard to recognize speech OR it's hard to wreck a nice beach

prepositional attachment

- the boy saw the man with a telescope

syntax / morphology

- time flies (noun / verb) like (verb / preposition) an arrow

parts-of-speech

- can we can fish in a can

semantic ambiguities

- bank

Heteronyms

content, noun

≠

content, adjective

Definition:

- The topics treated in a written work.

Example:

- The content is exquisitely written.

Definition:

- contented, satisfied

Example:

- She was content with her life as it was.

These are just two of the four possible meanings...
https://www.merriam-webster.com/dictionary/content

Outline

[1] Foundations

- Again: Why is natural language processing (NLP) difficult?
- **Evolution of NLP methods**
- Crash course: Elements of language
- Morphology

[2] Pre-processing

- Text segmentation
  - Sentence splitting
  - Tokenization
- Token normalization
  - Stemming
  - Lemmatization

[1] Foundations
Evolution of NLP Methods

Resolving these Problems (as a Human)

Rules and Memorization

- Current thinking in psycholinguistics: we combine rules and memorization
- Similar mechanism to Two System Thinking
  - System 1: fast, instinctive, and emotional
  - System 2: slow, deliberative, and logical
- Note: this is still **quite controversial** (and not our problem in this lecture)

Mechanism

- If there is an applicable rule, apply the rule
- If there is a memorized version, memory takes precedence.
- Example: recognizing past tenses of irregular verb forms
  - think → thought vs. blink → blinked

Resolving these Problems Computationally

Consecutive evolution of three major types of approaches

- Rule-based models (until the 1990s)
- Statistical models / corpus linguistics (until the 2010s)
- (Deep) neural network models (work in progress)

Rule-based NLP

- Languages are inherently rule-based (to a point)
- A skilled linguist can create rules that solve many problems in NLP quite well
- However, this does not scale, and meaning is not rule-based

→ Diminishing returns

Towards Modern Language Models (w/ Statistics)

Statistical language models and corpus linguistics

- The idea had been around longer, but in the 1990s, we reached thresholds for
  - computing power,
  - storage space, and
  - available text data.

Core idea: combine memorizing word cooccurrence patterns and (learned) rules

- How to do it:
  - Obtain large text collections (called **corpora**),
  - compute statistics over word (co)occurrences in those collections,
  - use these statistics to make “smarter” decisions with rules,
  - or learn rules directly from the statistics

Distributional Semantics

“You shall know a word by the company it keeps”

J. R. Firth, 1957

[to keep so company:
idiom. the influence of the people that someone spends time with]

(also known as the the most cliché quotation in all of NLP)

Corpus Linguistics Example: Grammar Checker

Task: Decide whether to use <principal> or <principle> in the sentence:

```
<Principal|Principle> Skinner attempts to institute discipline
at Springfield Elementary School, with an uptight attitude.
```

Solution: look at which words surround each use in the corpus:

- I am in my third year as the principal of Anamosa High School.
- School-principal transfers caused some upset.
- This is a simple formulation of the quantum mechanical uncertainty principle.
- Power without principle is barren, but principle without power is futile.

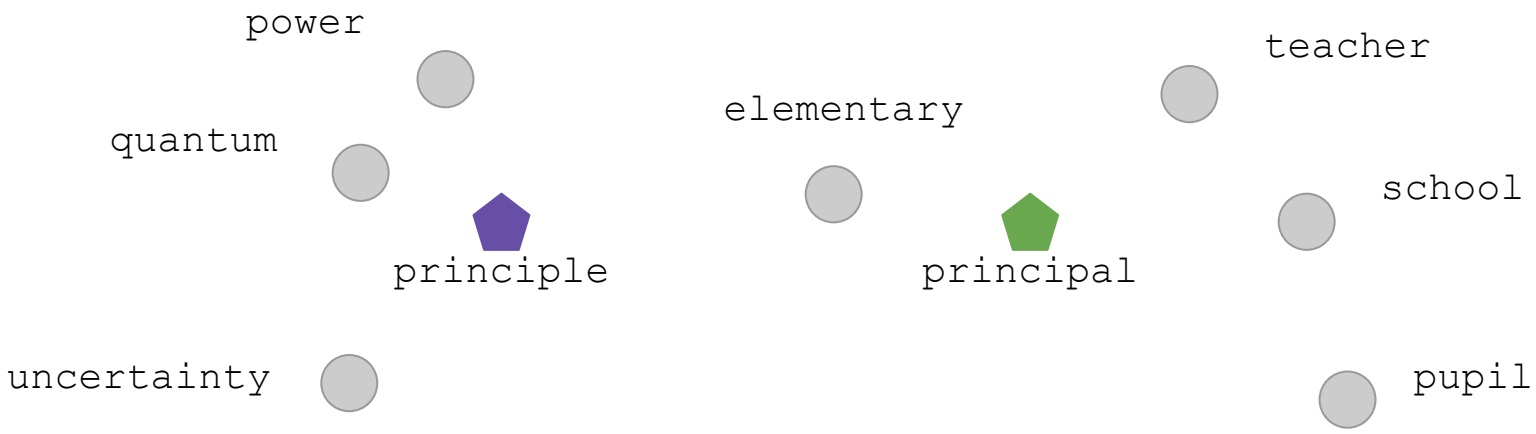
Corpus Linguistics Example: Grammar Checker

Algorithm:

Keep track of all words that cooccur with each spelling in the corpus, e.g.:

- Principal: school
- Principle: uncertainty

At grammar-check time, choose spelling best predicted by the “surrounding” words.



Why Size Matters (for Corpora)

The 207 most frequent word forms in German

30 forms constitute 31.8 % of the words:

der, die, und, in, zu, den, das, nicht, von, sie, ist, des, sich, mit, dem, dass, er, es, ein, ich, auf, so, eine, auch, als, an, nach, wie, im, für

70 forms constitute further 15.3 % of the words:

man, aber, aus, durch, wenn, nur, war, noch, werden, bei, hat, wir, was, wird, sein, einen, welche, sind, oder, zur, um, haben, einer, mir, über, ihm, diese, einem, ihr, uns, da, zum, kann, doch, vor, dieser, mich, ihn, du, hatte, seine, mehr, am, denn, nun, unter, sehr, selbst, schon, hier, bis, habe, ihre, dann, ihnen, seiner, alle, wieder, meine, Zeit, gegen, vom, ganz, einzelnen, wo, muss, ohne, eines, können, sei

107 forms constitute 7.25 % of the words:

ja, wurde, jetzt, immer, seinen, wohl, dieses, ihren, würde, diesen, sondern, weil, welcher, nichts, diesem, alles, waren, will, Herr, viel, mein, also, soll, worden, lassen, dies, machen, ihrer, weiter, Leben, recht, etwas, keine, seinem, ob, dir, allen, großen, Jahre, Weise, müssen, welches, wäre, erst, einmal, Mann, hätte, zwei, dich, allein, Herren, während, Paragraph, anders, Liebe, kein, damit, gar, Hand, Herrn, euch, sollte, konnte, ersten, deren, zwischen, wollen, denen, dessen, sagen, bin, Menschen, gut, darauf, wurden, weiß, gewesen, Seite, bald, weit, große, solche, hatten, eben, andern, beiden, macht, sehen, ganze, anderen, lange, wer, ihrem, zwar, gemacht, dort, kommen, Welt, heute, Frau, werde, derselben, ganzen, deutschen, lässt, vielleicht, meiner

Zipf‘s Law

Quelle: http://en.wikipedia.org

George K. Zipf (1902–1950, American linguist)

- Observation:
  few words are very frequent, most words are rather seldom

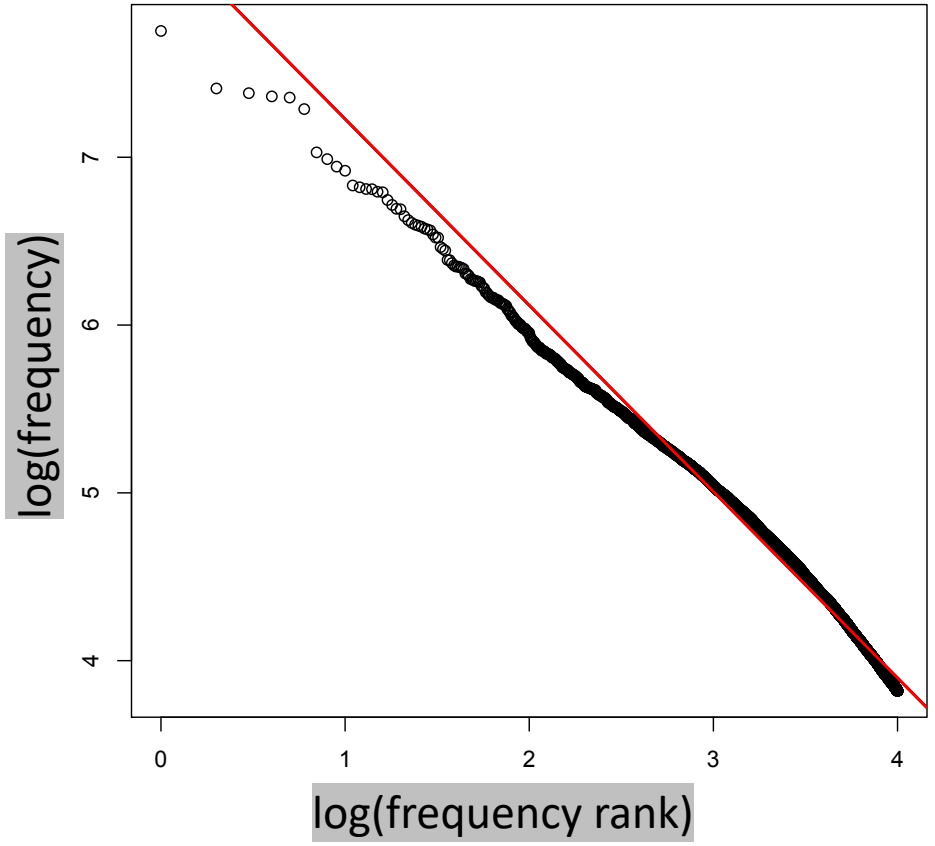
$$f(w) \propto \frac{1}{r(w)^{\alpha}}$$

Collection Frequency

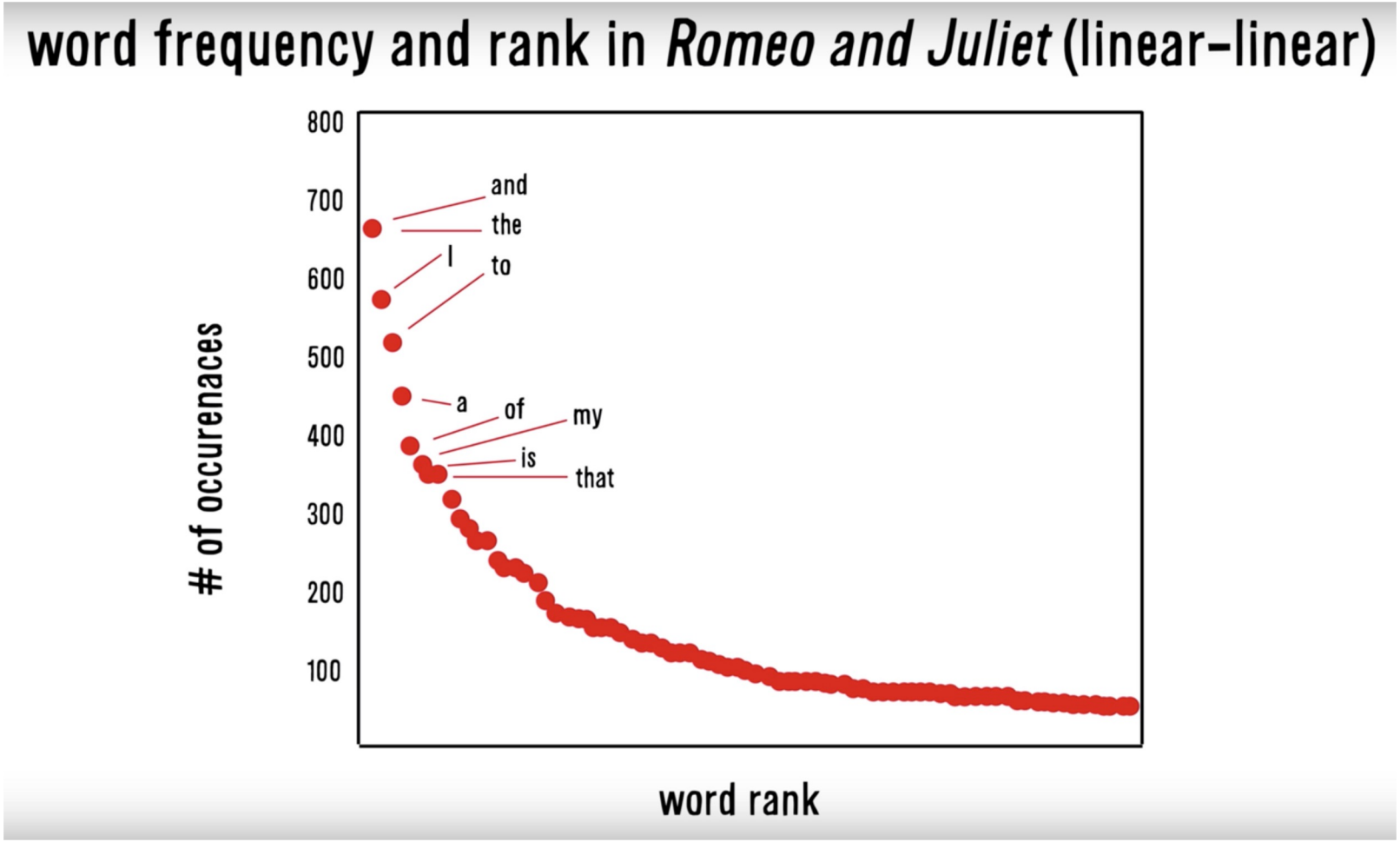
- frequency of a word *f(w)* (all occurrences in all documents) is inversely proportional to rank *r(w)* of that word
- Valid for most texts and languages (with different α)
- For English: α = 1

Zipf‘s Law

- **Word frequency** und **frequency rank** in **The New York Times** (1987-2007)



Zipf‘s Law



medium.datadriveninvestor.com

Deep Learning for NLP

Neural language models:

- In the early 2000s, we reached new thresholds for
  - **parallel** computing power,
  - storage space, and
  - available text data.
- Core idea: feed **web-scale corpora** of unlabeled text into (deep) neural networks with billions of parameters to **learn word cooccurrence** patterns
- Current deep neural language models:
  - can solve **many NLP tasks** better than humans
  - can solve tasks for which they have not or barely been trained (transfer learning, **zero-shot learning**)
  - can be quite **biased** (garbage in, garbage out)

Outline

[1] Foundations

- Again: Why is natural language processing (NLP) difficult?
- Evolution of NLP methods
- **Crash course: Elements of language**
- Morphology

[2] Pre-processing

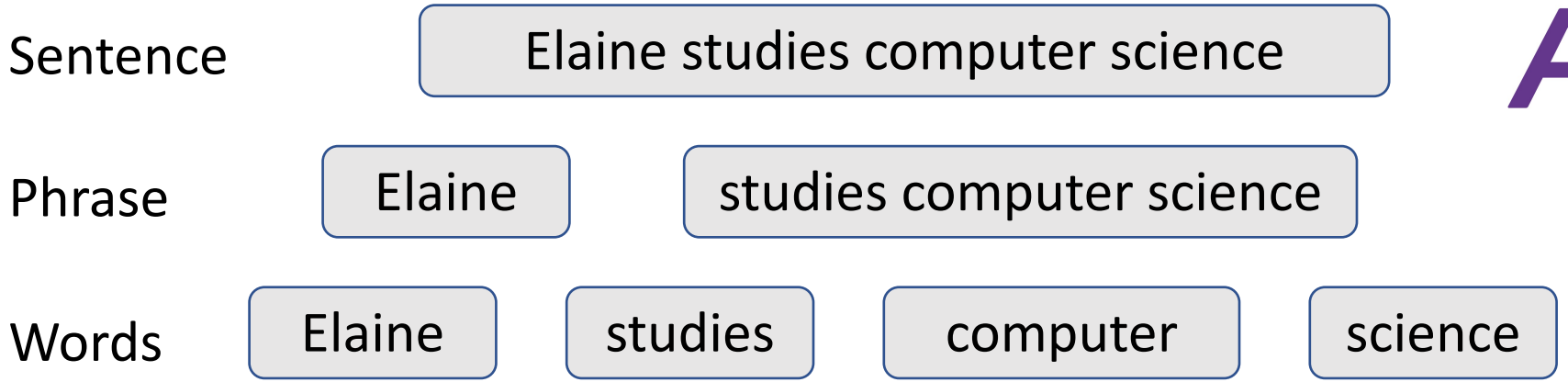
- Text segmentation
  - Sentence splitting
  - Tokenization
- Token normalization
  - Stemming
  - Lemmatization

[1] Foundations
Elements of Language

Elements of Language

Language	
Sound	1. Phonetics
Grammar	2. Phonology 3. Morphology 4. Syntax
Meaning	5. Semantics

NLP Hierarchy



Words can be subdivided further:

- Phonemes: studies → /ˈstʌd·iz/
- Characters: studies → s-t-u-d-i-e-s
- **Morphemes**: studies → studi-es
- Syllables: studies → stud-ies
- **Stem**: studies → studi
- **Lemma**: studies → study

Outline

[1] Foundations

- Again: Why is natural language processing (NLP) difficult?
- Evolution of NLP methods
- Crash course: Elements of language
- **Morphology**

[2] Pre-processing

- Text segmentation
  - Sentence splitting
  - Tokenization
- Token normalization
  - Stemming
  - Lemmatization

[1] Foundations Morphology

Morphology

JABBERWOCKY

Lewis Carroll

(from *Through the Looking-Glass and What Alice Found There*, 1872)

`Twas brillig, and the slithy toves
Did gyre and gimble in the wabe:
All mimsy were the borogoves,
And the mome raths outgrabe.

"Beware the Jabberwock, my son!
The jaws that bite, the claws that catch!
Beware the Jubjub bird, and shun
The frumious Bandersnatch!"

He took his vorpal sword in hand:
Long time the manxome foe he sought --
So rested he by the Tumtum tree,
And stood awhile in thought.

And, as in uffish thought he stood,
The Jabberwock, with eyes of flame,
Came whiffling through the tulgey wood,
And burbled as it came!

Morphology

JABBERWOCKY

Lewis Carroll

(from *Through the Looking-Glass and What Alice Found There*, 1872)

`Twas brillig, and the slithy toves
Did gyre and gimble in the wabe:
All mimsy were the borogoves,
And the mome raths outgrabe.

"Beware the Jabberwock, my son!
The jaws that bite, the claws that catch!
Beware the Jubjub bird, and shun
The frumious Bandersnatch!"

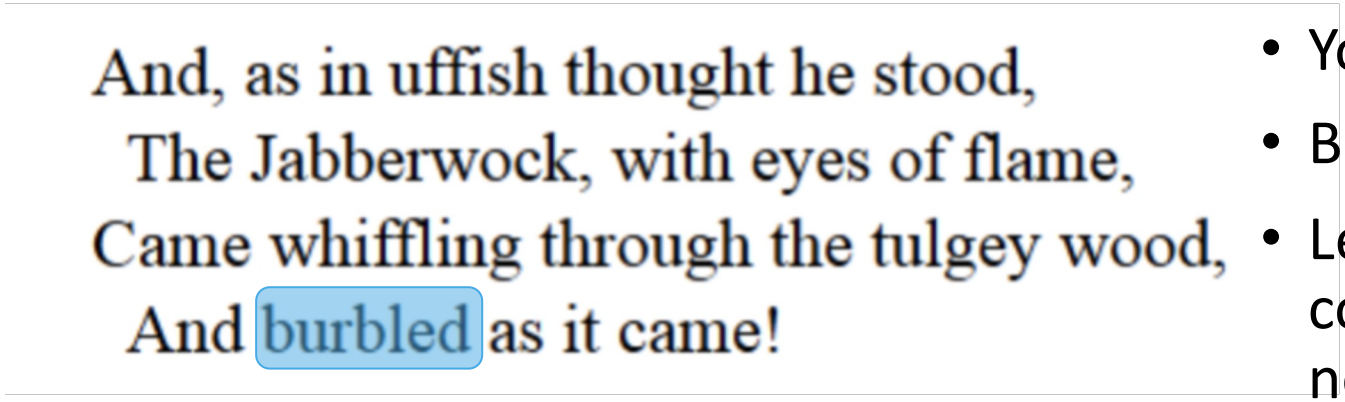
He took his vorpal sword in hand:
Long time the manxome foe he sought --
So rested he by the Tumtum tree,
And stood awhile in thought.

And, as in uffish thought he stood,
The Jabberwock, with eyes of flame,
Came whiffling through the tulgey wood,
And burbled as it came!

An analysis of the poem:

- This is clearly nonsense.
- ... but is it?
- This is clearly not English.
- ... but it is also kind of English:
  It is much more like English than it is like French or German or Chinese, etc.
- So why do we pretty much understand the words and can paint a mental picture?

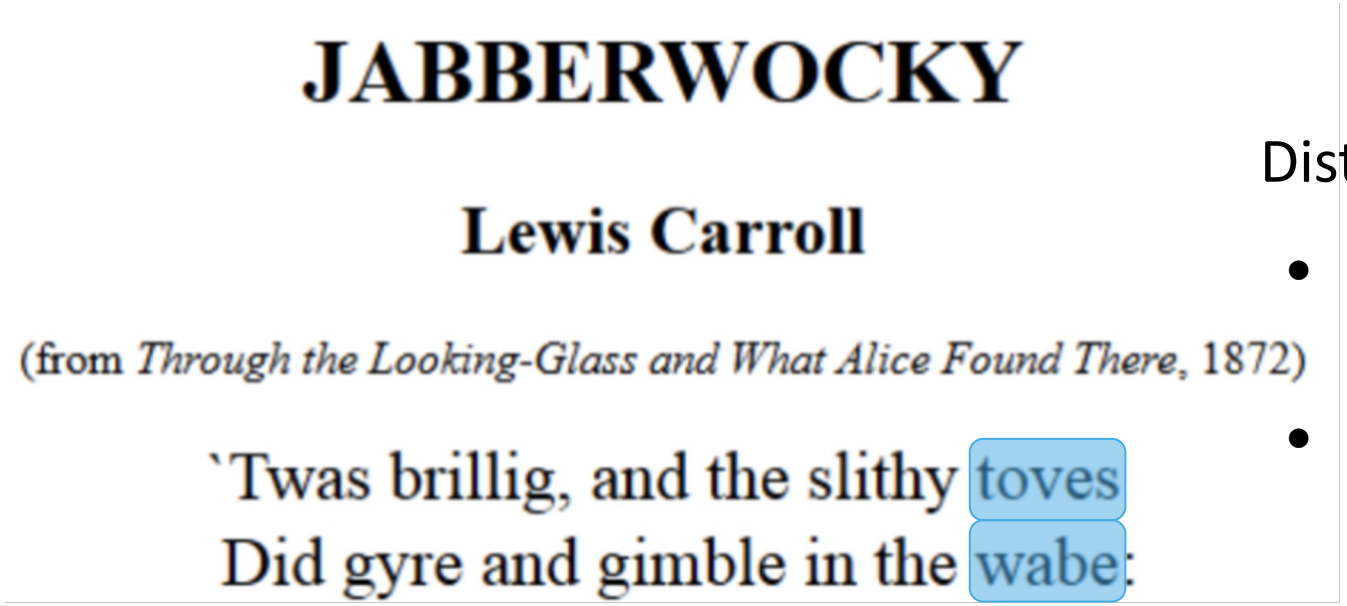
Morphology



We recognize combinations of morphemes.

- You may not know the word **burble**.
- But you know **bubble**, **blubber** and **gurgle**.
- Lewis Carroll is well known for creating such combination (**portmanteaus**), which are now in common use
- burble, verb
  bur·ble | 'bər-bəl
  Definition: to make a bubbling sound

Morphology



JABBERWOCKY

Lewis Carroll

(from *Through the Looking-Glass and What Alice Found There*, 1872)

`Twas brillig, and the slithy toves
Did gyre and gimble in the wabe:

Distributional semantics also help:

- Surrounding English words indicate the parts-of-speech of the nonsense words.
- **toves**: Something that can probably perform an action (because they did **gyre** and **gimble**)
- **wabe**: Probably a place (because the toves did something **in** the wabe)

Morphology (Terminology)

Morphology:

- The study of the way in which words are constructed from smaller units of meaning.

Morpheme:

- The smallest meaningful unit in the grammar of a language.

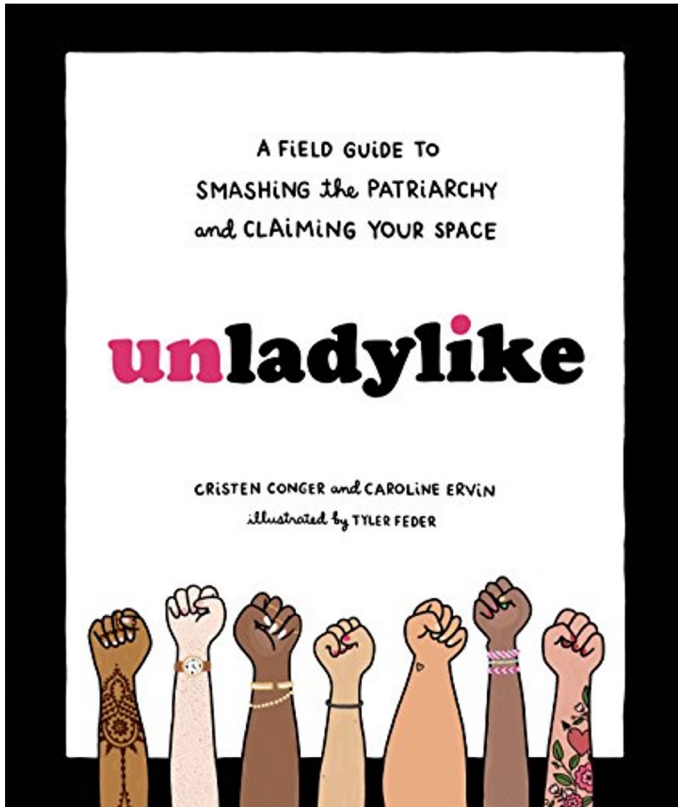
Contrasts in morphology:

- Derivational morphology vs. inflectional morphology
- Concatenative morphology vs. templatic morphology (root-and-pattern)

Morphology (Examples)

unladylike

- 3 morphemes, 4 syllables
  - un-: meaning “not”
  - lady: meaning “(well behaved) female adult human”
  - -like: meaning ”having the characteristics of”
- We cannot further break these down without distorting the meaning of the units



Inflectional vs. Derivational Morphology

Typical word classes (parts of speech): noun, verb, adjectives, etc.

Inflection:

- Variation in the form of a word, typically by means of an affix, that expresses a grammatical contrast.
- Does not change the word class
- Usually produces a predictable, non-idiosyncratic change of meaning.
- Example: run → runs | running

Derivation:

- The formation of a new word or inflectable stem from another word or stem.
- Example: compute → computer → computerization

Inflectional Morphology

Typically, inflectional morphology

- Does not change the word class
- Modifies words to serve a new grammatical role
- Adds morphemes that encode
  - tense, number, person, mood, aspect, etc.
- English example:
  - The pizza guy **arrives** at noon.
  - **arrive** is inflected for **person** (3rd person) and **number** (singular)
- Spanish example:
  - **Las manzanas rojas** son deliciosas. (The red apples are delicious).
  - **Las** and **rojas** are inflected for agreement with **manzanas** in grammatical gender by **-a** and in number by **-s**

Derivational Morphology

Nominalization: formation of nouns from other parts of speech (en: mainly verbs):

- compute → computerization
- appoint → appointee
- kill → killer
- fuzzy → fuzziness

Formation of adjectives (primarily from nouns)

- computer → computational
- clue → clueless
- embrace → embraceable

This process is not always entirely regular by simply adding morphemes (**comput**er → **comput**ational)

Morphemes (Terminology)

Root: The component of the word that
- is common to a set of derived or inflected forms, if any, when all affixes are removed
- cannot be split further into meaningful components
- carries the principal portion of meaning of the words
- For example: studying / studies → stud

Stem: The common part to all inflected words.
- Is not always a proper word from the dictionary
- For example: studying / studies → studi

Lemma: The base form of a word (which can be found in a dictionary)
- For example: studying / studies → (to) study

Morphemes (Terminology)

Affix: (prefix, infix, suffix)

- A bound morpheme that is joined before, after, or within a root or stem.
- English Example:
  - This is **un**believable! (The prefix **un-** modifies the word **believable**)

Clitic:

- A morpheme that functions syntactically like a word, but does not appear as an independent phonological word
- English Examples:
  - Hal**’s** shop (genitive marker ‘s)
  - What**’s** going on? (shortened form of is)

Outline

[1] Foundations

- Again: Why is natural language processing (NLP) difficult?
- Evolution of NLP methods
- Crash course: Elements of language
- Morphology

[2] Pre-processing

- Text segmentation
  - Sentence splitting
  - Tokenization
- Token normalization
  - Stemming
  - Lemmatization

From Last Week

1. Pre-processing

Thdistinctlyirememberitwasinthebleakdecemberandeachseparatedyingember
wroughtitsghostuponthefloореagerlyiwishedthemorrowvainlyIhadsoughttobor
rowfrommybookssurceaseofsorrowsorrowforthelostlenorefortherareandradia
ntm heangelsnamelenorenamelesshereforevermore

adj: distinct
adv: distinctly

Ah, distinctly I remember it was in the bleak December;
And each separate dying ember wrought its ghost upon the floor.
Eagerly I wished the morrow;—vainly I had sought to borrow
From my books surcease of sorrow—sorrow for the lost Lenore—
For the rare and radiant maiden whom the a Lenore—
Nameless here for evermore.

to seek
1. person sg.
past perfect

15. März 2023 Natural Language Processing @ HKA - Prof. Dr. Jannik Strötgen 10

Outline

[1] Foundations

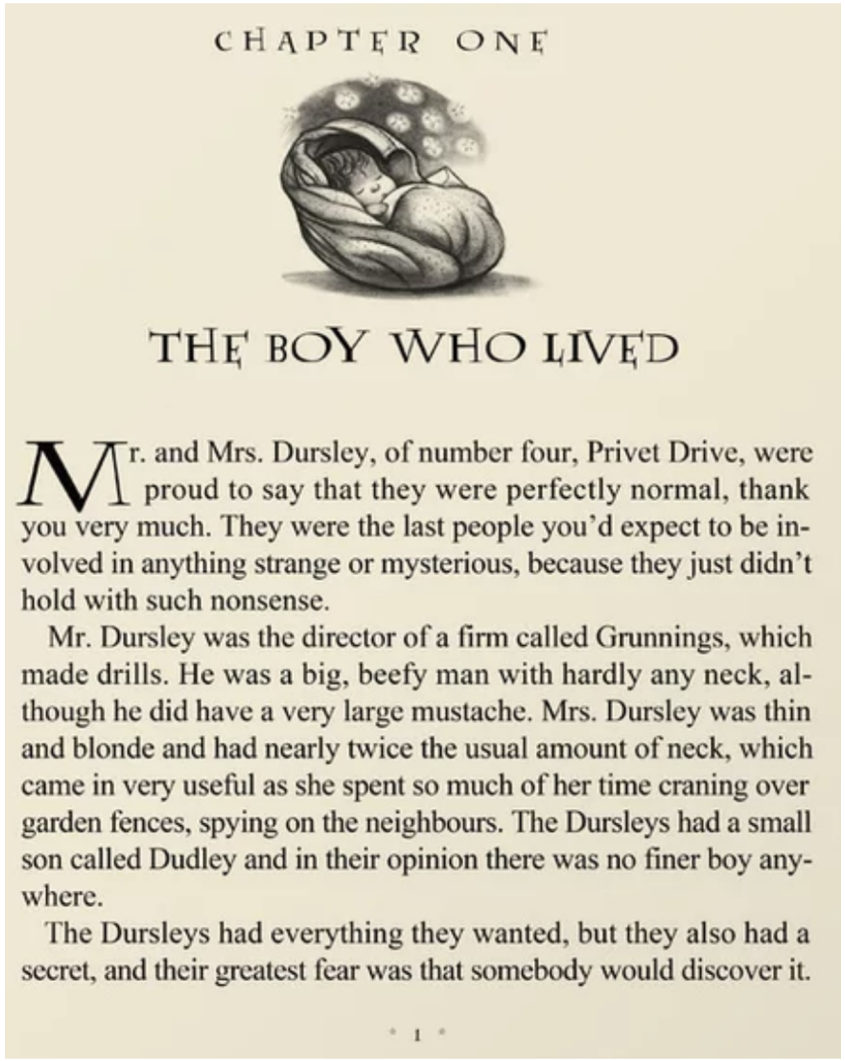
- Again: Why is natural language processing (NLP) difficult?
- Evolution of NLP methods
- Crash course: Elements of language
- Morphology

[2] Pre-processing

- **Text segmentation**
  - Sentence splitting
  - Tokenization
- Token normalization
  - Stemming
  - Lemmatization

[2] Pre-processing
Text Segmentation

Segments of Text



CHAPTER ONE

THE BOY WHO LIVED

Mr. and Mrs. Dursley, of number four, Privet Drive, were proud to say that they were perfectly normal, thank you very much. They were the last people you'd expect to be involved in anything strange or mysterious, because they just didn't hold with such nonsense.

Mr. Dursley was the director of a firm called Grunnings, which made drills. He was a big, beefy man with hardly any neck, although he did have a very large mustache. Mrs. Dursley was thin and blonde and had nearly twice the usual amount of neck, which came in very useful as she spent so much of her time craning over garden fences, spying on the neighbours. The Dursleys had a small son called Dudley and in their opinion there was no finer boy anywhere.

The Dursleys had everything they wanted, but they also had a secret, and their greatest fear was that somebody would discover it.

1

Text (in book form) could be segmented into

- Volumes
- Books
- Chapters
- Paragraphs
- **Sentences**
- **Tokens (and punctuation)**
- Characters

But other forms of text might be structured differently (e.g., websites, tweets, patents, etc.)

Sentence Splitting

Using whitespaces and punctuation, sentences should not be too difficult to detect… right?

- ! and ? are relatively unambiguous
- A period "." is quite ambiguous, but only has a few cases such as
  - Sentence boundaries
  - Abbreviations
  - Numbers
  - etc.

...at least as long as we have punctuation.

- Classical Greek and Latin texts use no punctuation
- Classical Chinese considers punctuation as optional
- Thai is using whitespaces instead of punctuation

Error Propagation

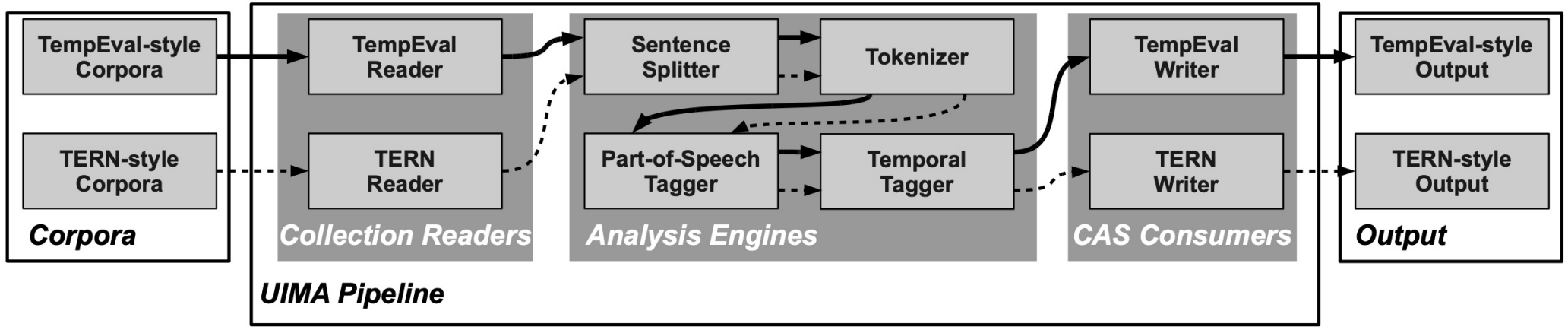


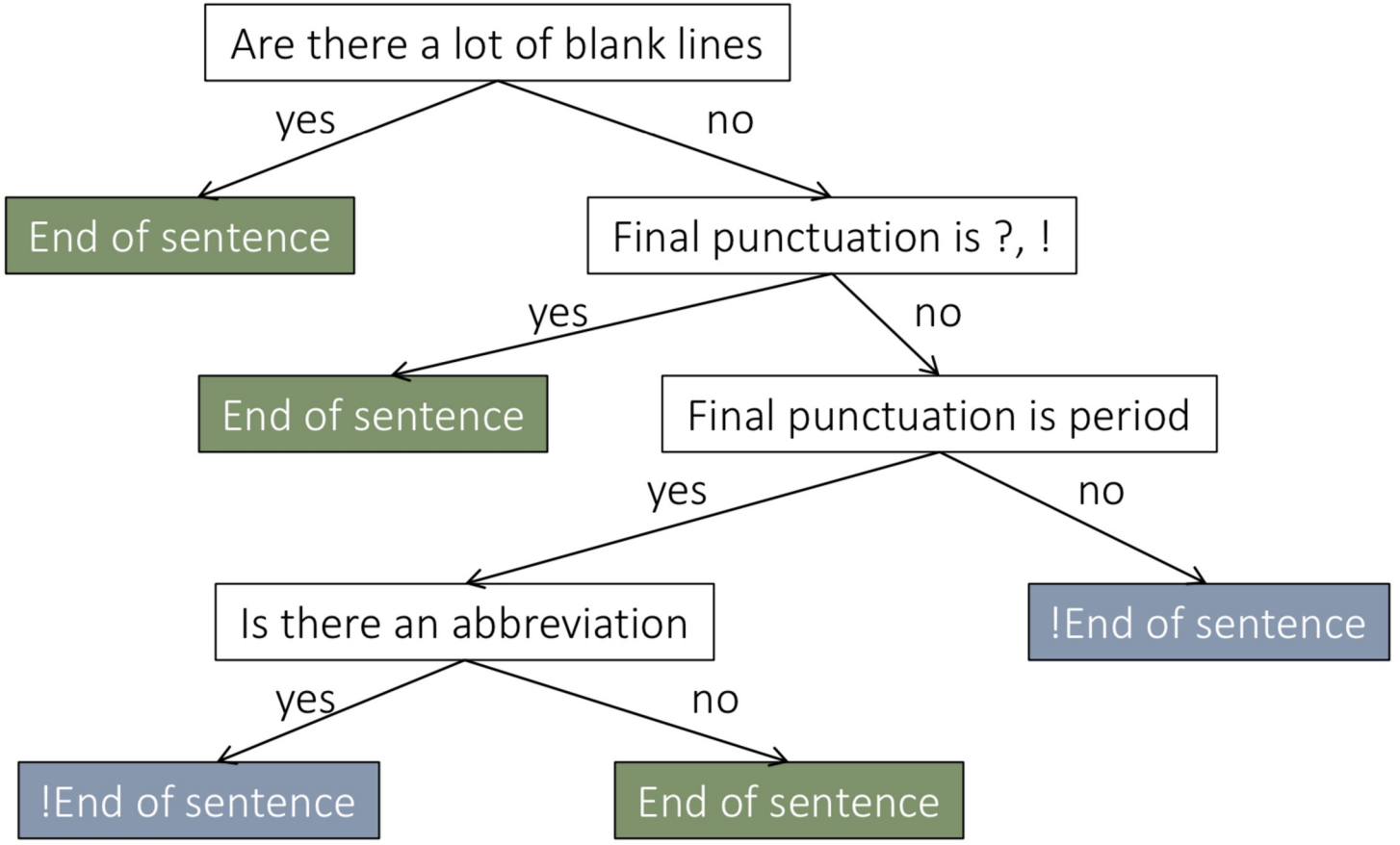
Figure 2.4: An example UIMA pipeline for temporal tagging corpora in two different formats.

Jannik Strötgen: ***Domain-sensitive Temporal Tagging for Event-centric Information Retrieval.*** PhD thesis, Institute of Computer Science, Heidelberg University, 2015.

Naïve Sentence Splitting Algorithm

We could consider a binary classifier:

- Sequentially iterate over all tokens and punctation
- Decide whether the sentence ends here



Date Expressions (in German)

"Ende August war da, der Hochzeitstag (3. Oktober) rückte näher, [...]"

Theodor Fontane: *Effi Briest* (ersch. 1896)

"Den 20. Jänner ging Lenz durchs Gebirg."

Georg Büchner: *Lenz* (ersch. 1839)

Sentence Splitting as an “Easy” Problem

Beware! Sentence splitting…

- Is often considered a solved problem
- has strong language dependencies,
- can still be quite complicated,
- often has varying performance and output when using different tools.

Sentence Boundary Detection: A Long Solved Problem?

Jonathon READ, Rebecca DRIDAN, Stephan OEPEN, Lars Jørgen SOLBERG
University of Oslo, Department of Informatics
{jread|rdridan|oe|larsjol}@ifi.uio.no

ABSTRACT
We review the state of the art in automated sentence boundary detection (SBD) for English and call for a renewed research interest in this foundational first step in natural language processing. We observe severe limitations in comparability and reproducibility of earlier work and a general lack of knowledge about genre- and domain-specific variations. To overcome these barriers, we conduct a systematic empirical survey of a large number of extant approaches, across a broad range of diverse corpora. We further observe that much previous work interpreted the SBD task too narrowly, leading to overly optimistic estimates of SBD performance on running text. To better relate SBD to practical NLP use cases, we thus propose a generalized definition of the task, eliminating text- or language-specific assumptions about candidate boundary points. More specifically, we quantify degrees of variation across ‘standard’ corpora of edited, relatively formal language, as well as performance degradation when moving to less formal language, viz. various samples of user-generated Web content. For these latter types of text, we demonstrate how moderate interpretation of document structure (as is now often available more or less explicitly through mark-up) can substantially contribute to overall SBD performance.

KEYWORDS: Sentence Boundary Detection, Segmentation, Comparability, Reproducibility.

Tokenization

Token:

- The occurrence of a word in a text

Tokenization:

- Segmentation of an input stream into an ordered sequence of tokens

Tokenizer:

- A system that splits texts into word tokens

Example:

- Input text: John likes Mary and Mary likes John.
- Tokens: {“John”, “likes”, “Mary”, “and”, “Mary”, “likes”, “John”, “.”}

Token vs. Type

Example:

- Input text: *John likes Mary and Mary likes John.*
- Tokens: {“John”, “likes”, “Mary”, “and”, “Mary”, “likes”, “John”, “.”}
- Types: {“John”, “likes”, “Mary”, “and”, “.”}

Or with the example of a prototypical NLP class

- Input text: *a rose is a rose is a rose*
- Tokens: {“a”, “rose”, “is”, “a”, “rose”, “is”, ”a”, “rose”}
- Types: {“a”, “rose”, “is”}

Naïve Tokenization Algorithm

Naïve approach:

- Split tokens at whitespace characters

Example:

- Input: Mr. Sherwood said, reaction to Sea Containers‘ proposal has been “very positive“. In New York Stock Exchange composite trading yesterday, Sea Containers closed at $62.625, up 62.5 cents.

- (Partial) output: {said, positive”. $62.625, cents.}

Tokenization Issues: Ambiguous Punctation

Periods are ambiguous

- In most of the cases: Final sentence punctuation symbol
- Could be part of an abbreviation: e.g., U.N., etc.
- Could be ordinal numbers: 21.
- Could be numbers with fractions: 3.14 (English) or 3,14 (German)
- Could be references to resource locators: www.h-ka.de
- ...

Whitespace characters cause similar issues

- Could be part of large numbers: 1 543
- Could be part of a multi-word expressions: New York *(if we want one or two tokens depends…)*

Similar issues with commas, dashes, semicolons, single quotes, etc.

Tokenization Issues: Special Cases

There tend to be irregular cases and spelling variations in any language that require us to make a decision, based on what we want to do with the tokenized data.
Consider the English examples:

- `Finland’s capital → Finland | Finlands | Finland’s`
- `What’re | I’m | isn’t → What are | I am | is not`
- `Hewlett-Packard → Hewlett Packard`
- `State-of-the-art → state of the art`
- `Lowercase → lower-case | lowercase | lower case`
- `San Francisco ->` one token or two tokens?
- `PhD. →` ??

Tokenization Issues: Agglutinative Languages

Some languages use agglutination as a morphological feature.

German is famous for long compound nouns:

- `[de] Lebensversicherungsgesellschaftsangestellter`
- `[en] life insurance company employee`

If we want to understand the meaning of such words or even just retrieve information from such texts, we also need to split such compound words.

Tokenization Issues: Sinitic Languages

Chinese Example: **乒乓球拍賣光了**

- No whitespaces
- Multiple characters can constitute a word
- Disambiguation requires contextual information

乒乓　　球拍　　賣光了
ping-pong　racket　sold out
(The ping-pong rackets have been sold out.)

乒乓球　　拍賣　　光了
ping-pong ball　auction　finish
(The auction of the ping-pong ball has finished.)

Tokenization Algorithm: Maximum Matching

Core idea:

- Consider the input sentence as a simple word sequence
- Use a dictionary
- Start at the beginning of the sentence
- Search for the longest matching string in the dictionary and cut
- Start matching against the dictionary again

(English) example inputs:

- `Thedoginthehat`
- `Thetabledownthere`

Tokenization Algorithm: Maximum Matching

Input: `Thedoginthehat`

`The|doginthehat`
`The dog|inthehat`
`The dog in|thehat`
`The dog in the|hat`
`The dog in the hat`

Input: `Thetabledownthere`

`Theta|bledownthere`
`Theta bled|ownthere`
`Theta bled own|there`
`Theta bled own there`

Instead of: `The table down there`

Obvious problem: Greedy matching makes locally optimal choices.

- Does not work well in English (where we should not need it)
- Works reasonably well in Chinese

Tokenization Libraries

In practice:

- Use one of the available NLP libraries
- Note: different libraries may produce different results!
- ...and so might different versions of the same library

	White spaces	Apache Open NLP 1.8.3 (Model)	Apache Open NLP 1.8.3 (Rules)	Stanford NLTK 3.8	Optimal
1		“	“	“	“
2	“I	I	I	I	I
3	said,	said	said	said	said
4		,	,	,	,
5			‘	‘	‘
6	‘what‘re	‘what	what	what	what
7			‘		
8		‘re	re	‘re	are
9	you?	you	you	you	you
10		?	?	?	?
11	Crazy?‘“	Crazy	Crazy	Crazy	Crazy
12		?	?	?	?
13		‘	‘	‘	‘
14		“	“	“	“
15	said	said	said	said	said
16	Sandowsky.	Sandowsky	Sandowsky	Sandowsky	Sandowsky
17		.	.	.	.
18		“	“	“	“
19	“I	I	I	I	I
20	can‘t	ca	can	ca	can
21			‘		
22		n‘t	t	n‘t	not
23	afford	afford	afford	afford	afford
24	to	to	to	to	to
25	do	do	do	do	do
26	that.“	that	that	that	that
27		.	.	.	.
28		“	“	“	“

68

Be Aware!

- Tokens are still the input for “advanced methods” (e.g., transformer-based models)

MEGABYTE: Predicting Million-byte Sequences with Multiscale Transformers

Lili Yu [*1] Dániel Simig [*1] Colin Flaherty [*2] Armen Aghajanyan [1] Luke Zettlemoyer [1] Mike Lewis [1]

Abstract

Autoregressive transformers are spectacular models for short sequences but scale poorly to long sequences such as high-resolution images, podcasts, code, or books. We propose MEGABYTE, a multi-scale decoder architecture that enables end-to-end differentiable modeling of sequences of over one million bytes. MEGABYTE segments sequences into patches and uses a *local* submodel within patches and a *global* model between patches. This enables sub-quadratic self-attention, much larger feedforward layers for the same compute, and improved parallelism during decoding—unlocking better performance at reduced cost for both training and generation. Extensive experiments show that MEGABYTE allows byte-level models to perform competitively with subword models on long context language modeling, achieve state-of-the-art density estimation on ImageNet, and model audio from raw files. Together, these results establish the viability of tokenization-free autoregressive sequence modeling at scale.

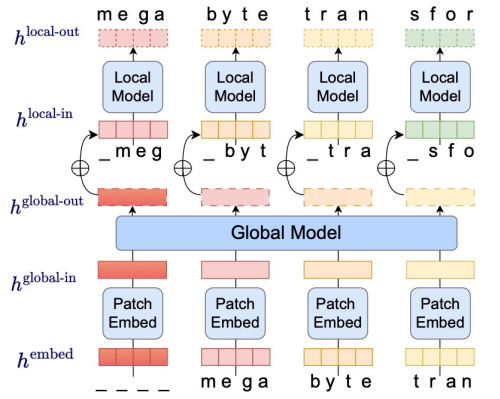


Figure 1. Overview of MEGABYTE with patch size $P = 4$. A small *local* model autoregressively predicts each patch byte-by-byte, using the output of a larger *global* model to condition on previous patches. Global and Local inputs are padded by P and 1 token respectively to avoid leaking information about future tokens.

7185v2 [cs.LG] 19 May 2023

Andrej Karpathy ✓
@karpathy

Promising. Everyone should hope that we can throw away tokenization in LLMs. Doing so naively creates (byte-level) sequences that are too long, so the devil is in the details.

Tokenization means that LLMs are not actually fully end-to-end. There is a whole separate stage with its own training and inference, and additional libraries. It complicates the ingest of additional modalities. Tokenization also has many subtle sharp edges. Few examples:

That "trailing whitespace" error you've potentially seen in Playground? If you end your (text completion API) prompt with space you are surprisingly creating a big domain gap, a likely source of many bugs: blog.scottlogic.com/2021/08/31/a-p...

Tokenization is why GPTs are bad at a number of very simple spelling / character manipulation tasks, e.g.: twitter.com/npew/status/15...

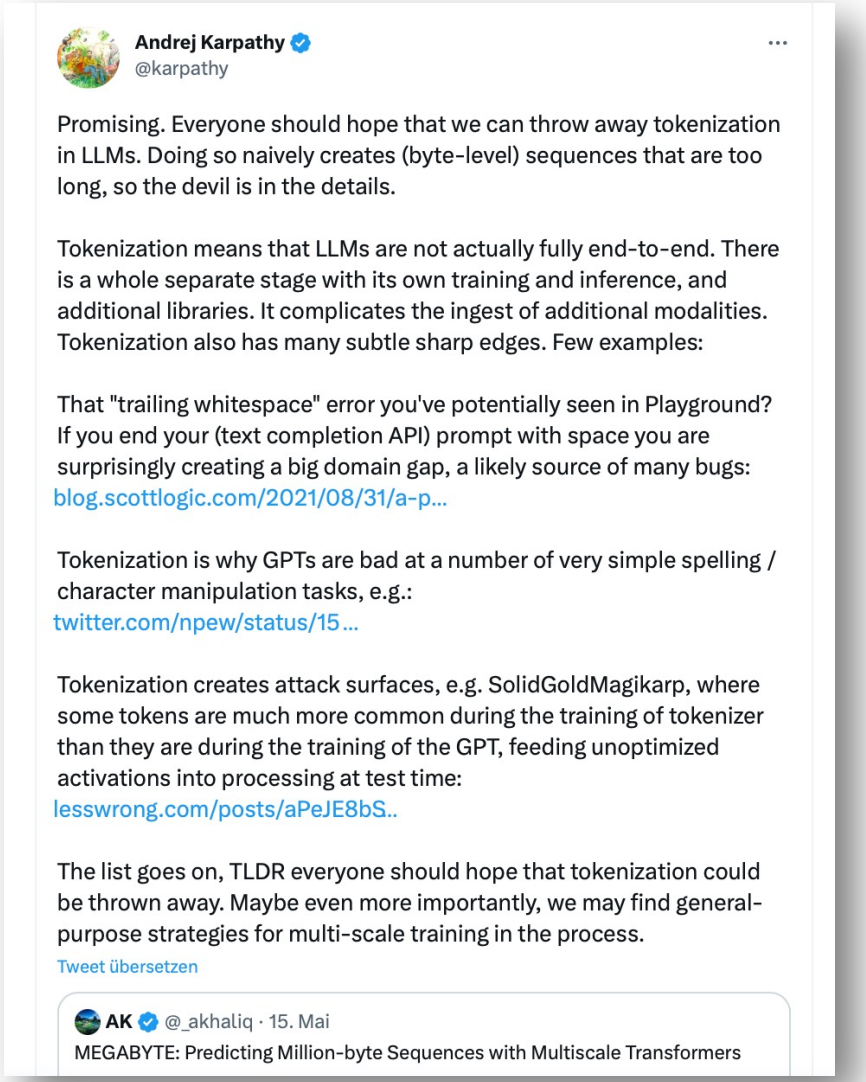
Tokenization creates attack surfaces, e.g. SolidGoldMagikarp, where some tokens are much more common during the training of tokenizer than they are during the training of the GPT, feeding unoptimized activations into processing at test time: lesswrong.com/posts/aPeJE8bS..

The list goes on, TLDR everyone should hope that tokenization could be thrown away. Maybe even more importantly, we may find general-purpose strategies for multi-scale training in the process.

Tweet übersetzen

AK ✓ @_akhaliq · 15. Mai
MEGABYTE: Predicting Million-byte Sequences with Multiscale Transformers

Be Aware!



Andrej Karpathy
@karpathy

Promising. Everyone should hope that we can throw away tokenization in LLMs. Doing so naively creates (byte-level) sequences that are too long, so the devil is in the details.

Tokenization means that LLMs are not actually fully end-to-end. There is a whole separate stage with its own training and inference, and additional libraries. It complicates the ingest of additional modalities. Tokenization also has many subtle sharp edges. Few examples:

That "trailing whitespace" error you've potentially seen in Playground? If you end your (text completion API) prompt with space you are surprisingly creating a big domain gap, a likely source of many bugs: blog.scottlogic.com/2021/08/31/a-p...

Tokenization is why GPTs are bad at a number of very simple spelling / character manipulation tasks, e.g.: twitter.com/npew/status/15...

Tokenization creates attack surfaces, e.g. SolidGoldMagikarp, where some tokens are much more common during the training of tokenizer than they are during the training of the GPT, feeding unoptimized activations into processing at test time: lesswrong.com/posts/aPeJE8bS..

The list goes on, TLDR everyone should hope that tokenization could be thrown away. Maybe even more importantly, we may find general-purpose strategies for multi-scale training in the process.

Tweet übersetzen

AK @_akhaliq · 15. Mai
MEGABYTE: Predicting Million-byte Sequences with Multiscale Transformers

Tokenization means that LLMs are not actually fully end-to-end.

Tokenization is why GPTs are bad at a number of very simple spelling / character manipulation tasks, e.g.:

The list goes on, TLDR everyone should hope that tokenization could be thrown away. Maybe even more importantly, we may find general-

Be Aware!

TADA: Efficient Task-Agnostic Domain Adaptation for Transformers

Chia-Chien Hung[1, 2, 3*], **Lukas Lange**[3], **Jannik Strötgen**[3, 4]
[1]NEC Laboratories Europe GmbH, Heidelberg, Germany
[2]Data and Web Science Group, University of Mannheim, Germany
[3]Bosch Center for Artificial Intelligence, Renningen, Germany
[4]Karlsruhe University of Applied Sciences, Karlsruhe, Germany
Chia-Chien.Hung@neclab.eu
Lukas.Lange@de.bosch.com jannik.stroetgen@h-ka.de

Abstract

Intermediate training of pre-trained transformer-based language models on domain-specific data leads to substantial gains for downstream tasks. To increase efficiency and prevent catastrophic forgetting alleviated from full domain-adaptive pre-training, approaches such as adapters have been developed. However, these require additional parameters for each layer, and are criticized for their limited expressiveness. In this work, we introduce TADA, a novel task-agnostic domain adaptation method which is modular, parameter-efficient, and thus, data-efficient. Within TADA, we retrain the embeddings to learn domain-aware input representations and tokenizers for the transformer encoder, while freezing all other parameters of the model. Then, task-specific fine-tuning is performed. We further conduct experiments with meta-embeddings and newly introduced meta-tokenizers, resulting in one model per task in multi-domain use cases. Our broad evaluation in 4 downstream tasks for 14 domains across single- and multi-domain setups and high- and low-resource scenarios reveals that TADA is an effective and efficient alternative to full domain-adaptive pre-training and adapters for domain adaptation, while not introducing additional parameters or complex training steps.

concerns. In practice, the absence of such domain-relevant information can severely hurt performance in downstream applications as shown in numerous studies (i.a., Zhu and Goldberg, 2009; Ruder and Plank, 2018; Friedrich et al., 2020).

To impart useful domain knowledge, two main methods of domain adaptation leveraging transformers have emerged: (1) *Massive pre-training from scratch* (Beltagy et al., 2019; Wu et al., 2020) relies on large-scale domain-specific corpora in-

Within TADA, we retrain the embeddings to learn domain-aware input representations and tokenizers for the transformer encoder, while freezing all other parameters of the model. Then, task-specific fine-tuning is per-

Domain Text: Acetaminophen is an analgesic drug

=> **TOK-1:** Ace #ta #mino #phen is an anal #gesic dr #ug (10 subwords)
=> **TOK-2:** Aceta #minophen is an anal #gesic drug (7 subwords)

Aggregation:	SPACE	DYNAMIC	TRUNCATION
TOK-1	[Ace #ta #mino #phen] is an [anal #gesic] [dr #ug]	[Ace #ta] [#mino #phen] is an anal #gesic [dr #ug]	[Ace] [#mino] is an anal #gesic [dr]
TOK-2	[Aceta #minophen] is an [anal #gesic] drug	Aceta #minophen is an anal #gesic drug	Aceta #minophen is an anal #gesic drug

Table 1: Examples of our proposed aggregation approaches for meta-tokenization: SPACE, DYNAMIC, TRUNCATION for a given text and two different tokenizers (TOK-1, TOK-2). The bottom of the table shows the results after aggregation. $[a\ b \ldots z]$ denotes the average of all embedding vectors corresponding to subword tokens a, b, …, z.

Outline – what we covered so far…

[1] Foundations

- **Again: Why is natural language processing (NLP) difficult?**
- **Evolution of NLP methods**
- **Crash course: Elements of language**
- **Morphology**

[2] Pre-processing

- **Text segmentation**
  - **Sentence splitting**
  - **Tokenization**
- Token normalization
  - Stemming
  - Lemmatization

Reminder

- Please join the ILIAS course:
  https://ilias.h-ka.de/goto.php?target=crs_853687&client_id=HSKA

Password:
NLP2024NLP

Thank you! Questions?