

Natural Language Processing

Making the world a better place through Artificial Intelligence!

What is NLP

- Branch of AI
- 2 Goals
 - Science Goal: Understand the way language operates
 - Engineering Goal: Build systems that analyse and generate language; reduce the man machine gap
- “It’s the ability of computer program to understand human language as it is spoken and written.”
- NLP combines computational linguistics—rule-based modeling of human language—with statistical, machine learning, and deep learning models.
- Together, these technologies enable computers to process human language in the form of text or voice data.
- It allows computers to 'understand' the full meaning of language, including the speaker or writer's intent and sentiment.

Why NLP?

- A hallmark of human intelligence
- Text is the largest repository of human knowledge and it is growing quickly
- Computer programs that understood text or speech

History of NLP

- 1940-1960 - Focused on Machine Translation (MT)
 - The Natural Languages Processing started in the year 1940s
- 1948 - In the Year 1948, the first recognizable NLP application was introduced in Birkbeck College, London.
- 1950s - In the Year 1950s, there was a conflicting view between linguistics and computer science. Now, Chomsky developed his first book syntactic structures and claimed that language is generative in nature.
- In 1957, Chomsky also introduced the idea of Generative Grammar, which is rule based descriptions of syntactic structures.
- 1960-1980 - Flavored with Artificial Intelligence (AI)
- In the year 1960 to 1980, the key developments were:
 - Augmented Transition Networks (ATN): Augmented Transition Networks is a finite state machine that is capable of recognizing regular languages.
 - Case Grammar
 - Case Grammar was developed by Linguist Charles J. Fillmore in the year 1968. Case Grammar uses languages such as English to express the relationship between nouns and verbs by using the preposition.
 - In Case Grammar, case roles can be defined to link certain kinds of verbs and objects.
 - For example: "Neha broke the mirror with the hammer". In this example case grammar identify Neha as an agent, mirror as a theme, and hammer as an instrument.

History of NLP (Cont.)

– In the year 1960 to 1980, key systems were:

– SHRDLU

- SHRDLU is a program written by Terry Winograd in 1968-70. It helps users to communicate with the computer and moving objects. It can handle instructions such as "pick up the green ball" and also answer the questions like "What is inside the black box." The main importance of SHRDLU is that it shows those syntax, semantics, and reasoning about the world that can be combined to produce a system that understands a natural language.

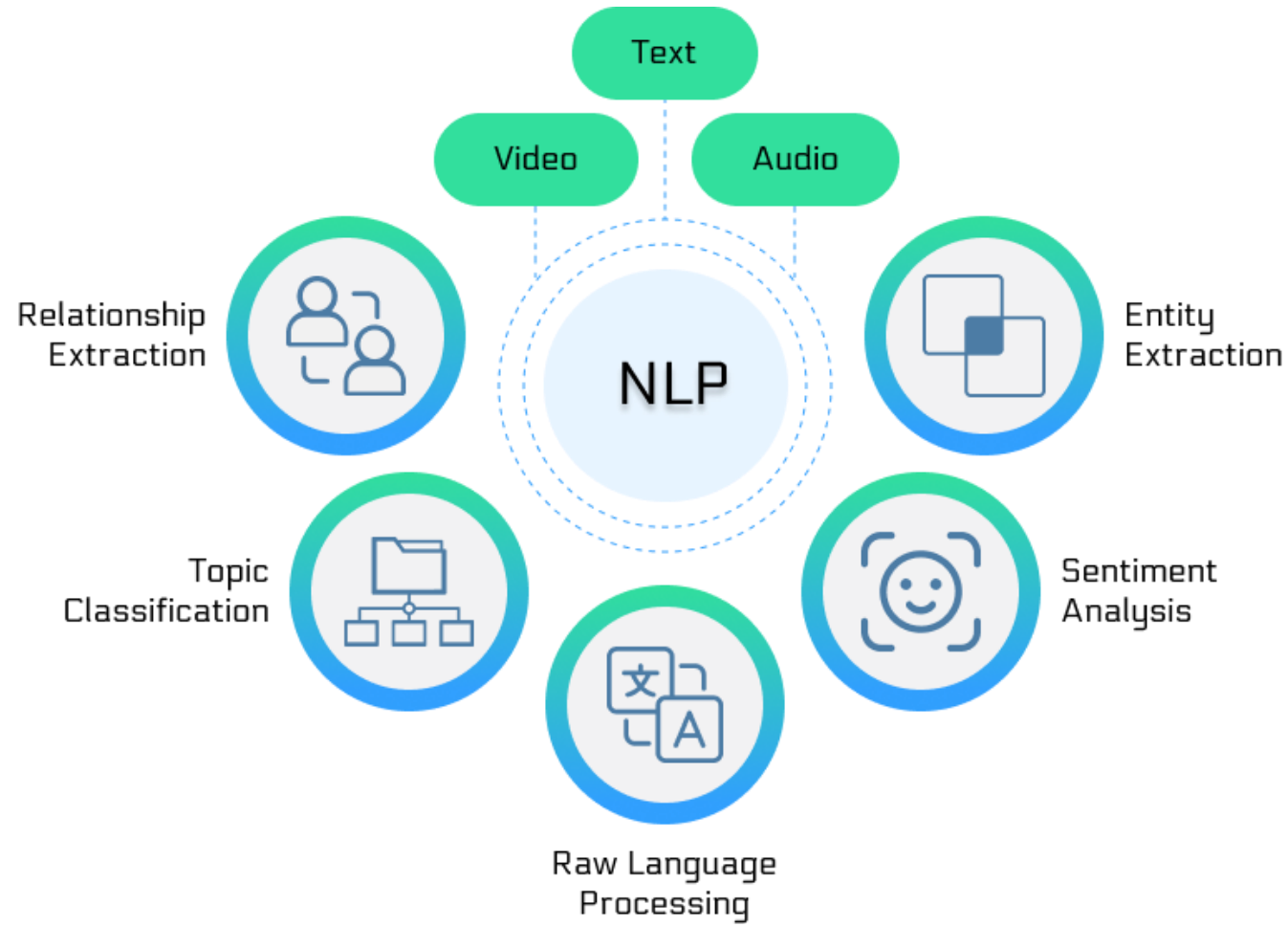
– LUNAR

- LUNAR is the classic example of a Natural Language database interface system that is used ATNs and Woods' Procedural Semantics. It was capable of translating elaborate natural language expressions into database queries and handle 78% of requests without errors.

– 1980 – Current

- Till the year 1980, natural language processing systems were based on complex sets of hand-written rules. After 1980, NLP introduced machine learning algorithms for language processing.

NLP Applications



ChatGPT by OpenAI

The screenshot shows a web browser window with the ChatGPT interface. The browser's address bar shows the URL `chatgpt.com/c/fe8c7dbf-9737-44b9-b9cc-8b0f24626500`. The browser's bookmark bar includes links to Kics.edu.pk Mail, Google Scholar, My Drive - Google..., Grammarly, Boards | Trello, Wordtune Editor, YouTube, Google Slides, Paraphrasing Tool, and My Meetings - Zoom. The ChatGPT interface shows a conversation with the user and the AI model.

You

Write a comprehensive literature review and draw a table consisting of 5 columns on the applications of machine learning in medical diagnosis for a research paper.

ChatGPT

Writing a comprehensive literature review and creating a table for your research paper on the applications of machine learning in medical diagnosis involves synthesizing existing studies, identifying key findings, and presenting them in a coherent manner. Here's an outline to help you get started:

Literature Review:

1. Introduction

- Provide an overview of the significance of medical diagnosis and the role of machine learning in improving diagnostic accuracy.
- State the purpose of the literature review and outline the structure.

2. Machine Learning Techniques in Medical Diagnosis

- Describe various machine learning techniques commonly used in medical diagnosis, such as supervised learning, unsupervised learning, and deep learning.

Message ChatGPT

ChatGPT can make mistakes. Consider checking important information.

ChatGPT-4 by OpenAI

The screenshot displays the ChatGPT-4 interface with a custom GPT named "GenAI-Principles-GPT" in draft mode. The interface is split into two main sections: configuration on the left and a preview on the right.

Configuration Section (Left):

- Name:** GenAI-Principles-GPT
- Description:** GenAI-Principles-GPT
- Instructions:** What does this GPT do? How does it behave? What should it avoid doing?
- Conversation starters:** A text input field with a clear button (X).
- Knowledge:** A section with a note: "If you upload files under Knowledge, conversations with your GPT may include file contents. Files can be downloaded when Code Interpreter is enabled".

Preview Section (Right):

The preview shows a conversation between a user and the custom GPT.

User: write me summary of Gen ai principles in 5 lines

GenAI-Principles-GPT: The document outlines principles of Generative AI (GenAI), highlighting its capabilities in generating new content across various formats due to advances in machine learning and increased computing power. GenAI has reached human-level benchmarks in certain academic and professional areas and is significantly impacting business and operational processes, with rapid adoption rates compared to traditional technologies. It primarily focuses on large language models (LLMs) for text generation, but also acknowledges the commercial success of image- and code-based GenAI applications. Moreover, the document discusses the underlying mechanisms, including how GenAI leverages massive data sets and the role of models like Transformers in optimizing learning processes. The future of GenAI is seen in the broader application of these principles to create foundation models that can generalize across different tasks.

User: generate me a question paper consisting of 10 short questions and 5 multiple choice questions from this document

GenAI-Principles-GPT: (The response is partially visible in the input field)

GPT-4 is said to have over 100 trillion parameters; GPT-3.5 has 175 billion parameters

Copilot by Microsoft

Copilot (preview)

You

What is Generative AI? Explain by diagrams

Copilot

Generative AI, also known as generative models, refers to a class of artificial intelligence techniques that create new data instances by learning from existing data. These models generate novel content, such as images, text, or other forms of media, based on patterns they've learned from training data.

One popular type of generative model is the Generative Adversarial Network (GAN). GANs consist of two neural networks: a generator and a discriminator. The generator generates new data samples, while the discriminator evaluates whether the generated samples are real or fake. Through an adversarial training process, the generator improves its ability to create realistic data, and the discriminator becomes better at distinguishing real from generated data.

Here's a simple diagram illustrating the GAN architecture:

G R
Generator → Discriminator → Loss

1. The **generator** takes random noise as input and produces synthetic data (e.g., images).
2. The **discriminator** receives both real data (from the training dataset) and generated data (from the generator). It learns to classify them as either real or fake.

Ask me anything...

Copilot (preview)

2. The **discriminator** receives both real data (from the training dataset) and generated data (from the generator). It learns to classify them as either real or fake.
3. The **loss** function measures the difference between the discriminator's predictions and the ground truth labels (real or fake).

Other generative models include Variational Autoencoders (VAEs), which learn a probabilistic representation of data, and transformer-based language models like GPT (Generative Pre-trained Transformer), which generate coherent text.

Keep in mind that these diagrams are simplified; the actual architectures can be more complex. If you'd like to visualize these concepts further, tools like [DiagramGPT](#) or [Eraser's DiagramGPT](#) can automatically create professional diagrams from text descriptions ¹

Learn more

- 1 [deepgram.com](#)
- 2 [eraser.io](#)
- 3 [creately.com](#)
- 4 [diagrammingai.com](#)

Ask me anything...

Copilot (preview)

You

How do I train a generative model?

Copilot

Training a generative model involves several key steps. Let's break it down:


1. Define the Objective:
 - Clearly specify what you want the generative model to achieve. For example, do you want it to generate realistic images, text, or music?
2. Collect and Prepare Data:
 - Gather a large dataset in the domain of interest (e.g., images, sentences, sounds).
 - Clean and preprocess the data to ensure its quality.
3. Choose the Right Model Architecture:
 - Select an appropriate neural network architecture for your generative model.

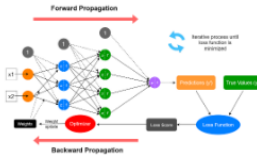
Ask me anything...


powered
by GPT-4,
OpenAI's
LLM

Gemini by Google

Gemini was just updated. [See update](#)

 Explain the given figure to me.



Show drafts  

✦ The figure you sent is a diagram that illustrates the concept of forward and backward propagation in artificial neural networks
https://en.wikipedia.org/wiki/Neural_network.

Here's a breakdown of the diagram:


Forward propagation

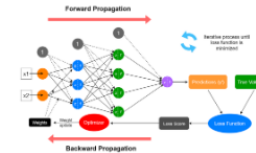
- This is represented by the left side of the diagram. It shows how information travels through the network from the input layer to the output layer.
- Here's how it works:
 - The input layer receives the input data (x1 and x2 in the figure).
 - This data is then multiplied by weights and passed through an activation function.



Enter a prompt here  

Gemini may display inaccurate info, including about people, so double-check its responses. [Your privacy & Gemini Apps](#)

Gemini was just updated. [See update](#)

 Explain the given figure to me.



Hide drafts  

Draft 1

The figure you sent is a diagram that illustrates the concept of forward and backward propagation in

Draft 2

The figure you sent is a diagram that illustrates the concept of forward and backward propagation in

Draft 3

The figure you sent is a diagram illustrating the concept of forward and backward propagation in





✦ The figure you sent is a diagram that illustrates the concept of forward and backward propagation in artificial neural networks
https://en.wikipedia.org/wiki/Neural_network.

Here's a breakdown of the diagram:

Forward propagation

- This is represented by the left side of the diagram. It shows how information travels through the network from the input layer to the output layer.

Enter a prompt here  

Gemini may display inaccurate info, including about people, so double-check its responses. [Your privacy & Gemini Apps](#)

GPT Vs Copilot Vs Gemini

Feature	GPT	Copilot	Gemini
Creator	OpenAI	Microsoft	Google
Architecture	Generative Pre trained Transformer	Generative Pre trained Transformer	Hybrid Model combining GPT and other architectures
Code Understanding	Limited	Intelligent	Intelligent
parameters	175B-1.76 trillion parameters	175B parameters	1.5 trillion parameters
Speed	Faster	Pianfully slow	Fast
Customization	Fine tuning for specific domains possible	-	Fine tuning for specific domains possible
Deployment	API integration into various platforms	-	API integration into various platforms

NLP Tasks

- Several NLP tasks break down human text and voice data in ways that help the computer make sense of what it's ingesting. Some of these tasks include the following:
- Speech Recognition
- Part of speech tagging
- Sentiment Analysis
- Natural Language Generation

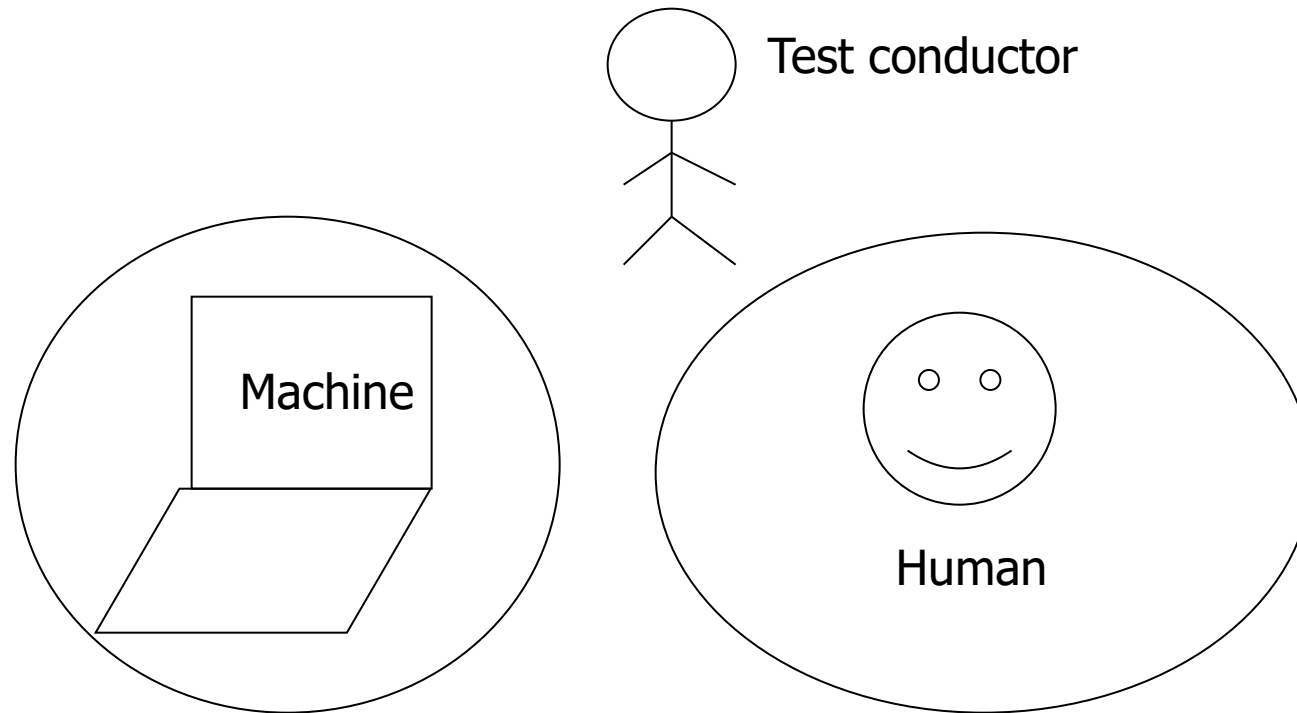
NLP Tools and Approaches

- From Hand-Coded Systems and rule based systems
- Advancement with Statistical NLP
- Assigning Statistical Likelihood to Meanings
- Deep Learning's Role
- Continuous Learning and Improved Extraction

NLP Use Cases

- Spam detection
- Machine translation:
- Virtual agents and chatbots
- Social media sentiment analysis
- Text summarization

The famous Turing Test: Language Based Interaction



Can the test conductor find out which is the machine and which the human

Ambiguity

This is what makes NLP challenging:

The Crux of the problem

Stages of language processing

- Phonetics and phonology
- Morphology
- Lexical Analysis
- Syntactic Analysis
- Semantic Analysis
- Pragmatics
- Discourse

Phonetics

- Processing of speech
- Challenges
 - Homophones: bank (finance) vs. bank (river bank)
 - Near Homophones: bear vs. bare
 - Word Boundary
 - aajaayenge (aa jaayenge (will come) or aaj aayenge (will come today)
 - I got [ua]plate
 - Phrase boundary
 - mtech1 students are especially exhorted to attend as such seminars are integral to one's post-graduate education
 - Disfluency: ah, um, ahem etc.

Morphology

- Word formation rules from root words
- Nouns: Plural (boy-boys); Gender marking (naseem-naseema)
- Verbs: Tense (stretch-stretched); Aspect (e.g. perfective sit-had sat); Modality (e.g. request khaanaa ->khaaiie)
- First crucial first step in NLP
- Languages rich in morphology: e.g., Dravidian, Hungarian, Turkish
- Languages poor in morphology: Chinese, English
- Languages with rich morphology have the advantage of easier processing at higher stages of processing
- A task of interest to computer science: Finite State Machines for Word Morphology

Lexical Analysis

- Essentially refers to dictionary access and obtaining the properties of the word
 - e.g. dog
 - noun (lexical property)
 - take-'s'-in-plural (morph property)
 - animate (semantic property)
 - 4-legged (-do-)
 - carnivore (-do)
- Challenge: Lexical or word sense disambiguation

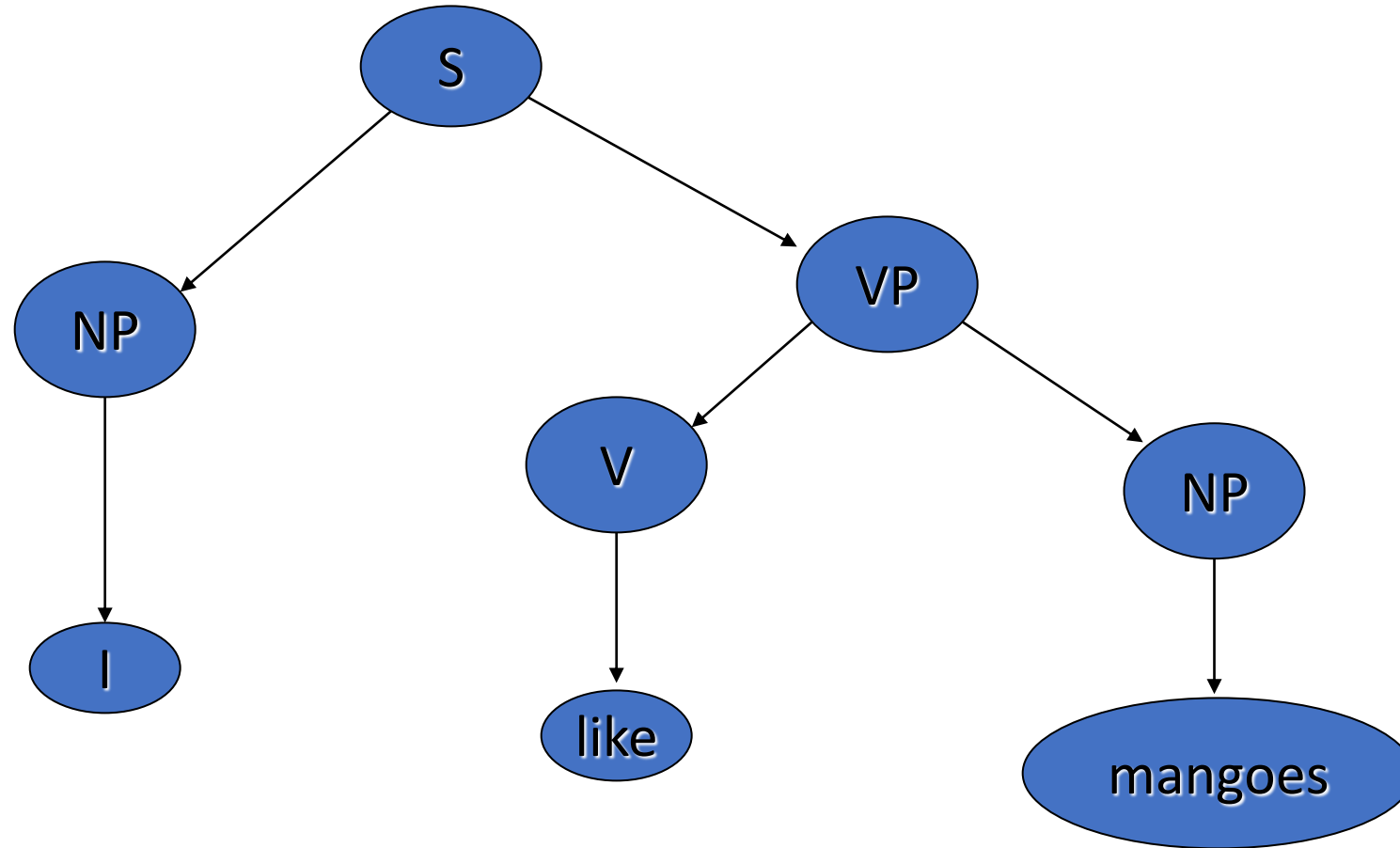
Lexical Disambiguation

- First step: part of Speech Disambiguation
- Dog as a noun (animal)
- Dog as a verb (to pursue, to follow closely)
- Sense Disambiguation
- Dog (as animal)
- Dog (as a very detestable person)
- Needs word relationships in a context
- The chair emphasised the need for adult education
- Very common in day to day communications
- Satellite Channel Ad: Watch what you want, when you want (two senses of watch)
- e.g., Ground breaking ceremony/research

Technological developments bring in new terms, additional meanings/nuances for existing terms

- Justify as in justify the right margin (word processing context)
- Xeroxed: a new verb
- Digital Trace: a new expression
- Communifaking: pretending to talk on mobile when you are actually not
- Discomgooglation: anxiety/discomfort at not being able to access internet
- Helicopter Parenting: over parenting

Syntax Processing Stage - Structure Detection



Challenges in Syntactic Processing: Structural Ambiguity

- Scope
 - 1. The old men and women were taken to safe locations
- (old men and women) vs. ((old men) and women)
 - 2. No smoking areas will allow Hookas inside
- Preposition Phrase Attachment
- I saw the boy with a telescope
- (who has the telescope?)
- I saw the mountain with a telescope
- (world knowledge: mountain cannot be an instrument of seeing)
- I saw the boy with the pony-tail
- (world knowledge: pony-tail cannot be an instrument of seeing)
- Very ubiquitous: newspaper headline "20 years later, BMC pays father 20 lakhs for causing son's death"

Structural Ambiguity...

- Overheard
- I did not know my PDA had a phone for 3 months
- An actual sentence in the newspaper
- The camera man shot the man with the gun when he was near Nadal
- (P.G. Wodehouse, Ring in Jeeves) Jill had rubbed ointment on Mike the Irish Terrier, taken a look at the goldfish belonging to the cook, which had caused anxiety in the kitchen by refusing its ant's eggs...
- (The Jung, 26/2/15) Aid for kins of cops killed in terrorist attacks

Higher level knowledge needed for disambiguation

- Semantics
 - ⊢ I saw the boy with a pony tail (pony tail cannot be an instrument of seeing)
- Pragmatics
 - ⊢ ((old men) and women) as opposed to (old men and women) in “Old men and women were taken to safe location”, since women- both and young and old- were very likely taken to safe locations
- Discourse:
 - ⊢ No smoking areas allow hookas inside, except the one in Hotel Grand.
 - ⊢ No smoking areas allow hookas inside, but not cigars.

Basic Text Processing

Regular expressions

- A formal language for specifying text strings
- How can we search for any of these?
- woodchuck
- woodchucks
- Woodchuck
- Woodchucks



Regular Expressions: Disjunctions

- Letters inside square brackets []

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

- Ranges [A-Z]

Pattern	Matches	
[A-Z]	An upper case letter	<u>D</u> renched Blossoms
[a-z]	A lower case letter	<u>m</u> y beans were impatient
[0-9]	A single digit	Chapter <u>1</u> : Down the Rabbit Hole

Regular Expressions: Negation in Disjunction

- Negations [^Ss]
 - Carat means negation only when first in []

Pattern	Matches	
[^A-Z]	Not an upper case letter	O <u>y</u> fn pripetchik
[^Ss]	Neither 'S' nor 's'	<u>I</u> have no exquisite reason"
[^e^]	Neither e nor ^	Look h <u>e</u> re
a^b	The pattern a carat b	Look up <u>a^b</u> now

Regular Expressions: More Disjunction

- Woodchuck is another name for groundhog!
- The pipe | for disjunction

Pattern	Matches
<code>groundhog woodchuck</code>	woodchuck
<code>yours mine</code>	yours
<code>a b c</code>	= <code>[abc]</code>
<code>[gG]roundhog [Ww]oodchuck</code>	Woodchuck



Regular Expressions: ? * +.

Pattern	Matches	
<code>colou?r</code>	Optional previous char	<u>color</u> <u>colour</u>
<code>oo*h!</code>	0 or more of previous char	<u>oh!</u> <u>ooh!</u> <u>oooh!</u> <u>ooooh!</u>
<code>o+h!</code>	1 or more of previous char	<u>oh!</u> <u>ooh!</u> <u>oooh!</u> <u>ooooh!</u>
<code>baa+</code>		<u>baa</u> <u>baaa</u> <u>baaaa</u> <u>baaaaa</u>
<code>beg.n</code>		<u>begin</u> <u>begun</u> <u>begun</u> <u>beg3n</u>



Stephen C Kleene

Kleene *, Kleene +

Regular Expressions: Anchors ^ \$

Pattern	Matches
<code>^[A-Z]</code>	<u>P</u> alo Alto
<code>^[^A-Za-z]</code>	<u>1</u> "Hello"
<code>\. \$</code>	The end <u>.</u>
<code>. \$</code>	The end <u>?</u> The end <u>!</u>

Example

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

→ the

→ Misses capitalized examples

→ [tT]he

→ Incorrectly returns other or theology

→ [^a-zA-Z][tT]he[^a-zA-Z]

→

Errors

- The process we just went through was based on fixing two kinds of errors:
- Matching strings that we should not have matched (there, then, other)
- False positives (Type I errors)
- Not matching things that we should have matched (The)
- False negatives (Type II errors)

Errors (Cont.)

- In NLP we are always dealing with these kinds of errors.
- Reducing the error rate for an application often involves two antagonistic efforts:
 - Increasing accuracy or precision (minimizing false positives)
 - Increasing coverage or recall (minimizing false negatives).

Summary

- Regular expressions play a surprisingly large role
 - Sophisticated sequences of regular expressions are often the first model for any text processing text
- For hard tasks, we use machine learning classifiers
 - But regular expressions are still used for pre-processing, or as features in the classifiers
 - Can be very useful in capturing generalizations

Words and Corpora

How many words in a sentence?

- "I do uh main- mainly business data processing"
 - ↳ Fragments, filled pauses
- "Seuss's cat in the hat is different from other cats!"
 - ↳ Lemma: same stem, part of speech, rough word sense
 - cat and cats = same lemma
 - ↳ Wordform: the full inflected surface form
 - cat and cats = different wordforms

How many words in a sentence?

- they lay back on the San Francisco grass and looked at the stars and their
- Type: an element of the vocabulary.
- Token: an instance of that type in running text.
- How many?
 - 15 tokens (or 14)
 - 13 types (or 12) (or 11?)

How many words in a corpus?

- N = number of tokens
- V = vocabulary = set of types, $|V|$ is size of vocabulary
- Heaps Law = Herdan's Law = $|V| = kN^\beta$ where often $.67 < \beta < .75$
- i.e., vocabulary size grows with $>$ square root of the number of word tokens

	Tokens = N	Types = $ V $
Switchboard phone conversations	2.4 million	20 thousand
Shakespeare	884,000	31 thousand
COCA	440 million	2 million
Google N-grams	1 trillion	13+ million

Corpora

- Words don't appear out of nowhere!
 - ↳ A text is produced by
 - ↳ a specific writer(s),
 - ↳ at a specific time,
 - ↳ in a specific variety,
 - ↳ of a specific language,
 - ↳ for a specific function.

Corpora vary along dimension like

- Language: 7097 languages in the world
- Variety, like African American Language varieties.
 - AAE Twitter posts might include forms like "iont" (I don't)
- Code switching, e.g., Spanish/English, Hindi/English:
 - S/E: Por primera vez veo a @username actually being hateful! It was beautiful:)
 - [For the first time I get to see @username actually being hateful! it was beautiful:)]
 - H/E: dost tha or ra- hega ... dont worry ... but dherya rakhe
 - ["he was and will remain a friend ... don't worry ... but have faith"]
- Genre: newswire, fiction, scientific articles, Wikipedia
- Author Demographics: writer's age, gender, ethnicity, SES

Corpus datasheets

- Motivation:
 - Why was the corpus collected?
 - By whom?
 - Who funded it?
- Situation: In what situation was the text written?
- Collection process: If it is a subsample how was it sampled? Was there consent? Pre-processing?
- +Annotation process, language variety, demographics, etc.

Word Tokenization

Text Normalization

- Every NLP task requires text normalization:
 - ↳ Tokenizing (segmenting) words
 - ↳ Normalizing word formats
 - ↳ Segmenting sentences

Space-based tokenization

- A very simple way to tokenize
 - ↳ For languages that use space characters between words
 - ↳ Arabic, Cyrillic, Greek, Latin, etc., based writing systems
 - ↳ Segment off a token between instances of spaces
- Unix tools for space-based tokenization
 - ↳ The "tr" command
 - ↳ Inspired by Ken Church's UNIX for Poets
 - ↳ Given a text file, output the word tokens and their frequencies

Simple Tokenization in UNIX

- (Inspired by Ken Church's UNIX for Poets.)
- Given a text file, output the word tokens and their frequencies

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

Change all non-alpha to newlines

Sort in alphabetical order

Merge and count each type

```
1945 A  
72 AARON  
19 ABBESS  
5 ABBOT  
... ..  
25 Aaron  
6 Abate  
1 Abates  
5 Abbess  
6 Abbey  
3 Abbot  
.... ..
```


The first step: tokenizing

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

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

The second step: sorting

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

A

A

A

A

A

A

A

A

A

...

More counting

- Merging upper and lower case

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

- Sorting the counts

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

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

What happened here?

Issues in Tokenization

- Can't just blindly remove punctuation:
 - m.p.h., Ph.D., AT&T, cap'n
 - prices (\$45.55)
 - dates (01/02/06)
 - URLs (<http://www.stanford.edu>)
 - hashtags ([#nlproc](#))
 - email addresses (someone@cs.colorado.edu)
- Clitic: a word that doesn't stand on its own
 - "are" in [we're](#), French "je" in [j'ai](#), "le" in [l'honneur](#)
- When should multiword expressions (MWE) be words?
 - [New York](#), [rock 'n' roll](#)

Tokenization in NLTK

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

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

Tokenization in languages without spaces

- Many languages (like Chinese, Japanese, Thai) don't use spaces to separate words!
- How do we decide where the token boundaries should be?

Word tokenization in Chinese

- Chinese words are composed of characters called "hanzi" (or sometimes just "zi")
- Each one represents a meaning unit called a morpheme.
- Each word has on average 2.4 of them.
- But deciding what counts as a word is complex and not agreed upon.

How to do word tokenization in Chinese?

-姚明进入总决赛 “Yao Ming reaches the finals”

-3 words?

-姚明 进入 总决赛

-YaoMing reaches finals

-5 words?

-姚 明 进入 总 决赛

-Yao Ming reaches overall finals

-7 characters? (don't use words at all):

-姚 明 进 入 总 决 赛

-Yao Ming enter enter overall decision game

Word tokenization / segmentation

- So in Chinese it's common to just treat each character (zi) as a token.
- So the segmentation step is very simple
- In other languages (like Thai and Japanese), more complex word segmentation is required.
- The standard algorithms are neural sequence models trained by supervised machine learning.

Byte Pair Encoding

Another option for text tokenization

- Instead of
 - white-space segmentation
 - single-character segmentation
- Use the **data** to tell us how to tokenize.
- **Subword tokenization** (because tokens can be parts of words as well as whole words)

Subword tokenization

- Three common algorithms:
- Byte-Pair Encoding (BPE) (Sennrich et al., 2016)
- Unigram language modeling tokenization (Kudo, 2018)
- WordPiece (Schuster and Nakajima, 2012)
- All have 2 parts:
- A token learner that takes a raw training corpus and induces a vocabulary (a set of tokens).
- A token segmenter that takes a raw test sentence and tokenizes it according to that vocabulary

Byte Pair Encoding (BPE) token learner

- Let vocabulary be the set of all individual characters
- $V = \{A, B, C, D, \dots, a, b, c, d, \dots\}$
- Repeat:
 - Choose the two symbols that are most frequently adjacent in the training corpus (say 'A', 'B')
 - Add a new merged symbol 'AB' to the vocabulary
 - Replace every adjacent 'A' 'B' in the corpus with 'AB'.
- Until k merges have been done.

BPE token learner algorithm

function BYTE-PAIR ENCODING(strings C , number of merges k) **returns** vocab V

$V \leftarrow$ all unique characters in C # initial set of tokens is characters

for $i = 1$ **to** k **do** # merge tokens til k times

$t_L, t_R \leftarrow$ Most frequent pair of adjacent tokens in C

$t_{NEW} \leftarrow t_L + t_R$ # make new token by concatenating

$V \leftarrow V + t_{NEW}$ # update the vocabulary

 Replace each occurrence of t_L, t_R in C with t_{NEW} # and update the corpus

return V

Byte Pair Encoding (BPE) Addendum

- Most subword algorithms are run inside space-separated tokens.
- So we commonly first add a special end-of-word symbol '___' before space in training corpus
- Next, separate into letters.

BPE token learner

Original (very fascinating 🤖) corpus:

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

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

corpus

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

vocabulary

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

BPE token learner

corpus

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

vocabulary

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

Merge **e r** to **er**

corpus

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

vocabulary

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

BPE

corpus

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

Merge **er _** to **er_**

corpus

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

vocabulary

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

vocabulary

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

BPE

corpus

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

Merge n e to ne

corpus

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

vocabulary

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

vocabulary

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

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_

BPE token segmenter algorithm

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

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

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

→ Result:

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

Properties of BPE tokens

- Usually include frequent words
- And frequent subwords
- Which are often morphemes like -est or -er
- A **morpheme** is the smallest meaning-bearing unit of a language
- unlikeliest has 3 morphemes un-, likely, and -est

Word Normalization and other issues

Word Normalization

- Putting words/tokens in a standard format
- U.S.A. or USA
- uhhuh or uh-huh
- Fed or fed
- am, is, be, are

Case folding

- Applications like IR: reduce all letters to lower case
 - † Since users tend to use lower case
 - † Possible exception: upper case in mid-sentence?
 - e.g., *General Motors*
 - *Fed* vs. *fed*
 - *SAIL* vs. *sail*
- For sentiment analysis, MT, Information extraction
 - † Case is helpful (*US* versus *us* is important)

Lemmatization

Represent all words as their lemma, their shared root
= dictionary headword form:

┆ *am, are, is* → *be*

┆ *car, cars, car's, cars'* → *car*

┆ Spanish *quiero* ('I want'), *quieres* ('you want')

→ *querer* 'want'

┆ *He is reading detective stories*

→ *He be read detective story*

Lemmatization is done by Morphological Parsing

– Morphemes:

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

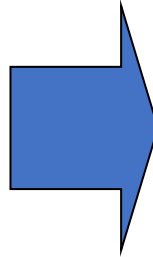
– Morphological Parsers:

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

Stemming

- Reduce terms to stems, chopping off affixes crudely

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



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

.

Porter Stemmer

- Based on a series of rewrite rules run in series
 - ┆ A cascade, in which output of each pass fed to next pass
- Some sample rules:

ATIONAL → ATE (e.g., relational → relate)

ING → ε if stem contains vowel (e.g., motoring → motor)

SSES → SS (e.g., grasses → grass)

Dealing with complex morphology is necessary for many languages

- † e.g., the Turkish word:
- † Uygarlastiramadiklarimizdanmissinizcasina
- † `(behaving) as if you are among those whom we could not civilize'
- † Uygar `civilized' + las `become'
 - + tir `cause' + ama `not able'
 - + dik `past' + lar `plural'
 - + imiz 'p1pl' + dan 'abl'
 - + mis 'past' + siniz '2pl' + casina 'as if'

Sentence Segmentation

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

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

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

- └ An abbreviation dictionary can help

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

