Natural Language Processing

1: Introduction

Objectives

to be able to answer questions

(1) to know about the course, connections to other courses in your programm

(2) What is Natural Language Processing (NLP)?

- common tasks
- challenges
- techniques
- typical applications of NLP

(3) Tokenization / Lemmatization / Edit distance (most likely, next time)

NLP Course

Syllabus Review

Grading

NLP Course Team

(1) PI: Vladimir Ivanov

- room 475
- @nomemm

(2) TAs:

- Lada Morozova
- Georgy Andryushchenko

Resources

Grading schema

Midterm: 30%

Assignments: 30%

Lab Participation: 10%

Final exam: 30%

Grade Letters

90-100: A

75-89: B

60-74: C

0-59: D

Moodle

Kaggle

Structure

- 12-14 Lectures: 2 hrs per week
- 3 Assignments
  - create a repo in your github
  - submit a link to moodle
- 12-14 Labs: 2 hours per week
- 1 Final Exam
- 0...n retakes

Course structure

	Lab / Assignment
Week 1: Introduction to NLP, Basics: tokenization	Lab 1
Week 2: Sentence Segmentation	Lab 2, A.1 out (Tweets Tokenization)
Week 3: Lexical analysis. POS-tagging	Lab 3
Week 4: Language modelling	Lab 4
Week 5: Language modelling (Ngram model)	Lab 5, A.1 due
Week 6: Parsing. Semantic Analysis	A.2 out (NEREL+codalab/kaggle)
Week 7: Distributional semantics. Embeddings	Lab 7
Week 8: Midterm (Moodle test + optional Oral)	no labs
Week 9: Knowledge Graphs	Lab 8, A.2 due
Week 10: RNNs for NLP. ELMo	Lab 9, A.3 out (ELMo)
Week 11: Transfer learning in NLP. Question answering	Lab 10
Week 12: Transformer-based architectures. BERT-like m	Lab 11
Week 13: Transformer-based architectures. GPT	Lab 12
Week 14: Generative Neural language models. LLMs	Lab 13, A.3 due

Books

(1) **(SLP)** Speech and Language Processing (3rd ed. draft). Dan Jurafsky and James H. Martin. https://web.stanford.edu/~jurafsky/slp3/ - **Core Course Book, new 2021-2023**

(2) **(HNLP)** Handbook Of Natural Language Processing, Second Edition Chapman & Hall Crc Machine Learning & Pattern Recognition 2010 **a bit outdated, but good**

(3) **(HoPNLP)** Kedia, Aman; Rasu, Mayank. Hands-On Python Natural Language Processing: Explore tools and techniques to analyze and process text with a view to building real-world NLP applications. Packt Publishing.

(4) **(HOML)** Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. Aurélien Géron — **ML/DL** + **NLP sections**

Courses

(1) https://huggingface.co/learn/nlp-course/chapter1/1

(2) https://web.stanford.edu/class/cs224n/

(3) https://web.stanford.edu/~jurafsky/slp3/

(4) ... many more

(5) https://t.me/DeepNLPspring2018

(6) Join the NLP Community: @natural_language_processing

What is Natural Language Processing (NLP)?

NLP

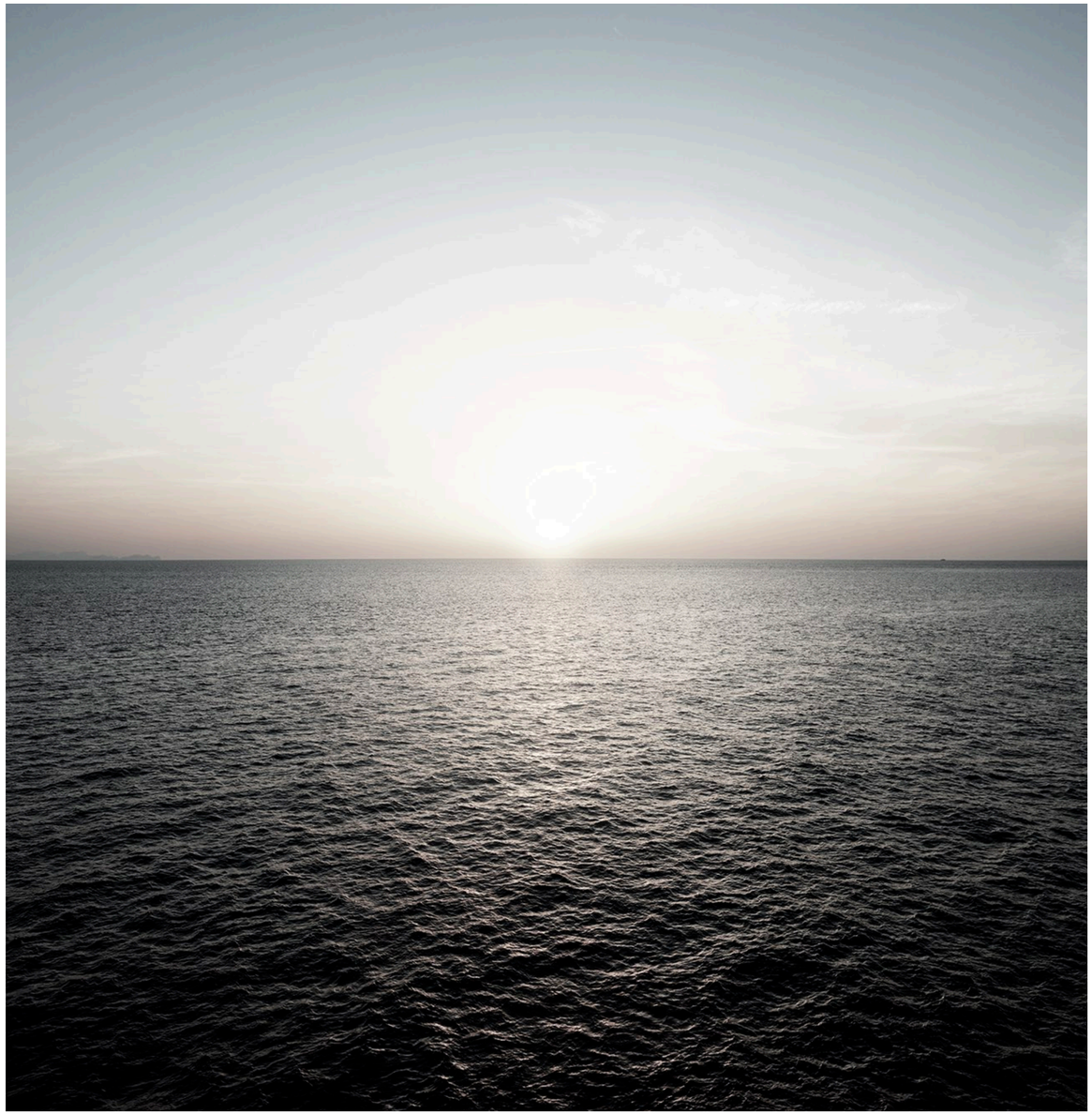
field of study

is a subfield of Artificial Intelligence

it combines

- Linguistics
- Computer Science
- Machine Learning

it makes human language intelligible to computers (yep, you can ask ChatGPT)



NLP

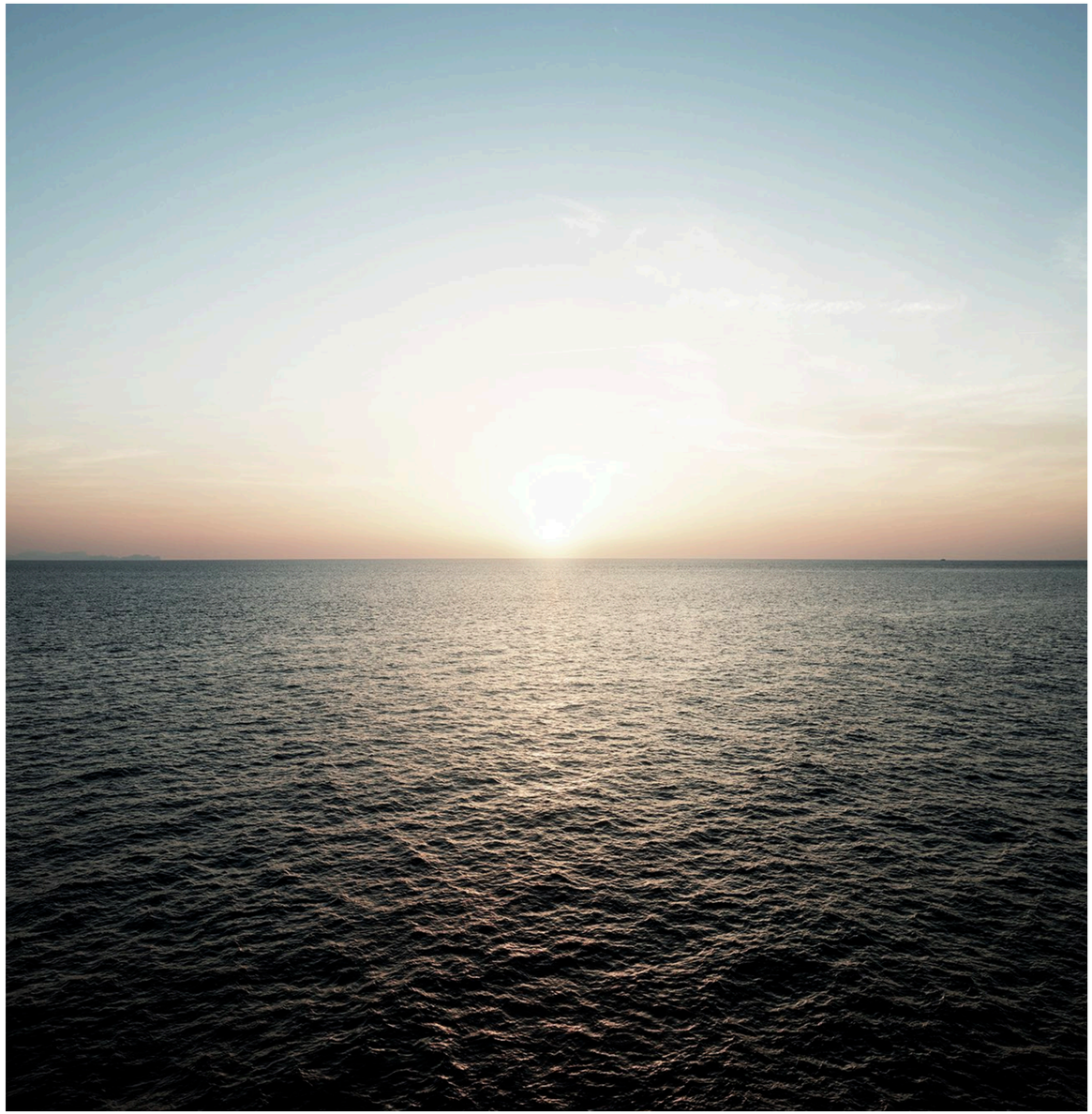
goals

understanding the structure and meaning of human language

develop algorithms / models

create computer systems capable of

- understanding
- analyzing
- extracting meaning from text and speech



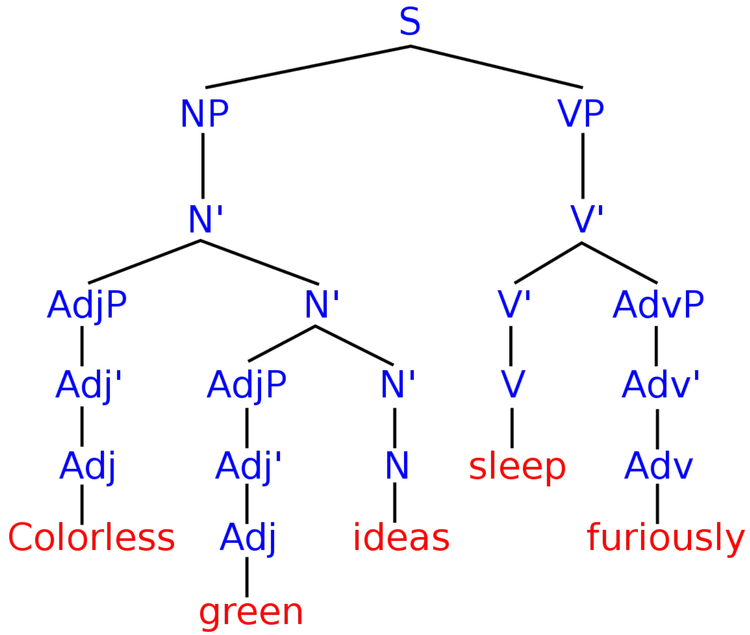
NLP

algorithms / models for

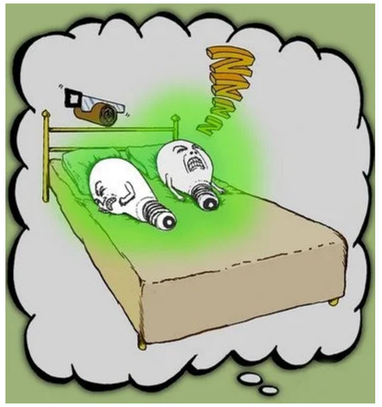
analysis of linguistic phenomena:

- morphology
- syntax
- semantics
- pragmatics

automatic preforming specific tasks to solve practical problems



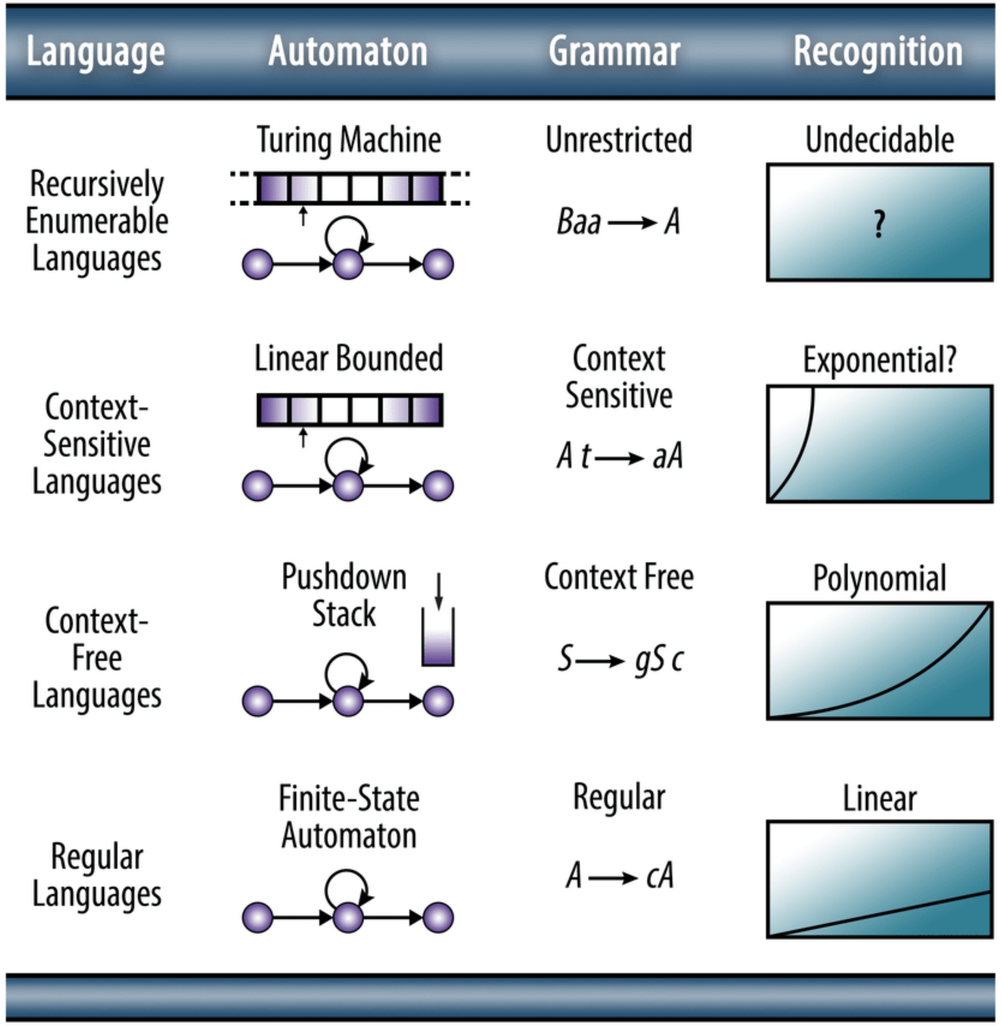
Colorless green ideas sleep furiously



NLP

benefits

- Large-scale text / speech analysis
- Automate business processes
- Common pipelines / tools can be tailored to specific application or industry (e.g. in Software Eng.)
- Studies of Cognition, Perception, Human Brain, ...



What are the challenges in NLP?

Main challenges

one word only: “ambiguity”

(1) **Ambiguity** is an intrinsic feature of human language

- meaning depends on context
- you cannot avoid it
- ambiguities present at all levels (syntax, semantics, etc.)
- tradeoff between complexity and expressiveness

(2) Infitine number of possible sentences in ANY natural language

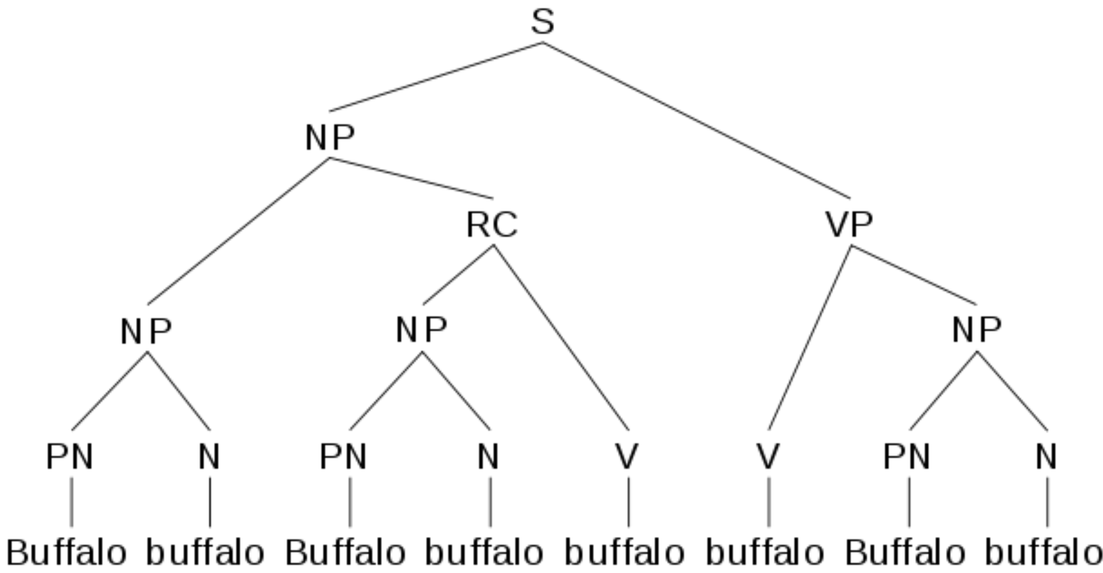
(3) Permanent changes of natural languages / variability

Examples of ambuguity

(1) One morning I shot an elephant in my pajamas.
How he got in my pajamas, I don't know.
-- Groucho Marx

(2) A: What has happened to the roast beef?
B: The dog is looking very happy.

(3)



https://en.wikipedia.org/wiki/Buffalo_buffalo_Buffalo_buffalo_buffalo_buffalo_Buffalo_buffalo

What are common tasks / techniques in NLP?

Tasks, aka Problems

Two large tasks

(1) Syntactic analysis

- Representing / Understanding structure

(2) Semantic analysis

- Representing / Understanding meaning

Many subtasks

in pactice

- Tokenization, Lemmatization, Stemming
- Part of Speech Tagging (PoS)
- Parsing: Dependency Parsing, Constituency Parsing
- Semantic Role Labeling
- Word Sense Disambiguation
- Information Extraction:
    (1) Named Entity Recognition (NER)
    (2) Relation / Event Extraction
- Entity Linking

What are typical applications of NLP?

Examples of applications

only few

(1) Text classification (email spam filters)

(2) Virtual assistants

(3) Search engines

(4) Autocorrect, grammar checkers

(5) Social media analysis (sentiment, trends, argument mining)

(6) Customer feedback, customer support

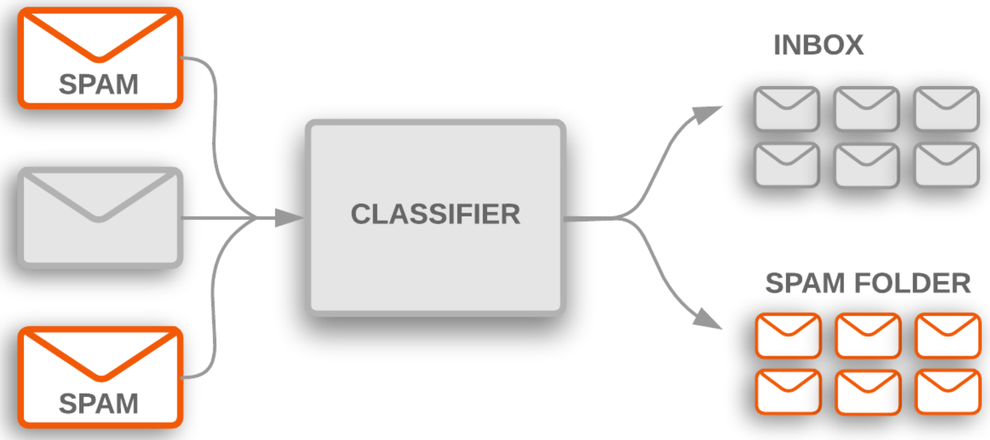
(7) Chatbots

(8) Machine translation

(9) Natural language generation

(10) Automatic text summarization, simplification

Text Classification



Binary: spam / non-spam

Mutli-class

- sport / politics / economics
- hierarchical

Multi-label

- toxic texts detection
- propaganda classification

Topic modeling

Technique	Train
Appeal to Authority	144
Appeal to Fear-Prejudice	294
Bandwagon/Reductio ad Hitlerum	72
Black and White Fallacy	107
Causal Oversimplification	209
Doubt	493
Exaggeration/Minimisation	466
Flag Waving	229
Loaded Language	2123
Name Calling/Labeling	1058
Repetition	621
Slogans	129
Thought-Terminating Cliches	76
Whataboutism/Straw Men/Red Herring	108

Virtual assistants

Let them talk!

Alexa, Alisa, Cortana, Google, Siri, ...

Processing voice requests

- speech to text
- recall previous interactions
- connect to apps
- control devices

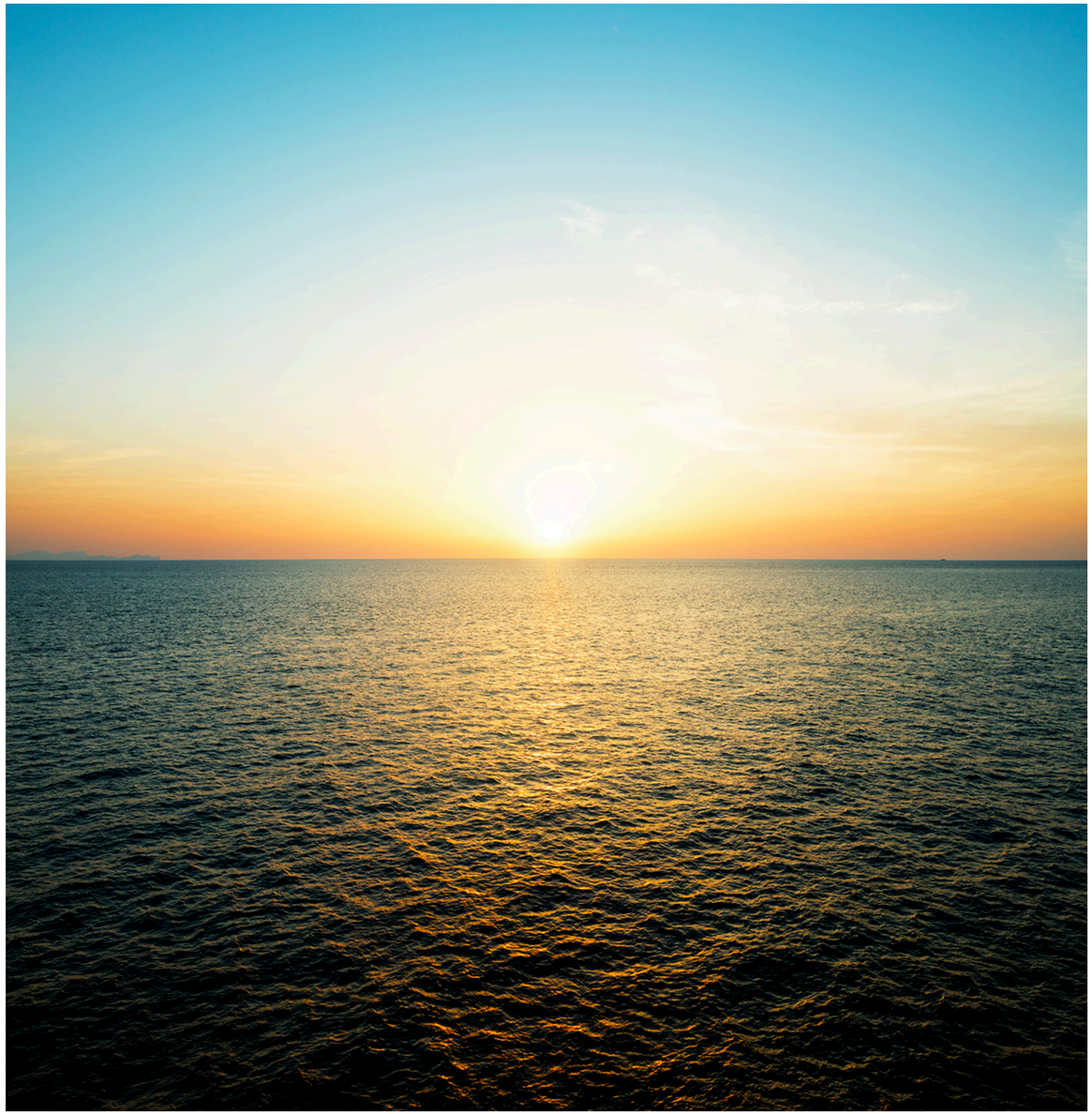
Search engines

Procetting of documents and queries

- clear content
- autocomplete
- query expansion

suggesting queries / topics

...

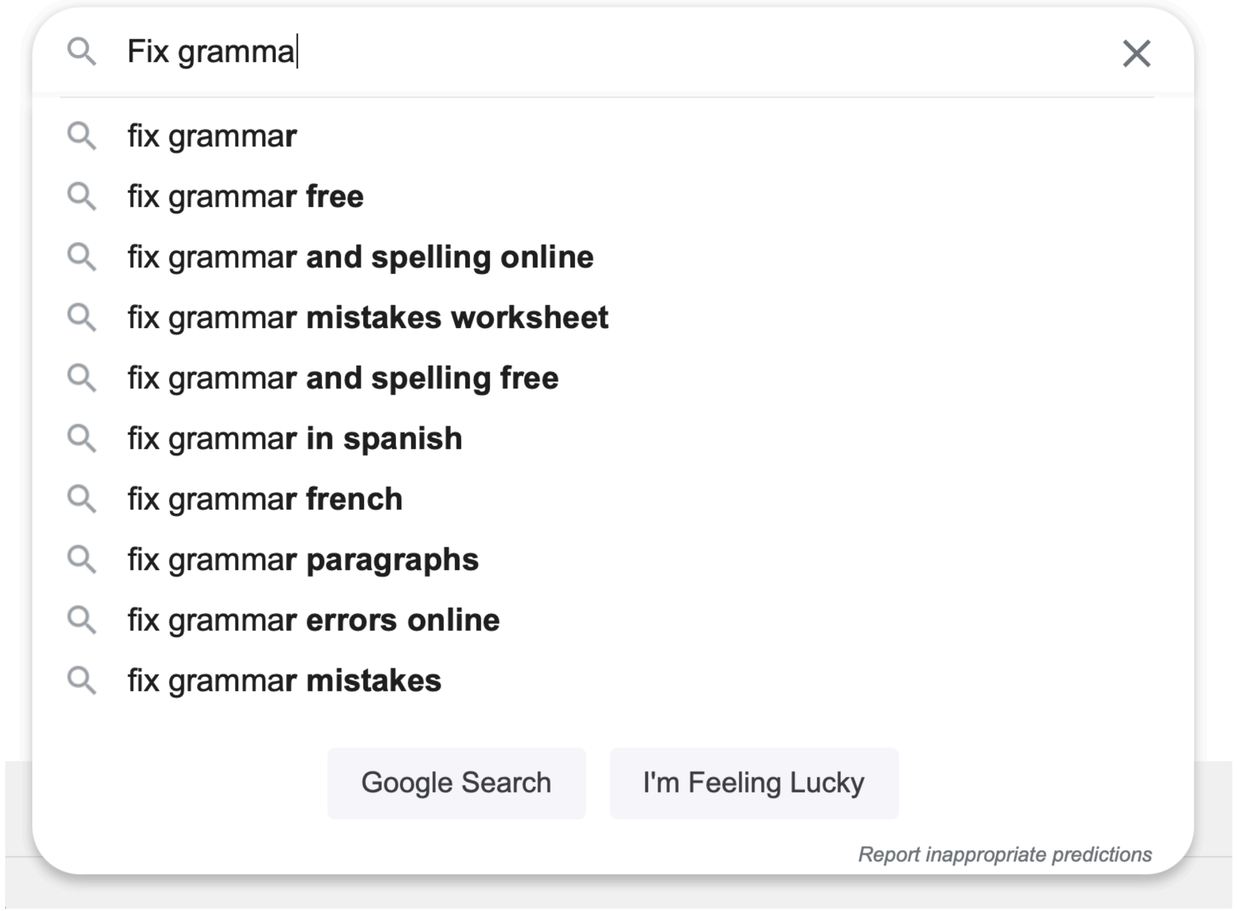


Autocorrect, grammar checkers

Predicts text as you type it

Fix grammar miskates

go back to grammar school, organic entity



Social media analysis

Monitoring society

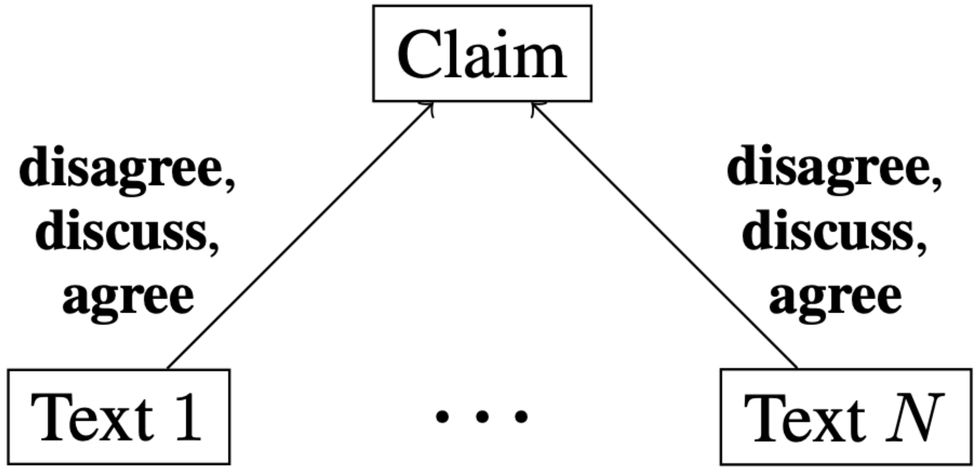
Sentiment analysis

Aspect-based sentiment

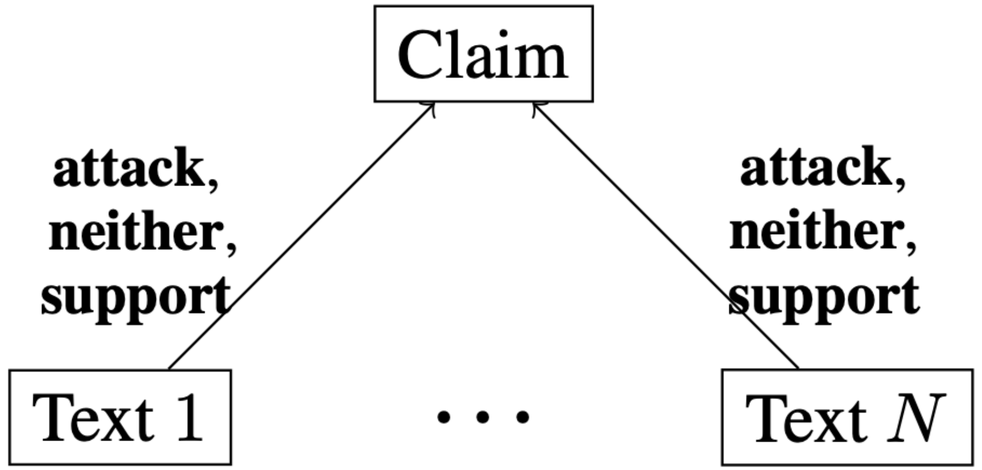
Opinion mining

Argument mining: classifying arguments and relationships between them

Brand tracking



(a) Stance Detection



(b) Relation-based Argument Mining

https://aclanthology.org/W19-4518.pdf

Customer feedback

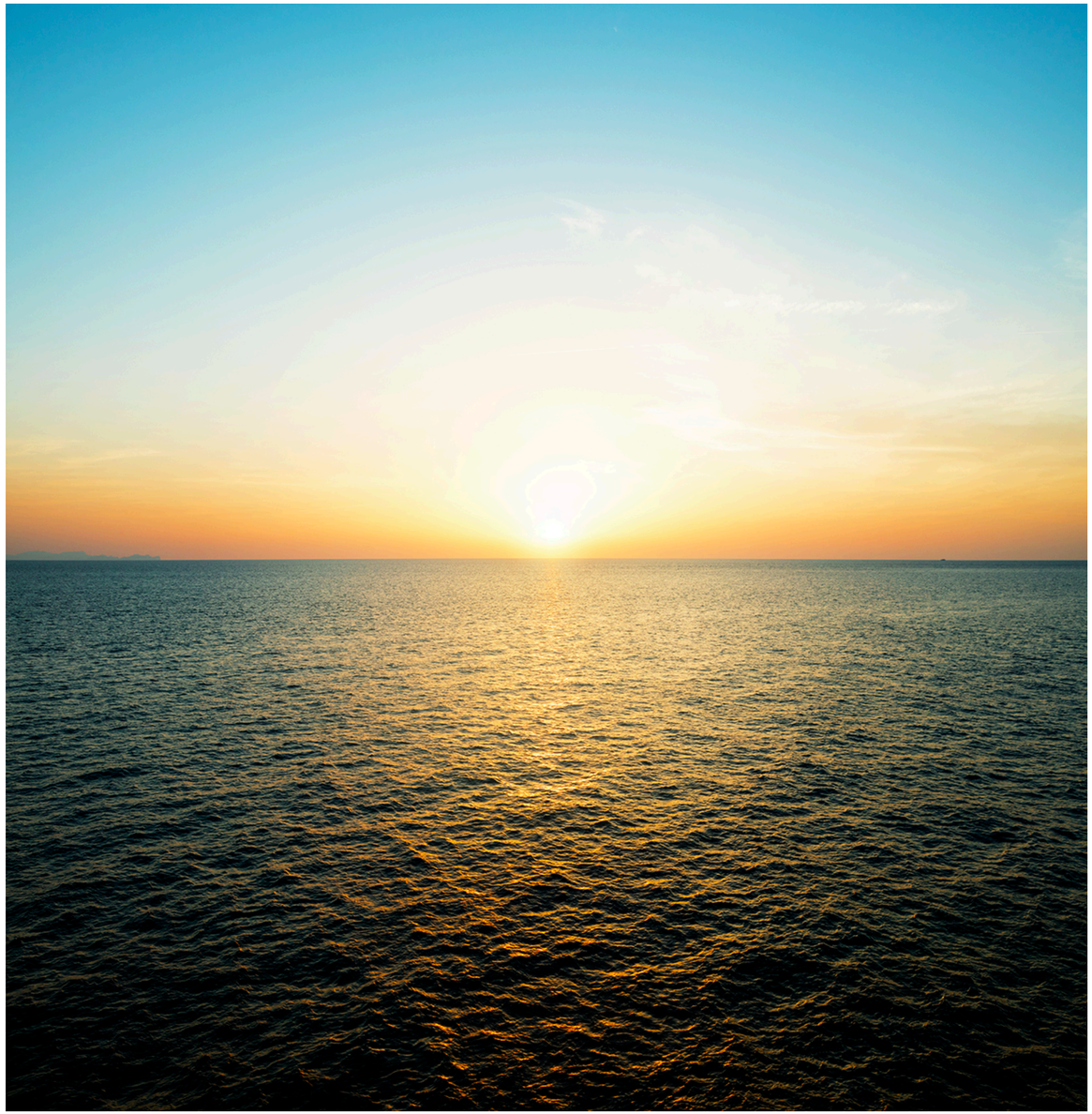
Customer analysis

User Feedback analysis

- Topics mentioned
- Response classification
- Rounting user questions
- Tickets in customer support

Question answering

- about products
- about company / processes



Chatbots

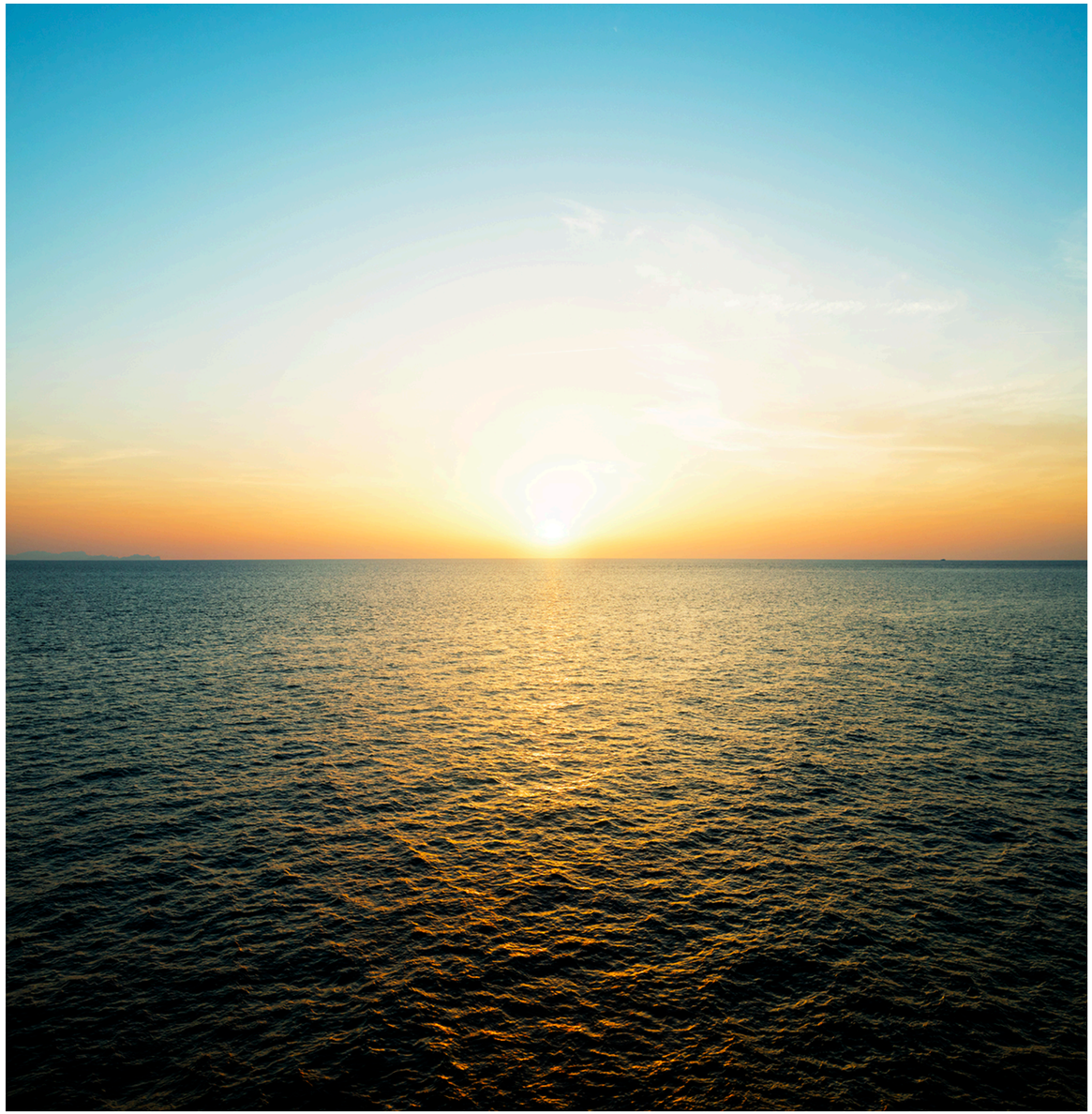
Simulates human conversation

Intent and emotion recongition

Auxilary tasks:

- recommmendation
- offer discounts
- make reservations

usually, simple tasks: if/then...



Machine translation

since 1950s

Milestones in MT:

- rule-based
- statistical MT
- neural MT

In 2019 Facebook AI (FAIR):

- English-German, English-Russian
- “superhuman” perfromance
- better than previous system by 4.5 BLEU

Key features of the MT model:

- cross-lingual pretraining and
- self-supervised learning

https://ai.facebook.com/blog/facebook-leads-wmt-translation-competition/

https://aclanthology.org/W19-5333.pdf

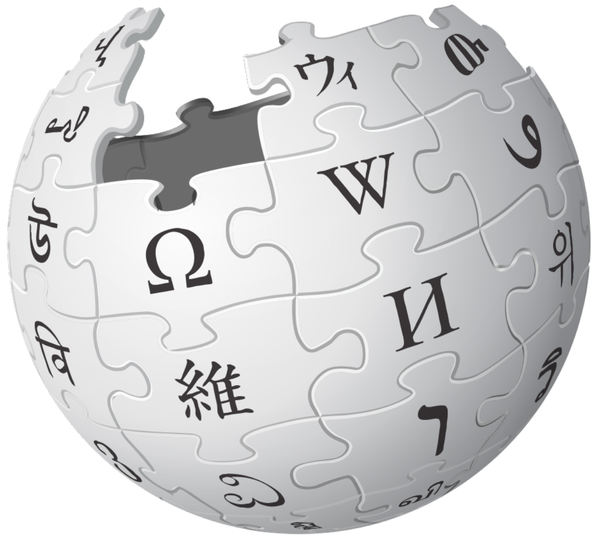
Automatic text summarization

keep it simple...

Extractive

Abstractive

Text simplification



Simple English
WIKIPEDIA

Natural language generation

Prompt is All you need (plus Attention)

Automatic generation of different texts

GPT-2 breakthrough (in 2019)

Generating whole stories
(e.g. about unicorns)

Fake news articles

Yes, there is GPT-3...

Yes, there is ChatGPT...

https://transformer.huggingface.co/doc/distil-gpt2

Write With Transformer

Get a modern neural network to
auto-complete your thoughts.

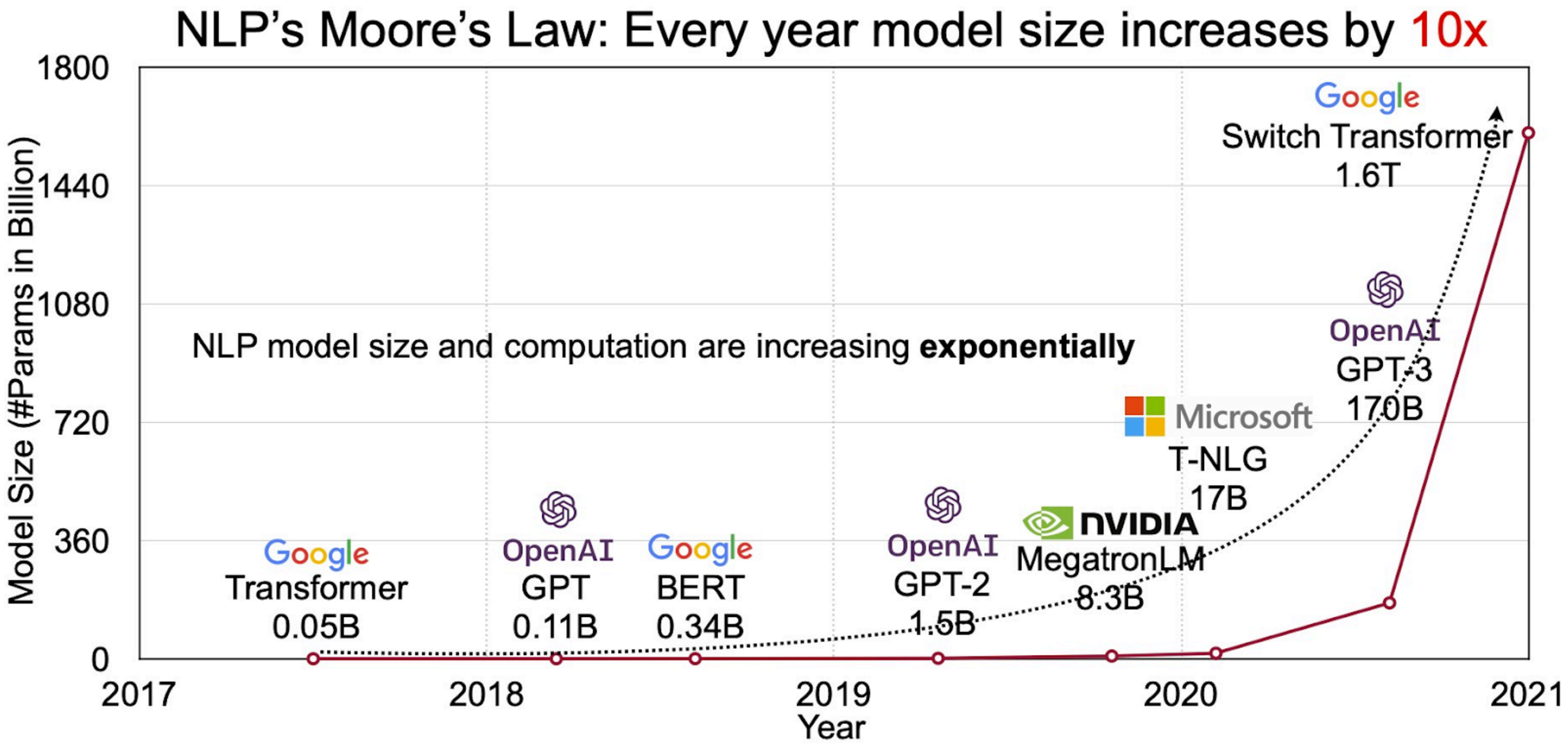
natural language generation is

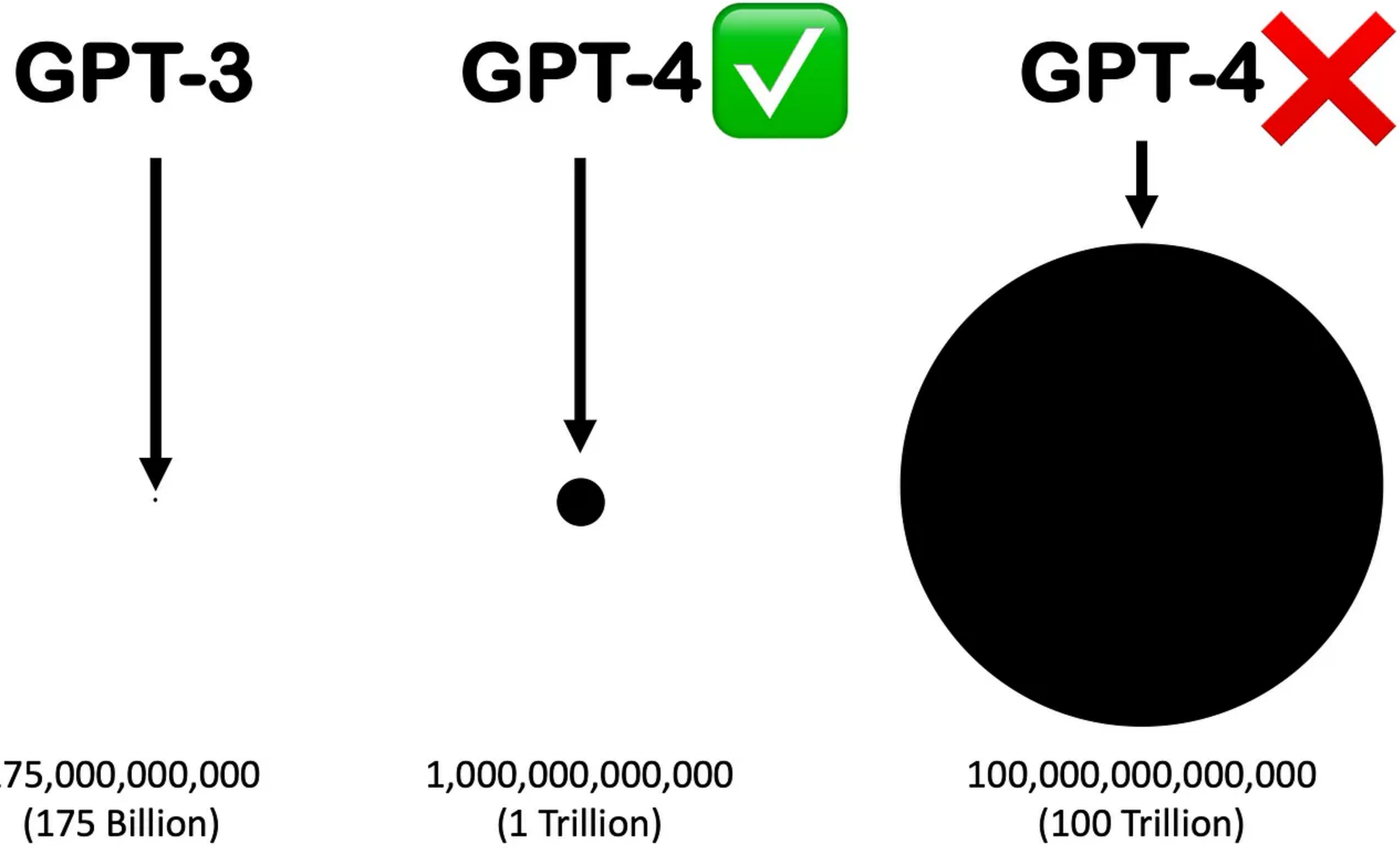
a good tool to understand language development.

"Give me a prompt and I shall move the earth!"

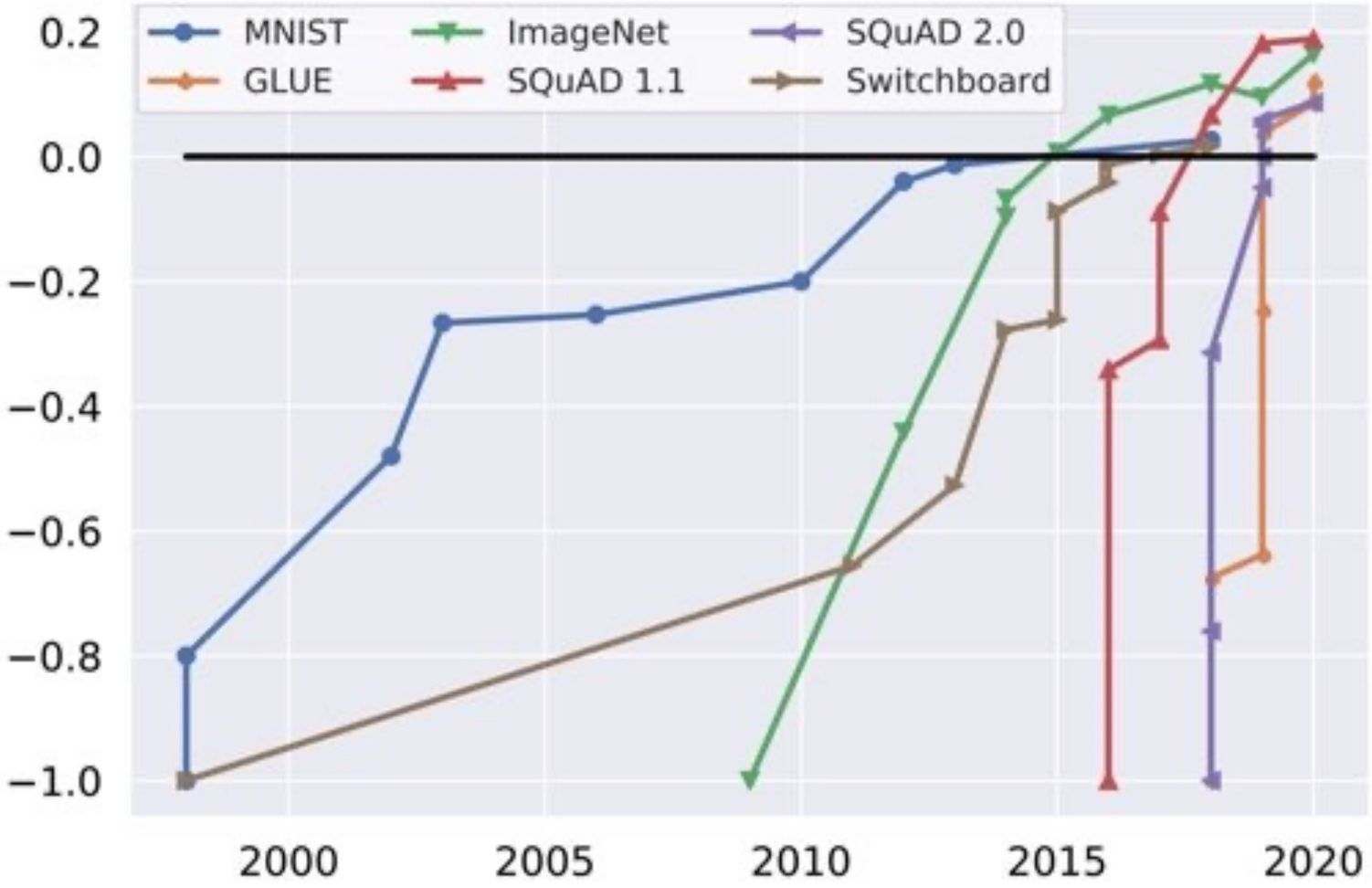
LLMs

Given the input sequence of tokens, a language model predicts the next ...




GPT-3
GPT-4
GPT-4
175,000,000,000
(175 Billion)
1,000,000,000,000
(1 Trillion)
100,000,000,000,000
(100 Trillion)

Benchmarks Saturation



Tokenization, Lemmatization, Stemming

Objectives

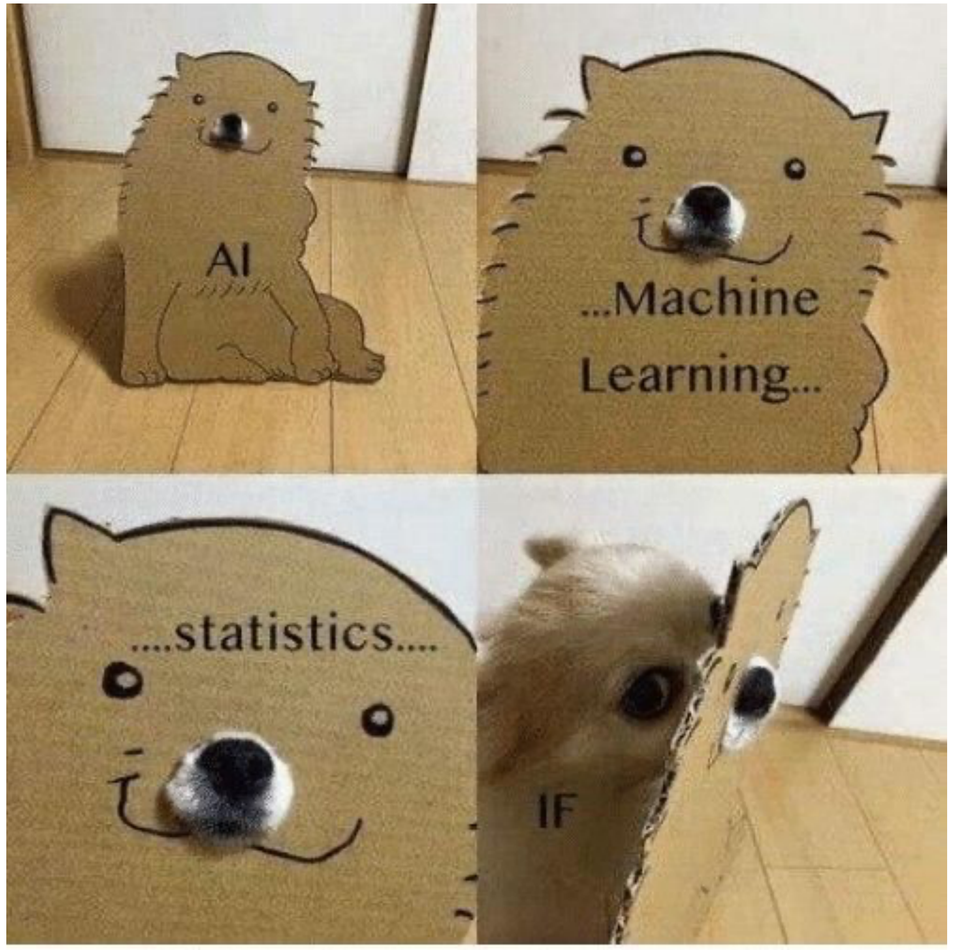
to be able to answer questions

(1) What methods exist for Tokenization and Segmentation?

(2) What is the place of a tokenizer in NLP pipeline?

(3) How does BPE work?

(4) Edit distance (optinally)



IF IF IF IF IF IF IF IF IF IF WE!

Text Normalization

Every NLP task requires text normalization:

- Tokenizing (segmenting) words
- Normalizing word formats
- Segmenting sentences

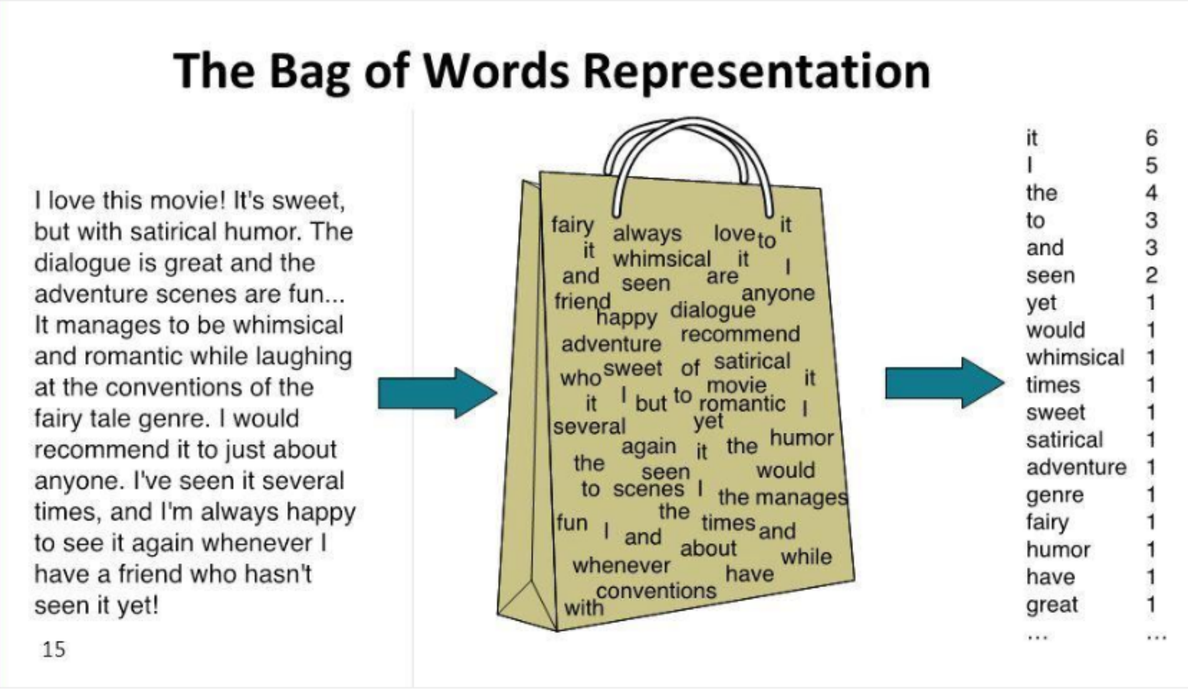
Bag of words (BoW)

(1) Machine learning algorithms cannot operate on raw text directly; text must be converted to numbers (more precisely, to vectors of numbers)

(2) BoW: representing text data when modeling text with ML

(3) Feature extraction or encoding.

(4) Ignoring the order of words in the text



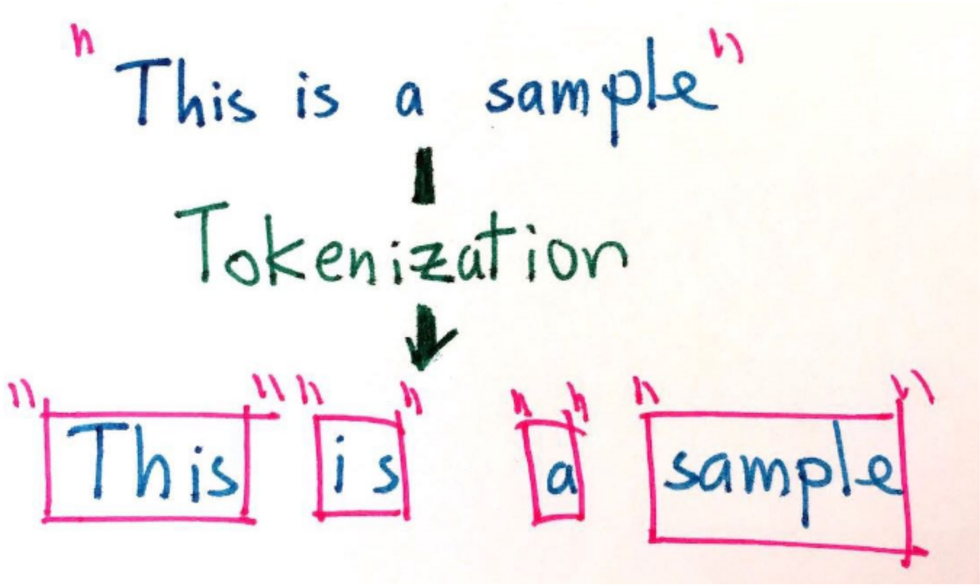
Tokenization

(1) Breaking the input text into meaningful elements, smaller pieces or "tokens".

(2) Tokens can be:

- elements for further semantic processing
- word, sentence, paragraph

(3) Certain characters, such as punctuation marks, can be removed at the same time.



Example

Input: "The San Francisco-based restaurant," they said,
"doesn't charge $10".

Output: "␣The␣San␣Francisco-based␣restaurant␣,␣"␣they␣said␣,␣
"␣does␣n't␣charge␣$␣10␣"␣.

Words and Corpora

How many words in a sentence?

(1)"I do uh main- mainly business data processing"

- Fragments, filled pauses

(2)"Seuss's cat in the hat is different from other cats!"

- Lemma: same stem, part of speech, rough word sense
  - **cat** and **cats** = same lemma
- Wordform: the full inflected surface form
  - **cat** and **cats** = different wordforms

How many words in a sentence?

- Example:

They lay back on the San Francisco grass and looked at the stars and their.

- Type: an element of the vocabulary
- Token: an instance of that type in running text
- How many?
  - 15 tokens (or 14)
  - 13 types (or 12) (or 11?)

How many words in a corpus?

N is a number of tokens

V is a vocabulary, a set of types, **|*V*|** is the size of the vocabulary

Heaps Law = Herdan's Law = $|V| = kN^{\beta}$ where often $.67 < \beta < .75$

i.e., vocabulary size grows with > square root of the number of word tokens

Corpus	Tokens = N	Types = |V|
Switchboard phone conversations	2.4 million	20 thousand
Shakespeare	884,000	31 thousand
COCA	440 million	2 million
Google N-grams	1 trillion	13+ million
Common Crawl / mC4	6.3 trillion	??

Corpora

corpora (= corpuses, pl. corpus)

Words don't appear out of nowhere!

A text is produced by

(1) a specific writer(s)

(2) at a specific time

(3) in a specific variety

(4) of a specific language

(5) for a specific function

Corpora vary along dimension

- Language: 7097 languages in the world
- Variety: like African American Language varieties.
  - AAE Twitter posts might include forms like "iont" (I don't)
- Code switching, e.g., Spanish/English, Hindi/English:

  S/E: Por primera vez veo a @username actually being hateful! It was beautiful:)
  [For the first time I get to see @username actually being hateful! it was beautiful:)]

  H/E: dost tha or ra- hega ... dont wory ... but dherya rakhe
  [“he was and will remain a friend ... don’t worry ... but have faith”]

- Genre: newswire, fiction, scientific articles, Wikipedia
- Author Demographics: writer's age, gender, ethnicity

Corpus datasheets

Motivation:

Why was the corpus collected?

By whom?

Who funded it?

Situation:

In what situation was the text written?

Collection process:

If it is a subsample how was it sampled? Was there consent? Pre-processing

Annotation process, language variety, demographics, etc.

Datasets vs. Corpora vs. Text Collections

(1) Discussion: What is the difference?

Space-based tokenization

(1) A very simple way to tokenize

- For languages that use space characters between words
- Segment off a token between instances of spaces

(2) Unix tools for space-based tokenization

- The "tr" command
- Inspired by Ken Church's UNIX for Poets
- Given a text file, output the word tokens and their frequencies

Simple Tokenization in UNIX

(1)(Inspired by Ken Church's UNIX for Poets.)

(2)Given a text file, output the word tokens and their frequencies

```
tr -sc 'A-Za-z' '\n' < shakes.txt | sort | uniq –c
```

```
1945 A
  72 AARON
  19 ABBESS
   5 ABBOT
 ... ...
```

```
25 Aaron
 6 Abate
 1 Abates
 5 Abbess
 6 Abbey
 3 Abbot
... …
```

The first step: tokenizing

```
tr -sc 'A-Za-z' '\n' < shakes.txt | head
```

```
THE
SONNETS
by
William
Shakespeare
From
fairest
creatures
We
...
```

The second step: sorting

```
tr -sc 'A-Za-z' '\n' < shakes.txt | sort | head

A
A
A
A
A
A
A
A
A
A
...
```

More counting

(1) Merging upper and lower case

```
tr 'A-Z' 'a-z' < shakes.txt | tr -sc 'A-Za-z' '\n' | sort | uniq -c
```

(2) Sorting the counts

```
tr 'A-Z' 'a-z' < shakes.txt | tr -sc 'A-Za-z' '\n' | sort | uniq -c | sort -n -r
```

```
  23243 the
  22225 i
  18618 and
  16339 to
  15687 of
  12780 a
  12163 you
  10839 my
  10005 in
   8954 d
```

What happened here?

Issues in Tokenization

(1)Can't just blindly remove punctuation:

- m.p.h., Ph.D., AT&T, cap'n
- prices ($45.55)
- dates (01/02/06)
- URLs (http://www.stanford.edu)
- hashtags (#nlproc)
- email addresses (someone@cs.colorado.edu)

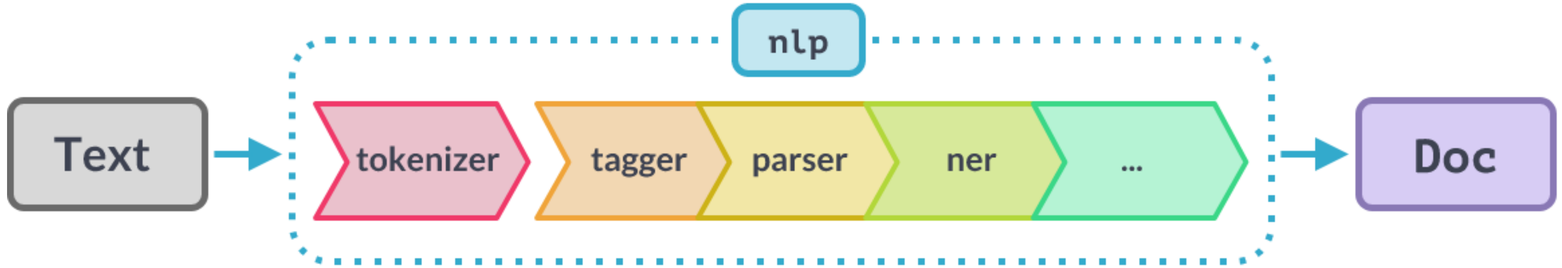
(2)Clitic: a word that doesn't stand on its own

- "are" in we're, French "je" in j'ai, "le" in l'honneur

(3)When should multiword expressions (MWE) be words?

- New York, rock 'n' roll

Tokenization in Spacy




```
import spacy
nlp = spacy.load("en_core_web_md")
doc = nlp("We are learning NLP using spaCy.")
print ([token.text for token in doc])
```

✓ 2.6s

```
['We', 'are', 'learning', 'NLP', 'using', 'spaCy', '.']
```

https://medium.com/analytics-vidhya/nlp-with-spacy-tutorial-part-2-tokenization-and-sentence-segmentation-352df790a214

https://machinelearningknowledge.ai/complete-guide-to-spacy-tokenizer-with-examples/

Example BERT tokenizer

- A sample sentence to tokenize with htoe.
- [‘a’, 'sample', 'sentence', 'to', 'token', '##ize', 'with', 'h', '##to', '##e', ‘.’]

- https://colab.research.google.com/github/SaifAlmaliki/Bert-NLP/blob/main/BERT_Vocabulary.ipynb#scrollTo=jZgZzZYdky0K

BPE

Byte Pair Encoding

encoding (rare) words via subword units

eliminating the need for large vocabularies

subword models achieve better accuracy for the translation of rare words than large-vocabulary models

Algorithm 1 Learn BPE operations

```
import re, collections

def get_stats(vocab):
  pairs = collections.defaultdict(int)
  for word, freq in vocab.items():
    symbols = word.split()
    for i in range(len(symbols)-1):
      pairs[symbols[i],symbols[i+1]] += freq
  return pairs

def merge_vocab(pair, v_in):
  v_out = {}
  bigram = re.escape(' '.join(pair))
  p = re.compile(r'(?<!\S)' + bigram + r'(?!\S)')
  for word in v_in:
    w_out = p.sub(''.join(pair), word)
    v_out[w_out] = v_in[word]
  return v_out

vocab = {'l o w </w>' : 5, 'l o w e r </w>' : 2,
         'n e w e s t </w>':6, 'w i d e s t </w>':3}
num_merges = 10
for i in range(num_merges):
  pairs = get_stats(vocab)
  best = max(pairs, key=pairs.get)
  vocab = merge_vocab(best, vocab)
  print(best)
```

https://arxiv.org/pdf/1508.07909.pdf

Byte Pair Encoding

Another option for text tokenization

Instead of

(1) white-space segmentation

(2) single-character segmentation

Use the data to tell us how to tokenize.

Subword tokenization (because tokens can be parts of words as well as whole words)

Subword tokenization

Four common algorithms:

- **Byte-Pair Encoding (BPE)** (Sennrich et al., 2016)
- **Unigram language modeling tokenization** (Kudo, 2018)
- **WordPiece** (Schuster and Nakajima, 2012)
- **SentencePiece** (Kudo and Richardson, 2018) https://arxiv.org/abs/1808.06226

All have 2 parts:

A **token learner** that takes a raw training corpus and induces a vocabulary (a set of tokens).

A **token segmenter** that takes a raw test sentence and tokenizes it according to that vocabulary

Byte Pair Encoding (BPE) token learner

Let vocabulary be the set of all individual characters that you meet in a corpus

= {A, B, C, D,..., a, b, c, d....}

(1) Repeat:

Choose the two symbols that are most frequently adjacent in the training corpus (say 'A', 'B')

Add a new merged symbol 'AB' to the vocabulary

Replace every adjacent 'A' 'B' in the corpus with 'AB'.

(2) Until *k* merges have been done.

BPE token learner algorithm

We shall consider the following text:

low low low low low lowest lowest newer newer newer newer newer newer wider wider wider new new

Algorithm 1 Learn BPE operations

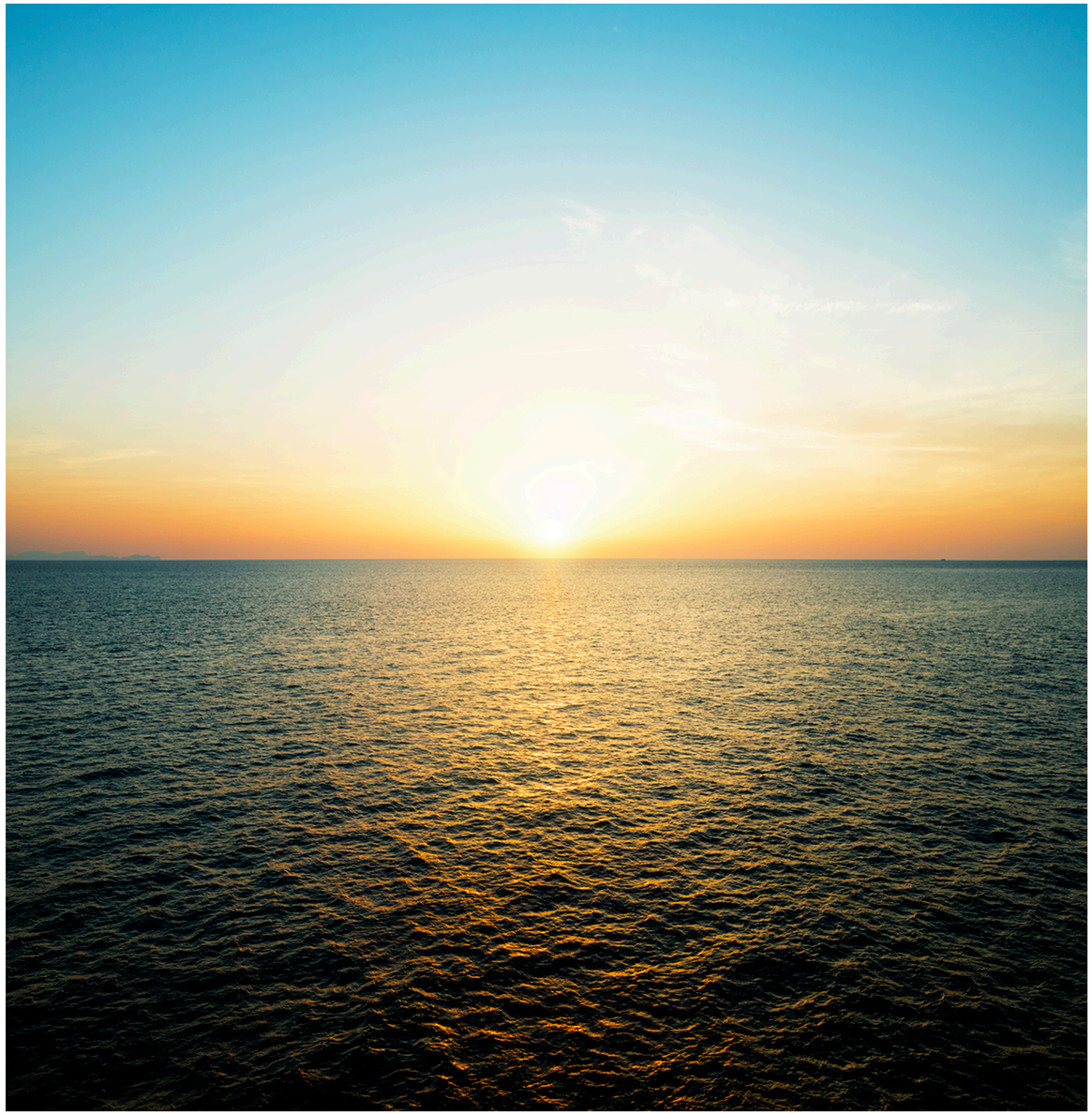
```
import re, collections

def get_stats(vocab):
  pairs = collections.defaultdict(int)
  for word, freq in vocab.items():
    symbols = word.split()
    for i in range(len(symbols)-1):
      pairs[symbols[i],symbols[i+1]] += freq
  return pairs

def merge_vocab(pair, v_in):
  v_out = {}
  bigram = re.escape(' '.join(pair))
  p = re.compile(r'(?<!\S)' + bigram + r'(?!\S)')
  for word in v_in:
    w_out = p.sub(''.join(pair), word)
    v_out[w_out] = v_in[word]
  return v_out

vocab = {'l o w </w>' : 5, 'l o w e r </w>' : 2,
         'n e w e s t </w>':6, 'w i d e s t </w>':3}
num_merges = 10
for i in range(num_merges):
  pairs = get_stats(vocab)
  best = max(pairs, key=pairs.get)
  vocab = merge_vocab(best, vocab)
  print(best)
```

Byte Pair Encoding (BPE)

Most subword algorithms are run inside space-separated tokens.

So we commonly first add a special end-of-word symbol '__' before space in training corpus

Next, separate words into letters.



BPE token learner

Original (very fascinating🙄) corpus:

low low low low low lowest lowest newer newer newer newer newer newer wider wider wider new new

Adding end-of-word tokens, resulting in this vocabulary:

corpus	vocabulary
5 l o w _	_, d, e, i, l, n, o, r, s, t, w
2 l o w e s t _	
6 n e w e r _	
3 w i d e r _	
2 n e w _	

BPE: 1st Merge operation

corpus

```
5   l o w _
2   l o w e s t _
6   n e w e r _
3   w i d e r _
2   n e w _
```

vocabulary

```
_, d, e, i, l, n, o, r, s, t, w
```

Merge e r to er

corpus

```
5   l o w _
2   l o w e s t _
6   n e w er _
3   w i d er _
2   n e w _
```

vocabulary

```
_, d, e, i, l, n, o, r, s, t, w, er
```

BPE: 2nd Merge operation

corpus

5 l o w _
2 l o w e s t _
6 n e w er _
3 w i d er _
2 n e w _

vocabulary

_, d, e, i, l, n, o, r, s, t, w, er

Merge er _ to er_

corpus

5 l o w _
2 l o w e s t _
6 n e w er_
3 w i d er_
2 n e w _

vocabulary

_, d, e, i, l, n, o, r, s, t, w, er, er_

BPE: 3rd Merge operation

corpus	vocabulary
5 l o w _	_, d, e, i, l, n, o, r, s, t, w, er, er_
2 l o w e s t _	
6 n e w er_	
3 w i d er_	
2 n e w _	

Merge n e to ne

corpus	vocabulary
5 l o w _	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne
2 l o w e s t _	
6 ne w er_	
3 w i d er_	
2 ne w _	

BPE

(1) The next merges are:

Merge	Current Vocabulary
(ne, w)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new
(l, o)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo
(lo, w)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low
(new, er_)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low, newer_
(low, _)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low, newer_, low_

BPE token segmenter algorithm

On the test data, run each merge learned from the training data:

- Greedily
- In the order we learned them
  (test frequencies don't play a role)

So: merge every e r to er, then merge er _ to er_, etc.

(1) Result:

- Test set "n e w e r _" would be tokenized as a full word
- Test set "l o w e r _" would be two tokens: "low er_"

Properties of BPE tokens

Usually include frequent words

And frequent subwords

(1)Which are often morphemes like *-est* or *–er*

A morpheme is the smallest meaning-bearing unit of a language

(2)*unlikeliest* has 3 morphemes *un-*, *likely*, and *-est*

Word Normalization and other issues

Word Normalization

(1)Putting words/tokens in a standard format

(1)U.S.A. or USA

(2)uhhuh or uh-huh

(3)Fed or fed

(4)am, is, be, are

Case folding

(1) Applications like IR: reduce all letters to lower case

- Since users tend to use lower case
- Possible exception: upper case in mid-sentence?
  (1) e.g., ***General Motors***
  (2) ***Fed*** vs. ***fed***
  (3) ***SAIL*** vs. ***sail***

(2) For sentiment analysis, MT, Information extraction

- Case is helpful (***US*** versus ***us*** is important)

Lemmatization

Represent all words as their lemma, their shared root

= dictionary headword form:

- *am, are, is → be*
- *car, cars, car's, cars' → car*
- Spanish quiero ('I want'), quieres ('you want')

  → querer 'want'
- *He is reading detective stories*

→ *He be read detective story*

Lemmatization is done by Morphological Parsing

(1)Morphemes:

- The small meaningful units that make up words
- **Stems**: The core meaning-bearing units
- **Affixes**: Parts that adhere to stems, often with grammatical functions

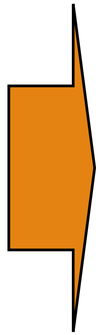
(2)Morphological Parsers:

- Parse *cats* into two morphemes *cat* and *s*
- Parse Spanish *amaren* ('if in the future they would love') into morpheme *amar* 'to love', and the morphological features *3PL* and *future subjunctive*.

Stemming

Reduce terms to stems, chopping off affixes crudely

This was not the map we found in Billy Bones's chest, but an accurate copy, complete in all things-names and heights and soundings-with the single exception of the red crosses and the written notes.



Thi wa not the map we found in Billi Bone s chest but an accur copi complet in all thing name and height and sound with the singl except of the red cross and the written note
.

Dealing with complex morphology is necessary for many languages

- 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 '2pl' + casina 'as if'

Sentence Segmentation

!, ? mostly unambiguous but **period** "." is very ambiguous

- Sentence boundary
- Abbreviations like Inc. or Dr.
- Numbers like .02% or 4.3

Common algorithm: Tokenize first: use rules or ML to classify a period as either (a) part of the word or (b) a sentence-boundary.

- An abbreviation dictionary can help

Sentence segmentation can then often be done by rules based on this tokenization.

Regular Expressions: Recap

Regular expressions

(1)A formal language for specifying text strings

(2)How can we search for any of these?

- woodchuck
- woodchucks
- Woodchuck
- Woodchucks

Regular Expressions: Disjunctions

(1)Letters inside square brackets []

Pattern	Matches
`[wW]oodchuck`	Woodchuck, woodchuck
`[1234567890]`	Any digit

(3)Ranges `[A-Z]`

Pattern	Matches	
`[A-Z]`	An upper case	`Drenched Blossoms`
`[a-z]`	A lower case letter	`my beans were impatient`
`[0-9]`	A single digit	`Chapter 1: Down the Rabbit Hole`

Regular Expressions: Negation in Disjunction

(1) Negations `[^Ss]`

- Carat means negation only when first in []

Pattern	Matches	
`[^A-Z]`	Not an upper case letter	`Oyfn pripetchik`
`[^Ss]`	Neither ‘S’ nor ‘s’	`I have no exquisite reason”`
`[^e^]`	Neither e nor ^	`Look here`
`a^b`	The pattern a carat	`Look up a^b now`

Regular Expressions: More Disjunction

(1)Woodchuck is another name for groundhog!

(2)The pipe | for disjunction

Pattern	Matches
`groundhog|woodchuck`	woodchuck
`yours|mine`	yours
`a|b|c`	= [abc]
`[gG]roundhog|[Ww]oodchuck`	Woodchuck

Regular Expressions: ? *+ .

Pattern	Matches	
`colou?r`	Optional previous char	color colour
`oo*h!`	0 or more of previous char	oh! ooh! oooh! ooooh!
`o+h!`	1 or more of previous char	oh! ooh! oooh! ooooh!
`baa+`		baa baaa baaaa
`beg.n`		begin begun begun

Regular Expressions: Anchors ^ $

Pattern	Matches
^[A-Z]	Palo Alto
^[^A-Za-z]	1 "Hello"
\.$	The end.
.$	The end? The end!

Example

Find me all instances of the word “the” in a text

```
the
```

Misses capitalized examples

```
[tT]he
```

Incorrectly returns `other` or `theology`

```
[^a-zA-Z][tT]he[^a-zA-Z]
```

Errors

The process we just went through was based on fixing two kinds of errors:

- Matching strings that we should not have matched (there, then, other)

False positives (Type I errors)

- Not matching things that we should have matched (The)

False negatives (Type II errors)

Errors cont.

(1) In NLP we are always dealing with these kinds of errors.

(2) Reducing the error rate for an application often involves two antagonistic efforts:

- Increasing accuracy or precision
  (minimizing false positives)

- Increasing coverage or recall
  (minimizing false negatives).

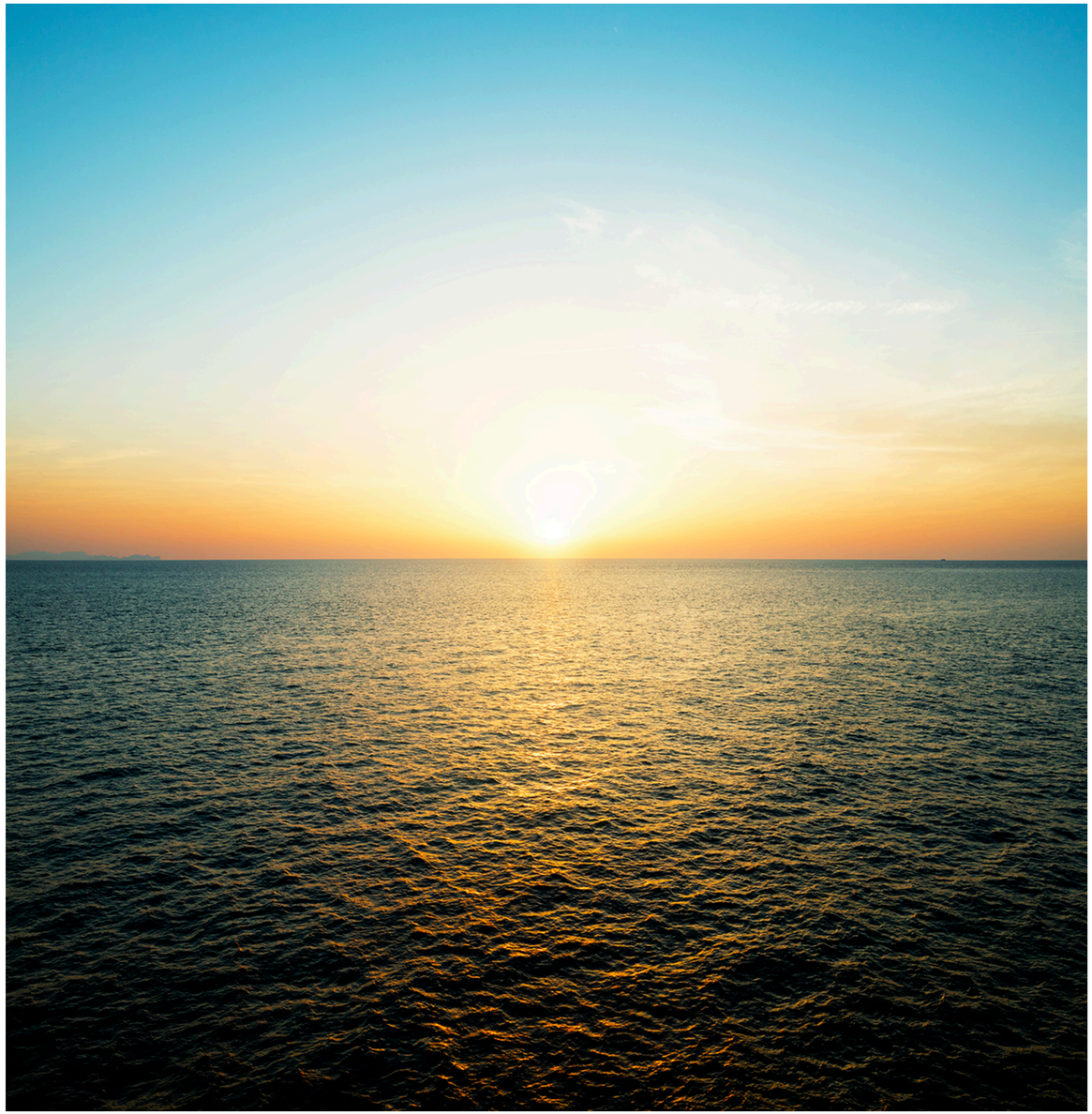
Summary

Regular expressions play a surprisingly large role

- Sophisticated sequences of regular expressions are often the first model for any text processing text

For hard tasks, we use machine learning classifiers

- But regular expressions are still used for pre-processing, or as features in the classifiers
- Can be very useful in capturing generalizations



End of the Week 1

Welcome to Labs!

Break, 1 week