InnoQuest Cohort 1
AI Bootcamp

NLP & LLMs

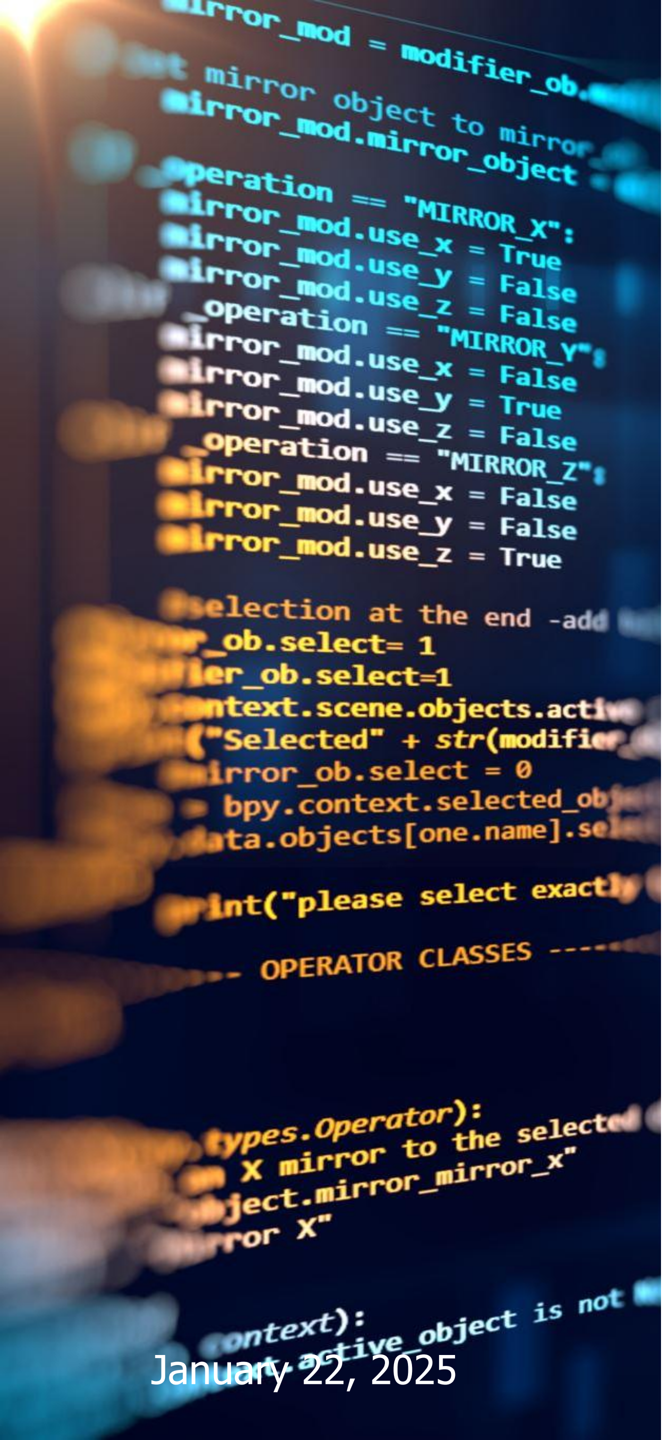
Intro to NLP

Dr Usman Zia
Director Technology, InnoVista
Asst Professor, SINES, NUST
usman.zia@sines.nust.edu.pk

My Profile

- NUST Graduate
- MS Computer Engg
- PhD Generative AI
- Software Developer/ Solution architect for past 18 years
- Development of enterprise applications
- Director Technology, InnoVista
- Asst Professor (Adjunct) at SINES, NUST
- Specialization in Deep Learning and LLM

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/



Goals of this Field

- Computers would be a lot more useful if they could handle our email, do our library research, talk to us …
- But they are fazed by natural human language.
- How can we tell computers about language? (Or help them learn it as kids do?)

Natural Language Processing (NLP)

Linguistics

- The study of natural language(s) in terms of form, meaning, and context

Computational linguistics

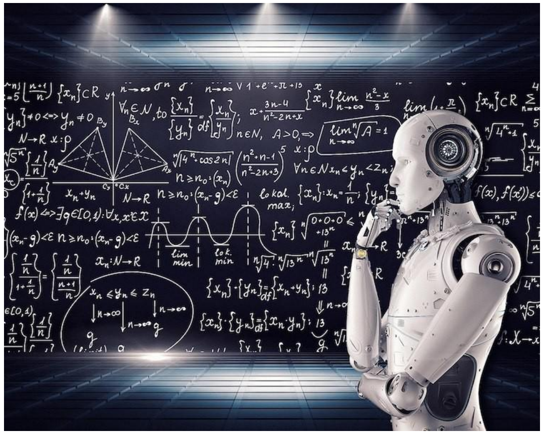
- Roughly, the intersection of computer science and linguistics
- Methods for tackling analysis and synthesis tasks from NLP
- Models to explain linguistic phenomena, using knowledge or statistics

Natural language processing

- The study of computational methods for understanding and generating human-readable text (or speech)

  We mostly speak about text only in this course.

- The goal is to decode structured information from language, or to encode it in language.



- NLP is a subfield of AI, and one part of *computational linguistics*.

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Natural Language Processing (NLP)

Analysis and Synthesis

Types of NLP tasks

- Analysis. The decoding of structured information from text
- Synthesis. The encoding of (structured) information into text

Aka *natural language understanding (NLU)* and *natural language generation (NLG)*

Selected analysis tasks

- Token and sentence splitting
- Stemming and lemmatization
- Part-of-speech tagging
- Constituency/Dependency parsing
- Named/Numeric entity recognition
- Reference resolution
- Entity/Temporal relation extraction
- Topic/Sentiment/Spam classification

Selected synthesis tasks

- Lexicon creation
- Free text generation
- Sentence composition
- Discourse composition
- Spelling correction
- Summarization
- Text style transfer
- Cluster labeling

... among many other tasks

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Natural Language Processing (NLP)

Example: Information Extraction

Task

- Identify entities, their attributes, and their relations in a given text
- Example. Extract company's founding dates from a news article

Time entity / Organization entity

" *2014 ad revenues of Google are going to reach*

Reference / Time entity

$20B. The search company was founded in '98.

Founded relation

Reference / Time entity

Its IPO followed in 2004. [...] "

Output: Founded("Google", 1998)

Possible approach

1. Lexical and syntactic preprocessing
2. Named and numeric entity recognition
3. Reference resolution

Natural Language Processing (NLP)

Example: Language modeling

Task

- Extend a given text word by word until a suitable ending is reached.
- Example. Answer a user's question to a chatbot

> H In one short sentence: What is natural language processing?

> Natural Language Processing (NLP) is a field of computer science and artificial intelligence that deals with the interaction between computers and humans through natural language.

Possible approach

1. Train general language model on huge amounts of text examples

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Natural Language Processing (NLP)

Terminology

Terms in NLP

- Task. A specific problem with a defined input and desired output
  Examples: Constituency parsing, summarization, ...
- Technique. A general way of how to analyze and/or synthesize a text
  Examples: Probabilistic parsing, language model, ...
- Algorithm. A specific implementation of a technique
  Examples: CKY parsing, GPT-3, ...
- Model. The configuration of an algorithm resulting from training
  Examples: CKY parsing on Penn Treebank, GPT-3 fine-tuned on a set of Q&A pairs, ...
- Approach. A computational method using model(s) to tackle a task
  Example: A method that fines phrases based on CYK parsing, ...
- Method. May refer to an algorithm, model, and/or approach
  Examples: As above
- Application. A technology that tackles a real-world problem using NLP
  Example: Watson, ChatGPT, ...

Applications

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Example Application: Watson

IBM Watson

- A technology for text analytics and decision support
- Originally: A focused question answering system
- First showcase was the “Jeopardy!” task

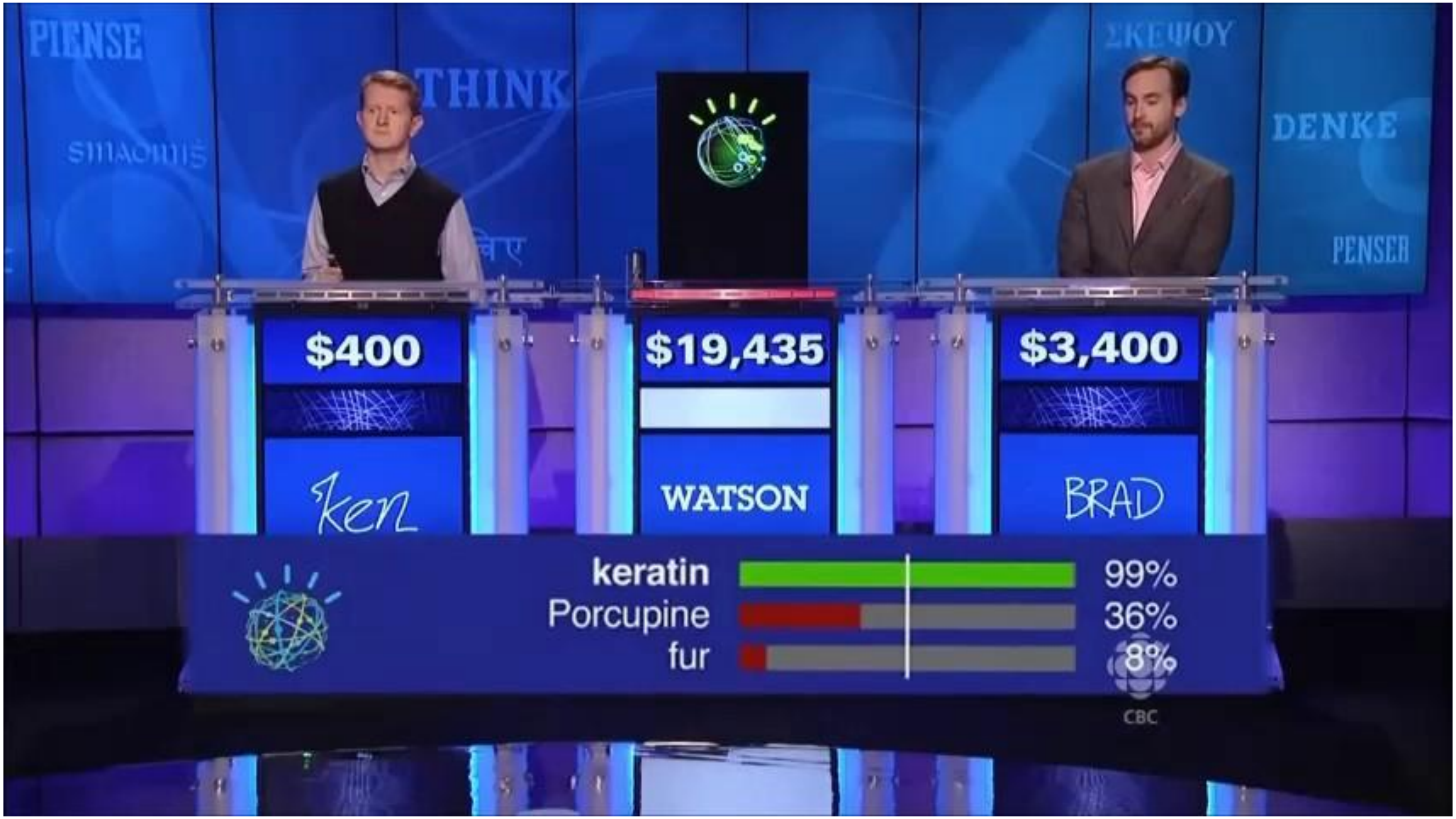
The IBM Challenge in 2011

- Watson plays against the best Jeopardy! champions
  https://www.youtube.com/watch?v=P18EdAKuC1U

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Example Application: Watson

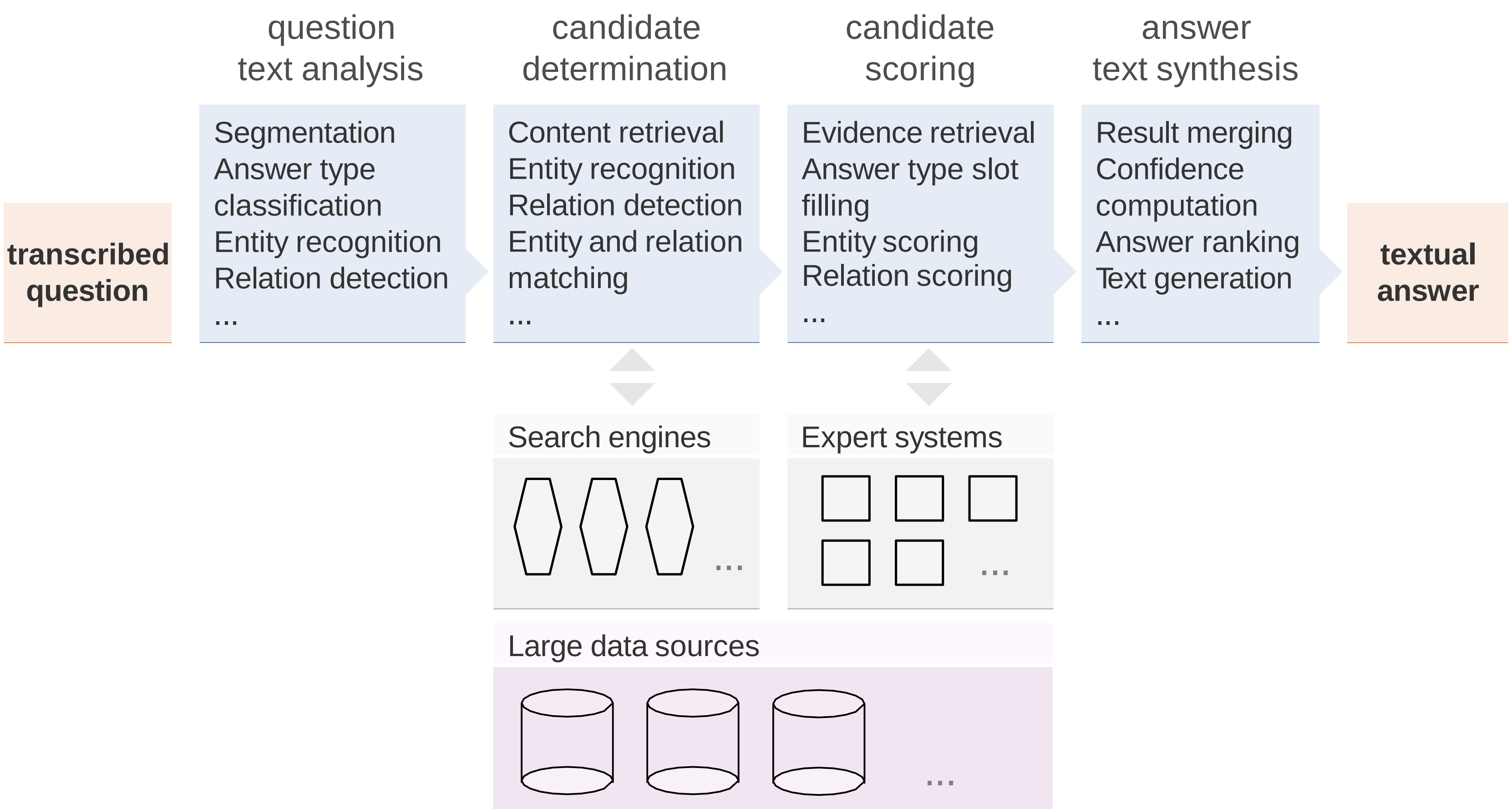
Watson's "Answer"



Example Application: Watson

NLP in Watson

Question answering process (simplified)



usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Applications

Evolution of NLP Applications

Selected milestones

- February 2011. Watson wins Jeopardy
  https://www.youtube.com/watch?v=P18EdAKuC1U
- October 2011. Siri starts on the iPhone
  https://www.youtube.com/watch?v=gUdVie_bRQo
- August 2014. Skype translates conversations in real time
  https://www.youtube.com/watch?v=RuAp92wW9bg
- May 2018. Google Duplex makes phone call appointments
  https://www.youtube.com/watch?v=pKVppdt_-B4
- February 2019. Project Debater competes in entire debates
  https://www.youtube.com/watch?v=nJXcFtY9cWY
- November 2022. ChatGPT leads conversations on any topic
  https://chat.openai.com

Observations

- NLP inside: All main analysis and synthesis tasks are tackled on text.
- None of these applications works perfectly.

Why is NLP Hard?

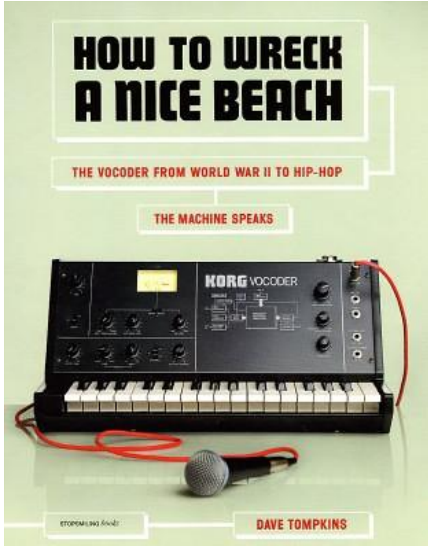
usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Challenges

Ambiguity

- Linguistic utterances allow for multiple interpretations.
- Fundamental challenge of processing natural language
- Pervasive across all language levels

Several types of ambiguity



https://flickr.com

- Phonetic. "wreck a nice beach"
- Word sense. "I went to the bank".
- Part of speech. "I made her duck."
- Attachment. "I saw a man with a telescope."
- Scope. "I didn't buy a car."
- Coordination. "If you love money problems show up."
- Speech act. "Have you emptied the dishwasher?"

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Challenges

Limitations of Focus on Text

Purpose of "I never said she stole my money."

I never said she stole my money.	• Someone else said it, but I didn't.
I *never* said she stole my money.	• I simply didn't ever say it.
I never *said* she stole my money.	• I might have implied it in some way. But I never explicitly said it.
I never said *she* stole my money.	• I said someone took it. But I didn't say it was her.
I never said she *stole* my money.	• I just said she probably borrowed it.
I never said she stole *my* money.	• I said she stole someone else's money.
I never said she stole my *money*.	• I said she stole something of mine. But not my money.

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Challenges

Non-Standard Language

Colloquial language

- Non-standard writing. “We‘re SOO PROUD of what youve accomplished! U taught us 2 #neversaynever”
- Informal use. “This is sh*t” vs. “This is the sh*t”

Special phrases

- Tricky entities. “Let it Be was recorded”, “mutation of the for gene”, ...
- Idioms. “get cold feet”, “lose face”, ...
- Neologisms. “unfriend”, “retweet”, “hangry”, ...

Tricky segmentation

- Hyphens. “the New York-New Haven Railroad”
- Punctuation. “She was a Dr. I was not.”
- Whitespaces. “   ”, “Just.Do.It.”

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Challenges

Language is dynamic

LOL	Laugh out loud
G2G	Got to go
BFN	Bye for now
B4N	Bye for now
Idk	I don't know
FWIW	For what it's worth

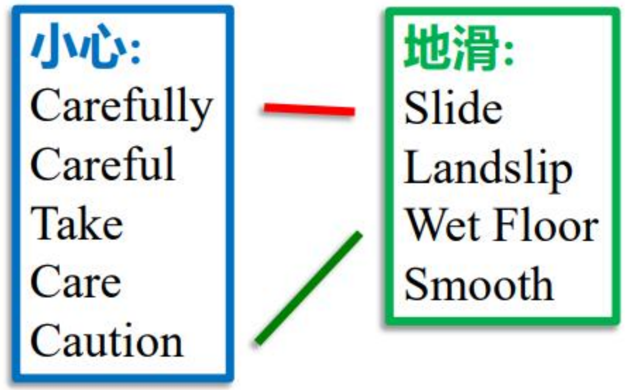
Challenges

Language is Compositional

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Challenges

Language is Compositional



usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Challenges

Scale

- Examples:
  - Bible (King James version): ~700K
  - Penn Tree bank ~1M from Wall street journal
  - Newswire collection: 500M+
  - Wikipedia: 2.9 billion word (English)
  - Web: several billions of words

Challenges

Practical Issues

Common practical issues

- NLP faces effectiveness, efficiency, and robustness issues in practice.
- How to deal with such issues will be discussed at the end of this course.

Effectiveness issues

- Effectiveness. The extent to which the output of a method is correct
- Methods may not be effective enough for use in real-life applications.

Efficiency issues

- Efficiency. The run-time, space, or energy consumption of a method
- Methods may not be efficient enough when applied to big text amounts.

Robustness issues

- Robustness. The effectiveness of a method across domains of text
- Methods may not be robust enough on data different from training data.

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

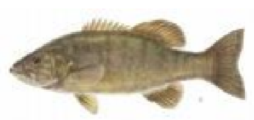
- Morphological and Lexical Analysis
- Syntactic Analysis
- Semantic Analysis
- Discourse Integration
- Pragmatic Analysis

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

STEPS of NLP

- Morphology: What is a word?
- 奧林匹克運動會（希臘語：Ολυμπιακοί Αγώνες，簡稱奧運會或奧運）是國際奧林匹克委員會主辦的包含多種體育運動項目的國際性運動會，每四年舉行一次。
- كبيوتها = "to her houses"

- Lexicography: What does each word mean?
  - He plays bass guitar.
  - That bass was delicious!



- Syntax: How do the words relate to each other?
  - The dog bit the man. ≠ The man bit the dog.
  - But in Russian: человек собаку съел = человек съел собаку

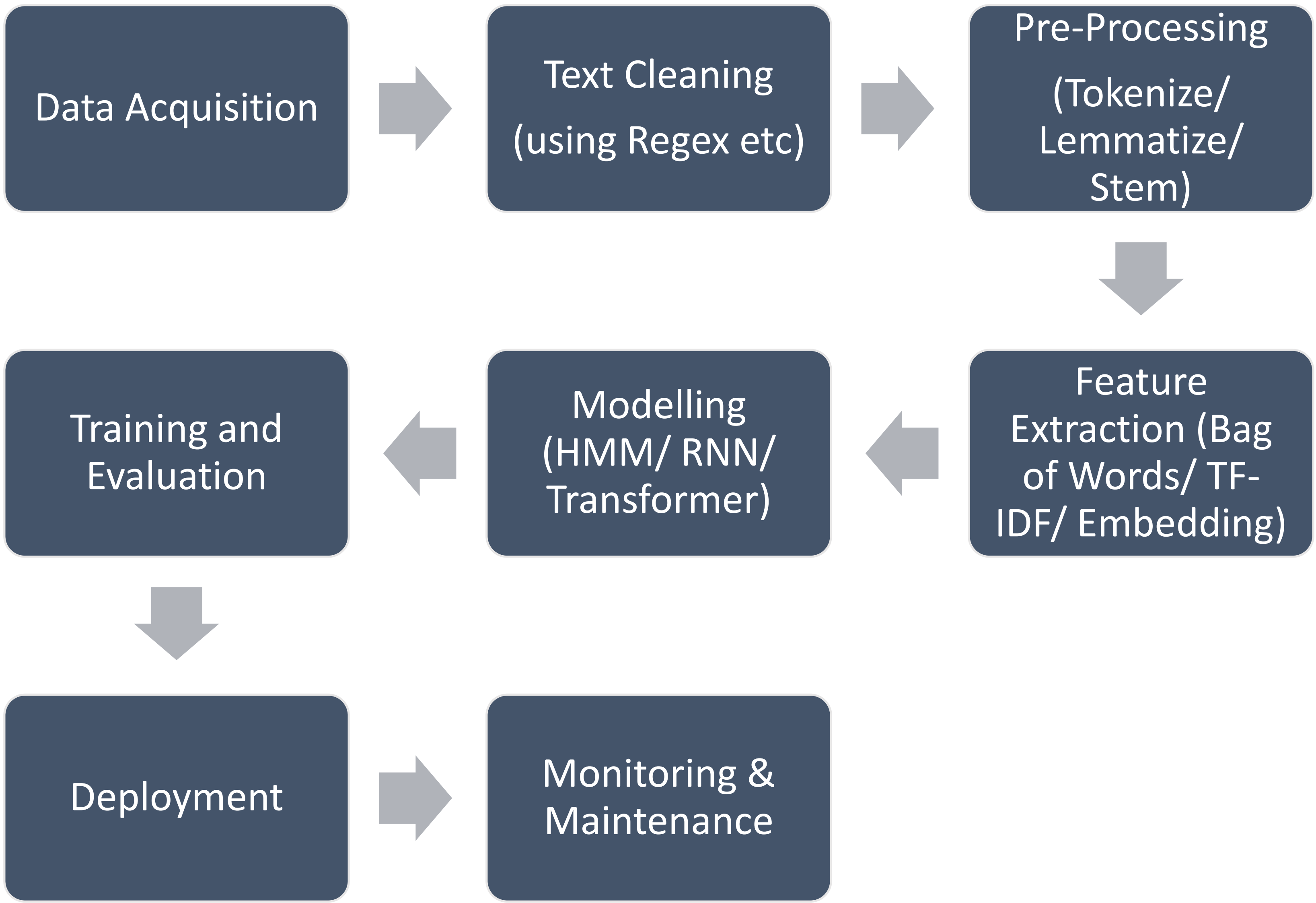
usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

STEPS of NLP

- Semantics: How can we infer meaning from sentences?
  - I saw the man on the hill with the telescope.
  - The ipod is so small! ☺
  - The monitor is so small! ☹
- Discourse: How about across many sentences?
  - President Bush met with President-Elect Obama today at the White House. He welcomed him, and showed him around.
  - Who is “he”? Who is “him”? How would a computer figure that out?

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

NLP Pipeline



usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

TEXT PRE-PROCESSING

Woodchuuucks

HOW MUCH WOOD WOULD A WOODCHUCK CHUCK,
IF A WOODCHUCK COULD CHUCK WOOD?

So much wood as a woodchuck could,
if a woodchuck would chuck wood.

It would chuck, if it would, as much wood as
it could, if a woodchuck could chuck wood.

A woodchuck would chuck no amount of wood,
since woodchucks can't chuck wood.

But if Woodchucks Could and Would Chuck Some Wood,
What Amount of Wood Would each Woodchuck Chuck?

... and so forth

Woodchuuucks

HOW MUCH WOOD WOULD A WOODCHUCK CHUCK,
IF A WOODCHUCK COULD CHUCK WOOD?

So much wood as a woodchuck could,
if a woodchuck would chuck wood.

It would chuck, if it would, as much wood as
it could, if a woodchuck could chuck wood.

A woodchuck would chuck no amount of wood,
since woodchucks can't chuck wood.

But if Woodchucks Could and Would Chuck Some Wood,
What Amount of Wood Would each Woodchuck Chuck?

... and so forth

If you don't know, maybe you know the difference between woodchucks and groundhogs?

There is none. A groundhog is a woodchuck.

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Woodchuuucks

Mining Woodchucks from Text

Woodchucks in the text above

- “Woodchuuucks”
- “WOODCHUCK”
- “woodchuck”
- “woodchucks”
- “Woodchucks”
- “Woodchuck”
- “groundhogs”
- “groundhog”

along with “It” and “it”

https://commons.wikimedia.org

Questions

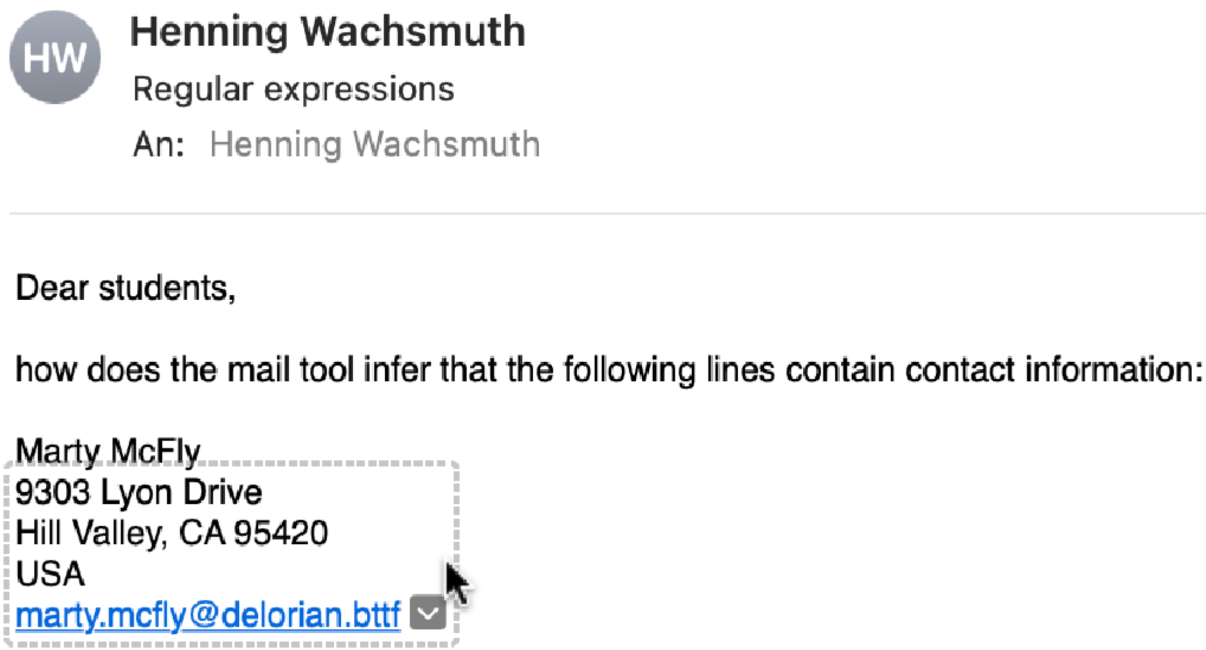
- How to find these and other references to woodchucks automatically?
- What makes woodchuck mining challenging?

Examples: Ambiguous cases, missing whitespaces, varying writings, ...

Contact Information

Finding contact information

- See the following e-mail:


HW
Henning Wachsmuth
Regular expressions
An: Henning Wachsmuth

Dear students,

how does the mail tool infer that the following lines contain contact information:

Marty McFly
9303 Lyon Drive
Hill Valley, CA 95420
USA
marty.mcfly@delorian.bttf

Questions

- How can we describe patterns in text sequences formally?
- What can be described well, what not?

Regular Expressions in NLP

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

NLP using Regular Expressions

Regular expression (regex)

- A sequence of characters that describes sequential text patterns
- A text can be matched against a regex to find all pattern instances

Use in NLP

- Effective for finding information that follows clear sequential structures
- Examples. Numeric entities, structural entities (e.g., e-mail addresses), lexico-syntactic relations (e.g., “<NN> is a <NN>”), ...

Numeric (and alphanumeric) entities

- Values, quantities, proportions, ranges, or similar
- This includes time periods, dates, monetary values, phone numbers, ...

“in this year” “2023-06-15” “$ 100 000” “762-123 77”

Numeric entity recognition

- The text analysis that mines numeric entities from text
- Used in NLP within many information extraction tasks

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Regular Expressions

Regular expression (regex)

- A regex defines a regular language over an alphabet Σ as a sequence of characters from Σ and *metacharacters*
- Metacharacters denote disjunction, negation, repetition, ... (see below)

Use of regular expressions

- Definition of patterns that generalize over structures of a language
- A pattern matches all spans of text that contain any of the structures.

Regular expressions in NLP

- Regexes are a widely used technique in NLP, particularly for the extraction of numeric and similar entities.
- In statistical NLP, regexes often take on the role of features.

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Regular Expressions

Characters and Metacharacters

Regular characters

- The default interpretation of a character sequence in a regex is a concatenation of each single character.

`woodchuck` matches “woodchuck”

Metacharacters

- A regex uses specific characters to encode specific regular language constructions, such as negation and repetition.
- The main metacharacters are the following (in Python notation):

`[]  -  |  ^  .  ( )  \  *  +  ?`

The characters used partly differ across literature and programming languages.

- Some languages also include certain *non-regular* constructions, e.g., `\b` matches if a word boundary is reached.

Regexes can solve this case only if given token information.

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Regular Expressions

Disjunction

Disjunction of patterns

- Brackets `[]` specify a character class.

`[wW]` matches “w” or “W”

`[wod]` matches “w” or “o” or “d”

- Disjunctive ranges of characters can be specified with a hyphen `-`.

`[a-zA-Z]` matches any letter

`[0-8]` matches any digit except for “9”

- The pipe `|` specifies a disjunction of string sequences.

`groundhog|woodchuck` matches “groundhog” and “woodchuck”

Notes on disjunctions

- Different disjunctions can be combined.

`[gG]roundhog|[wW]oodchuck` matches “groundhog”, “Woodchuck”, ...

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Regular Expressions

Negation, Choice, Grouping

Negation

- The caret ^ inside brackets complements the specified character class.

`[^0-9]` matches anything but digits

`[^wo]` matches any character but "w", "o"

- Outside brackets, the caret ^ is interpreted as a normal character.

`woodchuck^` matches "woodchuck^"

Free choice

- The period . matches any character.

To match a period, it needs to be escaped as: \.

`w..dchuck` matches "woodchuck", "woudchuck", ...

Grouping

- Parentheses () can be used to group parts of a regex. A grouped part is treated as a single character.

`w(oo)dchuck` matches any variation of the two o's in "woodchuck"

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Regular Expressions

Whitespaces and Predefined Character Classes

Whitespaces

- Different whitespaces are referred to with different special characters.
- For instance, `\n` is the regular new-line space.

Predefined character classes

- Several specific character classes are referred to by a backslash \ followed by a specific letter.

`\d`	Any decimal digit	equivalent to `[0-9]`
`\D`	Any non-digit character	equivalent to `[^0-9]`
`\s`	Any whitespace character	equivalent to `[\t\n\r\f\v]`
`\S`	Any non-whitespace character	equivalent to `[^\t\n\r\f\v]`
`\w`	Any alphanumeric character	equivalent to `[a-zA-Z0-9]`
`\W`	Any non-alphanumeric character	equivalent to `[^a-zA-Z0-9]`

- These classes can be used within brackets.

`[\s0-9]` matches any space and digit.

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Regular Expressions

Repetition

Repetition

- The asterisk `*` repeats the previous character zero or more times.

  `woo*dchuck` matches “wodchuck”, “woodchuck”, “wooodchuck”, ...

- The plus `+` repeats the previous character one or more times.

  `woodchu+ck` matches “woodchuck”, “woodchuuck”,“woodchuuuck”, ...

- The question mark `?` repeats the previous character zero or one time.

  `woodchucks?` matches “woodchuck” and “woodchucks”

Notes on repetitions

- Repetitions are implemented in a greedy manner in many programming languages, i.e., longer matches are preferred over shorter ones.

  `to*` matches “too”, not “too”, ...

- This may actually violate the regularity of the defined language.

  “woodchuck” needs to be processed twice for the regex `wo*odchuck`

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Regular Expressions

Examples

The

- Regex for all variations of “the” in news article text:

`the` (misses capitalized cases, matches “theology”, ...)

`[^a-zA-Z][tT]he[^a-zA-Z]` (requires a character before and afterwards)

Woodchucks

- Regex for all woodchuck cases from above (and for similar):

```
( [wW][oO][oO][dD][cC][hH][uU]+[cC][kK] | [Gg]roundhog ) [sS]?
```

E-mail addresses

- All e-mail addresses with the Pakistani educational top-level domain, whose text segments contain no special characters:

```
[a-zA-Z0-9]+ @ ([a-zA-Z0-9]+\.)* [a-zA-Z0-9][a-zA-Z0-9]+ \.edu.pk
```

Time Expression Recognition with Regular Expressions

Time expression

- An alphanumeric entity that represents a date or a period

> "Cairo, August 25th 2010 – Forecast on Egyptian Automobile industry
> [...] In the next five years, revenues will rise by 97% to US-$ 19.6 bn. [...]"

Time expression recognition

- The text analysis that finds time expressions in natural language text
- Used in NLP within temporal relation extraction and event extraction

Approach in a nutshell

- Model sequential structure of time expressions with a complex regex.
- Include lexicons derived from training data to match closed-class terms.
  Example closed classes: Month names, prepositions, ...
- Match regex with sentences of a text.

Notice

- The approach can easily be adapted to other types of information.

Time Expression Recognition with Regular Expressions

Pseudocode

Signature

- Input. A text split into sentences, and a regex
- Output. All time expressions in the text

```
extractAllMatches(List<Sentence> sentences, Regex regex)
    List<TimeExpression> matches ← ()
    for each sentence ∈ sentences do
        int index ← 0
        while index < sentence.length - 1 do
            int [] exp ← regex.match(sentence.sub(index))
            if exp ≠ ⊥ then    // ⊥ represents "null"
                matches.add(new TimeExpression(exp[0], exp[1]))
                index ← exp[1]
            index ← index + 1
    return matches
```

Notice

- Most programming languages provide explicit matching classes.

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

A Simple Exercise

- Write a regular expression to find all instances of the determiner "the":

The recent attempt by the police to retain their current rates of pay has not gathered much favor with the southern factions.

A Simple Exercise

- Write a regular expression to find all instances of the determiner “the”:

  /the/

The recent attempt by the police to retain their current rates of pay has not gathered much favor with

southern factions.

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

A Simple Exercise

- Write a regular expression to find all instances of the determiner "the":

  /the/

  /[tT]he/

The recent attempt by the police to retain their current rates of pay has not gathered much favor with the southern factions.

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

A Simple Exercise

- Write a regular expression to find all instances of the determiner "the":

  /the/

  /[tT]he/

*The recent attempt by the police to retain **the**ir current rates of pay has not ga**the**red much favor with the sou**the**rn factions.*

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

A Simple Exercise

- Write a regular expression to find all instances of the determiner "the":

  /the/

  /[tT]he/

  /\b[tT]he\b/

The recent attempt by the police to retain their current rates of pay has not gathered much favor with the southern factions.

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

A Simple Exercise

- Write a regular expression to find all instances of the determiner "the":

/the/

/[tT]he/

/\b[tT]he\b/

/(^|[^a-zA-Z])[tT]he[^a-zA-Z]/

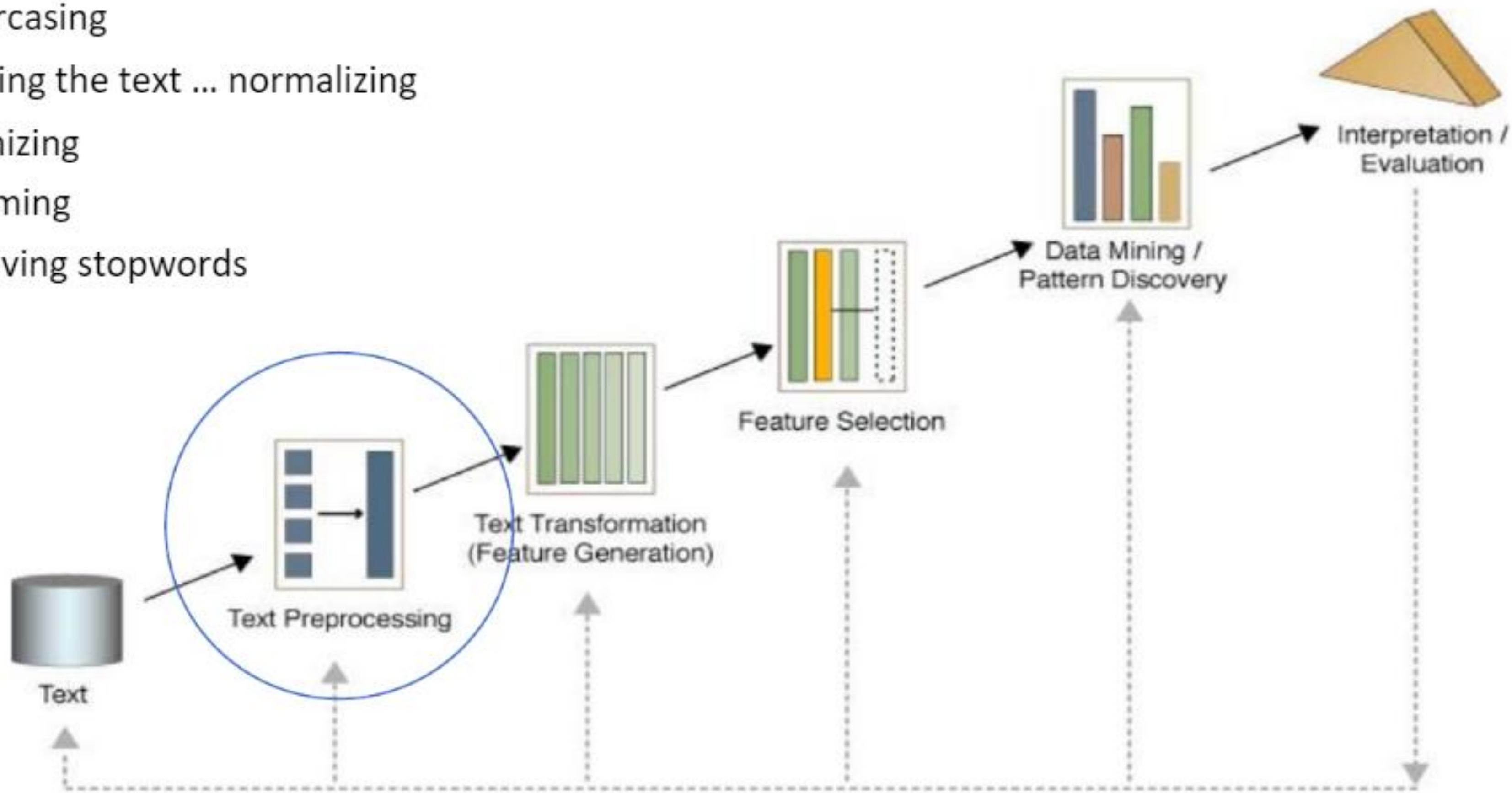
The recent attempt by the police to retain their current rates of pay has not gathered much favor with the southern factions.

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

TextPreprocessing

Text preprocessing is the task of transforming the text into a form that isanalyzablefor a given task.

- Lowercasing
- Cleaning the text ... normalizing
- Tokenizing
- Stemming
- Removing stopwords



Text Preprocessing: lowercasing

- **Lowercasing** all text, although commonly overlooked, is one of the simplest and most effective form of text preprocessing. It is applicable to most text mining problems and significantly helps with consistency of expected output.

- This is so that words like Skype and SKYPE are counted as the same thing. Case variations are so common (consider iPhone, iphone, and IPHONE) that case normalization is usually necessary.

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Text Preprocessing: lowercasing

- **Lowercasing** all text, although commonly overlooked, is one of the simplest and most effective form of text preprocessing. It is applicable to most text mining problems and significantly helps with consistency of expected output.

- This is so that words like Skype and SKYPE are counted as the same thing. Case variations are so common (consider iPhone, iphone, and IPHONE) that case normalization is usually necessary.

- Example of a task where lowercasing is not helpful:
  - distinguishing US and us,
  - predicting programming language of a source code file.
  - The wordSystem in Java is quite different from system in python. Lowercasing the two makes them identical, causing the classifier to lose important predictive features.

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Text Preprocessing: Cleaning the text & normalizing

Removing HTML tags

- If the reviews or texts are web scraped, chances are they will contain some HTML tags. Since these tags are not useful for text mining tasks, it is better to remove them.

Converting accented characters to ASCII characters

- Words with accents like "latté" and "café" can be converted and standardized to just "latte" and "cafe", else a model will treat them as different words even though they are referring to the same thing.

Expanding contractions

- Contractions are shortened words, e.g., don't and can't. Expanding such words to "do not" and "can not" helps to standardize text.

Standardizing different spelling ... abbreviations

- This is especially important for noisy texts such as social media comments, text messages and comments to blog posts where abbreviations and misspellings are common (2morrow and tomorrow).

Removing extra whitespaces

Removing punctation and special characters

- (matching USA and U.S.A.)

Converting number words to numeric form, removing numbers

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Code: Remove Punctuation

python Copy code

```python
import re

text = "Hello, world! This is some text with punctuation."
clean_text = re.sub(r'[^\w\s]', '', text)
print(clean_text)
```

Output:

vbnet Copy code

```vbnet
Hello world This is some text with punctuation
```


usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Code: Make All Text Lowercase

Input:

```
clean_text = clean_text.lower()
clean_text
```

Output:

```
'Hi Mr  Smith  I m going to buy some vegetables  tomatoes and cucumbers  from
the store  Should I pick up 2lbs of black eyed peas as well '
```

```
'hi mr  smith  i m going to buy some vegetables  tomatoes and cucumbers  from
the store  should i pick up 2lbs of black eyed peas as well '
```


usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Code: Remove Numbers

Input:

```
# Removes all words containing digits
clean_text = re.sub('\w*\d\w*', ' ', clean_text)
clean_text
```

Output:

```
'hi mr  smith  i m going to buy some vegetables  tomatoes and cucumbers  from
the store  should i pick up   of black eyed peas as well '
```


usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Preprocessing: Stop Words

Hi Mr. Smith! I'm going to buy some vegetables (tomatoes and cucumbers) from the store. Should I pick up some black-eyed peas as well?

What is the most frequent term in the text above? Is that information meaningful?

Stop words are words that have very little semantic value.

There are language and context-specific stop word lists online that you can use.

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Code: Stop Words

Input:

```
from nltk.corpus import stopwords
set(stopwords.words('english'))
```

Output:

```
{'but', 'isn', 'under', 'weren', 'those', 'when', 'why', 'few', 'for', 'it', 'of', 'down', 'ma',
'over', 'd', 'during', 'shouldn', 'did', 'above', 'below', 'myself', 'further', 'very', 'same',
'too', 'does', 'through', 'from', 'didn', 'whom', 'and', 'am', 'such', 'out', 'or', 'me', 'has',
'will', 'shan', 'on', 'then', 'here', 't', 'with', 'some', 'what', 'don', 'were', 'an',
'themselves', 'yourselves', 'off', 'being', 'more', 'they', 'ourselves', 'into', 'my', 'them',
'ain', 'a', 'wouldn', 'itself', 'i', 'hasn', 'her', 'their', 'mustn', 'our', 'herself', 'where',
'hers', 'once', 'any', 'theirs', 'before', 'most', 'other', 'not', 'himself', 'his', 'if', 'he',
'each', 'are', 'how', 'couldn', 'ours', 'doing', 'hadn', 'needn', 'again', 'these', 'wasn', 'nor',
'do', 'just', 'so', 'we', 'there', 'have', 'by', 'o', 'than', 're', 'while', 'your', 'at', 'him',
'own', 'can', 'you', 'll', 'between', 'been', 'that', 'is', 'she', 'yours', 'this', 'was', 'be',
'had', 'doesn', 'no', 'because', 'won', 'both', 'to', 'against', 'aren', 'y', 'after', 'all', 'up',
've', 'should', 'as', 'in', 'the', 'having', 'until', 'who', 'haven', 'only', 'm', 'yourself',
'about', 's', 'which', 'now', 'mightn', 'its'}
```


usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Text Preprocessing: tokenization

- **Tokenization** is a step which splits longer strings of text into smaller pieces, or tokens.

- Larger chunks of text can be tokenized into sentences, sentences can be tokenized into words, etc.

- Tokenization is also referred to as text segmentation or lexical analysis.

- Sometimes segmentation is used to refer to the breakdown of a large chunk of text into pieces larger than words (e.g. paragraphs or sentences), while tokenization is reserved for the breakdown process which results exclusively in words.

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Text Preprocessing: tokenization

- How are sentences identified within larger bodies of text?

- Using "sentence-ending punctuation," is ambiguous.

  The quick brown fox jumps over the lazy dog.

- But what about this one:

  Dr. Ford did not ask Col. Mustard the name of Mr. Smith's dog.

- Or this one:

  "What is all the fuss about?" asked Mr. Peters.

- And that's just sentences. What about words? Easy, right? Right?

  This full-time student isn't living in on-campus housing, and she's not wanting to visit Hawai'i.

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Text Preprocessing: tokenization in different language

- German noun compounds are not segmented
  - ***Lebensversicherungsgesellschaftsangestellter***
  - 'life insurance company employee'
  - German information retrieval needs **compound splitter**

French

- ***L'ensemble*** → one token or two?
  - ***L*** ? ***L'*** ? ***Le*** ?
  - Want ***l'ensemble*** to match with ***un ensemble***

- Chinese and Japanese no spaces between words:
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
  - 莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达
  - Sharapova now lives in US southeastern Florida
- Further complicated in Japanese, with multiple alphabets intermingled
  - Dates/amounts in multiple formats

フォーチュン500社は情報不足のため時間あた$500K(約6,000万円)

Katakana Hiragana Kanji Romaji

End-user can express query entirely in hiragana!

- Also called **Word Segmentation**
- Chinese words are composed of characters
  - Characters are generally 1 syllable and 1 morpheme.
  - Average word is 2.4 characters long.
- Standard baseline segmentation algorithm:
  - Maximum Matching (also called Greedy)

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Issues in Tokenization

- *Finland's capital* → *Finland Finlands Finland's ?*
- *what're, I'm, isn't* → *What are, I am, is not*
- *Hewlett-Packard* → *Hewlett Packard ?*
- *state-of-the-art* → *state of the art ?*
- *Lowercase* → *lower-case lowercase lower case ?*
- *San Francisco* → *one token or two?*
- *m.p.h., PhD.* → *??*

In Natural Language Generation we care about punctuation – and do not discard it!!!!

Finland's capital → *Finland ' s capital*

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Tokenization

A simple way: Split by spaces + rules to handle punctuations and special cases

"They currently play their home games at Acrisure Stadium." → "They" "currently" "play" "their" "home" "games" "at" "Acrisure" "Stadium" "."

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Tokenization

A simple way: Split by spaces + rules to handle punctuations and special cases

"They currently play their home games at Acrisure Stadium." → "They" "currently" "play" "their" "home" "games" "at" "Acrisure" "Stadium" "."

Challenges:

- Rules differ language to language
  - E.g. in English "don't", "cannot", "pre-training"
  - There are always edge cases
  - Some languages do not use space to separate words

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Tokenization

A simple way: Split by spaces + rules to handle punctuations and special cases

"They currently play their home games at Acrisure Stadium." → "They" "currently" "play" "their" "home" "games" "at" "Acrisure" "Stadium" "."

Challenges:

- Vocabulary explosion
  - Large vocabulary creates instability issue
  - Little signals for rare words in the long tail
    - Many of them are important, such as named entities ("Acrisure")

Tokenization

A simple way: Split by spaces + rules to handle punctuations and special cases

"They currently play their home games at Acrisure Stadium." → "They" "currently" "play" "their" "home" "games" "at" "Acrisure" "Stadium" "."

Challenges:

- Open vocabulary problem
  - Many words may never appear in training data. They become [UNK].
  - More severe in some languages
    - Low resource language
    - Language that have a large vocabulary, e.g., those that concatenate words

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Character-based tokenization

Solved the problem of missing words, as now we are dealing with characters that can be encoded using ASCII or Unicode.

Now it could generate embedding for any word.

Every character, whether it was a space, apostrophe, colon, or whatever can now be assigned a symbol to generate a sequence of vectors.

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Character-based tokenization

Solved the problem of missing words, as now we are dealing with characters that can be encoded using ASCII or Unicode.

Now it could generate embedding for any word.

Every character, whether it was a space, apostrophe, colon, or whatever can now be assigned a symbol to generate a sequence of vectors.

Challenges:

- Requires more computing resources
- Limits the type of NLP tasks we can perform. For applications like entity recognition or text classification, character-based encoding might turn out to be an inefficient approach
- Risk of learning incorrect semantics. Working with characters could generate incorrect spellings of words. Also, with no inherent meaning, learning with characters is like learning with no meaningful semantics.

Subword Tokenization

Tokenize sequences into sub-words

"They currently play their home games at Acrisure Stadium." → '▁They', '▁currently', '▁play', '▁their', '▁home', '▁games', '▁at', '▁A', 'cris', 'ure', '▁Stadium', '.'

- A dynamic tokenization:
  - Frequent words kept as whole
  - Rare words split into sub-words

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Subword Tokenization: Byte Pair Encoding (BPE)

Byte-pair encoding (BPE; Gage, 1994)

- Originally an algorithm for text compression
- Now it's the tokenization technique behind most language models, including GPTs
- First we have to *learn* a subword vocabulary (using corpus statistics), then we can tokenize

Subword Tokenization: Byte Pair Encoding (BPE)

Byte Pair Encoding: Construct subword vocabulary by learning to merge characters

- Inspiration from compression algorithms

Training Steps:

1. Start from single character vocabulary
    - E.g., in English, alphabets + punctuations
2. Merge the most frequent subword pair
    - Vocabulary size +1
3. Re-tokenize the corpus with the merged subword pair
    - Merge all appearances of that subword pair
4. Repeat step 2-3 till reached target vocabulary size

Subword Tokenization: Byte Pair Encoding (BPE)

Examples

- BERT : “Ex ##ample of token ##ization for IN ##55 ##50 .”
- XLM-R : “▁Exam ple ▁of ▁to ken ization ▁for ▁IN 55 50 .”
- ChatGPT : “Example Ġof Ġtoken ization Ġfor ĠIN 55 50 .”

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Subword Tokenization: Byte Pair Encoding (BPE)

Byte-pair encoding (BPE; Gage, 1994) – training

Text corpus: "AABCCAABAABCC"

- s_0: "A"
- s_1: "B"
- s_2: "C"

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Subword Tokenization: Byte Pair Encoding (BPE)

Byte-pair encoding (BPE; Gage, 1994) – training

Text corpus: 0 0 1 2 2 0 0 1 0 0 1 2 2

- s_0: "A"
- s_1: "B"
- s_2: "C"

Subword Tokenization: Byte Pair Encoding (BPE)

Byte–pair encoding (BPE; Gage, 1994) – training

Text corpus: 0 0–1 2 2 0 0–1 0 0–1 2 2

- s_0: "A"
- s_1: "B"
- s_2: "C"
- s_3: $s_0 + s_1 =$ "AB"

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Subword Tokenization: Byte Pair Encoding (BPE)

Byte-pair encoding (BPE; Gage, 1994) – training

Text corpus: 0 3 2 2 0 3 0 3 2 2

- s_0: "A"
- s_1: "B"
- s_2: "C"
- s_3: $s_0 + s_1 =$ "AB"

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Subword Tokenization: Byte Pair Encoding (BPE)

Byte-pair encoding (BPE; Gage, 1994) – training

Text corpus: 0–3 2 2 0–3 0–3 2 2

- s_0: "A"
- s_1: "B"
- s_2: "C"
- s_3: $s_0 + s_1$ = "AB"
- s_4: $s_0 + s_3$ = "AAB"

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Subword Tokenization: Byte Pair Encoding (BPE)

Byte-pair encoding (BPE; Gage, 1994) – training

Text corpus: 4 2 2 4 4 2 2

- s_0: "A"
- s_1: "B"
- s_2: "C"
- s_3: $s_0 + s_1$ = "AB"
- s_4: $s_0 + s_3$ = "AAB"

Subword Tokenization: Byte Pair Encoding (BPE)

Byte–pair encoding (BPE; Gage, 1994) – training

Text corpus: 4 2–2 4 4 2–2

- s_0: "A"
- s_1: "B"
- s_2: "C"
- s_3: $s_0 + s_1$ = "AB"
- s_4: $s_0 + s_3$ = "AAB"
- s_5: $s_2 + s_2$ = "CC"

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Subword Tokenization: Byte Pair Encoding (BPE)

Byte-pair encoding (BPE; Gage, 1994) – training

Text corpus: 4 5 4 4 5

- s_0: "A"
- s_1: "B"
- s_2: "C"
- s_3: $s_0 + s_1$ = "AB"
- s_4: $s_0 + s_3$ = "AAB"
- s_5: $s_2 + s_2$ = "CC"

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Subword Tokenization: Byte Pair Encoding (BPE)

Byte–pair encoding (BPE; [Gage, 1994]) – training

Text corpus: 4–5 4 4–5

- s_0: "A"
- s_1: "B"
- s_2: "C"
- s_3: $s_0 + s_1$ = "AB"
- s_4: $s_0 + s_3$ = "AAB"
- s_5: $s_2 + s_2$ = "CC"
- s_6: $s_4 + s_5$ = "AABCC"

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Subword Tokenization: Byte Pair Encoding (BPE)

Byte–pair encoding (BPE; Gage, 1994) – training

Text corpus: 6 4 6

- s_0: "A"
- s_1: "B"
- s_2: "C"
- s_3: $s_0 + s_1$ = "AB"
- s_4: $s_0 + s_3$ = "AAB"
- s_5: $s_2 + s_2$ = "CC"
- s_6: $s_4 + s_5$ = "AABCC"

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Subword Tokenization: Byte Pair Encoding (BPE)

Byte–pair encoding (BPE; Gage, 1994) – tokenization

Raw text: "CCAABCC"

Vocabulary / merge list

- s_0: "A"
- s_1: "B"
- s_2: "C"
- s_3: $s_0 + s_1$ = "AB"
- s_4: $s_0 + s_3$ = "AAB"
- s_5: $s_2 + s_2$ = "CC"
- s_6: $s_4 + s_5$ = "AABCC"

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Subword Tokenization: Byte Pair Encoding (BPE)

Byte–pair encoding (BPE; Gage, 1994) – tokenization

Raw text: "CCAABCC"

- 2 2 0 0 1 2 2

Vocabulary / merge list

- s_0: "A"
- s_1: "B"
- s_2: "C"
- s_3: $s_0 + s_1$ = "AB"
- s_4: $s_0 + s_3$ = "AAB"
- s_5: $s_2 + s_2$ = "CC"
- s_6: $s_4 + s_5$ = "AABCC"

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Subword Tokenization: Byte Pair Encoding (BPE)

Byte–pair encoding (BPE; Gage, 1994) – tokenization

Raw text: "CCAABCC"

- 2 2 0 0 1 2 2
- 2 2 0 3 2 2

Vocabulary / merge list

- s_0: "A"
- s_1: "B"
- s_2: "C"
- s_3: $s_0 + s_1$ = "AB"
- s_4: $s_0 + s_3$ = "AAB"
- s_5: $s_2 + s_2$ = "CC"
- s_6: $s_4 + s_5$ = "AABCC"

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Subword Tokenization: Byte Pair Encoding (BPE)

Byte–pair encoding (BPE; [Gage, 1994]) – tokenization

Raw text: "CCAABCC"

- 2 2 0 0 1 2 2
- 2 2 0 3 2 2
- 2 2 4 2 2

Vocabulary / merge list

- s_0: "A"
- s_1: "B"
- s_2: "C"
- s_3: $s_0 + s_1$ = "AB"
- s_4: $s_0 + s_3$ = "AAB"
- s_5: $s_2 + s_2$ = "CC"
- s_6: $s_4 + s_5$ = "AABCC"

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Subword Tokenization: Byte Pair Encoding (BPE)

Byte–pair encoding (BPE; Gage, 1994) – tokenization

Raw text: "CCAABCC"

- 2 2 0 0 1 2 2
- 2 2 0 3 2 2
- 2 2 4 2 2
- 5 4 5

Vocabulary / merge list

- s_0: "A"
- s_1: "B"
- s_2: "C"
- s_3: $s_0 + s_1$ = "AB"
- s_4: $s_0 + s_3$ = "AAB"
- s_5: $s_2 + s_2$ = "CC"
- s_6: $s_4 + s_5$ = "AABCC"

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

Subword Tokenization: Byte Pair Encoding (BPE)

Byte–pair encoding (BPE; Gage, 1994) – tokenization

Raw text: "CCAABCC"

- 2 2 0 0 1 2 2
- 2 2 0 3 2 2
- 2 2 4 2 2
- 5 4 5
- 5 6

→ 2 tokens: "CC" + "AABCC"

Vocabulary / merge list

- s_0: "A"
- s_1: "B"
- s_2: "C"
- s_3: $s_0 + s_1$ = "AB"
- s_4: $s_0 + s_3$ = "AAB"
- s_5: $s_2 + s_2$ = "CC"
- s_6: $s_4 + s_5$ = "AABCC"

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

BPE token learner

An example corpus :(

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

Add end-of-word tokens and segment:

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

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

BPE token learner

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

Merge e r to er

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

BPE

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

Merge er _ to er_

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

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

BPE

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

Merge n e to ne

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

BPE

The next merges are:

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

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/

BPE token learner algorithm

- On the test data, run each merge learned from the training data:
  - Greedily
  - In the order we learned them
  - (test frequencies don't play a role)
- So: merge every e r to er, then merge er _ to er_, etc.
- Result:
  - Test set "n e w e r _" would be tokenized as a full word
  - Test set "l o w e r _" would be two tokens: "low er_"

THANK YOU

usman.zia@sines.nust.edu.pk linkedin.com/in/usmanxia/