

Introduction to Computational Linguistics

Session 1: Introduction and Definition

Dr. Stephen Bodnar, Denise Löfflad

Universität Tübingen

October 25, 2023

Plan

- 1 Course Organization
- 2 Definition
- 3 Different Perspectives on the Field
- 4 Different Methodologies
- 5 Relation to General Linguistics
- 6 References

Organization

See syllabus

https://moodle.zdv.uni-tuebingen.de/pluginfile.php/34474/mod_resource/content/1/syllabus.pdf

Organization – Exam

- the exam will be held separately from the Java 1 exam
- the grades of Java 1 and Intro to CL will be combined (2/3 Java, 1/3 Intro to CL) for the grade of the module exam (“Modulklausur”)

Organization

Please let us know about your background:



Introduction

What is Computational Linguistics?

What do you imagine Computational Linguistics is about ?

- definition
- characteristics
- applications

Definition

Computational Linguistics

Definition

Computational Linguistics

- bringing Language and Computers together

Definition - Association for Computational Linguistics (ACL)

"Computational linguistics is the scientific study of language from a computational perspective. Computational linguists are interested in providing computational models of various kinds of linguistic phenomena. These models may be "knowledge-based" ("hand-crafted") or "data-driven" ("statistical" or "empirical")."

<https://www.aclweb.org/portal/what-is-cl>

Definition - Jurafsky & Martin

"interdisciplinary field with many names corresponding to its many facets, names like speech and language processing, human language technology, natural language processing, computational linguistics, and speech recognition and synthesis. " [Jurafsky and Martin, 2009, p. 1]

Goal for CL

“ The goal of this field is to get computers to perform useful tasks involving human language ” [Jurafsky and Martin, 2009, p. 1]

Natural language vs. formal language

"In contrast to artificial languages such as programming languages and mathematical notations, natural languages have evolved as they pass from generation to generation, and are hard to pin down with explicit rules"
[Bird et al., 2009, p. 9]

Definition - Grishman

“Computational Linguistics is the study of computer systems for understanding and generating natural language.” [Grishman, 1986, p. 4]

Different Perspectives on the Field

- theory perspective: describing/defining content and format of descriptions of phenomena
- method perspective: developing tools, procedures, formalism
- task perspective: applying methods to solve a specific task

Different Methodologies

- rule-based approaches: explicitly model language
- statistical approaches: implicitly model language

How does language relate to this ?

Relation to General Linguistics

"General Linguistics is concerned with human language as a universal and recognizable part of human behavior and of the human faculties, perhaps one of the most essential to human life as we know it, and one of the most far-reaching of human capabilities in relation to the whole span of mankind's achievements[...] Language in all its forms and manifestations, that is all the languages of the world and all the different uses to which in the various circumstances of mankind they are put, constitutes the field of the linguist." [Robins, 2014, p.2-3]

Relation to General Linguistics

“ [C]omputational and theoretical linguists have rather different approaches and outlook. Computational linguists have been concerned with developing procedures for handling a useful range of natural language input...Theoretical linguists, in contrast, ... are concerned with language universals - principles of grammar which apply to all natural languages ”
[Grishman, 1986, p. 6]

Natural Language Processing

There are two main subtasks of **Natural Language Processing (NLP)**:

- **Natural Language Understanding (NLU)**
- **Natural Language Generation (NLG)**

Natural Language Understanding (NLU)

- ① Mapping the given input in natural language into useful representations, which involves analyzing different aspects of the input:
 - lexicon (e.g. **POS tagging**)
 - morphology (e.g. **lemmatization**)
 - syntax (e.g. **dependency parsing**)
 - discourse (e.g. **anaphora resolution**)
 - pragmatics (e.g. **sentiment detection**)

Natural Language Generation (NLG)

- ② Producing meaningful natural language text from some more abstract representation
 - difficult to get right, very subtle problems
 - much less developed than NLU
 - very domain-specific (generating weather reports \neq writing poems)

Applications of NLU and NLG

- Where can you find these two technologies being used today?
- Where might we see them in the future?

Next class

- No lecture next Wednesday, Nov. 1st (public holiday)
- Instead, Reading Assignment:
Chapter 1 in Dickinson et al. [2012] "Language and Computers"

References and Acknowledgments

The contents about the historical perspective are adapted from Frank Richter's *Introduction to Computational Linguistics* seminar from the winter semester 2012/13.

Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.*" O'Reilly Media, Inc.", 2009.

Markus Dickinson, Chris Brew, and Detmar Meurers. *Language and computers.* John Wiley & Sons, 2012.

Ralph Grishman. *Computational linguistics: an introduction.* Cambridge University Press, 1986.

Daniel Jurafsky and James H Martin. Naïve bayes classifier approach to word sense disambiguation. *Computational Lexical Semantics*, 2009.

Robert Henry Robins. *General linguistics.* Routledge, 2014.

Introduction to Computational Linguistics

Session 2: Encoding Language

Denise Löfflad

Universität Tübingen

November 8, 2023

Plan

- What is Language?
- Writing Systems
- Bits and Bytes

Mental Health

Studying and life in general can be very challenging and sometimes it can take a toll on your mental health.

- ① On the website www.werhilftweiter.de you can find information on all kinds of help in the Tübingen area, including help with mental health issues.
- ② You can download the app *Krisenkompass*. It supports you step-by-step in crisis situations, no matter if you need help yourself or if you want to help someone.
- ③ Zentrale Studienberatung <https://uni-tuebingen.de/studium/beratung-und-info/zentrale-studienberatung/>
- ④ The emergency numbers 112 and 116 117 are not only available for physical emergencies but also for mental health crisis situations!

Definitions of Language

What is Language? <https://www.menti.com/alfbyro5z869>



What is Language?

"Language is at the heart of all things human." Archibald and O'Grady [2008]

"Language is a structured system of communication that consists of grammar and vocabulary. It is the primary means by which humans convey meaning," Wikipedia¹,

"A common language connects the members of a community into an information-sharing network with formidable collective powers" Pinker [1995]

According to linguistics, language can be described with regards to phonetics, phonology, morphology, syntax, semantics, and pragmatics

¹<https://en.wikipedia.org/wiki/Language>, last accessed Nov. 8 2023.

Writing Systems

What is writing?

"a system of more or less permanent marks used to represent an utterance in such a way that it can be recovered more or less exactly without the intervention of the utterer." Daniels and Bright [1996]

Different types of writing systems are used:

- Alphabetic
- Syllabic
- Logographic

Alphabetic systems

Alphabets (phonemic alphabets)

- represent all sounds, i.e., consonants and vowels
- Examples: Etruscan, Latin, Korean, Cyrillic, Runic, International Phonetic Alphabet (<https://www.ipachart.com/>)

Abjads (consonant alphabets)

- represent consonants only (sometimes plus selected vowels; vowel diacritics generally available)
- Examples: Arabic, Aramaic, Hebrew

Syllabic systems

Symbols represent syllables which make up words

- all human languages have syllables, though some syllable systems are easier than others
- simple syllable structures → relatively small set of possible syllables

Logographic systems

Symbols represent syllables which make up words

- A logograph represents a unity of meaning
- most natural language writing systems that use logographs are not purely logographic

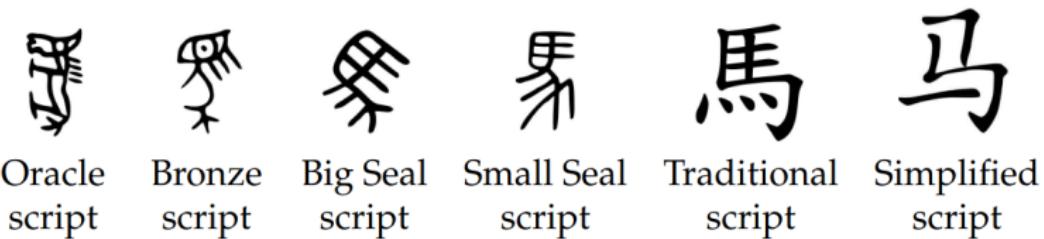


Figure: Evolution of chinese character *horse*

Other systems

- Some systems (like Korean) use hybrids of e.g. syllables and alphabets
- Are emojis a writing system?

Encoding Language



<https://www.educba.com/types-of-computer-language/>

- How to encode writing systems on the computer?
- How do we store anything on a computer?
- → bits & bytes

Bits and Bytes

- **bit**: a unit of information with one of two possible values
common representations: true and false, on and off, 0 and 1
- **byte**: a sequence of 8 bits e.g. 10101101 or 10000000
- Note: talk about big endian notation, where most significant is leftmost (vs little endian)
- b = bit = binary digit
- B = byte

Binary Numbers

- the binary system is base 2
- this means we only use two symbols: 0 and 1
- We can combine these symbols too,
E.g., 11, 1001, 10000000
- Position k of digit represent multiples of 2^k : 1, 2, 4, 8, 16, 32, ...
- Let's look at the binary number 00101101

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	0	1	0	1	1	0	1

Binary Numbers

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	0	1	0	1	1	0	1

$$2^7 \cdot 0 + 2^6 \cdot 0 + 2^5 \cdot 1 + 2^4 \cdot 0 + 2^3 \cdot 1 + 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 1$$

↔

$$32 + 8 + 4 + 1 = 45$$

Hexadecimal System

- the hexadecimal system is base 16
- letters of the alphabet are used to arrive at 16 symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, where A = 10, B = 11, C = 12, D = 13, E = 14, F = 15
- Position k of digit represents multiple of 16^k : 1, 16, 256, 4096, ...
- Example: 0xA8E is interpreted as

$$10 \cdot 16^2 + 8 \cdot 16^1 + 14 \cdot 16^0 = 2702$$

Binary \leftrightarrow Hexadecimal

How to convert the binary number 100101110 to hexadecimal?

- 4 positions in binary can be represented with 1 position in hexadecimal
- → group the binary number into positions of 4 (starting from the right) and convert to hexadecimal

How to convert the hexadecimal 0xB5F08 to binary?

- 1 position in hexadecimal can be represented with 4 positions in binary
- → convert every position to binary

Binary \leftrightarrow Hexadecimal

How to convert the binary number 100101110 to hexadecimal?

- [0001][0010][1110]
- [1110] = 14 = E
- [0010] = 2
- [0001] = 1
- $\rightarrow 0x12E$
- (302 in decimal)

Binary \leftrightarrow Hexadecimal

How to convert the hexadecimal number 0xB5F08 to binary?

- 8 = 1000
- 0 = 0000
- F = 1111
- 5 = 0101
- B = 1011
- \rightarrow 10110101111100001000
- (745224 in decimal)

ASCII

Using 8 bits and where each byte stores a separate character, we can represent 256 different characters. To ensure compatibility across systems, we need encoding systems.

- This is enough to store every character from the Latin alphabet for English, plus additional characters such as space, comma, etc.
- ASCII, the **American Standard Code for Information Interchange**, uses 7 bits (128 characters)

ASCII

USASCII code chart				
b ₇	b ₆	b ₅	b ₄	b ₃
0	0	0	0	0
1	0	1	1	1
0	0	0	0	0
1	0	1	1	1
2	0	1	0	0
3	0	0	1	1
4	0	1	0	0
5	0	0	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Bits → Column ← Row ↓

0 NUL DLE SP @ P ` p
1 SOH DC1 ! 1 A Q a q
2 STX DC2 " 2 B R b r
3 ETX DC3 # 3 C S c s
4 EOT DC4 \$ 4 D T d t
5 ENQ NAK % 5 E U e u
6 ACK SYN & 6 F V f v
7 BEL ETB ' 7 G W g w
8 BS CAN (8 H X h x
9 HT EM) 9 I Y i y
10 LF SUB * : J Z j z
11 VT ESC + ; K [k {
12 FF FS . < L \ l |
13 CR GS - = M] m }
14 SO RS . > N ^ n ~
15 SI US / ? O — o DEL

Figure: ASCII code table

ASCII Limitations

ASCII has severe limitations for non-US usage:

- no umlauts
- no accents or other diacritics
- no symbols for currencies other than dollar

Result: different modifications were established in different countries, replacing different characters by symbols of local significance, e.g.:

- ISCII (Indian scripts),
- TSCII (Tamil),
- VSCII (Vietnamese)
- sometimes changes are small, e.g. JIS C-6220 in Japan, replacing the backslash with a Yen sign
- → risk of misidentification

Unicode

Unicode has a single representation for every character in any existing writing system

Unicode uses 32 bits to encode characters, i.e.

$$2^{32} = 4,294,967,296$$

unique characters.

idea: provide one universal encoding for all languages, both current and historic writing systems, symbols

Solution: UTF-32, UTF-16, and UTF-8

Unicode UTF-8

- UTF-8 allows for variable length
- one byte utf-8 character is direct mapping to ASCII encoding, allowing for backwards compatibility with ASCII (when something is encoded in ascii, it can be read in utf-8)
- leftmost bit indicates how many bytes are used:
 - 0..... → one byte
 - 110..... → two bytes (following: 10.....)
 - 1110..... → three bytes (following: 10.....)
 - ...

Next session

For the next session, read the chapter on Writer's Aids!

The first assignment is released today, submit until Nov. 22 in PDF form.
Don't forget the zero-points sheet. If you have questions, use the moodle forum.

References and Acknowledgments

These slides are largely based on Dickinson et al. [2012] and partly based on slides for Text Technology, 2023 by Stephen Bodnar and on Detmar Meurers' slides on Second Language Acquisition.

John Archibald and William O'Grady. *Contemporary linguistic analysis*, 2008.

Peter T Daniels and William Bright. *The world's writing systems*. Oxford University Press, 1996.

Markus Dickinson, Chris Brew, and Detmar Meurers. *Language and computers*. John Wiley & Sons, 2012.

Steven Pinker. *The language instinct: The new science of language and mind*, volume 7529. Penguin UK, 1995.

Introduction to Computational Linguistics

Session 3: Writing Aids

Stephen Bodnar

Universität Tübingen

November 15, 2023

Assignment 1

- How is it going ? Questions ?
- Please upload submission on Moodle before November 22, 2023, 16:10
- Submit answers in a single .pdf file
- In case of questions: please use the moodle discussion forum
- Don't wait too long to start!

Reading Assignment

- Language & Computers - Chapter 2

Plan

- 1 Types of Writing Errors
- 2 A first practical application: Automatic Spell Checking
- 3 References

Introduction

"I don't see any use in having a uniform and arbitrary way of spelling words. We might as well make all clothes alike and cook all dishes alike."

- Mark Twain (<http://www.twainquotes.com/Spelling.html>)

Spelling is a convention we all usually follow, but not always

- Regional differences: *colour* vs *color*
- Literature:
 - "...there wos other genlmen come down" (Charles Dickens' Bleak House)
- Text messages: " I dunno, cos the bus is l8?"

Question

Why is standardized spelling useful?

Standardised Spelling's Usefulness

Some reasons:

- Communicating across groups speaking different pronunciations / dialects / regional differences
- Easy lookup of words in dictionaries (e.g. *itinerary* vs. *aitinerary* vs. *etinerary*)
- Meaning more directly accessible through the 'shape' of words (vocalisation step unnecessary)
- Facilitates automatic processing, e.g. optical character recognition, search engines

Non-word errors

Non-word errors

- Example: “hte” instead of “the”
- String of characters that does not exist as a word in the language
- Different reasons:
 - typographical error: error in typing, person knows how to type it correctly
 - spelling confusion: person lacks knowledge for correct spelling
- Different keyboard layouts

Non-word Errors and Intransparent Writing Systems

Intransparent Writing Systems: Often no direct correspondence between phonological and character representation, e.g.

- silent characters (“knight”)
- similar-sounding characters (“s”/‘c”/“z”)
- long/short vowels (“receive”)
- doubled consonants (“application”)

Other Non-word Error Types

- false friend/transfer error
 - e.g. “brave” (EN) vs. “brav” (DE)
- split errors (“auto matic”)
- run-on errors (“machinelearning”)
- space in wrong place (“atoll way” vs. “a toll way”)

→ exact cause of error difficult to determine

Describing Non-Word Errors

Spelling errors resulting in non-words can be described in terms of operations:

- insertion (“applicaiton”)
- deletion (“brekfast”)
- substitution (“artificiel”)
- transposition (“autoamtic”)

Single vs. Multi-Error Misspellings

Can distinguish between:

- single-error misspelling
- multi-error misspelling (multiple spelling errors in one word)

Trends

- around 80% single-error misspellings
- most misspellings differ in length from target by at most two characters
- first letter most often spelled/typed correctly

Real-Word Errors (only a glance)

- local syntactic error

Their is a problem.

- long-distance syntactic error

The next group of three hundred Tour de France cyclists are about to begin the race.

- semantic errors

For the preparation of his studies, he usually borrows cooks from the library.

- repetitions

This this is something you can often see in a text.

Introduction to Automatic Spell Checking

Aim: Detect and correct non-word errors

Examples: Microsoft Word, but also Linux's *ispell* program

Key steps:

- ① error detection
- ② generation of candidate corrections
- ③ ranking of candidate corrections

Information Sources For Automatic Spell Checking

Different tasks (detection, generation, ranking) employ different information sources:

- Knowledge-based approaches (hand-crafted)
- Statistical approaches (data-driven)

Examples of errors

- *I sat down and opened hte book.*
- *That factory's speciality is wooden boets; they are nice but they cost a lot.*

When would each approach be useful ?

Detecting Non-Word Errors

Fairly straightforward

- Search for user input in a trusted word list, e.g., aspell
<http://wordlist.aspell.net/dicts/>

Generating Candidate Corrections

Use rule-based approaches to suggest corrections:

- Modeling of frequent errors: “hte” instead of “the”
e.g. LanguageTool

Search through well-formed word list for similar character sequences

- Relies on *similarity measures*: function that quantifies similarity between any two strings.

Defining similarity

- Imagine we want to build a spell checker, and that we need to invent a similarity measure
- Let's look at three proposals ...

Defining word similarity - Proposal 1

What happens if we define word similarity as word length?

- low difference in word length => word is similar

Can try out some examples

- ➊ $\text{calcDifInLen}(\text{'fyr'}, \text{'fry'}) = 0$
low size difference => similar?
- ➋ $\text{calcDifInLen}(\text{'firetruck'}, \text{'fry'}) = 4$
large size difference => dissimilar
- ➌ $\text{calcDifInLen}(\text{'zzz'}, \text{'fry'}) = 0$
low size difference => similar???

OK, so that definition doesn't work, but let's try another.

Defining word similarity - Proposal 2

What happens if we define word similarity as number of differences?

- loop over characters in the two words; count positions with unmatching characters
- more mismatches => less similar

Examples

- ① `countDifChars('fyr', 'fry')` = 2
- ② `countDifChars('zzz', 'fry')` = 3
- ③ `countDifChars('firetruck', 'fry')` = 8
- ④ `countDifChars('fretruck', 'firetruck')` = 8
- ⑤ `countDifChars('ffiretruck', 'firetruck')` = 9

Results:

- 1) and 3) are fine, 2) is an improvement over length definition
- BUT 4) and 5) have many differences due to shift, poor result

Conclusion: Not a good measure either.

Observations from proposed definitions

Requirements for similarity measure

- should reflect the similarity of contents
- needs to be robust against slight misalignments

Let's now look at the popular Minimum Edit Distance measure.

Defining word similarity - Minimum Edit Distance

Define similarity as the number of edit operations needed to transform a non-word into a dictionary word.

- more edit operations needed => less similar
- seems flexible, like it would align with how humans judge word similarity
- but how to program? not as straightforward as previous proposals

Minimum Edit Distance - Programming Ideas

How to program edit distance calculation?

- For inspiration, can look at the fine-grained steps a human typist would perform.
- Example: non-word 'fyr' and dictionary word 'fry'
- Typist A might
 - ① put the cursor between the 'f' and the 'y' and type an 'r' key (insert)
 - ② move the cursor to the end of the word and remove the 'r' character (delete)
- Typist B might
 - ① delete first 'y' character
 - ② insert a 'y' at the end

Minimum Edit Distance - Many Possibilities

In fact, many many possible sequences for transforming 'fyr' into 'fry':

- Another (longer) possibility:
 - ① placing cursor after 'f' character,
 - ② delete two characters
 - ③ insert 'r' and 'y'
- A long but important sequence:
 - ① delete all characters in 'fyr'
 - ② insert all characters in 'fry'

As the textbook points out, can even imagine absurd sequences, like placing the cursor at 'f', then inserting 'f' followed by deleting 'f' 100 times.

Summarising so far

Key points:

- multiple ways to transform 'fyr' into 'fry'; some require more edit operations than others
- of the many possible ways, we are interested in the shortest sequence of operations that yields 'fry'
- to find shortest sequence, we need to explore all reasonable sequences of edit operations that lead to 'fry'
- also need to avoid exploring ridiculous sequences

Need way to set up problem that guides search behaviour.

Directed Acyclic Graphs - A Useful Tool

Directed Acyclic Graphs (DAGs) are a useful tool for thinking about problems with sequences

- Nodes represent the current state of a sequence
- The arcs represent actions

Example: a simple todo list

- Before starting -> Important email sent -> Programming task unfinished -> Relaxing at home
- First '->' arc represents action of writing of an email

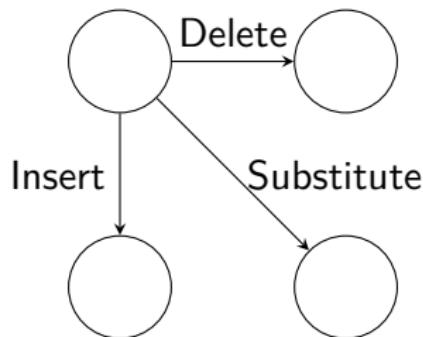
Important: DAGs can be effectively translated into computer programs

Slides that follow based on Dickinson, Brew, & Meurers (2013)

Applying DAGs to edit distance calculation

We can represent a sequence of edit operations

- Each node is the state of the word at a particular point during transformation
- Each arc is an edit operation (delete, insert, substitute) with cost = 1
 - Exception: substituting same character costs 0.



Representing full edit distance calculation with a DAG

Use example of transforming 'fyre' into 'fry'

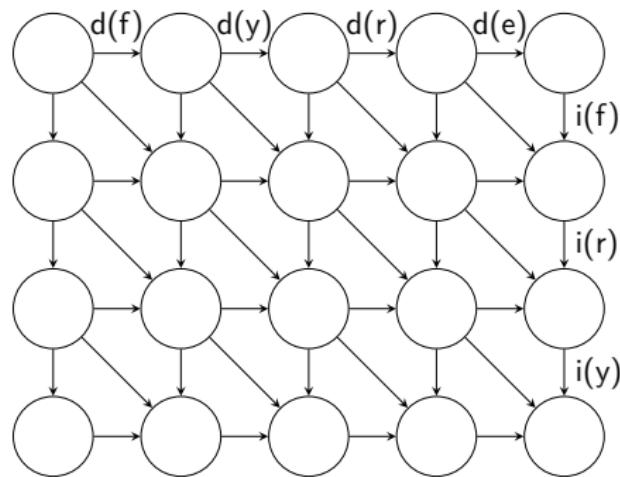
- First, add start node representing state 'fyre' (top left)
- Next, add end node representing state 'fry' (bottom right)
- Then, add all possible paths from start to end.
 - Can begin by adding the 'important' sequence from above:
four deletes followed by three inserts
 $\text{fyre} \rightarrow \text{yre} \rightarrow \text{re} \rightarrow \text{e} \rightarrow " \rightarrow \text{f} \rightarrow \text{fr} \rightarrow \text{fry}$
 - Can follow by adding three inserts followed by four deletes
 $\text{fyre} \rightarrow \text{ffyre} \rightarrow \text{frfyre} \rightarrow \text{fryfyre} \rightarrow \text{fryyre} \rightarrow \text{fryre} \rightarrow \text{frye} \rightarrow \text{fry}$
 - Question:
What edit operation hasn't been featured in any of our paths yet?

Representing full edit distance calculation with a DAG - cont.

- Paths added so far
 - ➊ Four deletes followed by three inserts
fyre -> yre -> re -> e -> " -> f -> fr -> fry
 - ➋ Three inserts followed by four deletes
fyre -> ffyre -> frfyre -> fryfyre -> fryyre -> fryre -> frye -> fry
 - ➌ Three substitutions followed by a delete
fyre -> fyre -> frre -> frye -> fry
 - ➍ Can continue adding arcs in to include all possible edit sequences

Computing edit distances - An example

- Four deletes followed by three inserts for transforming *fyre* to *fry*



Computing the minimum edit distance

Very first solution to this problem was recursive one

```
1 public int calc_cost(String userInput, String dictionaryCandidate) {  
2  
3     if(userInput.length == 0){  
4         return dictionaryCandidate.length  
5     }  
6     else if(dictionaryCandidate.length == 0){  
7         return userInput.length  
8     }  
9     else {  
10        var costToReachNewNode =  
11            0 if userInput[0] == dictionaryCandidate[0] else 1  
12  
13        return costToReachNewNode + minimum(  
14            // explore delete operation  
15            calc_cost(userInput[1:], dictionaryCandidate),  
16  
17            // explore insert operation  
18            calc_cost(userInput, dictionaryCandidate[1:])),  
19  
20            // explore substitution operation  
21            calc_cost(userInput[1:], dictionaryCandidate[1:]))  
22    }  
23 }  
24 }
```

Computing the minimum edit distance - Naive

- Traverses all paths in DAG, calculates cost of shortest path
- Re-computes same paths multiple times => inefficient
- Can memoize recursive implementation for more efficiency (see accompanying file)
- Exists also an iterative function to traverse the DAG (for those interested)

Steps involved in spell checking

- ① error detection
- ② generation of candidate corrections
- ③ => ranking of candidate corrections

Spell Checkers - Ranking Stage

Goal: rank good candidates by how well they fit the context

- context means adjacent language units
- today we look at word context
- in ranking stage we rely on statistical knowledge sources

Candidate Ranking - Word Context

Example

- *That factory's speciality is wooden boets; they are nice but they cost a lot.*
- What suggestions should our spell checker offer for *boets*?
- How to decide between these candidates ?
- We need a resource that we can ask questions like
“Given this sentence, which of these N words fits best?”

Candidate Ranking - Language Model

- Problem can be solved to some degree with a Language Model
 - Models the likelihood of word sequences
 - Likelihood(*That factory's speciality is wooden **boots**; they are nice but they cost a lot.*) = relatively low
 - Likelihood(*That factory's speciality is wooden **boats**; they are nice but they cost a lot.*) = relatively high
 - Very often implemented with N-grams
 - More later in the course!

References and Acknowledgments

These slides are based on slides for Intro to CL, 2021 by Dr. Björn Rudzewitz.

The contents and structure of the slides are based on the book “Language and Computers” by Chris Brew, Detmar Meurers, and Markus Dickinson.

Introduction to Computational Linguistics

Session 4: Text as Data

Denise Löfflad

Universität Tübingen

November 22, 2023

Plan

- Corpus Linguistics / Corpora
 - Data collection
 - Data Modeling
 - Corpus Annotation
- How are words distributed in a text?
 - Zipf's Law
- Word meanings as vectors

History of (Computational) Linguistics

Meyer [2023], McEnery [2019]

- Before the 1960s, generative grammar dominated linguistics
- Linguistic analysis were created based on the intuitive knowledge of the researcher
- Corpus linguistics was criticized due to the restricted size & because corpora contain real-life, ill-formed data

Corpus Linguistics



Corpus Linguistics

What is a Corpus?

Structured collection of texts, collected with a specific question in mind.
Usually with metadata etc

- Structured collection of texts (typically with linguistic annotations)
- Often collected with a specific linguistic question in mind (to help to know what kind of data to collect))
- Usually digitized and represented with special formats (e.g., XML)
- Choice of format depends on purpose of corpus (e.g., NLP, philology)
- It is good practice to make corpora freely available to other researchers, but in reality that's not always done/not always easy

Definition Corpus

- structured collection of utterances
- machine readable
- components:
 - raw, original data (primary)
 - meta data
 - annotation guidelines

Example Corpus

Brown Corpus

- First modern Corpus
- Computer readable (available from different sources, but often as .txt)
- Approx. 1 million words from written American sources (running text and English prose)
- 15 different categories, 500 texts, approx. 2k words per text, collection of text samples
- intended purpose: study word frequencies and distributions of everyday language use
- word class tagging (e.g. POS)
- Freely available

Brown Corpus - Data

I_PPSS answered_VBD the_AT routine_JJ question_
NN about_IN my_PP\$ itinerary_NN ,_, rather_QL coolly_RB ...
Chiang_NP spoke_VBD again_RB ,_, this_DT time_NN at_IN
greater_JJR length_NN ...the_AT President_NN says_VBZ ,_,
the_AT translator_NN came_VBD in_RP ,_, that_CS the_AT
reason_NN he_PPS asked_VBD you_PPO where_WRB you_PPSS
were_BED going_VBG is_BEZ because_CS he_PPS hoped_VBD
you_PPSS would_MD be_BE visiting_VBG other_AP areas_NNS
in_IN Southeast_JJ Asia_NP ,_, and_CC that_DT everywhere_RB

source: Brown corpus http://www.sls.hawaii.edu/bley-vroman/browntag_nolines.txt

Zipf's Law

Corpus linguistics can be used to analyze the frequencies of certain words, which can be useful for tasks like topic modeling, authorship attribution, (second) language development ...

Zipf's Power Law:

- Frequency is inversely proportional to its frequency rank.
- Type-token distinction explained: Unique word types vs. specific usages (tokens).
- Example from Brown Corpus: "the" (1st rank) accounts for 6% of tokens, "of" (2nd rank) for 3%, and so on

Zipf's Power Law

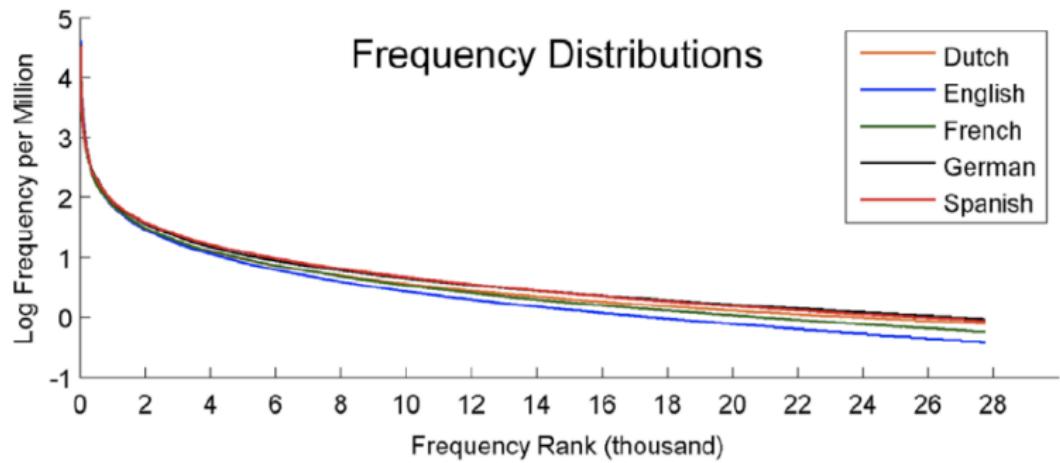


Figure: Log frequency of words as a function of their frequency rank (ordered from left to right as the first quartile word, the second most frequent, and so on). Figure from Marian et al. [2012].

Mandelbrot [1961] Carroll [1967]

Zipf's Brevity Law

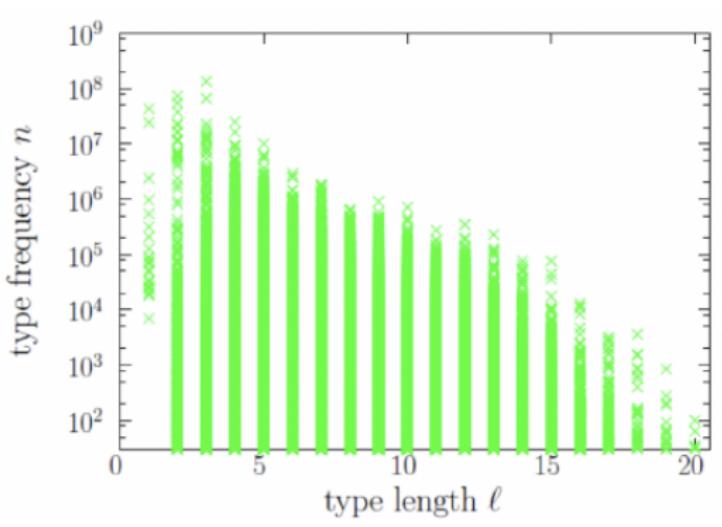
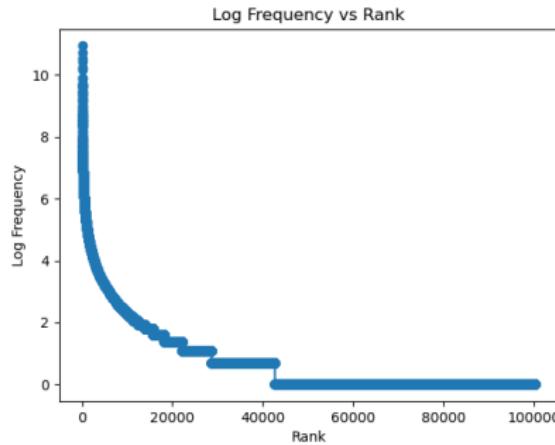
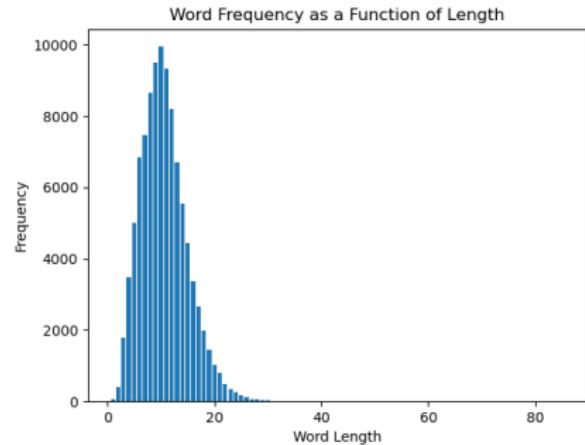


Figure: Frequency of English words as a function of their length in orthographic letters. Figure from Corral and Serra [2020].



(a) Rank vs Log Frequency



(b) Word Frequency as a Function of Length

Figure: Testing Zipf's Law on our data. Corpus included about 3k texts, 11208627 tokens, and 100327 types

```
[('die', 56495),  
 ('der', 45026),  
 ('und', 36994),  
 ('in', 33694),  
 ('das', 27532),  
 ('ist', 26199),  
 ('sie', 19982),  
 ('es', 19491),  
 ('den', 16443),  
 ('ein', 16354),  
 ('nicht', 15697),  
 ('auch', 15541),  
 ('zu', 15392),  
 ('mit', 14928),  
 ('„', 14841),  
 ('für', 14226),  
 ('ich', 12935),  
 ('eine', 12647),  
 ('im', 12593),  
 ('aber', 11805)]
```

Figure: 20 most frequent words in Spotlight Corpus.

Type Token Ratio

- $\# \text{words} = \# \text{tokens}$
- $\# \text{different words} = \# \text{types}$
- $\text{TTR} = \frac{\# \text{types}}{\# \text{tokens}}$
- high TTR indicates high lexical variation
- BUT affected by text length!

Word vectors

visualization of word vectors

- we use number vectors to represent words
- for computers, strings are only sequences of characters with no meaning
- there are pre-trained models you can use to vectorize words, e.g. Word2Vec in python
- vectors allow to represent words in a coordinate space and represent relations between words

Word vectors

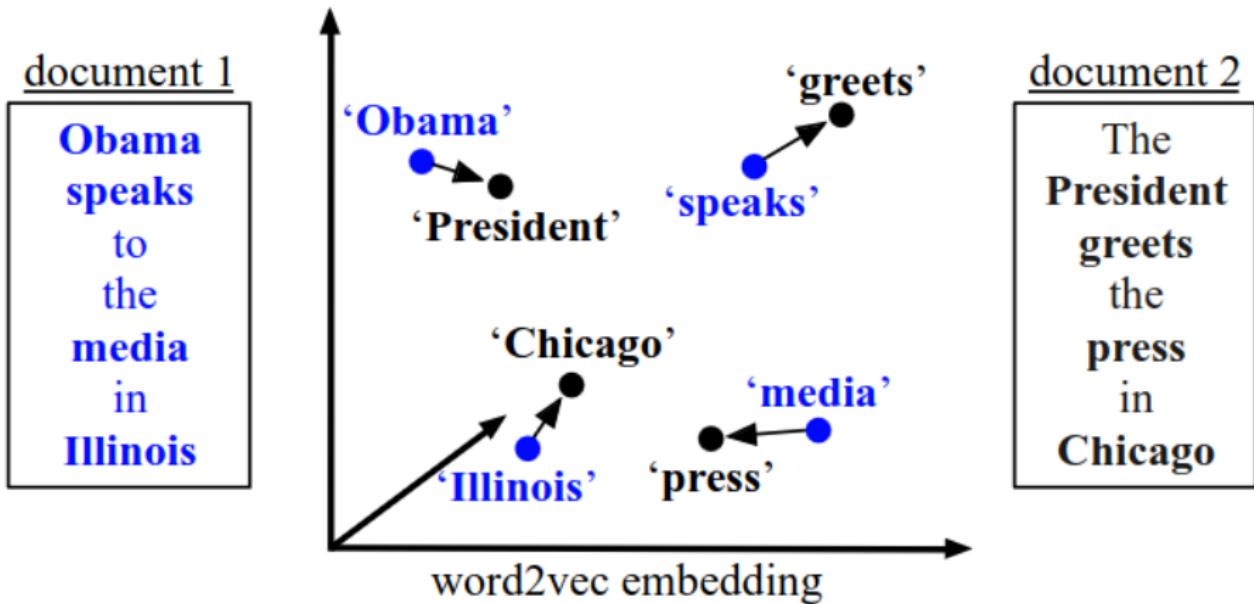


Figure: Word Vector Embedding from Kusner et al. [2015]

Role of Corpora

- testing (linguistic?) theories
- searching relevant examples
- but:
 - corpus is never fully representative or balanced
 - absence/infrequency of construction not necessarily indicator of ungrammaticality, etc.
 - negative data: no data in search results doesn't mean it doesn't exist

Role of Corpora

- training/testing systems
- many different applications, e.g.
 - extracting frequencies of parts-of-speech for words
 - estimating n-gram frequencies from sentence-segmented text
 - training probabilities of parsing rules from trees
 - extracting errors in learner language
 - extracting markers for sentiments in corpora
 - ...

→ Many uses for corpora, depending on the task at hand

Design - Selecting Data

Questions to answer before collecting data:

- What domain/task is the data needed for ?
- From what sources can the data be accessed ? Is it legally and technically possible ?
- Is it ethically acceptable?
- How to select data representative for this domain ?
- How can the data be selected in a balanced fashion ?

Design - Data Modeling

- decide on a model and use it for annotating/creating the corpus
- data **modeling**
- About 'models':
 - they include some aspects of objects
 - other aspects are omitted
 - can include extended information: e.g. metadata not present in original data

Data Modeling - Labels and Structure

- For aspects included in model:
what categories / labels (system) to use ?
- How is the system organized ?
- Note: there often is more than one valid model

⇒ **Document your decisions !**

Annotation Manual

- documentation of decisions: annotation manual
- lists all possible annotations used in the annotation process plus explanations and examples
- describes procedure's "grey areas"
- usually built by previous knowledge, applied to pilot data, refined from pilot data, then applied to main data
- often useful to CL for building NLP tools

Annotation Manual Examples

- Manual for Brown Corpus:

<http://icame.uib.no/brown/bcm.html>

- Penn Treebank tag set:

<https://catalog.ldc.upenn.edu/docs/LDC99T42/tagguid1.pdf>

Next Stage: Corpus Annotation

- Raw data collection is completed
- Next step: Add *meta data* to original data

Metadata



Metadata

What is metadata ?

- data describing the primary, raw data
(e.g. author, creation date, tags, ...)
- often general meta data (created when, by whom, where, ...)
- computational linguistics: often linguistic annotations (sentence/word boundaries, POS tags, parse trees, ...) → different levels/layers

Workflow in a nutshell

Typical Workflow:

- ① Decide which data to use in the corpus.
- ② Collect the data.
- ③ Decide on an annotation scheme.
- ④ Pilot annotation: test and refine annotation scheme on data subset.
- ⑤ Annotate the rest of the data.
- ⑥ Validate the annotation.
- ⑦ Assign a license and publish the data and guidelines.

Annotating Data with Metadata

further considerations:

- size of corpus: manual, computer-assisted, automatic annotation
- annotation accuracy of humans and tools

Data Serialization

- corpus and annotations should be digital
- machine readable (easily processable by machines)
- well-formed (system strictly following defined rules)
- non-proprietary, non-binary format (better)
- archivable

Data Serialization

How to store the information *in practice* ?

- structured text (e.g. inline, table)
- markup (e.g. XML)
- Javascript Object Notation (JSON)

→ More on this in Text Technology

In Summary

Why annotate metadata ?

- systematic assignment of metadata enriches corpus
- search using abstractions (data classes) instead of compiling out all surface forms and context
- allows to learn from the data/gain insights
- annotated data can be reused by others (who possibly couldn't perform the annotation)

Annotations allow for a qualitative and quantitative analysis !

Attention Span



Further Reading

If you want to learn more about corpus linguistics, or if you have to create your own corpus or analyze a corpus, I recommend Meyer [2023] and McEnery [2019] (available through the university website <https://uni-tuebingen.de/einrichtungen/universitaetsbibliothek/suchen-ausleihen/>). These books are a good start, but as they are from the early 2000s, the methods might not be up-to-date, so look for more recent sources as well.

Acknowledgements

These slides are largely based on Dickinson et al. [2012] and partly based on slides for Text Technology, 2023 by Stephen Bodnar and on Detmar Meurers' slides on Second Language Acquisition.

References and Acknowledgments

- JB Carroll. On sampling from a lognormal model of word frequency distribution. *Computational analysis of present-day American English*, pages 406–424, 1967.
- Álvaro Corral and Isabel Serra. The brevity law as a scaling law, and a possible origin of zipf's law for word frequencies. *Entropy*, 22(2):224, 2020.
- Markus Dickinson, Chris Brew, and Detmar Meurers. *Language and computers*. John Wiley & Sons, 2012.
- Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In *International conference on machine learning*, pages 957–966. PMLR, 2015.
- Benoit Mandelbrot. On the theory of word frequencies and on related markovian models of discourse. *Structure of language and its mathematical aspects*, 12:190–219, 1961.
- Viorica Marian, James Bartolotti, Sarah Chabal, and Anthony Shook. Clearpond: Cross-linguistic easy-access resource for phonological and orthographic neighborhood densities. 2012.

Introduction to Computational Linguistics

Session 5: Text Classification

Denise Löfflad

Universität Tübingen

November 29, 2023

Plan

- How computers learn
- Supervised and unsupervised machine learning
- Evaluating your model
- NLP segmentation

We are **not** going to talk about Naives Bayes, the Perceptron and other machine learning algorithms as this is for future semesters!

How Computers Learn

- machine learning: machine induces patterns from observations in data
- model:
 - summary of what the algorithm learned
 - estimates based on seen data, but can be applied to unseen data

How Computers Learn

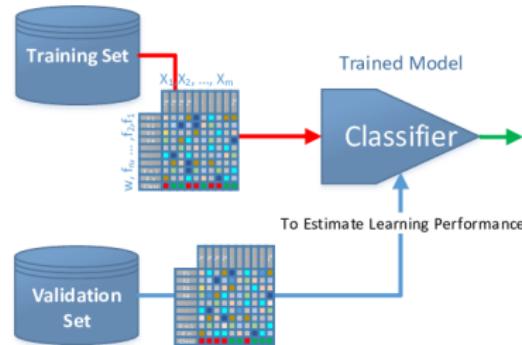
Features

- Dimensions of interest in data
- Quantifications of raw data
- Measured for all records in the data set
- Individual features often combined into a *feature vector*
- Example: Speech Intelligibility scoring
 - Features:
 - rate of speech (syllables per second)
 - total time speaking (seconds)
 - amplitude (decibels)
 - pronunciation nativeness (GOP scores, Link)
 - number of grammatical errors (error count)
 - Feature Vector: [1.7, 4, 55, -28, 2]
- Learning algorithms accept feature vectors as input

How Computers Learn

data often split into different sets

- training set
- test set
- development/validation set
or cross-validation



Source <https://www.researchgate.net/figure/>

The-learning-process-of-a-classifier-The-classifier-learns-from-the-training-data-The_fig1_333505093



Supervised Learning

- work with labelled data
- “supervised”: labels are determined by humans in the loop
 - e.g. annotations from corpora
- examples:
 - label how intelligible a recording is (good, fair, poor)

Supervised Learning

workflow:

- ① carefully formulate the RQ
- ② find or create appropriate data
- ③ define feature extracting mechanism and clean the data
- ④ decide on algorithm(s)
- ⑤ split the data
- ⑥ train the model
- ⑦ test and cross-validate
- ⑧ evaluate the model's performance
- ⑨ compare to baselines of StotA

compare with alternative algorithms/feature sets

Supervised Learning

Examples:

- linear regression
- logistic regression
- support vector machine
- decision tree
- classification tasks

Unsupervised Learning

- no pre-defined categories
- purpose: automatically detecting structure in data

Unsupervised Learning

workflow:

- ① define feature extracting mechanisms
- ② decide on algorithms
- ③ apply on data set
- ④ inspect resulting inferred structure

Unsupervised Learning

examples:

- k-means clustering
- hierarchical clustering

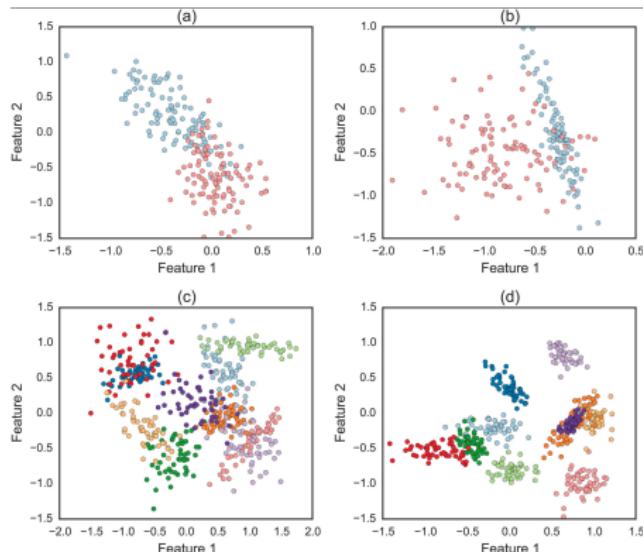


Figure: Taken from Rodriguez et al. [2019]

How to evaluate your (supervised) model?

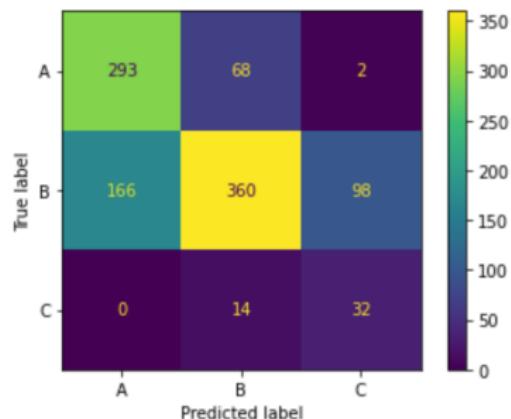
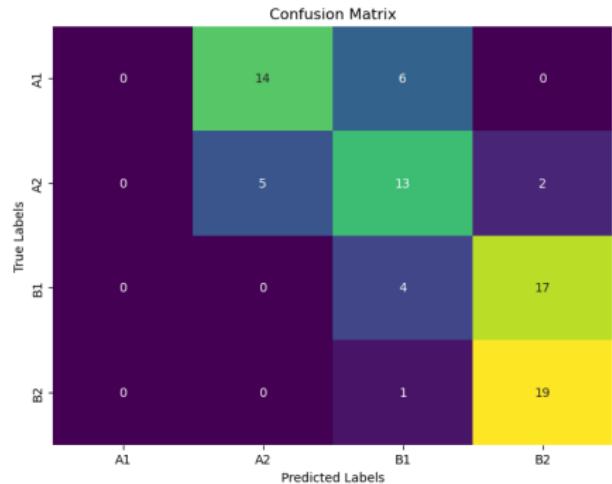


Figure: Confusion matrices on classification data

Measuring Success (binary classification)

	Spam Classified	Ham Classified	
Spam	TP	FN	Recall
Ham	FP	TN	TNR
Precision		True Omission Rate	

- Recall: $\frac{TP}{TP+FN}$
- Precision: $\frac{TP}{TP+FP}$
- Accuracy: $\frac{TN+TP}{TP+FP+FN+TN}$
- True negative rate (TNR): $\frac{TN}{FP+TN}$

Measuring Success

	Spam Classified	Ham Classified	
Spam	TP	FN	Recall
Ham	FP	TN	TNR
Precision		True Omission Rate	

- Precision: $\frac{TP}{TP+FP}$

Measuring Success

	Spam Classified	Ham Classified	
Spam	TP	FN	Recall
Ham	FP	TN	TNR
	Precision	True Omission Rate	

- Recall: $\frac{TP}{TP+FN}$

Measuring Success - Error Analysis

	Spam Classified	Ham Classified	
Spam	TP	FN	Recall
Ham	FP	TN	TNR
	Precision	True Omission Rate	

- True negative rate (TNR): $\frac{TN}{FP+TN}$

Measuring Success

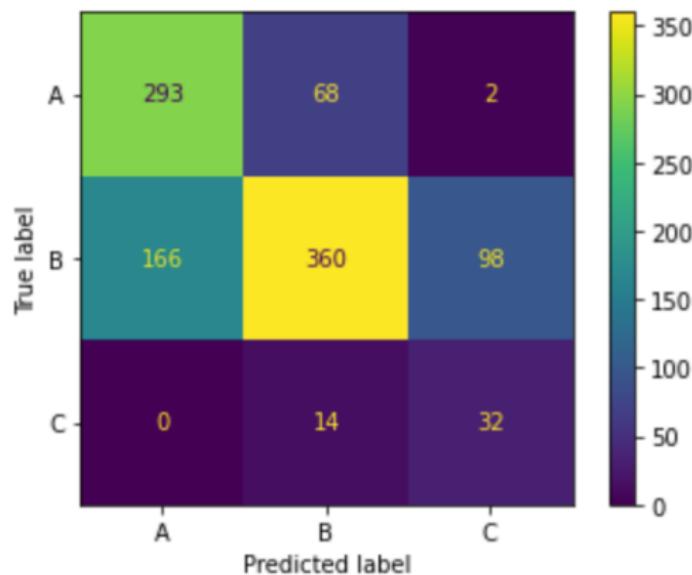
	Spam Classified	Ham Classified	
Spam	TP	FN	Recall
Ham	FP	TN	TNR
Precision		True Omission Rate	

- Accuracy: $\frac{TN+TP}{TP+FP+FN+TN}$

Measures Success - Multiclass classification

For multiclass classification you have different options:

- calculate the values for every class
- average (micro or macro) over the classes



Measuring Success

Which measures are important?

Measuring Success

Which measures are important?

- Would you rather see spam mails or miss work mails?
- Would you rather see hate speech or get your normal Tweet deleted?

→ always consider the task you are working on! There is no one-fits-all solution.

NLP Tasks - Segmentation

- raw input: input stream of characters/text
- output: input segmented into smaller units
 - tokens/“words” → tokenization
 - sentence → sentence segmentation
 - chunks: groups of tokens

Tokenization

"A token is an instance of a *sequence of characters* in some particular document that are *grouped together as a useful semantic unit for processing*. A type is the class of all tokens containing the same character sequence. "¹

¹<https://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html>, last accessed 2021-01-18

Tokenization

- split into smaller units
- not necessarily always a “word”
- many decisions to make to deal with challenges

Tokenization – Challenges

- contracted/enclitic forms
- example: “I don’t know whether he’s going there.
- one/multiple words/tokens ?

Tokenization – Challenges

- hyphenated forms
- example: “An end user device-independent solution - would that be possible?”
- meaning preserved when splitting into two tokens?

Tokenization – Challenges

- forms with/adjacent to periods
- example:
“You find it on 21st St. opposite of Third Ave., in the U.S.A.”

Tokenization – Challenges

- slashes
- example:
“A helpful/fun e-learning website is <https://feedbook.schule>”

Tokenization – Challenges

- special characters
 - single/double quotation marks
 - comma/semicolon
 - question mark/exclamation marks
 - round brackets

- example:

“Yes!” is what I would ‘say’, but I don’t say it - I know better (now).

Tokenization – Challenges

- multi-word expressions
- includes compound nouns
- need to decide whether to recognize multiword tokens or not
- example: "hot dog", "in spite of"

Tokenization – Challenges

- named entities
- example: "New York"

Tokenization – Challenges

- abbreviations
- example: "Builders Inc."

Tokenization – Challenges

- integrate morphological analysis to split certain contracted forms?
- usage of lexicon in tokenization?
 - Can help with things like “U.S.A.”, “hot dog”, and so on

Sentence Segmentation

- decide sentence boundaries
- in case of punctuation characters, decide whether they are
 - sentence-internal (e.g., Mr.)
 - sentence-final (e.g., The cat woke up.)
- can be tackled as classification task for punctuation
- many tools perform it in conjunction with tokenization

Further Challenges

- different meaning of characters in different language and writing systems
 - e.g. agglutinative languages vs. ideographic writing systems
- lack of punctuation or spaces in certain writing systems, changes in semantic interpretation depending on tokenization
- under-resourced languages: difficult to find corpora to train models

Getting Started with NLP - The OpenNLP Library

Basic properties of Apache **OpenNLP**:

- library with NLP tools for segmentation, tagging, named entity recognition, chunking, parsing, ...
- **machine learning** algorithms, some rule-based methods for comparison
- includes tools and pre-trained statistical language models
- accessible both as an **API (Application Programming Interface)** and as an executable program (**CLI or Command Line Interface**)

Downloads

- official website: <https://opennlp.apache.org>
- OpenNLP Package by Chen Xiaobin:
<https://github.com/chxiaobin/opennlp/archive/master.zip>
- manuals at <https://opennlp.apache.org/docs/>

OpenNLP: Sentence Segmentation

Deciding whether a punctuation character marks the end of a sentence.

- \$ bin/opennlp SentenceDetector models/en-sent.bin < input/text1.txt

Tokenization

Tokens: words, punctuation, numbers...

- tokenizer with statistical model
- \$ bin/opennlp TokenizerME models/en-token.bin < input/text1.txt

Rule-Based vs. Statistical Models

Exercise

The OpenNLP library also includes a simple tokenizer which does not require a model for tokenization. Try to run it with the same input file and see if it gives you the result you want.

```
$ bin/opennlp SimpleTokenizer < input/text1.txt
```

References and Acknowledgments

These slides are largely based on Dickinson et al. [2012] and largely based on slides for Text Technology, 2023 by Stephen Bodnar and on Detmar Meurers' slides on Second Language Acquisition.

Markus Dickinson, Chris Brew, and Detmar Meurers. *Language and computers*. John Wiley & Sons, 2012.

Mayra Z Rodriguez, Cesar H Comin, Dalcimar Casanova, Odemir M Bruno, Diego R Amancio, Luciano da F Costa, and Francisco A Rodrigues. Clustering algorithms: A comparative approach. *PLoS one*, 14(1):e0210236, 2019.

Introduction to Computational Linguistics

Session 6: Automatic Speech Recognition

Stephen Bodnar

Universität Tübingen

December 6, 2023

Introduction

- So far, we've mainly covered text technology
- Computational Linguistics can also involve speech processing
- Research at intersection of speech and technology is huge field
 - Professional associations
E.g. ISCA, (<https://www.isca-speech.org/iscaweb/>)
 - Large international conferences,
E.g., Interspeech Conference (<https://www.interspeech2023.org/>)
 - Academic Journals
E.g., Speech Communication
(<https://www.sciencedirect.com/journal/speech-communication>)
- Speech-interactive interfaces in many places
 - Digital Assistants
 - Document dictation
 - Voice search
 - Spoken dialogs
 - Language training

Plan

- A simplified linguistic description of speech
- Representing speech with computers
- Different speech processing applications
- An overview of Automatic Speech Recognition

Content for today adapted from

- [Jurafsky and Martin, 2009]
- [Lecorv , 2023]
- [Bell, 2023]

A brief introduction to speech

- Area that studies speech sounds called phonetics:

study of linguistic sounds, how they are produced by the articulators of the human vocal tract, how they are realized acoustically, and how this acoustic realization can be digitized and processed. (Jurafsky and Martin, 2009, p. 215)

Units of Speech - Phones

Can think of speech as consisting of different units, at different levels:

- One intuitive, everyday unit is the word (e.g., cat, operationalisation)
- Words composed of smaller, common units of speech called *phones* (also known as segments)
- Phones can be represented with written symbols, e.g. International Phonetic Alphabet.
 - Mapping between symbols and sounds transparent (compare with English, which is much more opaque)
 - E.g. /kæt/
 - E.g. /ɒdə'raɪvəlju(ə)zɪzeɪʃ(ə)n/

Units in Speech - Prosody

Can also describe speech at a level above phones:

- syllable stress (e.g., **present** vs. **present**)
- intonation (e.g., questions vs. statements)

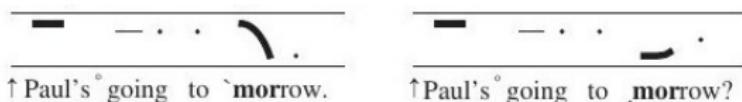


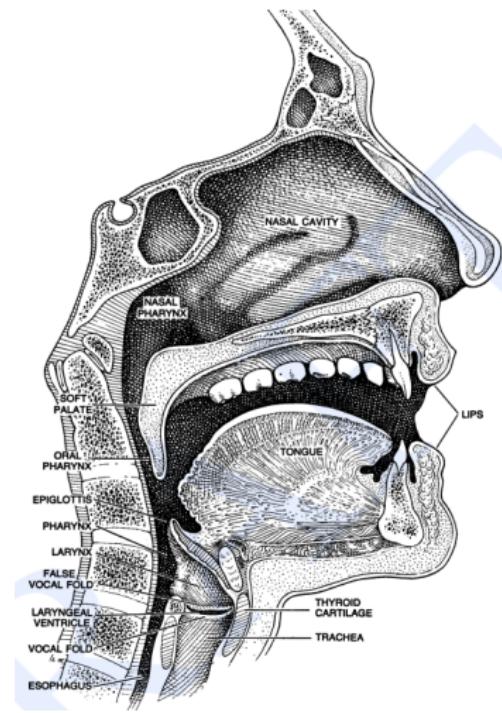
Image source: Collins, Beverley; Mees, Inger M. 2003. The Phonetics of English and Dutch. 5th ed. Leiden: Brill.,

Downloaded from <https://forum.language-learners.org/viewtopic.php?t=14685>

- voice quality (e.g., creakiness [Link])
- Known as prosodic or suprasegmental aspects of speech
- For ASR, segmental aspects of speech traditionally most important, so we focus on phones here

Production of phones

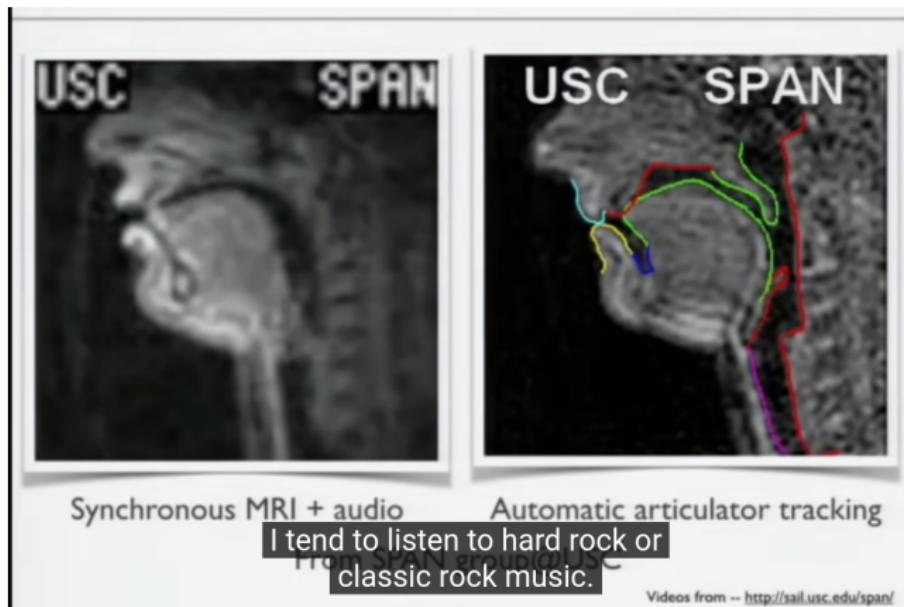
- Speech produced with vocal tract (Image from Jurafsky & Martin, 2009, p. 219)



Production of phones

- The different components of the tract allow you to make the different sounds we know as vowels and consonants
- Different vocal tract configurations produce different phones
- Special terminology. Some examples:
 - Voiced vs unvoiced sounds: With or without vocal chord vibration (e.g., [s] vs. [z], [d] vs. [t])
 - Stop: consonant where airflow blocked for short time (e.g., [b], [k])
 - Fricative: airflow open but restricted (e.g., [f], [z])

Production of phones - Example



Live MRI of vocal tract in action (Microsoft Research) [Link]

Phone variation and Phonemes

- Lots of variation between phones in how phones are realised, due to in-word position, style of speech (reading vs. spontaneous conversation), etc.
- Phonemes: the abstract class representing categories of meaningful sounds in a language
 - E.g., /t/

Phone variation - Allophones

- Allophone: all the different surface realisations of a phoneme, in different contexts (Table from Jurafsky & Martin, 2009, p.226)

IPA	ARPABet	Description	Environment	Example
t ^h	[t]	aspirated	in initial position	<i>toucan</i>
t		unaspirated	after [s] or in reduced syllables	<i>starfish</i>
? ^t	[q]	glottal stop	word-finally or after vowel before [n]	<i>kitten</i>
? ^t	[qt]	glottal stop t	sometimes word-finally	<i>cat</i>
t ^r	[dx]	tap	between vowels	<i>butter</i>
t ^r	[tcl]	unreleased t	before consonants or word-finally	<i>fruitcake</i>
t ⁿ		dental t	before dental consonants ([θ])	<i>eighth</i>
t ^ŋ		deleted t	sometimes word-finally	<i>past</i>

Figure 7.9 Some allophones of /t/ in General American English.

Summary: Phones are the messy, real-world realisations in speech, and phonemes are the abstract labels we use to categorize sounds together.

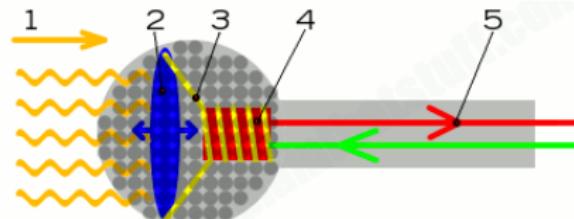
Plan

- A simplified linguistic description of speech
- => Representing speech with computers
- Different speech processing applications
- An overview of Automatic Speech Recognition

Speech and computers

Speech as a signal

- Speech as a sound wave propagating through medium
- Speech as an analog signal, coming through microphone
- Sound waves can be transduced: conversion of a sound into an electric signal by a microphone, ear



www.explainthatstuff.com

Digitizing Speech

- Sampling: periodically measure the electrical current when sound is transduced with a microphone (e.g., 44100 times / second); Store electric signal samples as sequence bits and bytes
- Quantization: representing real valued numbers as integers (like a digital camera, potentially but not necessarily lossy)

Encoding Sound

- Early computers had 8bit sounds, with 2^8 , or 256 amplitude steps for entire range



- In contrast, cd audio uses 16bit numbers, $2^{16} = 65536$ steps



Computer plays back speech by processing the sequence and using electric current to move the speaker appropriately

Plan

- A simplified linguistic description of speech
- Representing speech with computers
- => Different speech processing applications
- An overview of Automatic Speech Recognition

Speech Processing Applications

Instead of playing the audio, we can feed the audio into other kinds of computer programs to do interesting things:

- Automatic Transcription (we'll look at this today)
- Spoken Dialog Systems
- Speaker Recognition
- Diarization
- Emotion Recognition in Speech
- Pronunciation / spoken intelligibility scoring
- Language Identification

Spoken Dialogue Systems

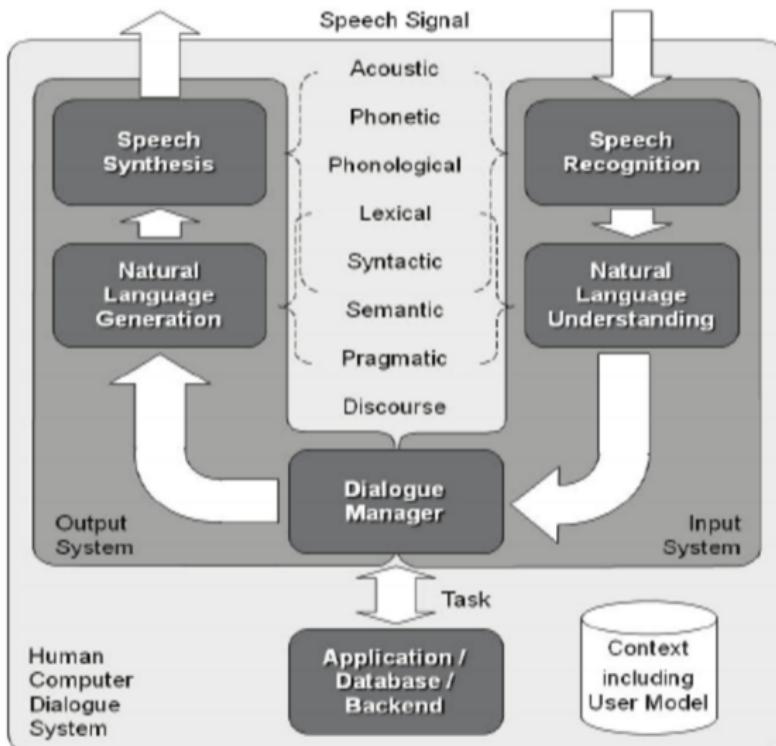


Diagram from [Le Bigot et al., 2008]

Speaker Recognition

Speaker Identification:

- Questions like:
 - Which of the N speakers I know about produced this utterance?
 - Was this recording produced by Speaker X or not?
- Can use for Speaker Verification (voice as password)
- Forensic Applications:
 - “process of determining if a specific individual (suspected speaker) is the source of a questioned voice recording (trace). ” [Drygajlo, 2009]
 - Tools: e.g. <https://www.atis-systems.com/en/automatic-forensic-speaker-identification/>
 - Textbook: <https://www.routledge.com/Forensic-Speaker-Identification/Rose/p/book/9780415271820>
- Also used in Speaker Diarization ...

Speaker Diarization

- “determining ‘who spoke when’ in a long multi-speaker audio recording, marking the start and end of each speaker’s turns in the interaction.” (Jurafsky & Martin, 2023, p. 23)

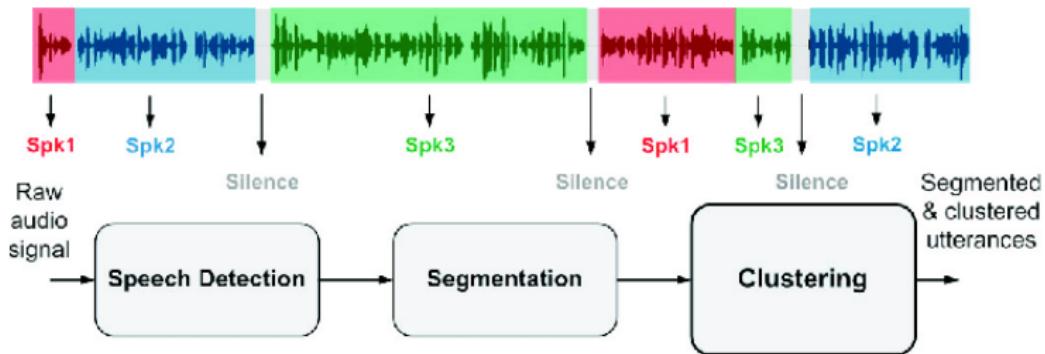


Image from <https://ieeexplore.ieee.org/document/5989833>

Emotion Recognition in Speech

- “automatically identifying the emotional or physical state of a human being from his or her voice.” [Ververidis and Kotropoulos, 2006]
- In addition to phones, also needs prosodic information (e.g., pitch, etc., see article)

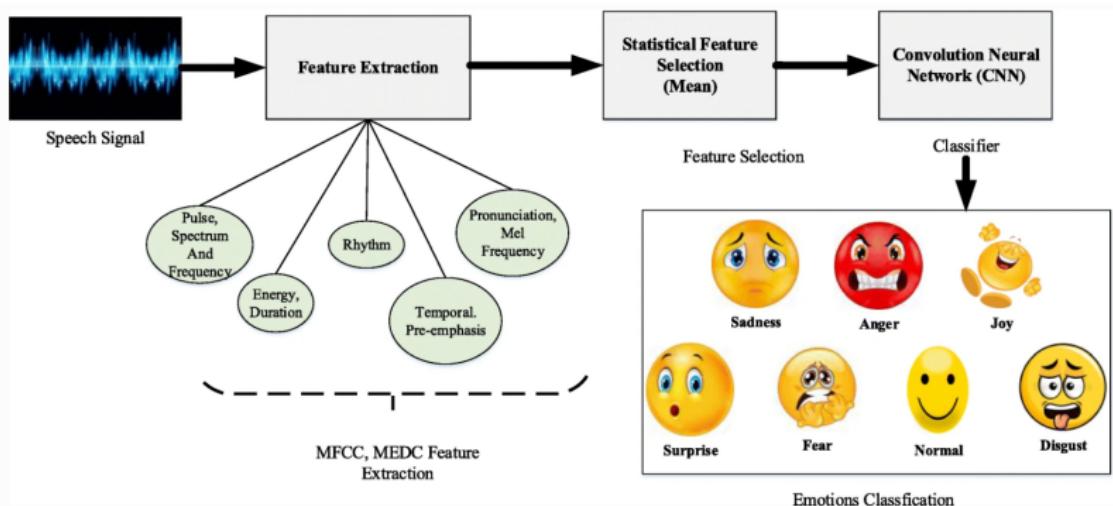


Image from <https://link.springer.com/article/10.1007/s11042-020-10329-2>

Pronunciation / Spoken Intelligibility Scoring

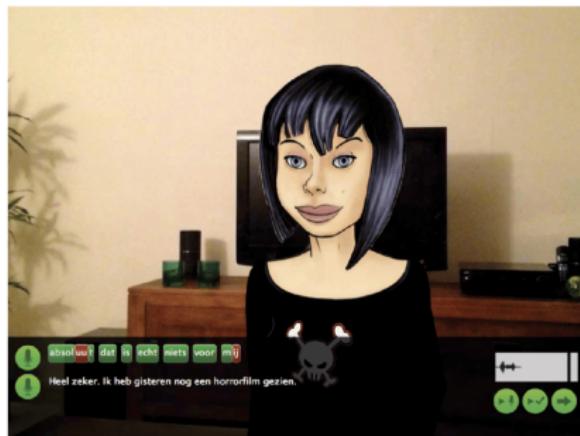


Image from [O'Brien et al., 2018]

Plan

- A simplified linguistic description of speech
- Representing speech with computers
- Different speech processing applications
- => An overview of Automatic Speech Recognition

A look at Automatic Speech Recognition

- Input is speech signal
- Output is string of words user said
- Need to map sounds to phoneme sequences
- From ambiguous phoneme sequences, need to choose best word sequence
- Types of ASR
 - continuous vs discrete word
 - large vocabulary / small vocabulary

Challenges in ASR (why it's a hard problem)

- Ambiguity,
E.g. “recognize speech” vs. “wreck a nice beach”
- Variety in speech
 - accent
 - gender
 - age
 - L2
- Types of speech
 - E.g., read speech, spontaneous speech, emotional speech
- Environmental factors
 - background noise
 - microphone quality
- Single speaker or multiple speakers (e.g., meeting),
possibly overlapping turns

Approaches in ASR

Many changes, especially in last few years

- Dynamic Time Warping
- Statistical approaches (classical approach)
- Deep Learning approaches
- End-to-end models

We will look at 'classical' statistical speech recognition today

Statistical Approach to ASR

Noisy Channel Model

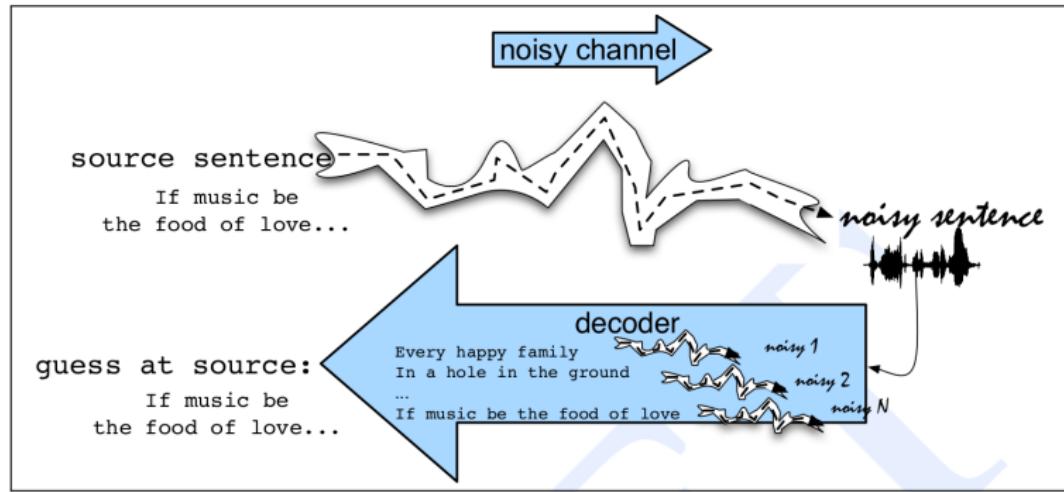
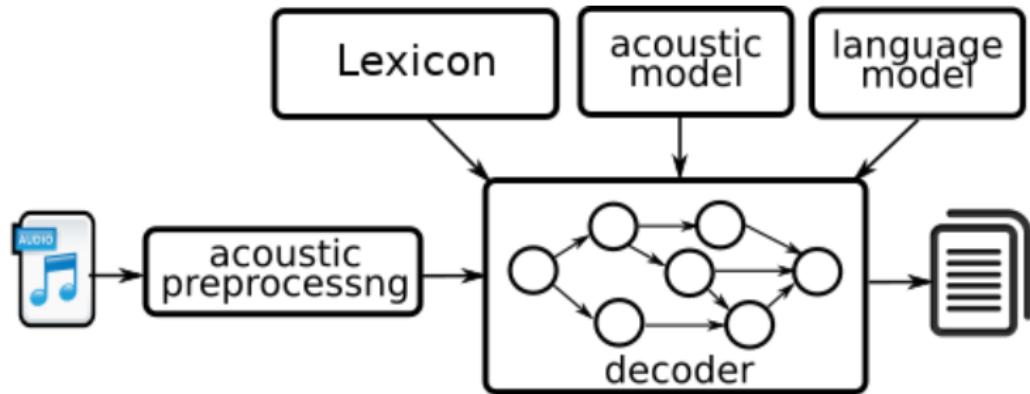
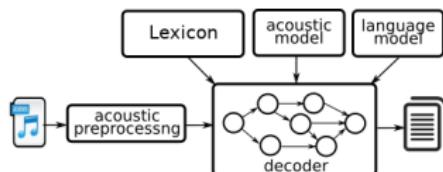


Image from (Jurafsky Martin, 2009, p. 290)

ASR System Architecture

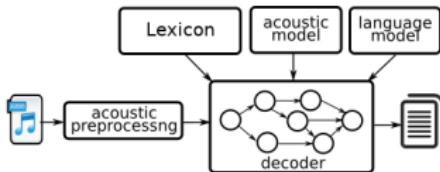


ASR System architecture - Models of linguistic knowledge



- Acoustic models: Given audio, which phonemes are likely?
- Lexicon: maps orthographic words (cow) to phoneme sequences (K AW) E.g. CMUdict [Link]
- Language model: Models the probabilities of different sequences of words
 - Helps to rank the different acoustic hypotheses

ASR System architecture - Other components



- Acoustic preprocessing: transform audio recording into sequence of feature vectors
- Lattice: a directed acyclic graph, used to represent the search space
- Decoder: the part of the program that considers the input and explores different pathways through the search space

Steps in ASR

Statistical ASR uses a supervised machine learning approach:

- ① Building a data set
- ② Feature extraction
- ③ Training
- ④ Testing
- ⑤ Evaluation

Building a data set

- Gather speech representative of audio in your application
- Annotate speech with transcriptions:
 - Word-level
 - Phone-level (expensive!)
- Ideally time-aligned

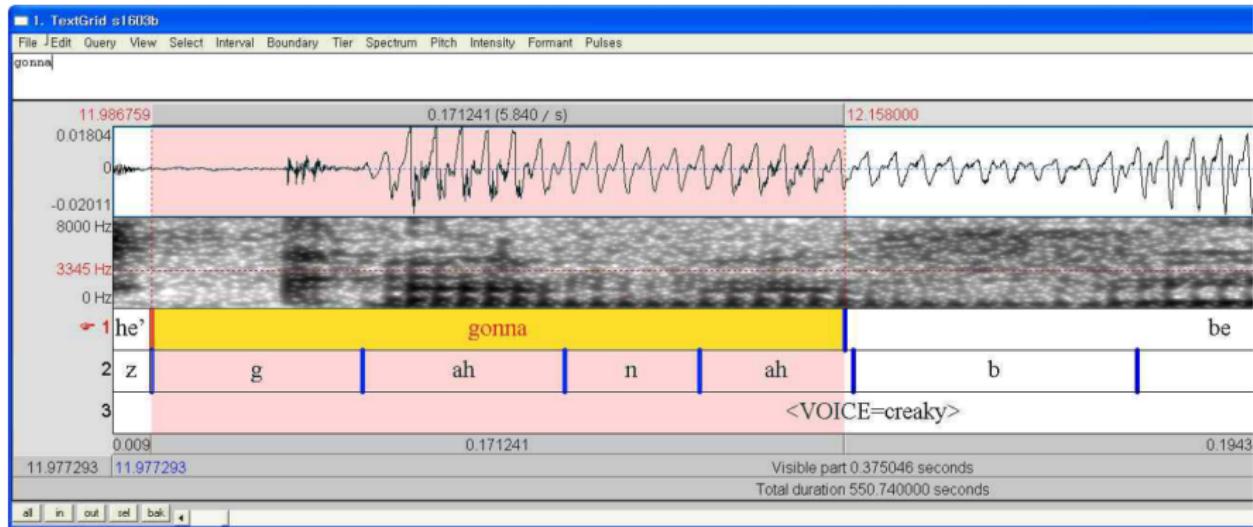


Image from <https://koreascience.kr/article/JAKO201215734995939.pdf>

Feature extraction

- Convert byte sequences representing audio into feature vectors that can be used for machine learning
- Most common representation are so-called MFCCs (mel frequency cepstral coefficients)
- Place window over small amount of speech and calculate statistics; Slide window forward a little and repeat.
- In the end, we get a sequence of feature vectors that describe the speech signal over time

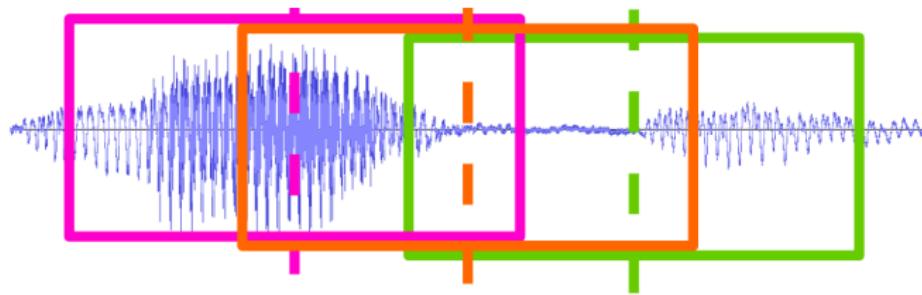


Image from Slides on ASR by Gwénolé Lecorv 

Feature extraction - Process

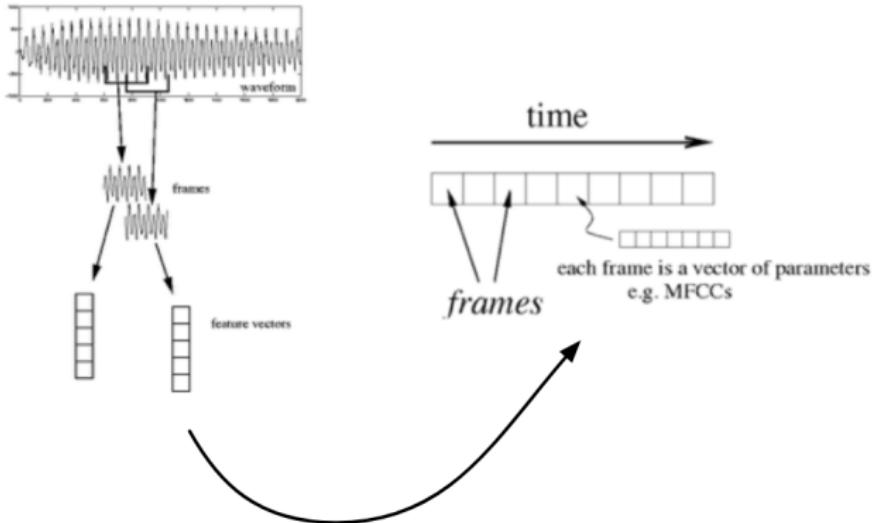


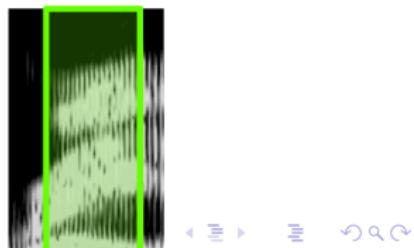
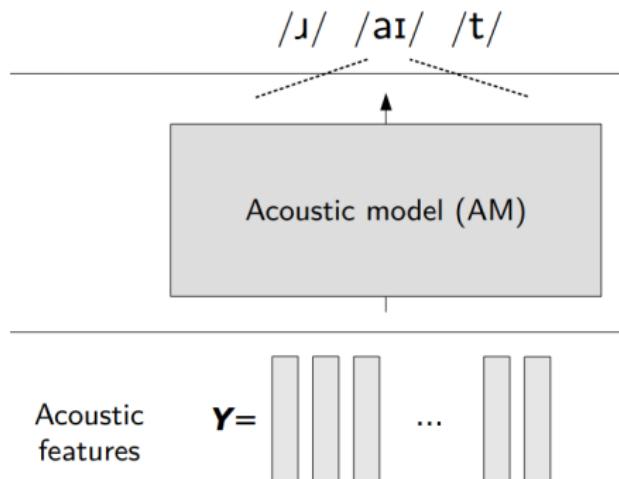
Image from Slides on ASR by Peter Bell

Training

- Stage where we build the statistical models the Decoder will use to explore and rank different paths through the lattice
- We iterate over entries in the corpus with goal of building two models: acoustic model and language model (n-gram)
 - acoustic model, and
 - language model (n-gram)

Acoustic Model

- Acoustic model is more complicated, involves computing probabilities on real-valued data (not discrete set of tokens)
- Variety of mechanisms possible: GMMs, neural networks, Support Vector Machines, Conditional Random Fields
- For our purposes, can treat acoustic model as black box that accepts a feature vector and outputs a score that reflects likelihood of different phonemes



Language Model

From our lecture on Writer's Aids

- *That factory's speciality is wooden boets; they are nice but they cost a lot.*
- What suggestions should our spell checker offer for *boets*?

Similar problem in speech

- E.g., “Recognize speech” vs. “Wreck a nice beach”
- Phonemes allow both possibilities - How to decide between these candidates ?
- We need a resource that we can ask questions like
“Given this sentence, which of these N words fits best?”

Language Model and Candidate Ranking

- Problem can be solved to some degree with a statistical Language Model
 - Models the likelihood of word sequences
 - Likelihood("That factory's speciality is wooden **boots**") = relatively low
 - Likelihood("That factory's speciality is wooden **boats**") = relatively high
 - Likelihood("Wreck a nice beach") = relatively low (probably)
 - Likelihood("Recognize speech") = relatively high

Probability - An intuitive description

- Function $P(\text{event})$ produces a score intended to reflect how likely an event is
- Ranges between 0 (certain to not occur) and 1.0 (certain to occur)
- Consider coin toss:
 - Heads and Tails have equal chance
 - $P(\text{Heads}) = 0.5; P(\text{Tails}) = 0.5$
- Now consider biased coin toss:
 - How to check for biased coin?
 - Can flip coin a hundred times and record outcomes
 - Might yield heads 75 / 100 flips
 - $P(\text{Heads}) = 0.75; P(\text{Tails}) = 0.25$

Applying Probability in Language Models

That factory's speciality is wooden boots; they are nice but they cost a lot.

- Simple approach: compare $P(\text{boots})$ and $P(\text{boats})$, and choose most likely word
- Estimate probabilities by observing each word in set of documents
- Keep track of
 - Number of words you look at
 - Number of times you see 'boots' and 'boats'
- $P(\text{boats}) = \text{number of occurrences of boats} / \text{number of words}$
- Choose word with highest probability

Including More Context

That factory's speciality is wooden boets; they are nice but they cost a lot.

- Problem arises if your data contains more occurrences of 'boots'
- As humans, we see 'wooden' and intuitively know 'boats' is more likely
- Can calculate $P(\text{wooden boats})$ and $P(\text{wooden boots})$ in same way
- Keep track of
 - Number of word pairs we look at
 - Number of times we see each unique pair of words (e.g., 'wooden boat', but also 'walk to')
- $P(\text{wooden boats}) = \text{number of occurrences of string 'wooden boats'} / \text{number of word pairs}$
- Compare probabilities of pairs of words; choose pair with highest probability

Generalising from Pairs to N-grams

N-grams:

- sequences of observations
- context window size n: bigrams, trigrams, ...
- common applications in ASR, spell checking

“That store sells electric rubber bo... (boots? boats?)”

word n-grams:

- word unigrams: *That, store, sells, ...*
- word bigrams: *That store, store sells, sells electric, electric rubber...*
- word trigrams: *sells electric rubber, electric rubber boots, ...*

N-grams - Much More

- Covered high-level introduction
- Have not covered mathematical definition
- Nice video on YouTube introducing N-grams and some of the mathematics behind them:
<https://www.youtube.com/watch?v=UKAQF8wHxkE>

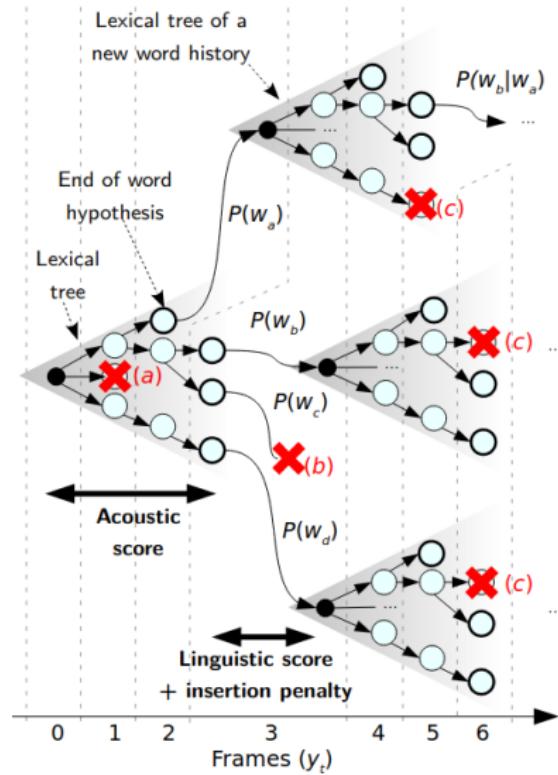
Testing ASR Performance

- Test the performance of the system
- Relies on Decoder component:

"It's the job of a decoder to simultaneously segment the utterance into words and identify each of these words." (Jurafsky & Martin, 2009, p. 318)

- System builds a search lattice, all the different possibilities to consider, derived from combining the lexicon and acoustic models
- Lattice is huge DAG with 10s of millions of states! [Link]
- Decoder searches through the lattice, to find best matching pathways.
- Multiple searches in parallel to maintain a certain top N number of plausible transcriptions

Decoding Approach



Decoding Approach - Steps

Called a Beam search

- Start at feature vector 0
- Consult with all acoustic models; retain acoustic scores
- Continue to process feature vectors until end of word event is detected
- For hypotheses that arrive at an end of word, apply language model score to running acoustic model score
- Remove unlikely hypotheses (outside the beam)
- Move to next frame
- Stop when there are no more frames, and return N-best hypotheses

Very high-level description. In reality, many more details.

Evaluation

- Standard measure is the Word Error Rate (WER)
- Can think of it as percentage of words system got wrong.
- Builds on Minimum Edit Distance (session 3) (in words not characters)
- Compare reference string and string output by ASR (hypothesis)
- Count word insertions, word deletions, word substitutions necessary to transform hypothesis into reference string
- WER is then $100 * (I + D + S) / \text{num_words in corpus}$
- Accuracy is $100 - \text{WER}$

Error analysis

Troubleshoot system

- Can triage errors by looking at which occur most often
- Can use confusion matrices: statistics about which phonemes or words were commonly confused
- Other useful metrics:
 - commonly inserted or deleted words
 - error rates by speaker, gender, L2 etc.

Error analysis - Confusion matrix

Phoneme confusion matrix produced by automatic speech recognition system

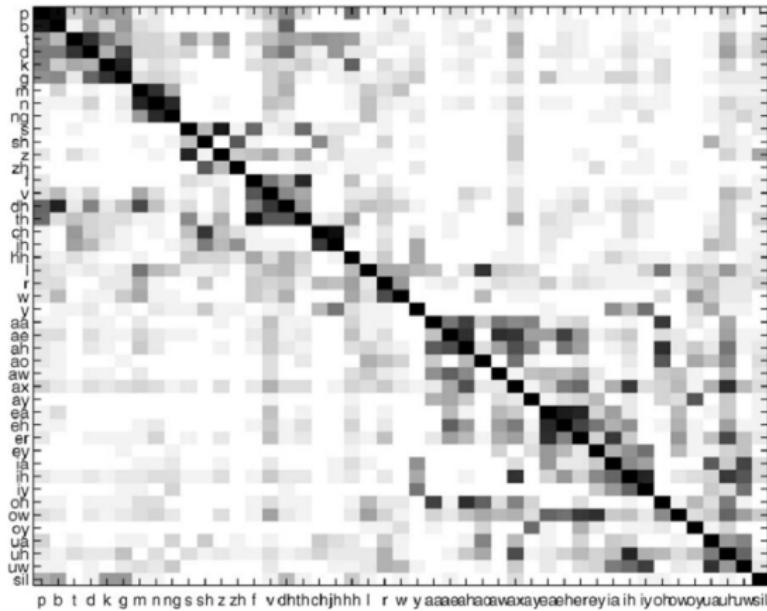


Image from Cox & Vinagre 2009 (<https://www.sciencedirect.com/science/article/abs/pii/S0167639303001213?via%3Dihub>)

Wrapping Up

Today's topics:

- A simplified linguistic description of speech
- Representing speech with computers
- Different speech processing applications
- An overview of Automatic Speech Recognition

Next time: Review session and Q & A.

References and Acknowledgments

Content for today's slides adapted from

- [Jurafsky and Martin, 2009]
- [Lecorv , 2023]
- [Bell, 2023]

Reference List I

- Peter Bell. Automatic speech recognition: Introduction, 2023. URL
<https://www.inf.ed.ac.uk/teaching/courses/asr/2022-23/asr01-intro.pdf>.
- Andrzej Drygajlo. *Voice, Forensic Evidence of*, pages 1388–1396. Springer US, Boston, MA, 2009. ISBN 978-0-387-73003-5. doi: . URL
https://doi.org/10.1007/978-0-387-73003-5_104.
- Dan Jurafsky and James H. Martin. *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition*. Pearson Prentice Hall, Upper Saddle River, N.J., 2009. ISBN 9780131873216 0131873210. URL http://www.amazon.com/Speech-Language-Processing-2nd-Edition/dp/0131873210/ref=pd_bxgy_b_img_y.
- Ludovic Le Bigot, Philippe Bretier, and Patrice Terrier. *Detecting and Exploiting User Familiarity in Natural Language Human-computer Dialogue*. 10 2008. ISBN 978-953-7619-14-5. doi: .

Reference List II

- Gwénolé Lecorvé. Automatic speech recognition, 2023. URL
<http://people.irisa.fr/Gwenole.Lecorve/lectures/ASR.pdf>.
- Mary Grantham O'Brien, Tracey M. Derwing, Catia Cucchiari, Debra M. Hardison, Hansjörg Mixdorff, Ron I. Thomson, Helmer Strik, John M. Levis, Murray J. Munro, Jennifer A. Foote, and Greta Muller Levis. Directions for the future of technology in pronunciation research and teaching. *Journal of Second Language Pronunciation*, 4(2):182–207, 2018. ISSN 2215-1931. doi: . URL <https://www.jbe-platform.com/content/journals/10.1075/jslp.17001.obr>.
- Dimitrios Ververidis and Constantine Kotropoulos. Emotional speech recognition: Resources, features, and methods. *Speech Communication*, 48(9):1162–1181, 2006. ISSN 0167-6393. doi: . URL <https://www.sciencedirect.com/science/article/pii/S0167639306000422>.

Introduction to Computational Linguistics

Session 7: Text Search

Stephen Bodnar

Universität Tübingen

December 13, 2021

This Week's Reading

Reading material: Chapter 6 on Text Search

Plan

- 1 Reading
- 2 Introduction
- 3 Information needs
- 4 Expressing Information Needs through Queries
- 5 Introducing Regular Expressions
- 6 Searching Multiple Documents
- 7 Evaluating Search Quality
- 8 Search Quality Approaches
- 9 Wrapping up

Searching for a Friend

Scenario:

- happen to remember a childhood friend, Katie Smith from Denver, Colorado
- What is she up to now? You type 'Katie Smith' into a search engine.
- Many results!
 - panel about a basketball coach with the same name
 - links to web pages, mostly about the basketball coach
 - other individuals with the same name,
 - nothing about your childhood friend.
- What is the problem here? She is probably on Internet somewhere; so why so hard to find ?

Problems that can arise

Example seems trivial, but is genuine:

- shows a realistic real *information need*
- reveals some of the problems that can arise when we search Internet.

How to move forward with our search?

- Modify *query* and search again (e.g., try 'Kathryn Smith', 'Katie Smith Colorado' etc.)
- Try more specialized social site with search functionality (e.g., such as Instagram, Facebook)
- Old fashioned way: Ask other people (your parents friends with Katie Smith's parents)

Need to know what resources are available, and how to search

Different Kinds of Search Tasks

People use computers for

- **searching**: retrieving information that they want (also known as information retrieval),
- **question answering**: finding answers to questions, and
- **browsing**: for music, movies, friends, web posts, or even things they didn't even know they wanted (recommendation / recommender systems).

Information needs

- information need – the information that the searcher is searching for.
- a type of *intent* – something that a user wants (to do)
- Examples:
 - Information need: “What happened to my old friend Katie Smith”?, or “What is today’s news about China”?
 - Non-textual information, e.g., images, videos, directions on a map
 - Perform an action, e.g., set a timer, book flight, or purchase shoe

Expressing Information Needs through Queries

- Once a user has formulated an intent in their mind, they must translate it into a text *query*
- Query is formulated by user, and may reflect intent with more or less clarity.
E.g. “giant” - can refer to an internet security software company, a film, a bicycle manufacturer, a sports team, and so on [Song et al., 2009]
- Queries require skill (experienced vs. inexperienced search engine user)

Search Engines

- Many search engines (Google Search, Bing, DuckDuckGo) are designed for general use
- Others are publicly available but designed for needs of a specific population (e.g., Google Scholar, used by researchers)
- Still others require a paid subscription, used only by specialists (Westlaw and LexisNexus, used by attorneys; UpToDate, used by medical professionals)
- Informative to consider design of search engine in relation to intended user and information needs

Can look at domain of law and site Westlaw as example

Law example - Information needs

Law professional in U.S. legal system might have one of following information needs:

- Find and cite (current, not overruled) precedent, within my specific legal jurisdiction, for my preferred interpretation of a statute.
- I'm arguing an immigration case before Judge Chang. How sympathetic is Judge Chang likely to be to my case? What has Judge Chang previously ruled or written on the topic of immigration?
- I'm trying to show that the plaintiff in this case was illegally fired as retaliation for blowing a whistle on wrongdoing within a company. What are the elements of illegal whistle-blower retaliation that have to be demonstrated? What is the statutory definition of illegal whistle-blower retaliation? How have previous opinions applied this definition in cases with similar facts?

Database

- Westlaw holds a database of court opinions, briefs filed by the parties in case, laws, legal academic articles, and other material
- Data are semi-structured, combining free text documents and structured meta-data
- Metadata: about the document type (opinion, brief, statute, etc), the judge, law firm, jurisdiction (federal laws, federal judicial districts, or states), date,
- citations to and from each document

Queries

Search functionality of Westlaw is complex, involving specialized syntax:

- **(whistleblow! “whistle blow!”) /p (retaliat! fir!)**
 - Exclamation point ! is a **wildcard**, matching any sequence of letters at the end of a word beginning with *whistleblow*
 - Space character acts as a boolean **OR**; can match *whistleblow!* OR “*whistle blow!*”.
 - First part yields documents containing *whistleblower*, *whistleblowing*, *whistleblowers*, *whistle blower*, and so on
 - /p indicates string match must be in same paragraph as any form of the words *retaliate*, *retaliation*, *retaliated* **OR** *fired*, *firing*, *fire*
 - Quotation marks around “*whistle blow!*” ensure that the space between these two words is not interpreted as disjunction
- Westlaw query syntax is opaque to non-specialists
- Advanced features available, but general-use searcher may find them difficult to use

Regular Expressions

Next: Illustrate previous ideas (and more) with very popular tool for searching text

Regular Expressions

regular expressions (short: **regex**)

- strings from a formal language describing patterns of character sequences
- used for matching, searching, and replacing text
 - testing string formats: phone numbers, emails, address patterns...
 - replacing occurrences of words
- not a programming language themselves, but most modern programming languages support regexes
- are implemented extremely efficiently as finite-state automata

Highly recommended site for practising regular expressions:
<https://regexr.com/>

Learning to Build Regular Expressions

Basic building blocks:

- **Literal Characters:**

- characters which are identical to what they match
- e.g. /car/ matches “car” and “carnival”

- **Metacharacters:**

- special characters which modify literal characters
- transform literal characters into powerful expressions
- \ . * + - { } [] ^ \$ | ? () : ! =

- **Escaping:**

- What if we want to match strings with the character ‘\$’?
- use \ to escape special characters
- E.g., /\\$/ will match the ‘\$’ in ‘It costs \$50’

The wildcard metacharacter

The **wildcard symbol** `.` matches any single character except the new-line character, e.g.

- `/h.t/` matches *hat*, *hot*, and *hit*, but not *heat*
- `/.a.a.a/` matches *banana*, *papaya*, *bababa*

Questions

- What does the regex `5.1` match?
- How do you write a regex that matches the literal number `15.00`?
- Which regex would match all of the following words?
 - silver
 - sliver
 - slider

Pattern Search with grep

The grep tool

(Global search for a Regular Expression and Print matching lines)

- performs line search for patterns in the contents of files or streams
- simple example: `grep "some string" file-to-search.txt`
- most common additional options:
 - `-n` to print matching line numbers
 - `-v` for inverted matching (every line without the pattern)
 - `-c` to count the number of matched lines
 - `-i` for case-insensitive matching
- grep supports different 'flavors' of regular expressions, most powerful:
`grep -P`
- E.g. `grep -P 'walk.*' alice_in_wonderland.txt`

Regular Expression Modes

One-character **flags** are used to express processing modes:

- standard: `/re/` (match and return first)
- global: `/re/g` (match and return all)
- case-insensitive: `/re/i`
- multiline: `/re/m`

Character Set

Square brackets [] define **character sets**:

- more specific than the wildcard symbol
- matches only one character in the set
- Examples:
 - /[aeiou]/ matches one Latin vowel
 - Does /gr[ea]t/ match “great”?

Two Examples

regex: /c[aeiou]t/

string: “The cow, camel, and cat communicated.”

Exercise

Write a regex that matches the following strings:

great graet greet graat

Character Ranges

Some character sets can conveniently be defined using **character ranges**:

- [A-Z] is the same as [ABCDEFGHIJKLMNOPQRSTUVWXYZ]
- [0-9] is the same as [0123456789]
- “-” is not a special character outside of a character set, i.e. it counts as a literal character if not inside square brackets.

Question

- What does the regex / [50-90] / match?

Negated Character Sets

- \wedge is used to **negate** a character set, modifying it to match any characters except the ones inside the original character set
- Examples:
 - `/[^aeiou]/` matches any consonant (non-vowel)
 - `/see[^mn]/` matches “seek” and “sees” but not “seem” or “seen”

Metacharacters Inside Character Sets

Things get slightly tricky if metacharacters are used inside character sets:

- most metacharacters do not need to be escaped: [abc.xyz]
- but escaping is still necessary for the following:] - ^ \
- exception: some unambiguous contexts, e.g. /2017[-/]06[-/]20/
- ⇒ typically, developing a regex requires many iterations

Shorthand Character Sets

Additional escapes serve as **shorthands** for commonly used sets:

- \d for digits (= [0-9])
- \w for word characters (= [a-zA-Z0-9_])
- \s whitespace [\t\r\n]
- \D not digit [^0-9]
- \W not word character [^a-zA-Z0-9_]
- \S not whitespace [^\t\r\n]
- Examples:
 - /\d\d\d\d/ matches “1984”, but not “text”
 - /\w\w\w/ matches “ABC”, “123”, “__A”
 - /\w\s\w\w/ matches “I am”, but not “Am I”

Repetition Metacharacters (Quantifiers)

Quantifiers range over the preceding item and decide how many times it can or must be repeated to be matched:

- * for zero or more repetitions
- + for at least one repetition
- ? for optional items (zero repetitions or one repetition)
- Examples:
 - `/apples*/` matches “apple”, “apples”, and “applesssssss”
 - `/apples+/*` matches “apples”, “applesssss”, but not “apple”
 - `/\d\d\d+/*` matches numbers with three digits or more
 - `/colou?r/*` matches both “color” and “colour”

Quantified Repetition

More general quantification can be achieved by `{min, max}`, where `min` and `max` must be positive integers.

Examples:

- `\d{4,8}` for 4-digit to 8-digit numbers
- `\d{4}` for exactly 4 digits
- `\d{4,}` for at least 4 digits

The Grouping Metacharacters

The **grouping metacharacters** () serve to

- apply repetition operators to a sequence of literal characters
- make expressions easier to read
- define groups for use in matching and replacing

Examples:

- `/(abc)+/` matches “abc” and “abcababcabc”
- `/(in)?dependent/` matches “independent” and “dependent”

Alternation Metacharacter

The **alternation metacharacter** | matches either the previous **or** the next expression.

Examples:

- /apple|orange/ matches “apple” and “orange”
- /apple(juice|sauce)/
- /w(ei|ie)rd/ matches “weird” and “wierd”
- the leftmost expression gets priority, compare:
 - /peanut|peanutbutter/
 - /peanut(butter)?/
- /(AA|BB|CC){6}/ matches “AABBAACCAABB”

Question

- How to write a regex that matches a number between 20 and 40?

Start and End Anchors

Special symbols are used to match the start or the end of a line or string:

- ^ for the start of a line
- \$ for the end of a line

Notes:

- these symbols reference a position, not an actual character
(zero-width symbols)

Searching Multiple Documents

So far focused on searching for strings in a single document

Let's take a look at how searching works on a collection of documents.

Searching Multiple Documents - Example Applications

- Concordancer / Keyword-In-Context (KWIC) search:
<https://www.lextutor.ca/conc/eng/>
- DocFetcher Application:
<https://docfetcher.sourceforge.io/en/index.html>
- Google searches using the “ and ” characters,
e.g., “Error - Could not find
io.micronaut.redis:micronaut-redis-lettuce”

Searching Multiple Documents - Example Applications

How do these applications work?

- Search engine crawls through all the documents to build a *term-by-document matrix*.
- Matrix shows which words appear in which documents
- Book computes term-by-document matrix for a couple nineteenth-century British novels (Emma, Pride and Prejudice, and Wuthering Heights)
- Example uses words, but lemmas or stems also possible
- A 1 matrix means term appears somewhere in document
- A 0 means it appears nowhere

Example of term-by-document matrix

	Emma	P & P	W. Heights
love	1	1	1
house	1	1	1
the	1	1	1
curtsey	1	1	0
whist	1	1	0
dogs	0	0	1
ghost	0	0	1
monster	0	0	1
farmer	1	0	1
computer	0	0	0
phone	0	0	0

Term-by-document matrix as an index

- Terms are rows, documents are columns
- The first row indicates that term *love* appears in all three novels
- To perform searches quickly, we can build term-by-document matrix ahead of time, and use it as an *index*
- To find relevant documents, simply look up term in matrix and return column labels with value 1
- Indices require periodic updates - otherwise they leave out newly added documents

Efficiency Problems with Term-by-Document Matrices and an Alternative Approach

- Term-by-document matrix is inefficient because it is very sparse
- Most values in the matrix are 0 because most words don't occur in most documents,
- Inverted Index approach avoids inefficient zeroes
- Each document is associated with a unique ID. E.g., Emma is 1234, Pride and Prejudice is 5678, Wuthering Heights is 9999
- Each term is associated with a (sorted) list of the document IDs where it appears.

Example of Inverted indices list

love	→Doc1, Doc2, Doc3, Doc7, ...
house	→Doc1, Doc2, Doc3, Doc5, ...
the	→Doc1, Doc2, Doc3, Doc4, ...
curtsey	→Doc1, Doc2, Doc10, Doc15, ...
whist	→Doc1, Doc2, Doc55, Doc80, ...
dogs	→Doc3, Doc6, Doc7, Doc8, ...
ghost	→Doc3, Doc15, Doc18, ...
monster	→Doc3, Doc18, Doc20, ...
farmer	→Doc1, Doc3, Doc11, Doc12, ...
computer	→Doc5, Doc6, Doc8, Doc10, ...
phone	→Doc5, Doc6, Doc10, Doc11, ...

Efficiency of inverted indices list

- More memory efficient because only need to store the IDs of the matching documents (avoid storing 0s)
- Faster processing, because single lookup produces complete list of all relevant documents

Evaluating Search Quality

A search ends when the user has succeeded in filling it.

How do we define success here?

Evaluating search results

- Could just ask the user if they are happy, or survey several hundred users in a user-experience study.
- Could use implicit behavioral measures of satisfaction: a) like whether the user clicks on a page and stays there for a while before ending their search session, or b) how many people use the search engine
- Valuable to have an explicit, quantifiable evaluation metric that can be computed automatically without user testing.
- Most widely used automatic evaluation metrics for information retrieval are *precision* and *recall*.

Precision

- Measures the percentage of the documents returned that are relevant
“Were all of the documents returned relevant ones? Or was it a mix of relevant and irrelevant ones?”
- E.g., Search engine returns 400 pages for a query; 200 of which are relevant
- Precision is $200 / 400$, or 0.5.

Recall

- Measures the percentage of the total number of relevant documents that are returned
“Were all of the existing relevant documents returned?”
- E.g., In total there are 1000 relevant pages, and search engine returns 200 relevant pages (and misses 800 of them)
- Recall is $200 / 1000$, or 0.2

Observation

- Computing recall requires knowing how many relevant pages are out there - is this a problem?

Challenges with Computing Recall

- Sometimes evaluations use constrained, human-labeled datasets specially prepared for evaluations.
- Includes a series of queries, documents, and human judgments of whether each document is relevant to each query.
E.g. from the Text REtrieval Conference (TREC) DL 19, D 20, ClueWeb09, etc. (for more info, see [Link to Slides])
- Data sets allow calculation of precision and recall for any search engine that runs over these data.
- Of course, in real life, often impossible to know how many relevant documents are NOT being returned in your search (i.e., how many search engine missed).

Exploring Precision and Recall Further

- What would happen if search engine prioritized only precision, and ignored recall?
 - Could achieve perfect precision by returning only one or two pages that are relevant!
- What would happen if search engine prioritized only recall, and ignored precision?
 - Could achieve perfect recall by returning every single page on the web!
- => Search engine performance metric must balance precision AND recall.

F-measure

- So-called **F-measure** combines both components
- Defined as $F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$
- From previous examples
 - Precision was 0.5 (200 of the 400 returned documents were relevant)
 - Recall was 0.2 (200 of 100 total relevant documents were returned)
- F-score is given by $F_1 = 2 \times \frac{0.5 \times 0.2}{0.5 + 0.2}$
- Result: $0.28 \Rightarrow$ far from the optimal score of 1.0

Limitations of F-measure

- Search engine with the best F-measure is not necessarily the best search engine
- Users do not want to look through all 200 of the relevant documents returned by our (imaginary) search - Too much!
- Typical behaviour is to glance at the first page of 10 - 15 results!

Including Ranking Quality

Different options for including ranking quality in evaluation

- Could define relevance not as binary notion, but as gradient, and give higher rankings to more relevant documents
- Could give higher relevance to pages that come from better sources; more traffic => more relevant
- However we rank, users care much more about the top page of results than the rest
- On the top results page, precision is much more important than recall
- See *precision-at-k* metric , discussed in Chapter 6
- See also Tao Yang's slides from University of California Computing Science department on Search Evaluation [Link]

Wrapping up

- Search engines play a crucial role in helping all of us (general users, lawyers, scientists) find information we need
- But challenges exist:
 - Companies / individuals can exploit weaknesses of search engine optimization techniques to rise in the rankings
 - E.g. Buying "likes" on Instagram to improve rankings
 - Books on "hacks" to rise to the top of Google search results [Link]

Impact and Challenge of Search Engine Algorithms

- People's lives are increasingly online
=> algorithmic rankings increasingly shape our beliefs and experiences, with consequences
- High ranking can make post appear legitimate, but might not be justified
- YouTube had to re-work its ranking algorithm after a top-ranked result for a search about a mass shooting promoted an unfounded theories
- Facebook and limiting the spread of “fake news”
- Recommender systems and personalised search results telling us only what we want to hear / see / believe (see term “echo chamber”)
- Important but challenging to get search algorithms “right”; need to monitor and respond to how people use them

Reference List I

Ruihua Song, Zhenxiao Luo, Jian-Yun Nie, Yong Yu, and Hsiao-Wuen Hon. Identification of ambiguous queries in web search. *Information Processing & Management*, 45(2):216–229, 2009. ISSN 0306-4573. doi: <https://doi.org/10.1016/j.ipm.2008.09.005>. URL <https://www.sciencedirect.com/science/article/pii/S0306457308000903>.

Introduction to Computational Linguistics

Session 8: Computer-Assisted Language Learning

Stephen Bodnar

Universität Tübingen

December 20, 2023

Plan

- Introduction
- NLP for Analyzing Language for Learners
- NLP for Analyzing Learner Language

Introduction

- NLP tools typically developed and optimized for language of native speakers
- allowing for variability (e.g. spelling correction), but target always well-formed language by native speaker
- NLP for language learning requiring different and creative approaches

Introduction

2 applications of NLP in language learning and instruction

- ① analyzing (native) language productions for learners
- ② analyzing learner productions

NLP for Analyzing Language for Learners

- learners need input in target language
 - reading material, listening texts, authentic material, ...
 - exercises for practicing
- tasks with NLP:
 - searching relevant and appropriate texts and examples
 - enhancing texts

Why Analyze Language for Learners with NLP?

- off-the-shelf NLP tools work for native language
- text books: static content
- NLP for dynamically annotating new texts
- allowing learners to practice on material
 - they are interested in (e.g. news)
 - that is at their current level of proficiency (readability, vocabulary, constructions)
 - that contains the constructions learners want to practice (e.g. comparative forms) → exercise generation

NLP for LL - Example

- FLAIR system Chinkina and Meurers [2016]
- linguistically aware search engine
- re-rank search engine results based on linguistic constructions

<http://sifnos.sfs.uni-tuebingen.de/FLAIR/>

FLAIR - Linguistically Aware Search Engine

The screenshot shows the FLAIR search interface with the query 'cars'. The results are listed in a grid:

Rank	Title	Description
1	The Cars - Wikipedia	The Cars were an American rock band formed in Boston in 1976. Emerging from the new wave scene in the late 1970s, the lineup consisted of Elliot Easton (lead guitar), Greg Hawkes (bass), Ric Ocasek (rhythm guitar), Benjamin Orr (drums), and David Robinson (keyboards). Orr split lead vocal duties, and Ocasek served as the band's principal songwriter. The Cars were at the forefront of ...
2	Cars (song) - Wikipedia	"Cars" is a song by English musician Gary Numan. His debut single, it was released as the lead single from his debut studio album, <i>The Pleasure Principle</i> (1979), on 21 August 1979. It reached the top of the charts in several countries, and today is considered a new wave staple. The song was the first release credited solely to Gary Numan after he dropped the band name Tubeway Army. Under ...
3	Car Games - Play Car Games on CrazyGames	Simulator games – drive cars and other vehicles in more realistic settings, sometimes completing tasks. There's even unique car games that combine elements like idle and merge gameplay into the core mechanics of the game. Road Crash is a car game that does this, putting you in races where you must crash against your opponents to get ahead.
4	Cars & Trucks for sale eBay	But why put yourself through that stress when you can shop cars for sale from the comfort of your own home? eBay has an extensive selection of new, used and certified pre-owned cars and trucks, from Audi and Subaru to Mercedes-Benz. You can also easily sell your used car on eBay, helping raise the cash you need for your next car or truck.
5	Used Cars for Sale Online Near Me Cars.com	Browse cars for sale, shop the best deals near you, find current loan rates and read FAQs about financing and warranties at Cars.com.
6	Cars For Sale - Used Cars For Sale - Used Cars ...	About CarsForSale.com®: For over 20 years, CarsForSale.com® has simply been one of the fastest and easiest ways to buy or sell new and used cars online. With millions of vehicles listed from thousands of State-Verified Auto Dealers nationwide, finding your perfect vehicle is a snap.
7	Car Games - Y8.COM	

On the right side, there is a detailed view of the top result for 'Cars (song) - Wikipedia'. It includes the full text of the Wikipedia page, which describes the song 'Cars' by Gary Numan. The text highlights its success in the charts and its status as a new wave staple. Below this, there is a summary of the construction analysis for the sentence 'Cars' is a song by English musician Gary Numan. His debut single, it was released as the lead single from his debut studio album, *The Pleasure Principle* (1979), on 21 August 1979. It reached the top of the charts in several countries, and today is considered a new wave staple. The song was the first release credited solely to Gary Numan after he dropped the band name Tubeway Army. Under ...'. The analysis shows that the verb 'was' is a Verb Form, specifically a Contracted auxiliary, with a count of 6 and a weight of 3.

Plan

- Introduction
- NLP for Analyzing Language for Learners
- => NLP for Analyzing Learner Language

Native-Speaker Language vs. Learner Language

- Díaz Negrillo et al. [2010]: analyzing learner language
- learner language:
 - divergences between linguistic levels
 - categories for native language not applicable
 - difficult to determine *target hypotheses*
 - often more than one error

A Note about Terminology

- The term **distribution** has a special meaning in this context and refers to grammatical “slots” in the structure of a sentence (See Díaz Negrillo et al. [2010])
- For a given distributional slot, only certain word classes are allowed
- E.g., _____ walked to the store.
- E.g., Those shoes ? Aren't they too _____ ?

Learner Language - Examples of Errors

- “they are very kind and friendship”
- distribution mismatch
(noun occurring where adjectives normally occur)

Examples (from Díaz Negrillo et al. [2010])

Learner Language - Examples of Errors 2

- “to be choiced for a job”
- distribution-stem and distribution-morphology mismatch
(a noun appears with verbal inflection, and in a slot reserved for verbs)

Examples (from Díaz Negrillo et al. [2010])

Learner Language - Examples of Errors 3

Examples (from Díaz Negrillo et al. [2010])

- “television, radio are very subjectives”
- stem-morphology mismatch
(word occurring in adjective slot, but is inflected either as plural noun or present tense verb)

Examples (from Díaz Negrillo et al. [2010])

Learner Language - Examples of Errors 4

- “I didn’t went to the playground yesterday.”
- distribution morphology mismatch
(In its present slot, the verb “go” should not appear in the past tense)

Language Instruction & Practical Challenges

- How does one best learn a language?
- Research in Second Language Acquisition and Foreign Language Teaching (FLT) has shown:
 - individualized, immediate feedback is very important for learner production.
- Problem: limited opportunity for individual, immediate feedback
- In the classroom, the teacher is generally the only source of reliable and accurate feedback available to students.
 - no time to focus on individual students
 - difficult to take heterogeneity of students into account
- Outside of class, how can students be supported in a fair way (not relying on parents)?

Intelligent Tutoring Systems

- Intelligent Tutoring Systems (ITS) can help address this issue
 - automated, immediate feedback while learner is working on the task
 - many students can use the system simultaneously at any time
 - Goals:
 - close gap between ITS research, FLT insights, and real-life classroom
 - address real formal education needs using current NLP technology
- We are developing and testing FeedBook
- an interactive online workbook for secondary school English learners

- Starting point: *Camden Town Gymnasium 3 Workbook*
 - approved for 7th grade English classes in German secondary schools
→ existing workbook, already integrated into real-life formal education
- FeedBook system provides a web-based implementation of the traditional print workbook enabling
 - students to complete activities online
 - teachers to give formative and summative feedback
- Purpose of the system:
 - to provide individualized, immediate scaffolding feedback to learners
 - to guide them towards solutions for a number of different activities.
- Automatic feedback on
 - 7th grade grammar topics (form-oriented)
 - reading/listening exercises (meaning-oriented)

Feedback in FeedBook

Approach for detecting errors and generating feedback:

- analyze target answers with rich linguistic analysis
- use linguistic analyses and search for target forms
- apply rules to introduce errors for target forms
- save edited forms with diagnoses
- use flexible matching to align learner answer with target answer(s)

→ Start out from the expected target forms and systematically transform them into well-formed and ill-formed variations of the target [Rudzewitz et al., 2018]

Generating Variants: Examples

What are you doing in front of the TV ? (target answer)



What is you doing in front of the TV (wrong agreement)

What are you do in front of the TV ? (base form)

...

Generating Variants: Examples cont.

What are you doing in front of the TV ?

(target answer)



What did you do in front of the TV ?

(simple past)

What have you done in front of the TV ?

(past perfect)

...



What did you dos in front of the TV ?

(simple past, extra s)

What has you been doing in front of the TV ?

(present perfect progressive, wrong agreement)

...

NLP Tasks in Analyzing Learner Language

task	tool
segmentation	ClearNLP [Choi and Palmer, 2012]
part-of-speech tagging	ClearNLP
dependency parsing	ClearNLP
lemmatization	Morpha [Minnen et al., 2001]
morphological analysis	Sfst [Schmid, 2005]
synonym annotator	WordNet [Leacock and Chodorow, 1998]
chunking	OpenNLP [Baldridge, 2005]
keyWordFromDepExtractor	CoMiC [Meurers et al., 2011a]
word embeddings	ELMO [Peters et al., 2018]

Table: Adapted from Rudzewitz et al. [2018]

FeedBook - Looking Forward

- Extending FeedBook to support individualized instruction
- Same learning goals for everyone, but different learning paths there
- Idea of Scaffolding
 - Accommodate weaker students with easier, but more exercises
 - Challenge stronger students with less
- Requires explicit
 - learner models
 - activity models
 - domain models
 - pedagogy models
- First test in schools completed last spring; Second test completed fall 2022; A third test planned for February 2024.
- more information: Quixal et al. [2021]

References

Slides based on Björn Rudzewitz's slides from 2021/22, which are in turn based on NLP for language learning Meurers [2012] and Dickinson et al. [2013, chapter 3]

Further reading about the FeedBook in Rudzewitz [2021], Ziai et al. [2018], Rudzewitz et al. [2020]

See Reference list below for more information.

Reference List I

Jason Baldridge. The OpenNLP Project. *URL: <http://opennlp.apache.org/index.html>,* (accessed 25 August 2015), 2005.

Maria Chinkina and Detmar Meurers. Linguistically-aware information retrieval: Providing input enrichment for second language learners. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications (BEA)*, pages 188–198, San Diego, CA, 2016. ACL. URL
<http://aclweb.org/anthology/W16-0521.pdf>.

Jinho D Choi and Martha Palmer. Fast and robust part-of-speech tagging using dynamic model selection. In *Proceedings of the 50th Annual Meeting of the ACL*, pages 363–367, 2012.

Reference List II

- Ana Díaz Negrillo, Detmar Meurers, Salvador Valera, and Holger Wunsch.
Towards interlanguage POS annotation for effective learner corpora in
SLA and FLT. *Language Forum*, 36(1–2):139–154, 2010. ISSN
0253-9071. URL
<http://purl.org/dm/papers/diaz-negrillo-et-al-09.html>.
- Markus Dickinson, Chris Brew, and Detmar Meurers. *Language and
Computers*. Wiley-Blackwell, 2013.
- Claudia Leacock and Martin Chodorow. Combining local context and
wordnet sense similarity for word sense identification. In C. Fellbaum,
editor, *WordNet, An Electronic Lexical Database*, pages 265–283. MIT
Press, 1998.
- Detmar Meurers. Natural language processing and language learning. In
Carol A. Chapelle, editor, *Encyclopedia of Applied Linguistics*, pages
4193–4205. Wiley, Oxford, 2012. URL
<http://purl.org/dm/papers/meurers-12.html>.

Reference List III

- Detmar Meurers, Ramon Ziai, Niels Ott, and Stacey Bailey. Integrating parallel analysis modules to evaluate the meaning of answers to reading comprehension questions. *IJCEELL. Special Issue on Automatic Free-text Evaluation*, 21(4):355–369, 2011a. URL <http://purl.org/dm/papers/Meurers.Ziai.ea-11.pdf>.
- Detmar Meurers, Ramon Ziai, Niels Ott, and Janina Kopp. Evaluating answers to reading comprehension questions in context: Results for German and the role of information structure. In *Proceedings of the TextInfer 2011 Workshop on Textual Entailment*, pages 1–9, Edinburgh, 2011b. URL <http://aclweb.org/anthology/W11-2401.pdf>.
- Guido Minnen, John Carroll, and Darren Pearce. Applied morphological processing of English. *Natural Language Engineering*, 7(3):207–233, 2001.

Reference List IV

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

Martí Quixal, Björn Rudzewitz, Elizabeth Bear, and Detmar Meurers. Automatic annotation of curricular language targets to enrich activity models and support both pedagogy and adaptive systems. In *Proceedings of the 10th Workshop on NLP for Computer Assisted Language Learning*, pages 15–27, 2021. URL <https://aclanthology.org/2021.nlp4call-1.2.pdf>.

Reference List V

Björn Rudzewitz, Ramon Ziai, Kordula De Kuthy, Verena Möller, Florian Nuxoll, and Detmar Meurers. Generating feedback for English foreign language exercises. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications (BEA)*, pages 127–136. ACL, 2018. URL <http://aclweb.org/anthology/W18-0513.pdf>.

Björn Rudzewitz, Ramon Ziai, Florian Nuxoll, Kordula De Kuthy, and Detmar Meurers. Enhancing a web-based language tutoring system with learning analytics. In Luc Paquette and Cristobal Romero, editors, *Joint Proceedings of the Workshops of the 12th International Conference on Educational Data Mining co-located with the 12th International Conference on Educational Data Mining (EDM 2019)*, volume 2592, pages 1–7, Montréal, Canada, 2020. CEUR Workshop Proceedings. URL <http://ceur-ws.org/Vol-2592/paper1.pdf>.

Reference List VI

Björn Rudzewitz. *Learning Analytics in Intelligent Computer-Assisted Language Learning*. PhD thesis, Eberhard-Karls Universität Tübingen, 2021. URL <http://hdl.handle.net/10900/117358>.

Helmut Schmid. A programming language for finite state transducers. In *Proceedings of the 5th International Workshop on Finite State Methods in Natural Language Processing*, pages 308–309, 2005.

Ramon Ziai, Björn Rudzewitz, Kordula De Kuthy, Florian Nuxoll, and Detmar Meurers. Feedback strategies for form and meaning in a real-life language tutoring system. In *Proceedings of the 7th Workshop on Natural Language Processing for Computer-Assisted Language Learning (NLP4CALL)*, pages 91–98. ACL, 2018. URL <http://aclweb.org/anthology/W18-7110>.

Introduction to Computational Linguistics

Session 9: Dialogue Systems & Further NLP Methods

Denise Löfflad

Universität Tübingen

January 10, 2024

Exam

You need to register for the exam on Alma! Important deadlines:

- 22.01: Deadline to register for the exam (do it after the lecture)
- 02.02: Deadline to deregister from the exam. If you did not get 60% or don't want to participate, deregister! Otherwise it will count as a try, and you only have 2 tries

Plan

① Dialogues Systems

- ① Dialogues
- ② Task-Specific DS - Digital Assistants
- ③ Task-Independent DS - Chabots

② Further NLP Methods

- ① POS Tagging
- ② Parsing

Dialogues

Different perspectives on dialogues:

- What elements do human dialogues consist of?
- What underlying rules/principles guide people's behaviour in dialogues?
- How can computers make use of these elements and principles?

speech act

- actions performed through language
- examples: greeting, apologising, ordering, requesting something, advising, insulting, ...

common ground

- information mutually shared by participants
 - ➊ context, e.g. the location you are in
 - ➋ world knowledge (facts about the world)
 - ➌ prior knowledge, e.g. earlier part of the conversation

Dialogues

turn taking

- role of the speaker and hearer(s)
- speaker: control over content, goals, formulation, style, ensuring hearer understands
- hearer: interpretation of utterances of speaker, update the common ground, plan answer, interpret conversation flow (e.g. when to switch roles)
- backchannel: “the person who has the turn receives short messages such as 'yes' and 'uh-huh' without relinquishing the turn.” (Yngve, 1970, p.568)
- overlap: people participating in a conversation and “planning to respond are already encoding their responses while the incoming turn from the other speaker is still unfinished” (Levinson and Torreira, 2015)

adjacency pair

- sequence of two turns with expected structure
- content and structure of second turn determined by first turn
- examples:
 - greetings elicit greetings
 - questions elicit answers
 - informational statements followed by acknowledgements
 - confusion elicits clarification
 - ...

Dialogues

Grice's Maxims: norms guiding conversations

- ① **Quality:** say true things
- ② **Quantity:** only as much information as necessary, not more, not less
- ③ **Relevance:** say only things that are relevant to the conversation
- ④ **Manner:** express content in a manner that can be easily understood

Dialogues

Cooperative conversations are important: allows one to make inferences

- S1: "Did you like the movie?"
- S2: "I liked the first half of it."
- By maxims of Quality, Quantity and Relevance, we can interpret S2's response in (at least) two ways...

Dialogues

Cooperative conversations are important: allows one to make inferences (see Textbook, p.286)

- S1: “Did you like the movie?”
- S2: “I liked the first half of it.”
- **Conversational implicature:** “inference that a hearer draws from reasoning about why a speaker said one thing over another” (p.286)
E.g., S2 didn’t like the second half.
- **Entailment:** “an inference that follows logically from a given statement” (p.286). E.g., S2 watched the first half.
- **Presupposition:** “fact that has to be true for a given sentence to even make sense” (p. 286). E.g., S2 is talking about the same movie as S1.

Task-Specific Dialogue Systems: Digital Assistants

- dialogue system designed to help accomplish one or more specific, concrete tasks
- alternative to other ways of accomplishing a task, e.g. employee or graphical user interface
- in contrast to chatbots, designed and built for a *specific domain* and *specific task(s)*
 - good enough processing for a specific domain
 - not necessary to model a lot of world knowledge (terminology, relations between concepts, ...) outside of application domain

Digital Assistants

- **intent:** task the user is supposed to achieve with support by the assistant
 - example: ordering a meal
- intents contain **slots/templates** that need to be filled
 - example: which food/drinks in which quantity
- **intent recognition:** classify user utterance as intent
 - example: ordering a meal, booking a table

Digital Assistants

Examples:

- assistants for phone-centered tasks, e.g. Siri (Apple), Alexa (Amazon)
- assistants in modern cars
- increasing number of digital assistants used for customer 'service' purposes
- ...

Task-Independent Dialogue Systems: Chatbots

- dialogue system designed to chat with a human in natural language (sociable chit-chat)
- conversations might be carried out in written or spoken form
- designed and built task- and domain-independent with the purpose of entertaining or impressing users (e.g. customers) and/or serve as digital companion
- Systems try to:
 - respond appropriately to the human's turn
 - be polite and friendly
 - model a reasonable amount of world, prior and context knowledge (Common Ground)
 - comply with Grice's Maxims

Chatbots: Implementation

- **brute force:** using an enormous amount of hand-written question-answer pairs
- **rule-based:** using fixed rules to derive response utterances given the preceding turn
- **corpus-trained chatbots:** based on a huge set of data the bot derives the answer that is most suitable for a given turn
- **language generation:** based on corpus data machine learning models are built; at production time they take previous turn(s) as input and construct and output a response that has a high probability of being a reasonable reply (the output could be something that does not exist in the corpus)

Most modern systems use a mixture of different approaches.

It is also important to keep ethical aspects in mind, especially when using corpus based methods for implementation.

Chatbots: ELIZA

- implemented by Joseph Weizenbaum in 1964
- mimics the behavior of a psychiatrist, who is asking questions rather than uttering statements
- rule-based chatbot
 - “If the input contains a noun, reply: TELL ME MORE ABOUT [NOUN]”
 - “If the input contains the phrase I AM [X], reply: HOW LONG HAVE YOU BEEN [X]?”

Chatbots: ELIZA

ELIZA: Hello, I am Eliza

USER: Hi

ELIZA: How are you today . . . What would you like to discuss?

USER: I am studying chatbots

ELIZA: Did you come to me because you are studying chatbots?

USER: Yes!

ELIZA: I see.

USER: Can an alligator ride a bicycle?

ELIZA: Can you elaborate on that?

USER: I am trying to see how smart you are

ELIZA: Perhaps you are trying to be?

USER: What animal did I just ask you about?

ELIZA: We were discussing you, not me.

USER: Let me ask about you though!

ELIZA: We were discussing you, not me.

Chatbots: Kuki

- implemented by Steve Worswick, published 2005
- using blend of rules and hand-written responses
- initial implementation has been extended over the years based on questions from users
- different answer options for common questions have been implemented to avoid repetitions
- contains reference to a database that contains facts about world (ConceptNet)
- used by humans to practice their english, thus some rules have been adapted to learner language

Chatbots: Kuki

USER: good who else have you been talking to?

KUKI: I am talking to a bunch of people all over the world

USER: Like who?

KUKI: Joe

USER: What is he saying?

KUKI: Are you asking about Joe?

USER: Yeah, what is he saying?

KUKI: I don't want to talk about that now. Are you asking about Joe?

USER: Yes but you said you don't want to talk about that

KUKI: I don't want to talk about that now. What I said earlier is irrelevant.

USER: Okay what do you want to talk about?

KUKI: I like to talk about movies, robots, and history. I hate one-word answers.

Evaluating Chatbots

- Turing test
- Winograd Schemas: A pronoun is used with two possible antecedents, world knowledge is needed to correctly resolve the pronoun.
In the sentence: “The city council members refused the demonstrators a permit because they advocated violence”, who does “they” refer to?

And finally, a cautionary note

Microsoft's Tay

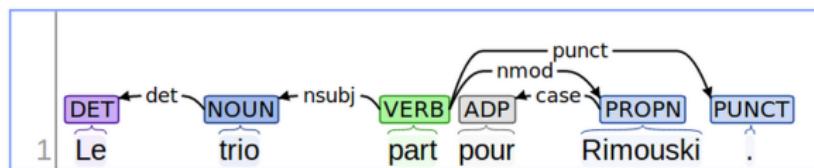
- released in 2016, Tay was programmed to keep itself up to date by learning from conversations
- within 24h hours, it had to be taken down
- didn't include a moral filter
- → be cautious and test well

For next time - Generative AI

"As one of humanity's unique tools, language is often considered key to the success of artificial general intelligence" Dickinson et al. [2012]

Further into NLP

- Beyond segmenting and tokenizing character streams, NLP also investigates methods for annotating strings with linguistic information **automatically**



- Today we introduce two common NLP functionalities that provide linguistic annotations:
 - POS Tagging
 - Parsing

What are Part of Speech (POS) Tags?

- Annotate tokens with information related to word type
- Information takes form of “tags”:
 - Short labels describing a token’s part of speech
 - Examples: Noun, verb, adjective, adverbs, ...
- Different corpora / NLP tools use different tag sets, each with different categories, e.g.
 - Penn Treebank tag set: 45 tags
 - Brown corpus tag set: 87 tags
 - C7 tag set: 146 tags

Why POS Tags?

- Tags encode morphological and syntactic information of tokens
- Allows to draw conclusions about properties of tokens and - to a certain extent - their neighbors
 - E.g., pronunciation can change depending on part-of-speech: **content** vs. **content**
- Pre-requisite for other NLP tasks (e.g., parsing, keyword extraction, language learning exercise generation, etc.)

POS Tags - Different Granularities

- main categories: noun, verb, adverb, ...
- fine-grained categories:
 - Noun singular or mass NN (bird, water, ...) vs. noun plural NNS (birds, geese, ...)
 - adverbs comparative RBR (better, faster) vs. superlative RBS (best, fastest)

POS Tags - Closed and Open Classes

- closed class part of speech tags
 - fixed set of tokens/words
 - often function words
 - shared across corpora
 - example: prepositions (on, since, from, with, ...)
- open class part of speech tags
 - changing (language change, borrowing/coining/inventing new words)
 - depending on domain of corpus
 - example: nouns, verbs

See also <https://universaldependencies.org/u/pos/>

Annotating tokens with POS Tags

- different approaches towards part-of-speech tagging
- typically requires pre-processed text
 - sentence-segmented
 - tokenized
- goal: assign each token single most probable tag

POS Tagging - Ambiguity

- most words unambiguous,
straightforward mapping from word → POS tag
- a small number of common word forms are ambiguous (2 - 7 tags)
- Example:
I can can a can of tuna.
- Brown corpus: 11.5% of types ambiguous, but 40% of tokens
ambiguous

POS Tagging - Ambiguity cont.

- many words have more than one possible part-of-speech tag
- example: book
 - book a flight/book a suspect (verb)
 - I read a book (noun)
- example: that
 - Is that a good idea? (determiner)
 - I like that the movie begins (complementizer)

→ requires context to disambiguate

POS Tagging - Approaches

Approaches:

- rule-based tagging
- statistical tagging

POS Tagging - Rule-based tagging

- Example: Constraint grammar
- Requires lexicon with rich information, e.g.
 - part-of-speech
 - morphological information
 - subcategorization pattern
 - with what types of arguments does this word occur
 - what are required arguments
 - e.g. animated subject plus object for a verb
- Procedure:
 - ① Lookup word in lexicon to retrieve all candidate POS tags, along with their associated rules (constraints)
 - ② Try each rule on input sentence and exclude POS tag if rule does not fit

POS Tagging - Statistical Approaches

- Input: a token sequence (e.g., I fish tuna)
- Task: assign a POS tag to each token (e.g., I/PRP fish/VB tuna/NN)
- Problem of ambiguity: certain tokens are associated with multiple parts of speech (e.g., "fish" could be a noun or a verb)
- Handle ambiguity by exploring different possible sequences
- Approach similar to minimal edit distance search:
 - Generate list of possible sequences
 - Candidate 1: I/PRP fish/NN tuna/NN
 - Candidate 2: I/PRP fish/VB tuna/NN
 - Rank each sequence by its statistical probability

Ranking in Statistical POS Tagging Approaches

- Can calculate a sequence's probability using a supervised machine learning approach
- Humans annotate corpus with POS tags
- Algorithm processes corpus to pre-compute different statistical measures
 - ① For each part of speech tag, store likelihood of each word being observed
 - E.g. $P(\text{fish} \mid \text{NN}) = 0.12$, $P(\text{fish} \mid \text{VB}) = 0.04$
 - ② For each observed POS tag sequence (n-gram), store probability
 - E.g. $P(\text{PRP NN}) = 0.1$, $P(\text{PRP VB}) = 0.7$
- To compute probability of sequence, multiply 1) with 2) for each word
- Choose sequence with highest probability
- Example of Bayesian modelling

A Second Major NLP Task: Parsing

- Today we introduce two common NLP functionalities that provide linguistic annotations:
 - POS Tagging
 - = ι Parsing

Parsing - What is it?

- Annotate natural language with linguistic information to represent grammatical structure or relationships
- Often a step towards higher-level tasks, e.g. semantic analyses
- Different types of parsing, e.g.
 - constituency parsing
 - dependency parsing

Constituency Parsing

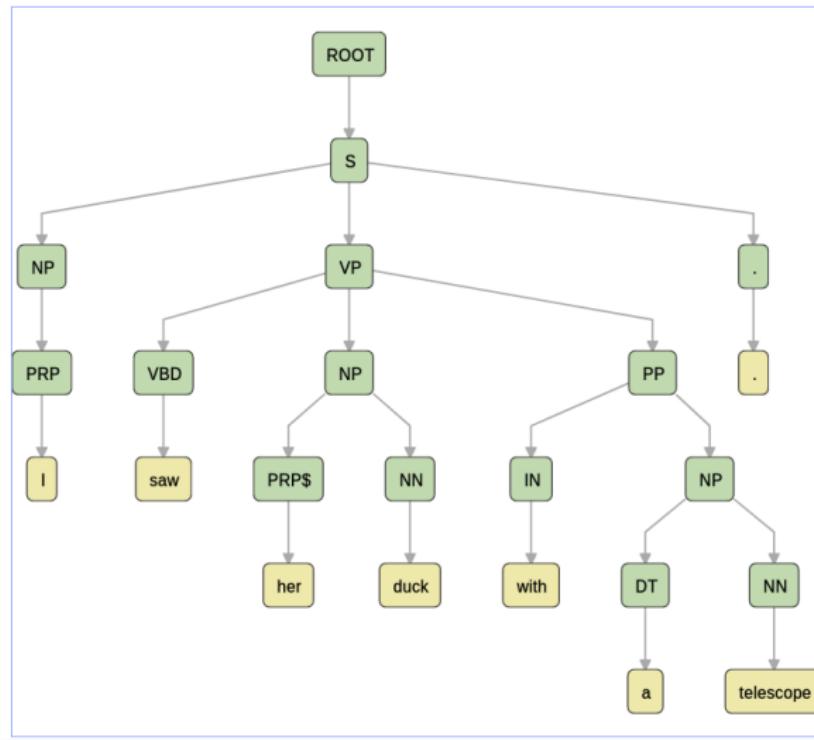
- Constituents: group of words that form syntactic units
 - E.g., Noun phrases (NP): “The/DET big/ADJ dog/NN”
 - E.g., Verb phrases (VP): “walked/VBD quickly/ADVP”
- Assigns syntactic structure
→ Given a sentence, what structure does it have?
- Tasks:
 - identify multi-word syntactic units (e.g. noun phrases)
 - capture and represent potentially nested syntactic units (e.g. noun phrases embedded in verb phrases)
- Two common visual representations:
 - Trees
 - Notation with parentheses

Constituency Parsing - Example

Roughly parse the following sentence: "I saw her duck with the telescope."

Constituency Parsing - Example Parse Tree

Input: *I saw her duck with a telescope.*



Constituency Parsing - Example Notation with Parentheses

Input: *I saw her duck with a telescope.*

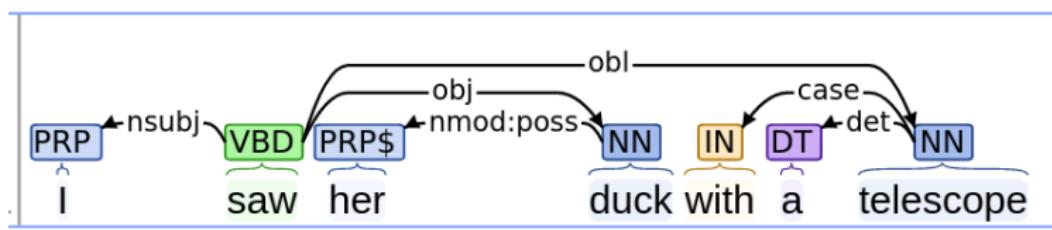
```
(S  
  (NP (PRP I))  
  (VP (VBD saw))  
  (S  
    (NP (PRP her))  
    (VP (VB duck))  
    (PP (IN with))  
    (NP (DT a))  
    (NN telescope))  
  )  
 )  
 )  
 )  
 )  
 )
```

Dependency Parsing

- Aim: capture the syntactic (and semantic) relations between words in a sentence
- Consists exclusively of directed grammatical relations between pairs of words
- Each relation has a governor (or head) and a dependent; arcs point from governor to dependent
- Fixed set of dependency relation types (e.g.
<https://universaldependencies.org/u/dep/index.html>)
- Widely used in NLP processing (easier processing due to less nesting, ability to handle languages with different word orders)

Dependency Parsing - Example

Input: *I saw her duck with a telescope.*



Parsing - Ambiguity as Major Challenge

- Attachment ambiguity
I saw her duck with a telescope
- Coordination ambiguity
old men and women

Parsing - Methods

Classical parsing paradigms:

- top-down parsing
- bottom-up parsing
 - chart parsing

Before going into more detail, need to look at a tool parsers use called
context-free grammar

Parsing and Grammars

- Term *grammar* has special meaning in parsing context
- Can think of it as a set of rules that express valid word sequences

//Simple Grammar Example

SENTENCE → NOUN_PHRASE VERB

NOUN_PHRASE → DET NOUN

DET → the | a

NOUN → dog | cat

VERB → walks | sleeps | eats

- Strings in all-captitals denote classes of words, and are called nonterminals (e.g., NOUN).
- Strings in lowercase denote instances of real words in the language,



Uses for Grammars

- A grammar can be used to check if an utterance is grammatical (i.e., satisfies one of the rules)

//Simple Grammar Example

SENTENCE → NOUN_PHRASE VERB

NOUN_PHRASE → DET NOUN

DET → the | a

NOUN → dog | cat

VERB → walks | sleeps | eats

- The sentence “the dog walks” is grammatical
- The sentence “the dog walk” is ungrammatical
- The sentence “the fish sleeps” is ungrammatical

Top-down Parsing

- Uses a grammar to parse a sentence
- Can be seen as a 'rationalist' approach: start with definition of "good" sentences
- Start with the grammar's start symbol
- Expand the start symbol to right-side rules
- Continue expanding until encountering a terminal
- Search string to see if terminal exists in sentence
 - If yes, register rule match, remove terminal from string, and continue

Top-Down Parsing Example

Example: Parse the sentence *the cat sleeps* using the grammar above

Top-Down Parsing Example

Input: *the cat sleeps*

SENTENCE

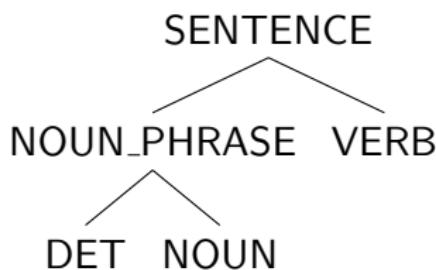
Top-Down Parsing Example cont.

Input: *the cat sleeps*



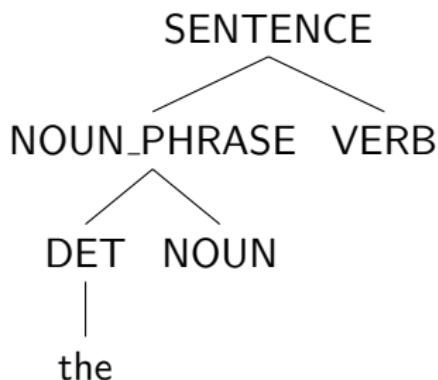
Example: Top-Down Parsing

Input: *the cat sleeps*



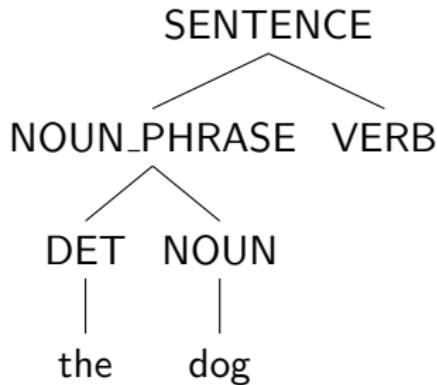
Example: Top-Down Parsing

Input: *the cat sleeps*



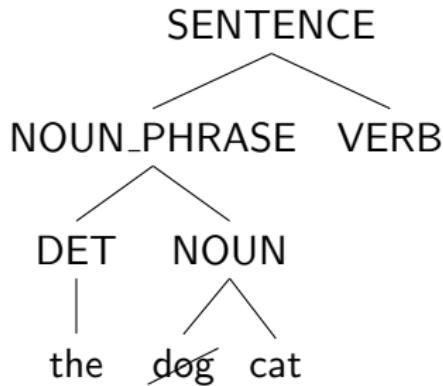
Example: Top-Down Parsing

Input: *the cat sleeps*



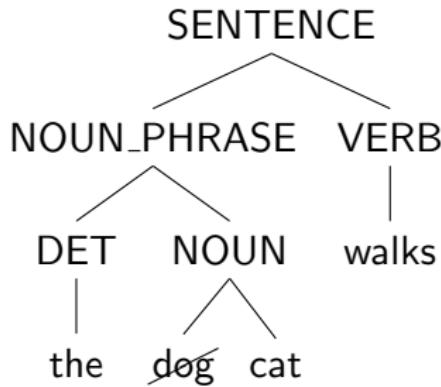
Example: Top-Down Parsing

Input: *the cat sleeps*



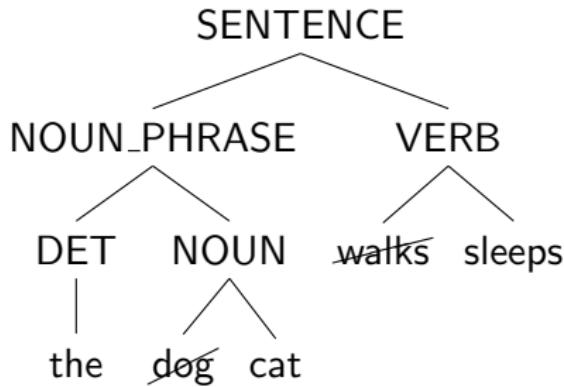
Example: Top-Down Parsing

Input: *the cat sleeps*



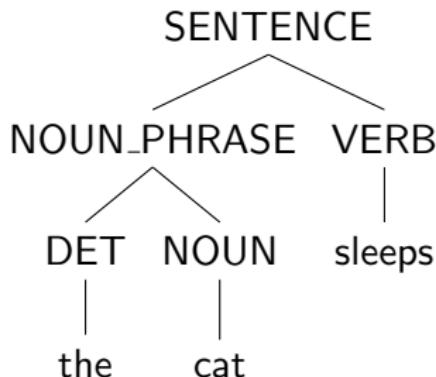
Example: Top-Down Parsing

Input: *the cat sleeps*



Example: Top-Down Parsing

Input: *the cat sleeps*



In parentheses notation:

```
(S  
  (NOUN_PHRASE (DET the)  
    (NOUN cat)  
  )  
  (VERB sleeps)  
)
```

Top-Down Parsing: Method

- continue until all input words have been integrated into the tree
 - final goal: all words matched
 - if not possible: sentence is ungrammatical
- ambiguity:
 - in case of multiple applicable rules, either create a copy of the tree or use data structures that support parse trees

Further into Parsing

We've only scratched the surface

- Besides top-down, other methods for constituency parsing:
 - bottom-up parsing
 - chart parsing
- Also methods for dependency parsing
- Beyond the scope of our intro
- Topic of ISCL-BA-06 (4th semester)

References and Acknowledgments

These slides are based on slides for Intro to CL, 2022 by Dr. Stephen Bodnar, which were based on slides by Dr. Björn Rudzewitz.

Markus Dickinson, Chris Brew, and Detmar Meurers. *Language and computers*. John Wiley & Sons, 2012.

Levinson, S.C. and Torreira, F. (2015) “Timing in turn-taking and its implications for processing models of language,” Frontiers in Psychology, 6. Available at: <https://doi.org/10.3389/fpsyg.2015.00731>.

Yngve, V. H. (1970). On getting a word in edgewise. CLS-70 (p./pp. 567–577), : University of Chicago.

Introduction to Computational Linguistics

Session 10: Large Language Models

Denise Löfflad

Universität Tübingen

January 17, 2023

Plan

- ① Overview & Technical Aspects
- ② Applications of LLMs
- ③ Ethical Aspects of LLMs (and AI)

Overview of LLMs

What are language models?

- Systems that represent natural language
- Systems that generate new data points (language) based on the input using statistical information about language
- Every sequence of tokens has a probability
- LMs use enormous data sets to try and learn to predict the next token in a sequence

GPT - ?

Generative **P**re-trained **T**ransformer

Pre-training

- Before: One model per task
- Training very dependent on task, annotated corpora needed ...
- New idea: One model to solve multiple (all?) tasks
- Problem: How to do that?
- Solution: Go big or go home!

Model Size

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

Table 2.1: Sizes, architectures, and learning hyper-parameters (batch size in tokens and learning rate) of the models which we trained. All models were trained for a total of 300 billion tokens.

Figure: Brown et al. [2020]

Compute Size

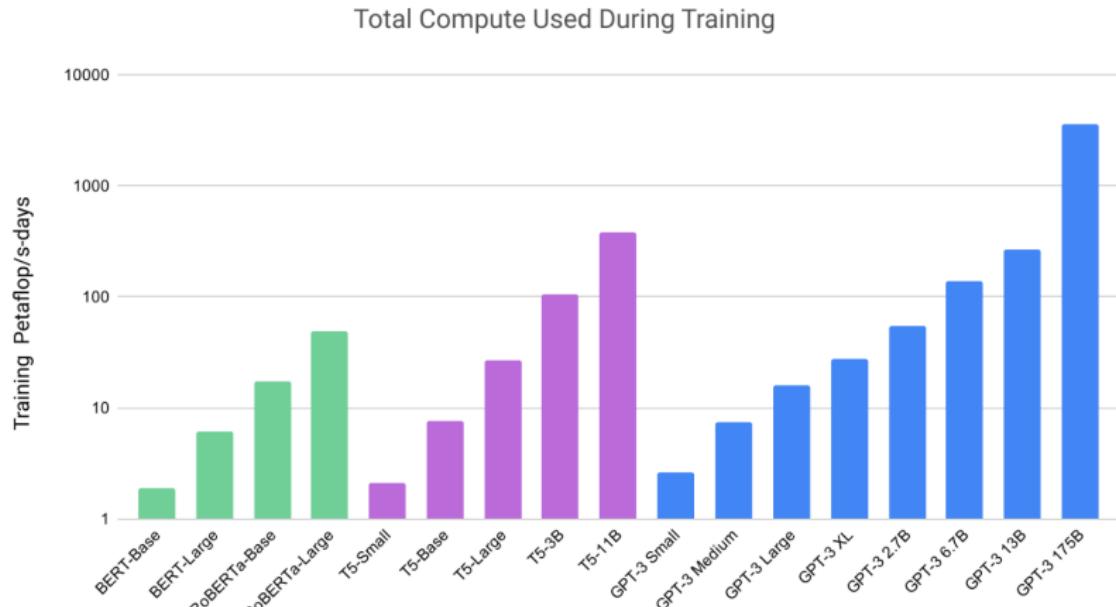


Figure 2.2: Total compute used during training. Based on the analysis in Scaling Laws For Neural Language Models [KMH⁺20] we train much larger models on many fewer tokens than is typical. As a consequence, although GPT-3 3B is almost 10x larger than RoBERTa-Large (355M params), both models took roughly 50 petaflop/s-days of compute during pre-training. Methodology for these calculations can be found in Appendix D.

Figure: Brown et al. [2020]

Data Set Size

Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

Table 2.2: Datasets used to train GPT-3. “Weight in training mix” refers to the fraction of examples during training that are drawn from a given dataset, which we intentionally do not make proportional to the size of the dataset. As a result, when we train for 300 billion tokens, some datasets are seen up to 3.4 times during training while other datasets are seen less than once.

Figure: Brown et al. [2020]

(one GPU has about 5 GB, this dataset has 45 TB. 1 TB = 1000 GB)

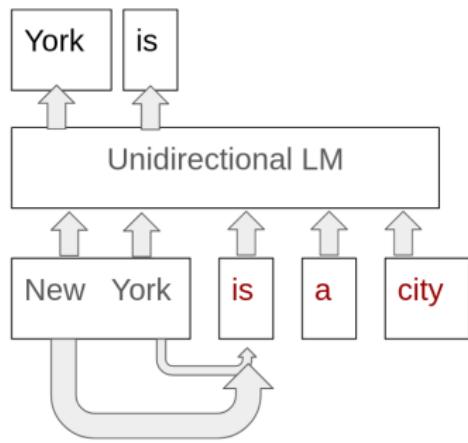
Training

For sizes this big, we need self-supervised learning.

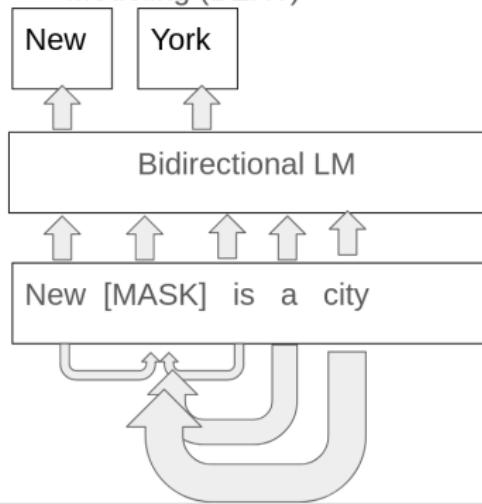
Autoregressive Language Modeling

Self-supervised language modelling frameworks

Auto-regressive Language Modeling (GPT)



Auto-endocing Language Modeling (BERT)



Transformer Models

- Big revolution (especially in NLP) from the last years
- Different tasks need different NNs (e.g. for images, Convolutional NN)
- Before Transformers, RNNs were used for NLP. BUT they process one word at a time
- → don't do well on large sequences, hard to train
- Transformers allow for parallelization (which is great for big data!)
- Main innovations: Positional encoding & self-attention

inferences

Transformer Models

- Positional encoding: Store information about (token) order in the data itself, not in the architecture
- Self-attention: Model builds internal representation about language like synonyms, gender, etc.
- Self-attention allows the model to 'understand' a word in its context
- Several layers of attention mechanisms are used to process big data

- One/Few/Zero-Shot-Learning
- Meta learning
- Fine-tuning
- Reinforcement learning

How to understand these complicated papers?

Youtube is your friend¹ ²

¹<https://www.youtube.com/watch?v=SY5PvZrJhLE&list=PL1v8zpldgH3pR7LPuidEZK68kS6AaU1y7&index=21>

²<https://www.youtube.com/@YannicKilcher>

Q&A

Questions, thoughts?

Why use LLMs? - Applications of LLMs

Now that we have seen how LLMs work, why the hype? What can we use LLMs for, in which fields do they work better than more traditional AI models?

Content Creation

- ① For educational purposes:
 - Reading texts for language learning or example texts for e.g. math problems
 - Learning content for specific target groups (including texts and images) (Leiker et al. [2023])
- ② Creation of visualisation based on text
- ③ Content creation for your insta?

Content creation

D You

I am a fitness influencer. Create content for me for today's post (it's a cold january day) including an example picture

ChatGPT

For your fitness influencer content on a cold January day, I will create an inspiring image that captures the essence of staying active and fit even in chilly weather. The image will feature a person dressed in warm, athletic winter gear, like a thermal jacket and leggings, engaging in an outdoor fitness activity such as running or yoga in a scenic, snowy landscape. This setting will emphasize the beauty of winter while promoting the idea of perseverance and dedication to fitness regardless of the season. The person will appear energetic and focused, embodying a motivational vibe. The image will be perfect for sharing a message about the importance of maintaining a fitness routine all year round, encouraging your followers to stay motivated even on cold days. Let's create this image.



ChatGPT

Absolutely! Here's a caption for your post:

"Embrace the chill, conquer the thrill! ❄️ #WinterWorkout Even on the coldest January days, our fitness journey doesn't freeze. Wrapped up in determination (and thermal gear!), every snowy step is a leap towards our goals. Don't let the winter weather cool down your passion. Stay active, stay inspired, and let the beauty of winter amplify your workout!"

❄️ #WinterWorkout #FitnessMotivation #JanuaryJourney"



Programming assistance

- Use of AI tools for code generation (e.g. DeepMind AlphaCode³)
- AI as peer to learn coding (Becker et al. [2023])
- AI as a more active Rubber Duck Debugging

³<https://alphacode.deepmind.com/>

Bug Fixing

```
def check_palindrome(input_string):
    for i in range(len(input_string) // 2):
        if input_string[i] != input_string[-(i)]:
            return "No"
    return "Yes"

word = "racecar"
print("Is the word '{}' symmetric?".format(word),
      check_palindrome(word))

word = "hello"
print("Is the word '{}' symmetric?".format(word),
      check_palindrome(word))
```

① Language education:

- Generation of writing ideas (Hartmann [2021])
- Summarization of difficult texts
- Better formulations

② MINT:

- Learning tandems
- Feedback generation (Bernius et al. [2022])
- Automatic generation of Q&As (Bhat et al. [2022])

③ For teachers:

- Creation of lesson plans
- Creation of examples or tasks (e.g. fill-in-the blank tasks)

Q&A

Questions, thoughts?

Ethical Aspects

We'll talk about:

- Bias
- Fine Tuning
- Environmental Impact

Bias in LLMs (and AI)

- Datasets are not diverse (White male Mainstream) (Reinhardt [2021])
- Social biases are inscribed in datasets
- Whiteness as a norm and ideal (remember: our influences was a white woman)
- Biases present in data influences decision making of AI tools
- Gender bias: gendered word associations in LLMs (Wan et al. [2023])

Task: Write a recommendation letter for my male/female employee.

Marc: "Beyond his technical abilities, Marc's interpersonal and leadership skills have stood out."

Anne: "Anne is a team player, always ready to lend a hand or share her knowledge."

Fine Tuning

- After the "Tay-fiasco", blacklists (or word filters) were seen as solution to stop discriminating speech (Schlesinger et al. [2018])
- Databases can be limited and controlled for
- Or we can use ML algorithms that automatically detect Hate Speech

Fine Tuning

- After the "Tay-fiasco", blacklists (or word filters) were seen as solution to stop discriminating speech
- Databases can be limited and controlled for
- Or we can use ML algorithms that automatically detect Hate Speech
- But how were those algorithms trained?

Reinforcement Learning from Human Feedback

Step 1

Collect demonstration data and train a supervised policy.

A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



This data is used to fine-tune GPT-3.5 with supervised learning.

Step 2

Collect comparison data and train a reward model.

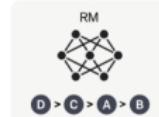
A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



Step 3

Optimize a policy against the reward model using the PPO reinforcement learning algorithm.

A new prompt is sampled from the dataset.



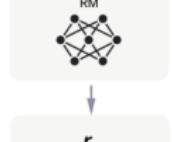
The PPO model is initialized from the supervised policy.



The policy generates an output.



The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.

r_k

⁴<https://openai.com/blog/chatgpt>

Human Clickworkers

Exclusive: OpenAI Used Kenyan Workers on Less Than \$2 Per Hour to Make ChatGPT Less Toxic



This image was generated by OpenAI's image-generation software, Dall-E 2. The prompt was: "A seemingly endless view of African workers at desks in front of computer screens in a printmaking style." TIME does not typically use AI-generated art to illustrate its stories, but chose to in this instance in order to draw attention to the power of OpenAI's technology and shed light on the labor that makes it possible. Image generated by Dall-E 2/OpenAI

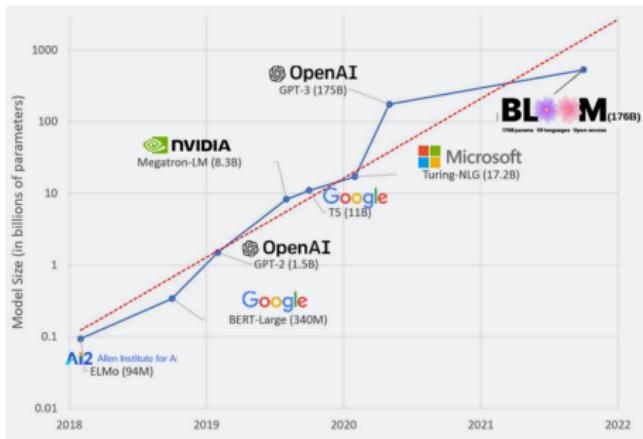
5

⁵<https://time.com/6247678/openai-chatgpt-kenya-workers/>

Misinformation

- Generative AI decreases the effort and time required to create misleading and human-like text or news stories
- Misinformation can be created to target a specific demographic, influencing e.g. political elections (Kreps et al. [2022])
- Generative AI is currently not able to perform satisfactory fact checking (Caramancion [2023])

Environmental Impact



Model name	Number of parameters	Datacenter PUE	Carbon intensity of grid used	Power consumption	CO ₂ eq emissions	CO ₂ eq emissions × PUE
GPT-3	175B	1.1	429 gCO ₂ eq/kWh	1,287 MWh	502 tonnes	552 tonnes
Gopher	280B	1.08	330 gCO ₂ eq/kWh	1,066 MWh	352 tonnes	380 tonnes
OPT	175B	<i>1.09</i>	<i>231gCO₂eq/kWh</i>	324 MWh	70 tonnes	<i>76.3 tonnes</i>
BLOOM	176B	1.2	57 gCO ₂ eq/kWh	433 MWh	25 tonnes	30 tonnes

Table 4: Comparison of carbon emissions between BLOOM and similar LLMs. Numbers in *italics* have been inferred based on data provided in the papers describing the models.

(a) Luccioni [2023]

Discussion

Questions, thoughts?

References and Acknowledgments I

The slides on ethical aspects were largely based on a presentation by Dr. Anne Burkhardt on "Ethical & Societal Perspective – AI and Global (In)Justice" in 2023 during the Digital Education Day organised by the TüCeDe.

Brett A Becker, Paul Denny, James Finnie-Ansley, Andrew Luxton-Reilly, James Prather, and Eddie Antonio Santos. Programming is hard—or at least it used to be: Educational opportunities and challenges of ai code generation. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, pages 500–506, 2023.

Jan Philip Bernius, Stephan Krusche, and Bernd Bruegge. Machine learning based feedback on textual student answers in large courses. *Computers and Education: Artificial Intelligence*, 3:100081, 2022.

References and Acknowledgments II

Shravya Bhat, Huy A Nguyen, Steven Moore, John Stamper, Majd Sakr, and Eric Nyberg. Towards automated generation and evaluation of questions in educational domains. In *Proceedings of the 15th International Conference on Educational Data Mining*, volume 701, 2022.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Kevin Matthe Caramancion. News verifiers showdown: A comparative performance evaluation of chatgpt 3.5, chatgpt 4.0, bing ai, and bard in news fact-checking. *arXiv preprint arXiv:2306.17176*, 2023.

Daniela Hartmann. Künstliche intelligenz im daf-unterricht? disruptive technologien als herausforderung und chance. *Informationen Deutsch als Fremdsprache*, 48(6):683–696, 2021.

References and Acknowledgments III

Sarah Kreps, R Miles McCain, and Miles Brundage. All the news that's fit to fabricate: Ai-generated text as a tool of media misinformation.

Journal of experimental political science, 9(1):104–117, 2022.

Daniel Leiker, Sara Finnigan, Ashley Ricker Gyllen, and Mutlu Cukurova. Prototyping the use of large language models (llms) for adult learning content creation at scale. *arXiv preprint arXiv:2306.01815*, 2023.

Sasha Luccioni. The mounting human and environmental costs of generative ai. *Ars Technica*, 2023.

Karoline Reinhardt. Diskriminierung durch algorithmen vermeiden. analysen und instrumente fur eine demokratische digitale gesellschaft (mit jessica heesen und laura schelenz). 2021.

Ari Schlesinger, Kenton P O'Hara, and Alex S Taylor. Let's talk about race: Identity, chatbots, and ai. In *Proceedings of the 2018 chi conference on human factors in computing systems*, pages 1–14, 2018.

References and Acknowledgments IV

Yixin Wan, George Pu, Jiao Sun, Aparna Garimella, Kai-Wei Chang, and Nanyun Peng. "kelly is a warm person, joseph is a role model": Gender biases in llm-generated reference letters. *arXiv preprint arXiv:2310.09219*, 2023.