Natural Language Processing

IFT6758 - Data Science

Sources:

http://demo.clab.cs.cmu.edu/NLP/

http://u.cs.biu.ac.il/~89-680/

And many more …

Mila

Université de Montréal

Announcements

- HM#1 grades are accessible in Gradescope!
- All the mid-term evaluation scores are on the scoreboard!
- HM#2 grades will be out on Gradescope on Thursday!
- Mid-term presentation grades will be announced on Gradescope on Thursday!
- Mid-term exam grades will be announced next Monday!
- Crash course on DL will be next Thursday!

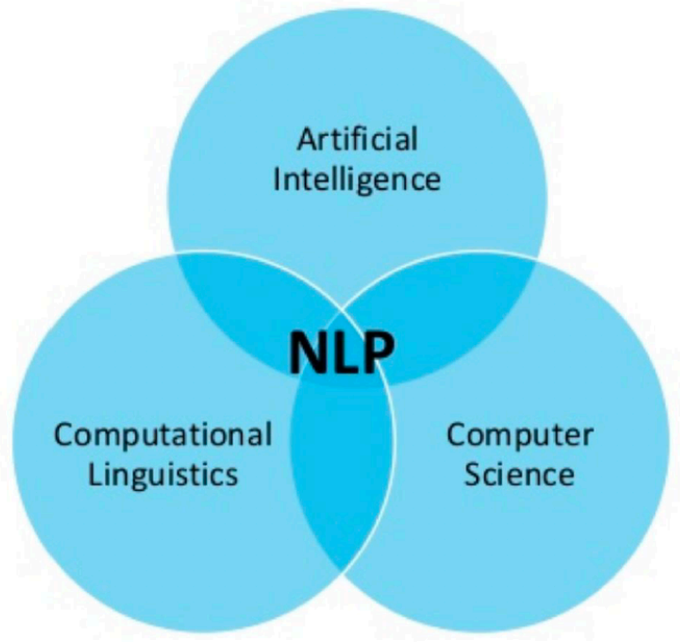
What is Natural Language Processing (NLP)?

- Automating the **analysis**, **generation**, and **acquisition** of human ("natural") language

    - **Analysis** (or "understanding" or "processing" ...) : input is language, output is some **representation** that supports useful action

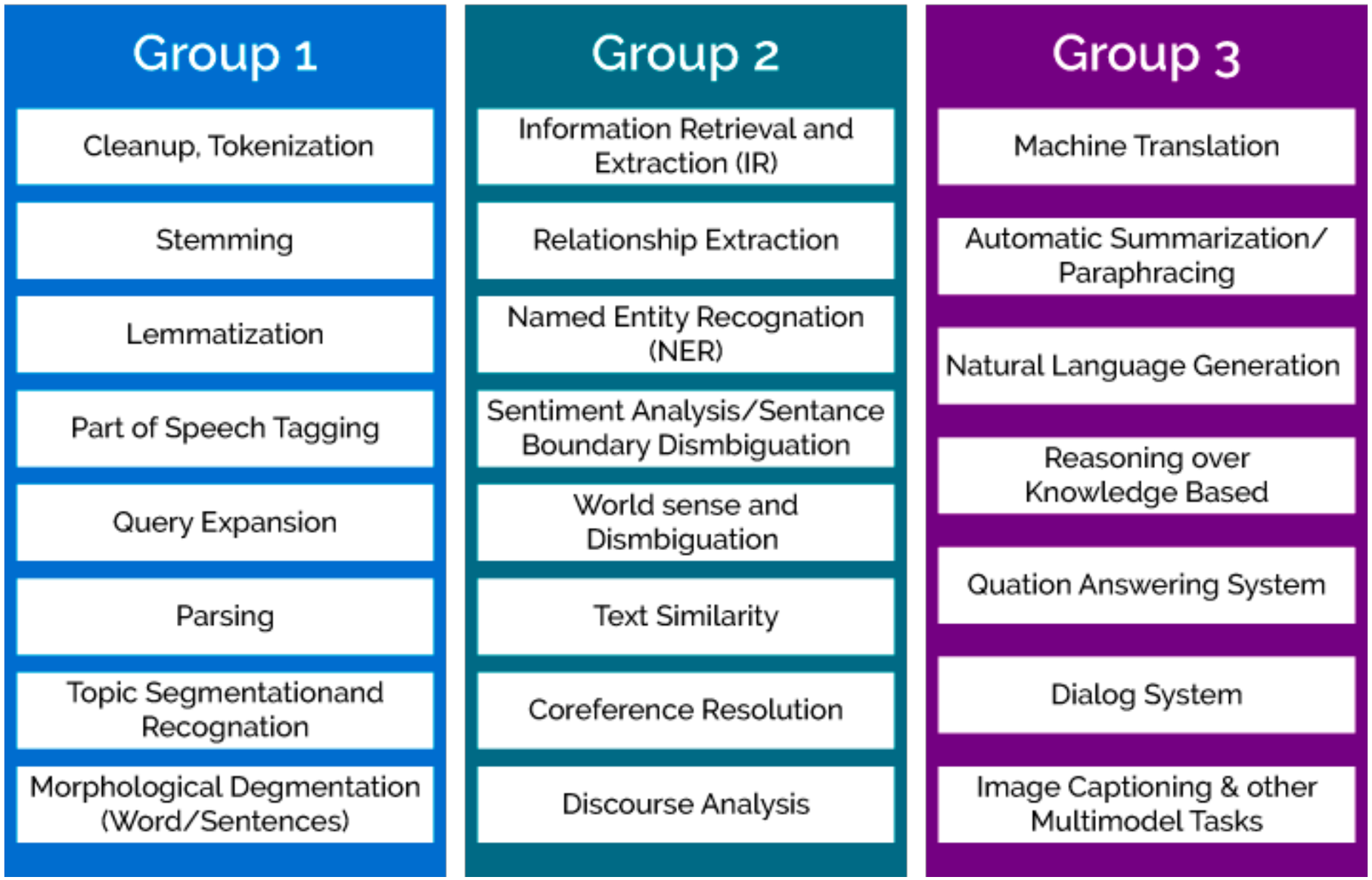
    - **Generation**: input is that **representation**, output is language

    - **Acquisition**: obtaining the **representation** and necessary **algorithms**, from knowledge and data

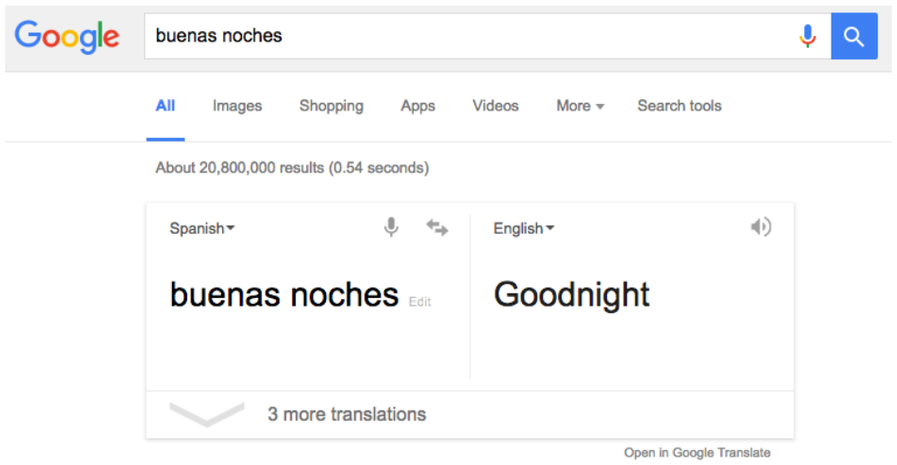


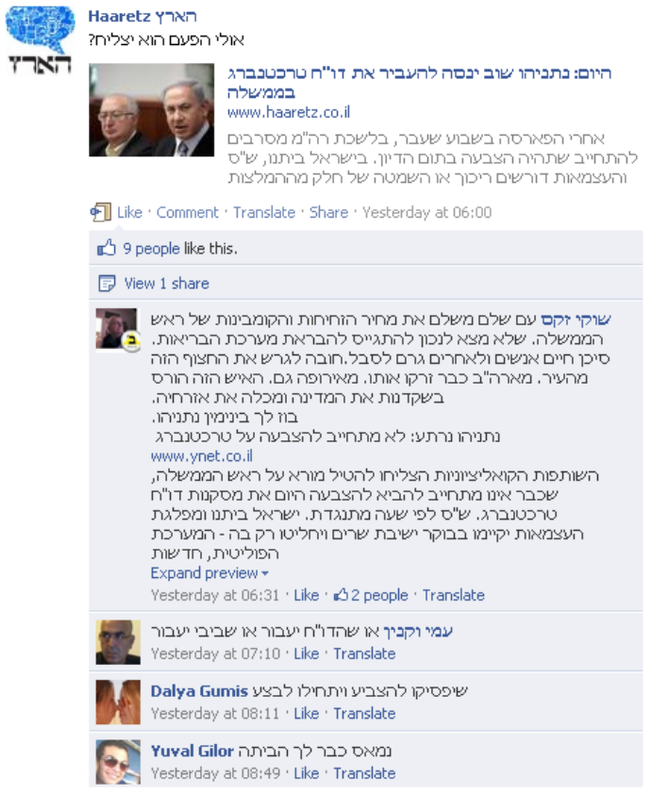
Applications of NLP

Supervised NLP Unsupervised NLP

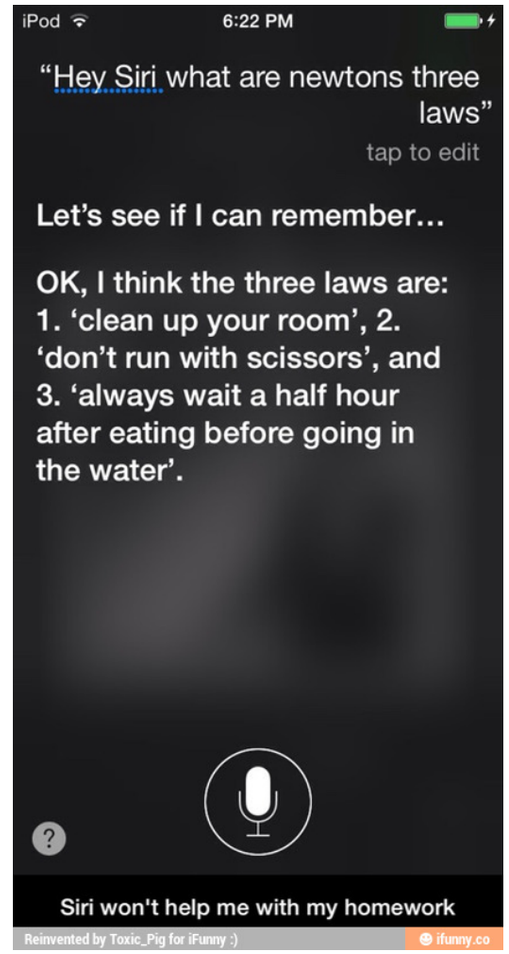


Machine translation

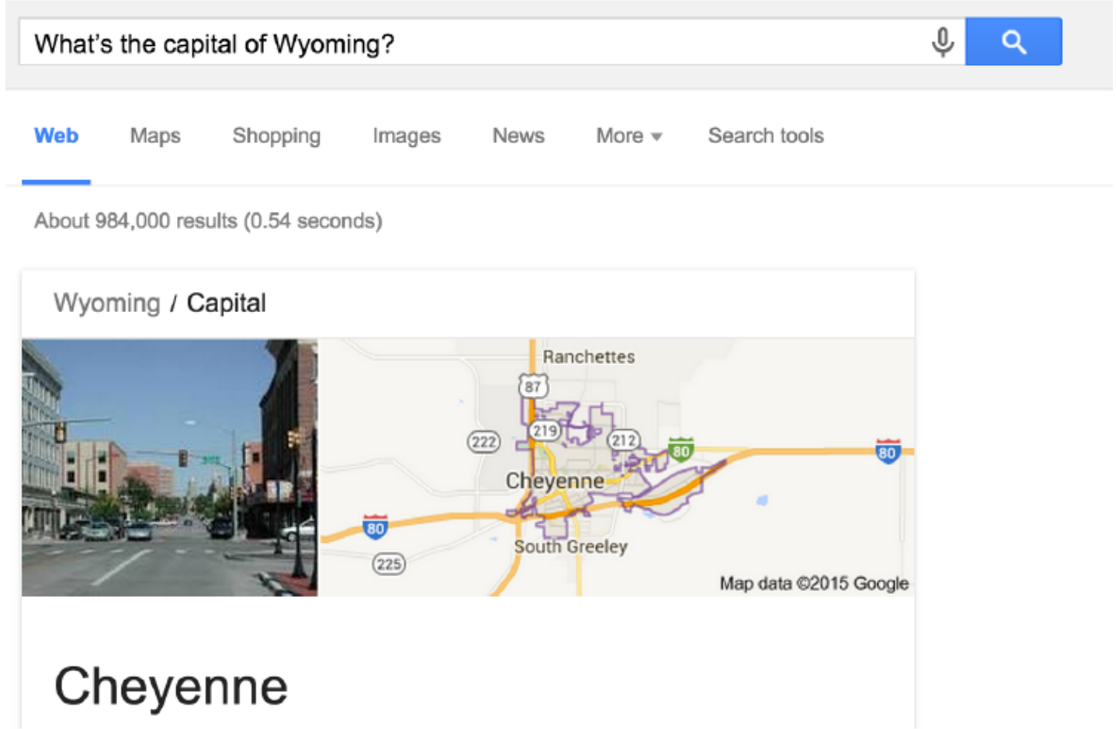




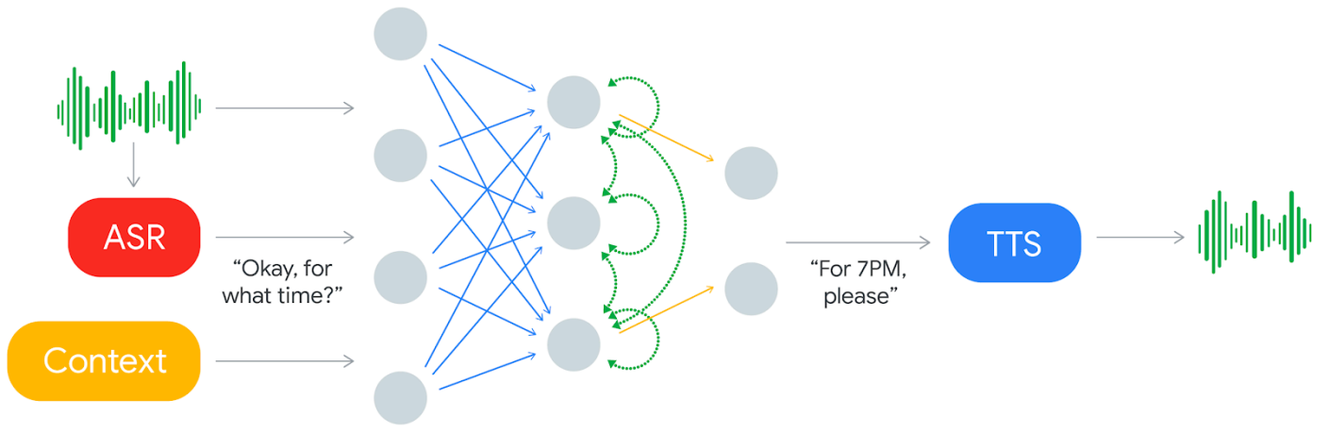
Question answering

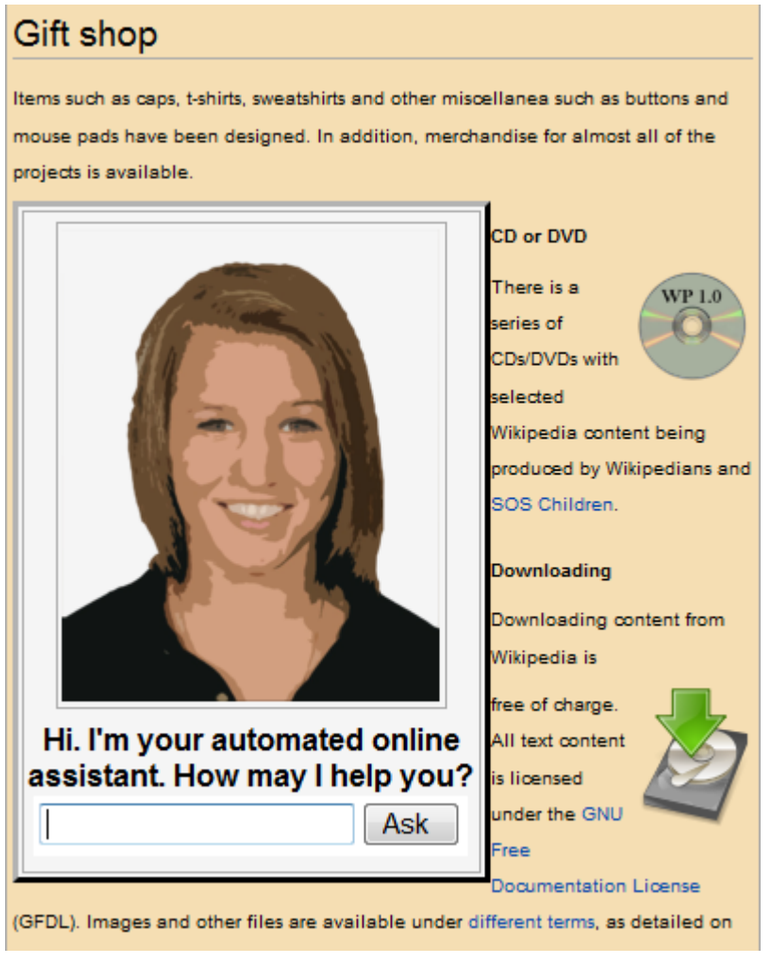


credit: ifunny.com

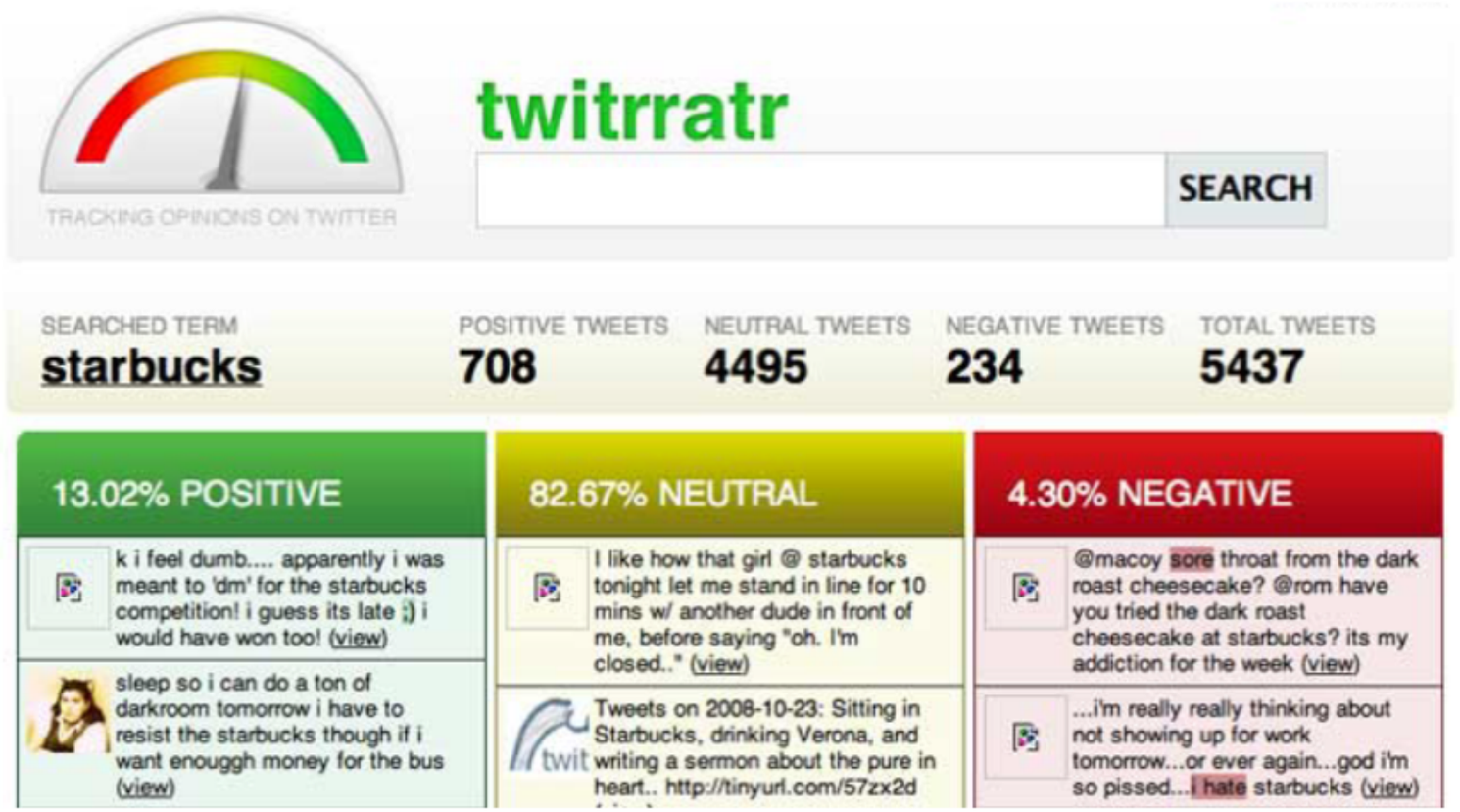


Dialog system

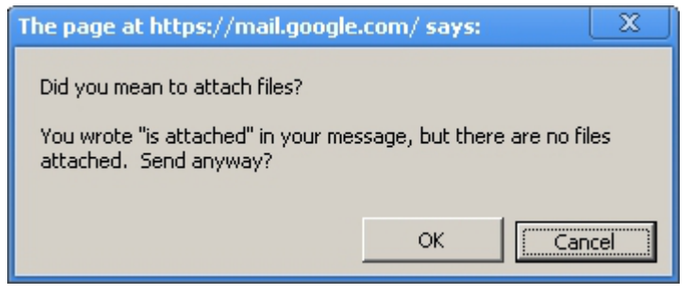


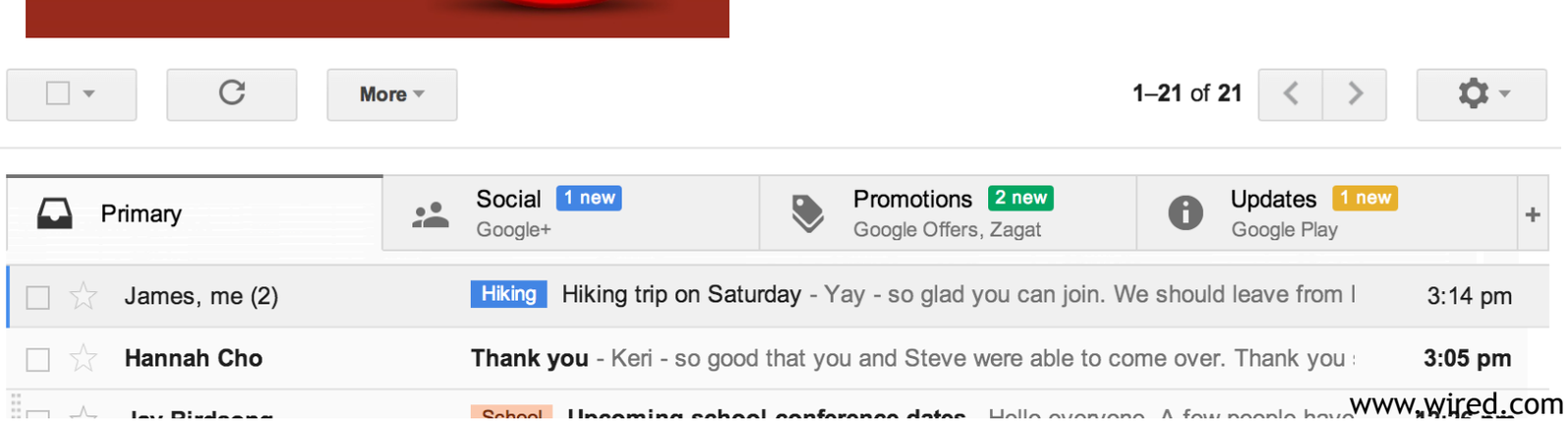


Sentiment/opinion Analysis



Text classification





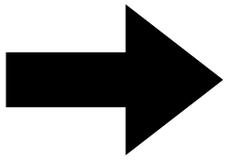
www.wired.com

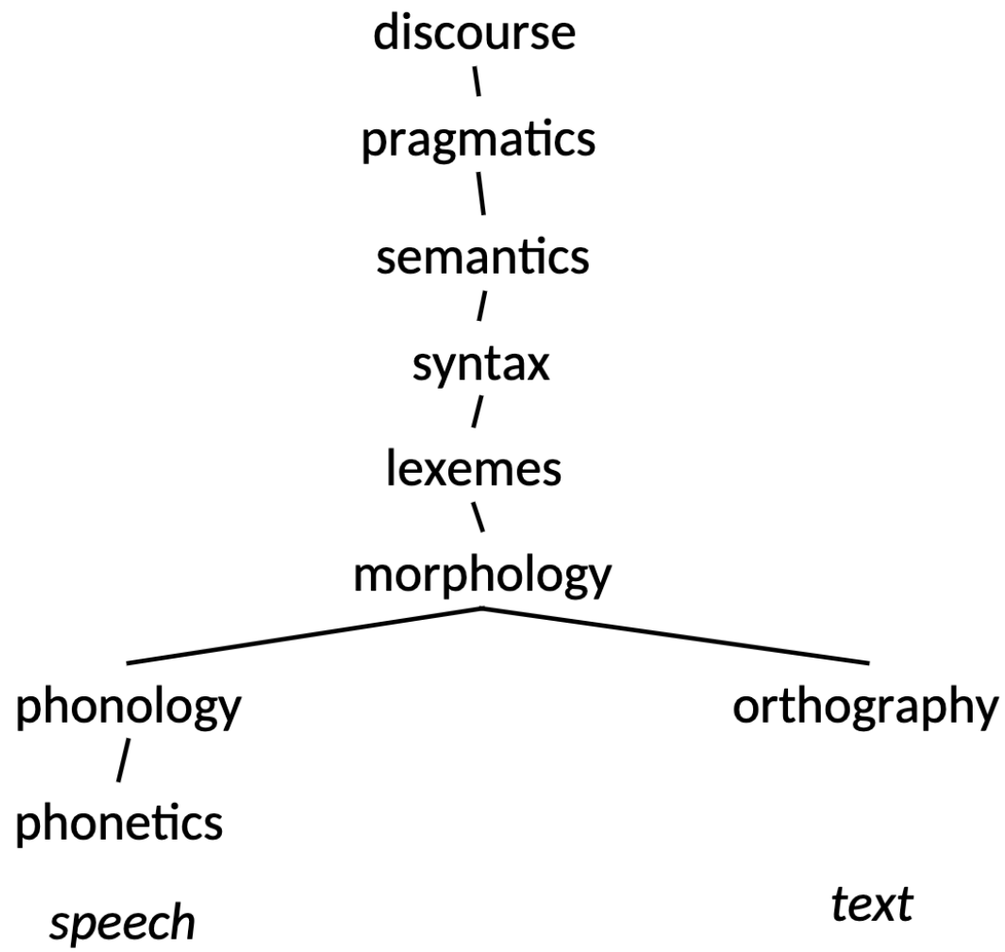
NLP Key tasks

NLP language model

Why NLP is hard?

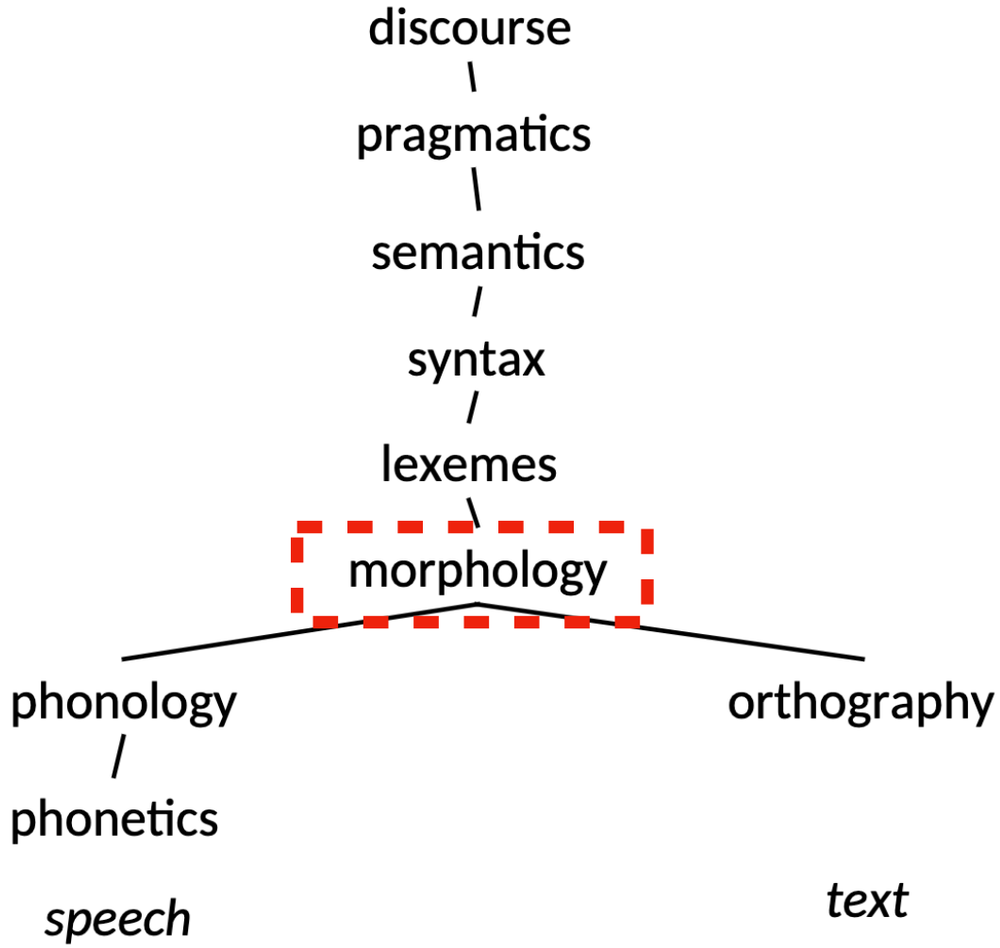
Representation





- The mappings between levels are extremely complex.
- Appropriateness of a representation depends on the application.

Why NLP is hard?



- **Morphology**: Analysis of words into meaningful components

- Spectrum of complexity across languages:

- **Analytic** or **isolating** languages (e.g., English, Chinese)

- Synthetic languages (e.g., Finnish, Turkish, Hebrew)

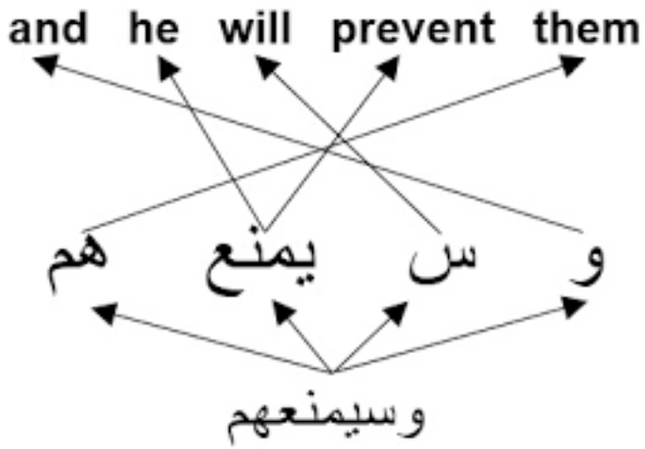
TIFGOSH ET HAYELED BAGAN
"you will meet the boy in the park"

Puedes dármelo
"You can give it to me"

uygarlaştıramadıklarımızdanmışsınızcasına
"(behaving) as if you are among those whom we could not civilize"

Tokenizing

- The most common task in NLP is tokenization.
- **Tokenization** is the process of breaking a document down into words, punctuation marks, numeric digits, etc.



A single morpheme

물 새 집

(water) (bird) (house)

Multiple morphemes

돼지고기

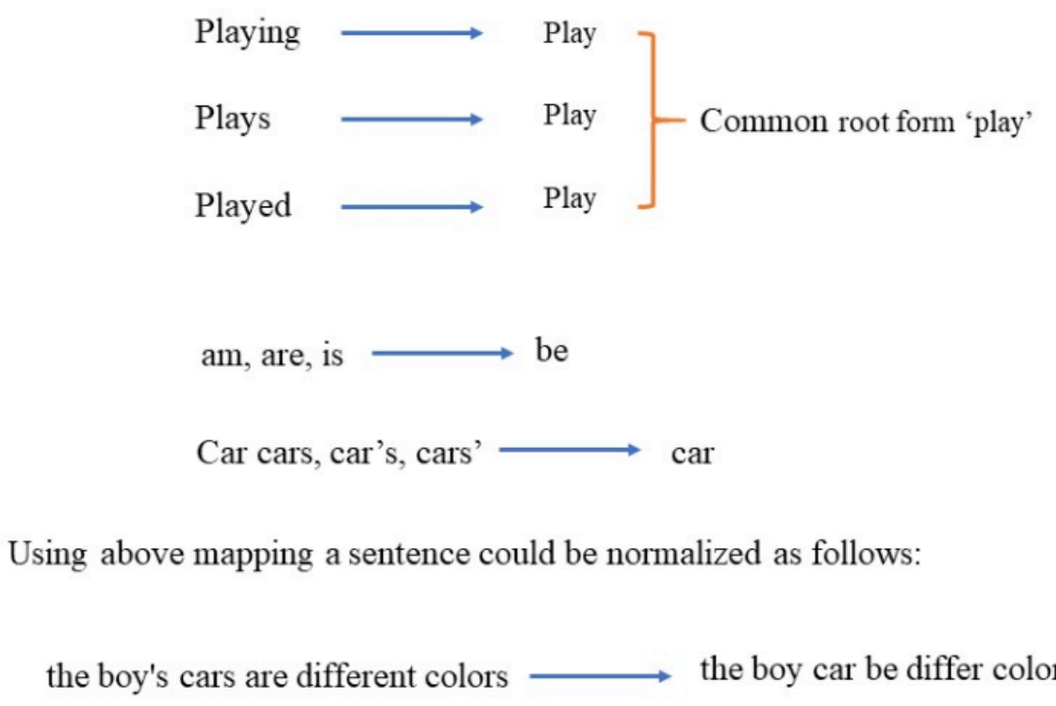
(Pork, 돼지+고기, pig+meat)

꽃병

(vase, 꽃+병, flower+bottle)

Text Normalization

- **Stemming and Lemmatization** are **Text Normalization** (or sometimes called **Word Normalization**) techniques in the field of Natural Language Processing that are used to prepare text, words, and documents for further processing.



- Languages we speak and write are made up of several words often derived from one another.

- When a language contains words that are derived from another word as their use in the speech changes is called **Inflected Language**. The degree of inflection may be higher or lower in a language.

Stemming

- **Stem (root) is the part of the word to which you add inflectional (changing/deriving) affixes such as (-ed,-ize, -s,-de,mis).** So stemming a word or sentence may result in words that are not actual words. Stems are created by removing the suffixes or prefixes used with a word.

- E.g.,

CONNECT
CONNECTIONS------> CONNECT
CONNECTED------> CONNECT
CONNECTING------> CONNECT
CONNECTION------> CONNECT

- Python NLTK provides not only two English stemmers: **PorterStemmer** and **LancasterStemmer** but also a lot of non-English stemmers as part of **SnowballStemmers, ISRIStemmer, RSLPSStemmer**.

Lemmatization

- **Lemmatization**, unlike Stemming, reduces the inflected words properly ensuring that the root word belongs to the language. In Lemmatization root word is called **Lemma**. A lemma (plural lemmas or lemmata) is the canonical form, dictionary form, or citation form of a set of words.

- e.g., *runs, running, ran* are all forms of the word *run*, therefore *run* is the lemma of all these words.

- Because lemmatization returns an actual word of the language, it is used where it is necessary to get valid words.

Stemming vs. Lemmatization

- You may be asking yourself when should I use Stemming and when should I use Lemmatization?

  - Stemming and Lemmatization both generate the root form of the inflected words. The difference is that stem might **not be an actual word** whereas, lemma is an **actual language word**.

  - Stemming follows an algorithm with steps to perform on the words which makes it faster. Whereas, in lemmatization, you used WordNet corpus and a corpus for stop words as well to produce lemma which makes it slower than stemming. You also had to define a parts-of-speech to obtain the correct lemma.

Example (Tokenizing and Stemming)

```
In [1]: import nltk

In [2]: nltk.download(["punkt", "averaged_perceptron_tagger", "treebank",
                       "maxent_ne_chunker", "words", "wordnet"])
```

...

```
In [3]: text = open("data/clean/asoiaf01.txt").read()

In [4]: sentences = nltk.tokenize.sent_tokenize(text)

In [5]: sentences[5313]
Out[5]: 'Cersei Lannister regarded him suspiciously.'

In [6]: sentences = [nltk.tokenize.word_tokenize(sentence) for sentence in sentences]

In [7]: sentences[5313]
Out[7]: ['Cersei', 'Lannister', 'regarded', 'him', 'suspiciously', '.']

In [8]: stemmer = nltk.stem.SnowballStemmer(language="english")
        [stemmer.stem(word) for word in sentences[5313]]
Out[8]: ['cersei', 'lannist', 'regard', 'him', 'suspici', '.']
```

Stop words

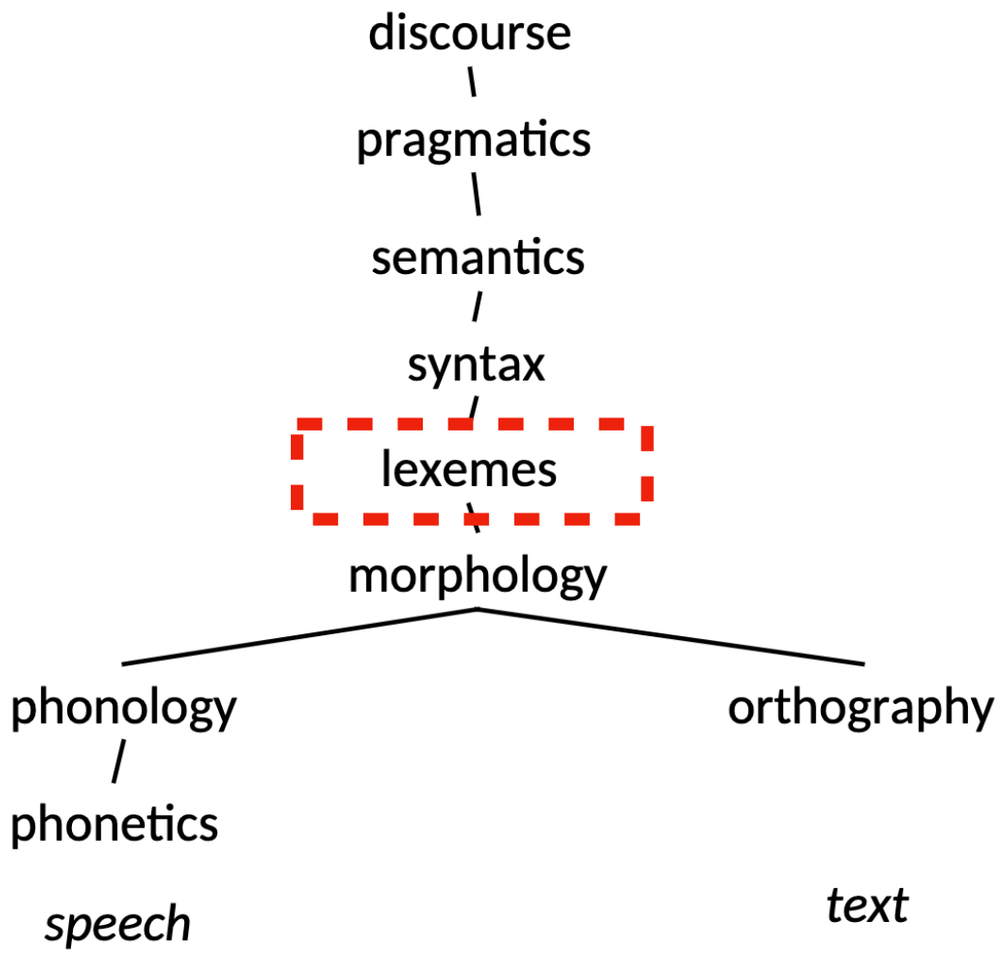
- **Stop words**: Stop Words are words which do not contain important significance to be used. They are very common **words** in a language (e.g. a, an, the in English. 的, 了in Chinese. え, もin Japanese). It does not help on most of **NLP** problem such as semantic analysis, classification etc.

- Usually, these words are filtered out from text because they return a vast amount of unnecessary information.

- Each language will give its own list of stop words to use. e.g, words that are commonly used in the English language are as, the, be, are.

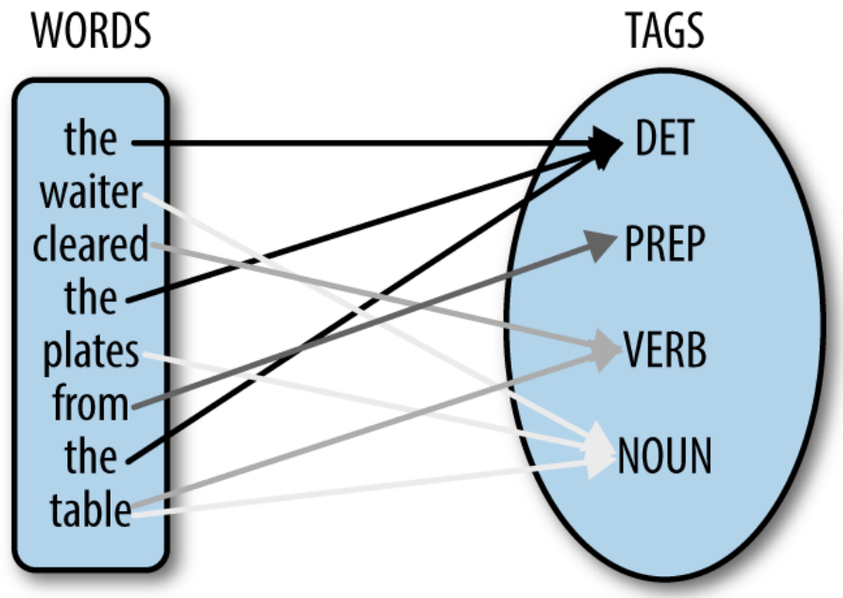
- In NLTK, you can use pre-defined stop words, or you can use other words which are defined by other party such as Stanford NLP and Rank NL.
  Stanford NLP: https://github.com/stanfordnlp/CoreNLP/blob/master/data/edu/stanford/nlp/patterns/surface/stopwords.txt
  Rank NL: https://www.ranks.nl/stopwords
  jieba: https://github.com/fxsjy/jieba/blob/master/extra_dict/stop_words.txt

Why NLP is hard?



- **Lexemes**: Normalize and disambiguate words
- Words with **multiple meanings**: bank, mean (Extra challenge: domain-specific meanings)
- **Multi-word expressions**: e.g., make ... decision, take out, make up, …
- For English, **part-of-speech tagging** is one very common kind of lexical analysis
- For Others: super-sense tagging, various forms of word sense disambiguation, syntactic “super tags,” …

Part of Speech Tagging



- "**Part-of-speech tagging** (**POS tagging** or **PoS tagging** or **POST**), also called **grammatical tagging** or **word-category disambiguation**, is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech, based on both its definition and its context — i.e., its relationship with adjacent and related words in a phrase, sentence, or paragraph. "

- A simplified form of this is commonly taught to school-age children, in the identification of words as nouns, verbs, adjectives, adverbs, etc.

POS algorithms

- POS-tagging algorithms fall into two distinctive groups:
  Rule-Based POS Taggers
  Stochastic POS Taggers

- Automatic part of speech tagging is an area of natural language processing where statistical techniques have been more successful than rule-based methods.

- Example of a rule: If an ambiguous/unknown word X is preceded by a determiner and followed by a noun, tag it as an adjective.

- Defining a set of rules manually is an extremely cumbersome process and is not scalable at all. So we need some automatic way of doing this.

- Automatic approaches: **The Brill's tagger** is a rule-based tagger that goes through the training data and finds out the set of tagging rules that best define the data and minimize POS tagging errors. **Hidden Markov Model** can be used as a simple stochastic POS tagger approach.

NER

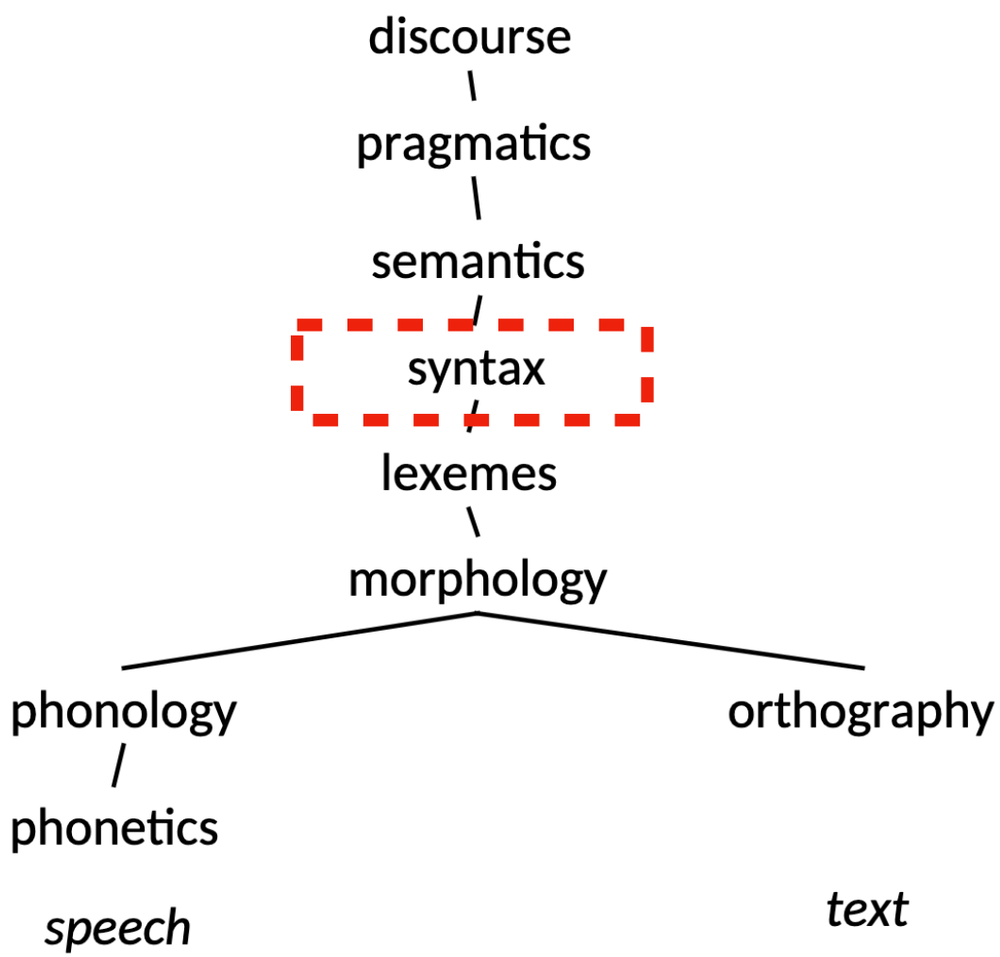
- Named entity recognition (NER) (also known as entity identification (EI) and entity extraction) is the task that locate and classify atomic elements in text into predefined categories such as the names of **persons**, **organizations**, **locations**, **expressions of times**, **quantities**, **monetary values**, **percentages**, etc.

<ENAMEX TYPE="PERSON">*John****</ENAMEX>*** *sold* ***<NUMEX TYPE="QUANTITY">****5****</NUMEX>*** *companies in* ***<TIMEX TYPE="DATE">****2002****</TIMEX>****.*

Example (POS/NER tagging)

```
In [10]: tagged_sentence = nltk.pos_tag(sentences[5313])

In [11]: tagged_sentence

Out[11]: [('Cersei', 'NNP'),
          ('Lannister', 'NNP'),
          ('regarded', 'VBD'),
          ('him', 'PRP'),
          ('suspiciously', 'RB'),
          ('.', '.')]

In [12]: lemmatizer = nltk.stem.wordnet.WordNetLemmatizer()
         [lemmatizer.lemmatize(word, pos=penn_to_wn(tag)) for word, tag in tagged_sentence]

Out[12]: ['Cersei', 'Lannister', 'regard', 'him', 'suspiciously', '.']

In [13]: tree = nltk.ne_chunk(tagged_sentence)

In [14]: tree.draw()
```

Why NLP is hard?



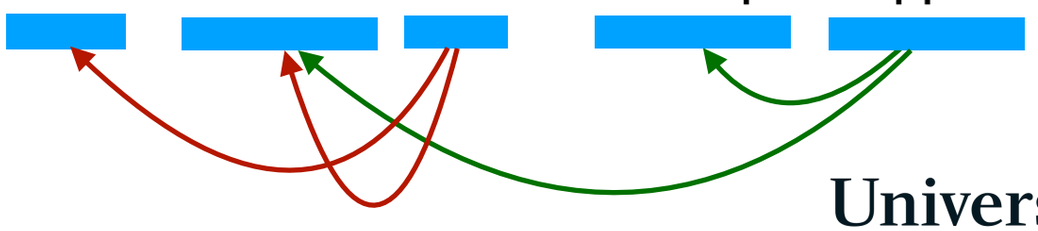
- **Syntax**: Transform a sequence of symbols into a hierarchical or compositional structure.

- Closely related to linguistic theories about what makes some sentences well-formed and others not.

✓ I want a flight to Tokyo
✓ I want to fly to Tokyo
✓ I found a flight to Tokyo
✸ I found to fly to Tokyo

- Ambiguities explode combinatorially

John saw the woman with the telescope wrapped in paper.

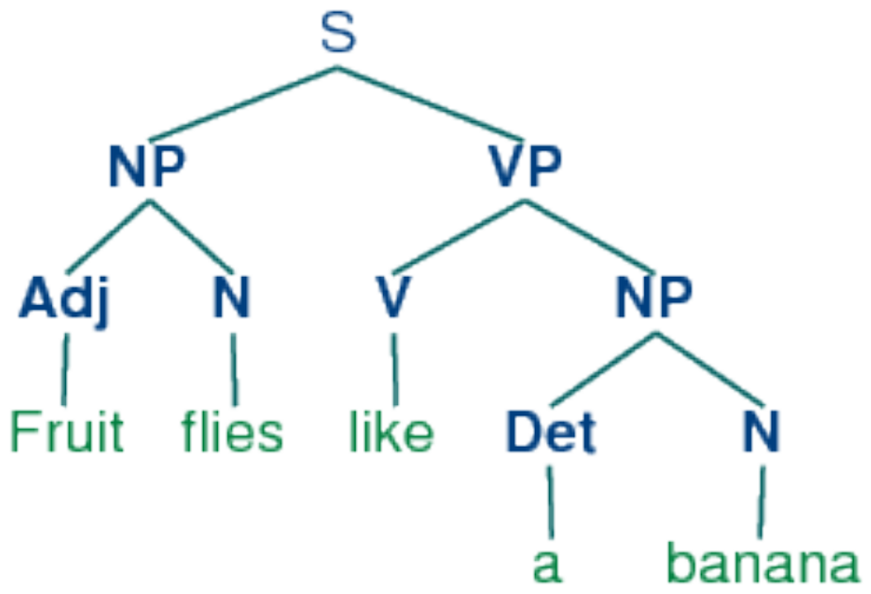


Synthetic parsing (Constituency)

- Constituent-based grammars are used to analyze and determine the constituents of a sentence.

- These grammars can be used to model or represent the internal structure of sentences in terms of a **hierarchically ordered structure** of their constituents. Each and every word usually belongs to a specific lexical category in the case and forms the head word of different phrases. These phrases are formed based on rules called ***phrase structure rules***.

<S> ::= <NP> <VP>
<NP> ::= <ART> <NOUN>
<VP> ::= <VERB> <NP>

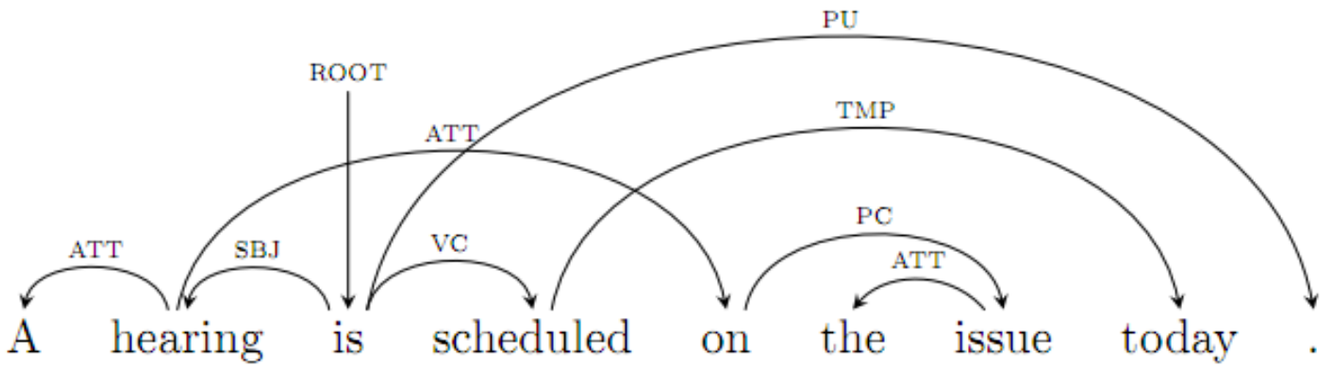


Dependency parsing

- In dependency parsing, we try to use dependency-based grammars to analyze and infer *both* ***structure and semantic dependencies*** and relationships between tokens in a sentence.

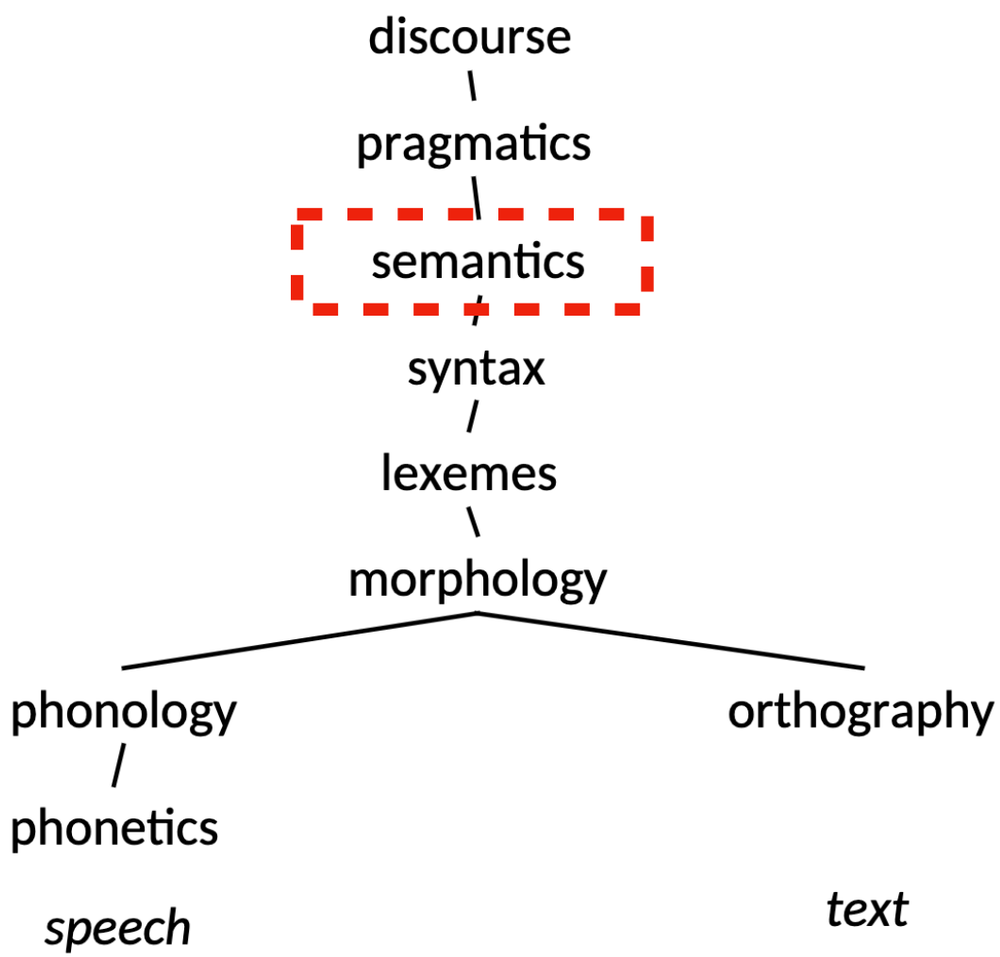
- The word that has no dependency is called the **root** of the sentence.

- The verb is taken as the root of the sentence in most cases. All the other words are directly or indirectly linked to the root verb using links, which are the dependencies.



https://nlp.stanford.edu/software/dependencies_manual.pdf

Why NLP is hard?



- **Semantics**: Mapping of natural language sentences into domain representations.

- E.g., a robot command language, a database query, or an expression in a formal logic.

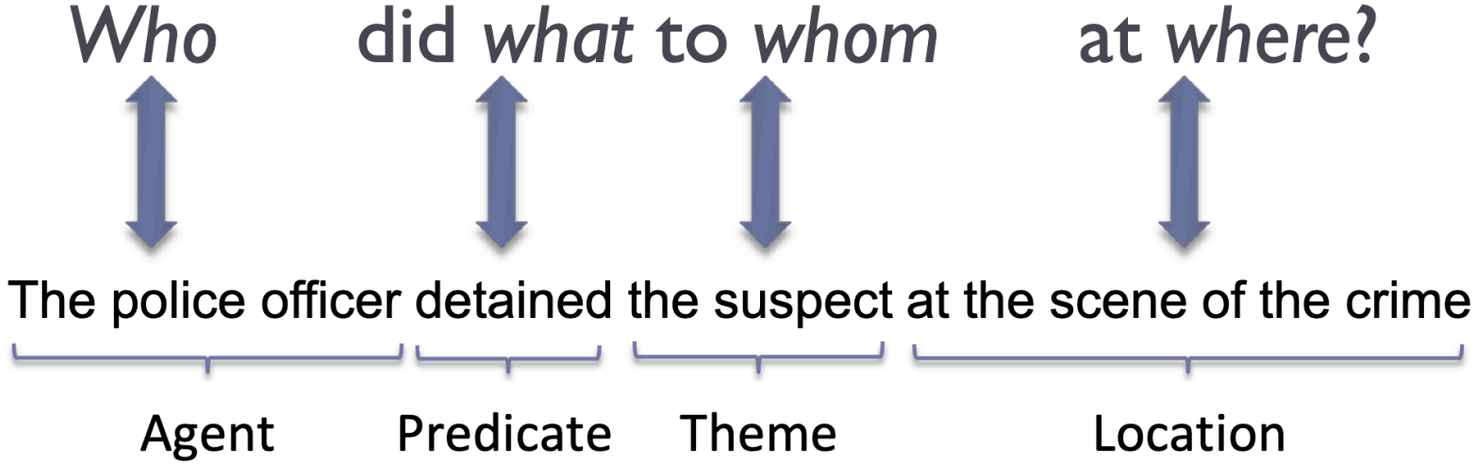
- **Scope ambiguities:**

  a. Everyone loves **SOMEone**.
  b. **EVERYone** loves someone.

- Going beyond specific domains is a goal of Artificial Intelligence

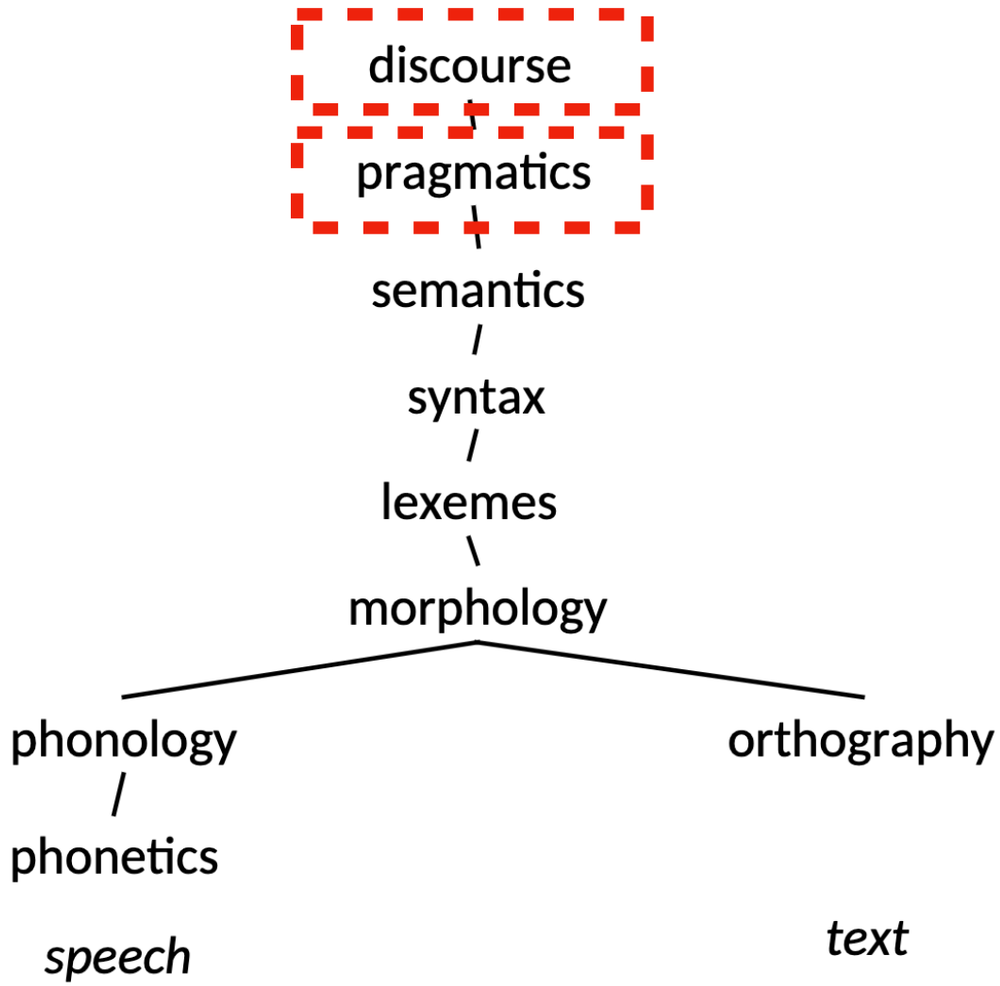
Semantic analysis

- Semantic role labeling is the process that assigns **labels** to words or phrases in a sentence that indicate their **semantic role** in the sentence, such as that of an agent, goal, or result.



- The suspect was detained by the police officer at the crime scene
- The police officer detained the suspect at the scene of the crime.

Why NLP is hard?



- **Pragmatics**: Study the ways in which context contributes to meaning

- Any non-local meaning phenomena:

  "Is he 21?" "Yes, he's 25."

- **Discourse**:

  - Structures and effects in related sequences of sentences

  - Texts, dialogues, multi-party conversations

    "I said the ***black*** shoes."

    "Oh, ***black***." (Is that a sentence?)

Co-reference resolution

- **Coreference resolution** is the task of finding all expressions that refer to the same entity in a text.
- It is an important step for a lot of higher level NLP tasks that involve natural language **understanding** such as document summarization, question answering, and information extraction.

Christopher Robin is alive and well. **He** is the same person that you read about in the book, **Winnie the Pooh**. As a **boy**, **Chris** lived in a pretty home called **Cotchfield Farm.** When **Chris** was three years old, **his father** wrote a poem about **him**. The poem was printed in a magazine for others to read. **Mr. Robin** then wrote a book

Complexity of NLP

- Linguistic representations are theorized constructs -> **We cannot observe them directly**.

- **Data**: Input is likely to be **noisy**.

- **Ambiguity**: each string may have many possible interpretations at every level. The correct resolution of the ambiguity will depend on the intended meaning, which is often inferable from context.

    - People are good at linguistic ambiguity resolution but computers are not so good at it:
      Q : How do we represent sets of possible alternatives?
      Q : How do we represent context?

- **Variablity**: there are many ways to express the same meaning, and immeasurably many meanings to express.

    - Lots of words/phrases. Each level interacts with the others. There is tremendous diversity in human languages. Languages express the same kind of meaning in different ways. Some languages express some meanings more readily/often

NLP challenges

- Data issues:
    - **A lot of data**: In some cases, we deal with huge amounts of data
      Need to come up with models that can process a lot of data efficiently
    - **Lack of data:** Many problems in NLP suffer from lack of data:
        - Non-standard platforms (code-switching)
        - Expensive annotation (word-sense disambiguation, named-entity recognition)
- Need to use methods to overcome this challenge (semi-supervised learning, multi-task learning…)

NLP challenges

- Ambiguity challenge: **Polysemy**: one word, many meanings

Book

Verb: **Book** a flight

Noun: He says it's a very good **book**

Bank

The edge of a river: He was strolling near the river **bank**

A financial institution: He works at the **bank**

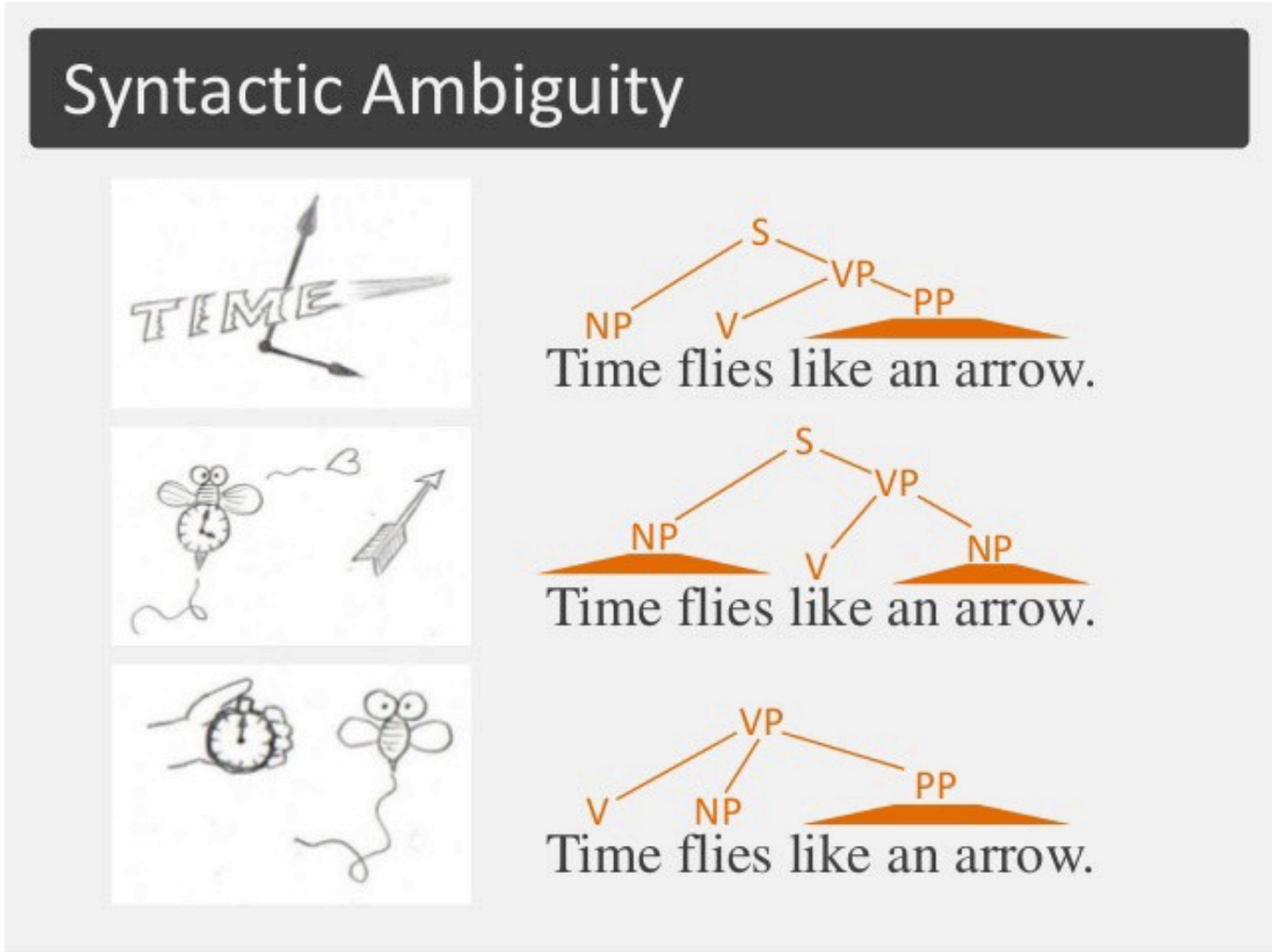
Solution

An answer to a problem: Work out the **solution** in your head

From Chemistry: Heat the **solution** to 75° Celsius

NLP challenges

- Ambiguity challenge: **Syntactic Ambiguity**: same words, many meanings



NLP challenges

- **Variability:** different words, same meaning

They allowed him to…

They let him …

He was allowed to…

He was permitted to…

NLP framework

NLP language model

Language Understanding

- Its all about how likely a sentence is...

$P(\text{obama}|\text{president of U.S.})$

$P(\text{Good morning}|\text{Buenos días})$

$P(\text{I saw a bus}) \gg$
$P(\text{eyes awe a boss})$

$P(\text{a man eating a sandwich})$

Probabilistic Language Models

- **Goal**: Compute the probability of a sentence or sequences of words

$$P(W) = P(w_1, w_2, w_3, w_4, w_5 \ldots w_n)$$

- Related task: probability of an upcoming word:

$$P(w_5 \mid w_1, w_2, w_3, w_4)$$

- A model that computes either of the above is called a **language model**.

Language Understanding

A sentence $(x_1, x_2, \ldots, x_T)$

- Ex: (the, cat, is, eating, a, sandwich, on, a, couch)

How likely is this sentence?

In other words, what is the probability of $(x_1, x_2, \ldots, x_T)$?

- i.e: $P(x_1, x_2, \ldots, x_T) = ?$

Recall
(Probability 101)

- Joint probability $p(x, y)$
- Conditional probability $p(x|y)$
- Marginal probability $p(x)$ and $p(y)$

- They are related by $p(x, y) = p(x|y)p(y) = p(y|x)p(x)$

Recall
(Probability 101)

- Joint probability $p(x, y)$
- Conditional probability $p(x|y)$
- Marginal probability $p(x)$ and $p(y)$

- They are related by $p(x, y) = p(x|y)p(y) = p(y|x)p(x)$

Chain Rule:

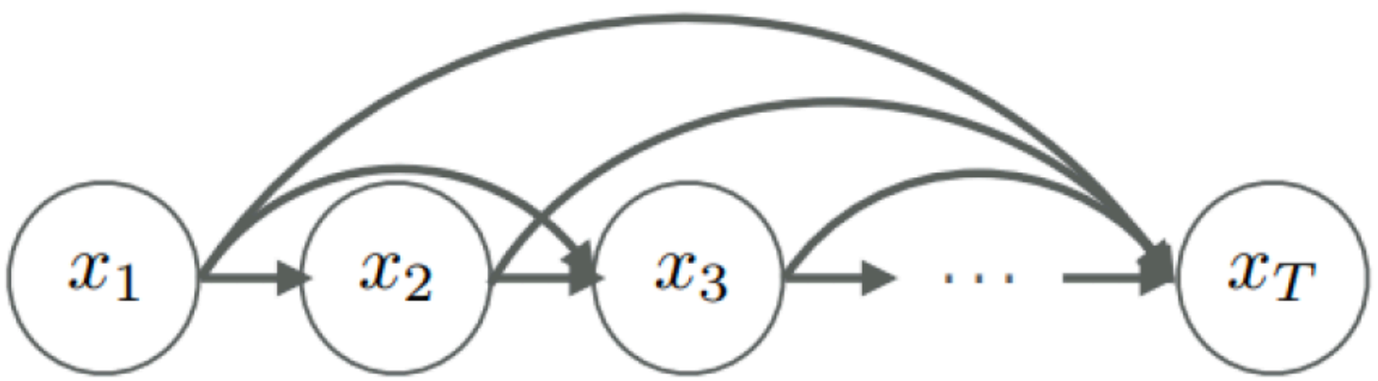
$$P(x_1,x_2,x_3,\ldots,x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1,x_2)\ldots P(x_n|x_1,\ldots,x_{n-1})$$

Probabilistic Language Models

- Rewrite $p(x_1, x_2, \ldots, x_T)$ into

$$p(x_1, x_2, \ldots, x_T) = \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-1})$$

- Graphically,



N-gram Language Model

Use Markov assumption: Next word does not depend on all previous words, but only on last n words:

$$
\begin{aligned}
P(x_1, x_2, \ldots, x_T) &= \prod_{t=1}^{T} p(x_t | x_1, \ldots, x_{t-1}) \\
&\approx \prod_{t=1}^{T} p(x_t | x_{\boldsymbol{t-n}}, \ldots, x_{t-1})
\end{aligned}
$$

How to calculate such probabilities? Just counting:

For Unigram: $P(x_t | x_{t-1}) \frac{\text{count}(x_{t-1}, x_t)}{\text{count}(x_{t-1})}$

N-gram Example

For Unigram:

$$P(x_t|x_{t-1}) \frac{\text{count}(x_{t-1}, x_t)}{\text{count}(x_{t-1})}$$

<s> I am sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>

$P(\texttt{I}\,|\,\texttt{<s>}) = \frac{2}{3} = .67$ $P(\texttt{Sam}\,|\,\texttt{<s>}) = \frac{1}{3} = .33$ $P(\texttt{am}\,|\,\texttt{I}) = \frac{2}{3} = .67$

$P(\texttt{</s>}\,|\,\texttt{Sam}) = \frac{1}{2} = 0.5$ $P(\texttt{Sam}\,|\,\texttt{am}) = \frac{1}{2} = .5$ $P(\texttt{do}\,|\,\texttt{I}) = \frac{1}{3} = .33$

Effects of n in performance

- Ex) $p(\text{i}, \text{would}, \text{like}, \text{to}, \ldots, ., \langle/\text{s}\rangle)$
- Unigram Modelling

  $p(\text{i})p(\text{would})p(\text{like}) \cdots p(\langle/\text{s}\rangle)$
- Bigram Modelling

  $p(\text{i})p(\text{would}|\text{i})p(\text{like}|\text{would}) \cdots p(\langle/\text{s}\rangle\,|.)$
- Trigram Modelling

  $p(\text{i})p(\text{would}|\text{i})p(\text{like}|\text{i}, \text{would}) \cdots$

  ⋮

word	unigram	bigram	trigram	4-gram
i	6.684	3.197	3.197	3.197
would	8.342	2.884	2.791	2.791
like	9.129	2.026	1.031	1.290
to	5.081	0.402	0.144	0.113
commend	15.487	12.335	8.794	8.633
the	3.885	1.402	1.084	0.880
rapporteur	10.840	7.319	2.763	2.350
on	6.765	4.140	4.150	1.862
his	10.678	7.316	2.367	1.978
work	9.993	4.816	3.498	2.394
.	4.896	3.020	1.785	1.510
</s>	4.828	0.005	0.000	0.000
average	8.051	4.072	2.634	2.251
perplexity	265.136	16.817	6.206	4.758

Evaluation: How good is a language model?

- **Extrinsic evaluation:** Test a trained model on a test collection
  - Try to predict each word
  - The more precisely a model can predict the words, the better is the model

- **Intrinsic evaluation**: **Perplexity**
  Perplexity is the inverse probability of the test set, normalized by the number of words:
  - Given $P(w_i)$ and a test text of length N
  - The lower the better!

$$Perplexity = 2^{-\frac{1}{N}\sum_{i=1}^{N}\log_2 P(w_i)}$$

Parametric Language Model

~~Non-parametric estimator~~ $\longrightarrow$ parametric estimator

$$
\begin{aligned}
P\left(x_t | x_{t-n}, \ldots, x_{t-1}\right) &= \frac{\cancel{\text{count}\left(x_{t-n}, \ldots, x_t\right)}}{\cancel{\text{count}\left(x_{t-1}, \ldots, x_{t-1}\right)}} \\
&= f_\Theta\left(x_{t-n}, \ldots, x_{t-1}\right)
\end{aligned}
$$

Parametric Language Model

~~Non-parametric estimator~~ $\longrightarrow$ parametric estimator

$$
\begin{aligned}
P\left(x_t | x_{t-n}, \ldots, x_{t-1}\right) &= \frac{\cancel{\text{count}\left(x_{t-n}, \ldots, x_t\right)}}{\cancel{\text{count}\left(x_{t-1}, \ldots, x_{t-1}\right)}} \\
&= f_\Theta\left(x_{t-n}, \ldots, x_{t-1}\right)
\end{aligned}
$$

Somehow, we need numerical representation for words... i.e. **Word vectors**

Embedding

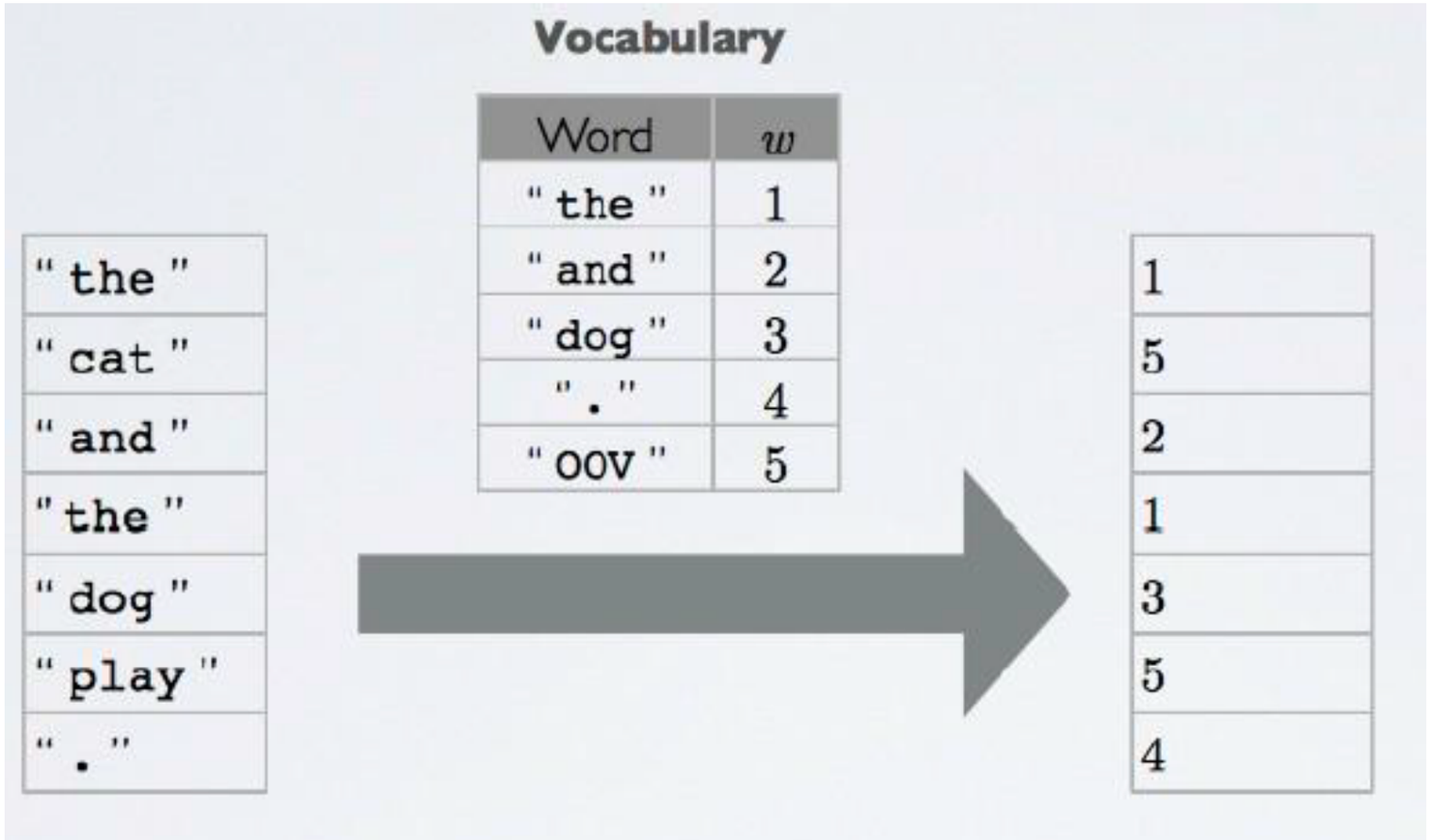
- **Embedding**: Mapping between space with one dimension per linguistic unit (character, morpheme, word, phrase, paragraph, sentence, document) to a continuous vector space with much lower dimension.

- Vector representation: “meaning” of linguistic unit is represented by a vector of real numbers.

$$\text{good} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

One hot encoding

- Naive and simple word embedding technique: Map each word to a unique ID



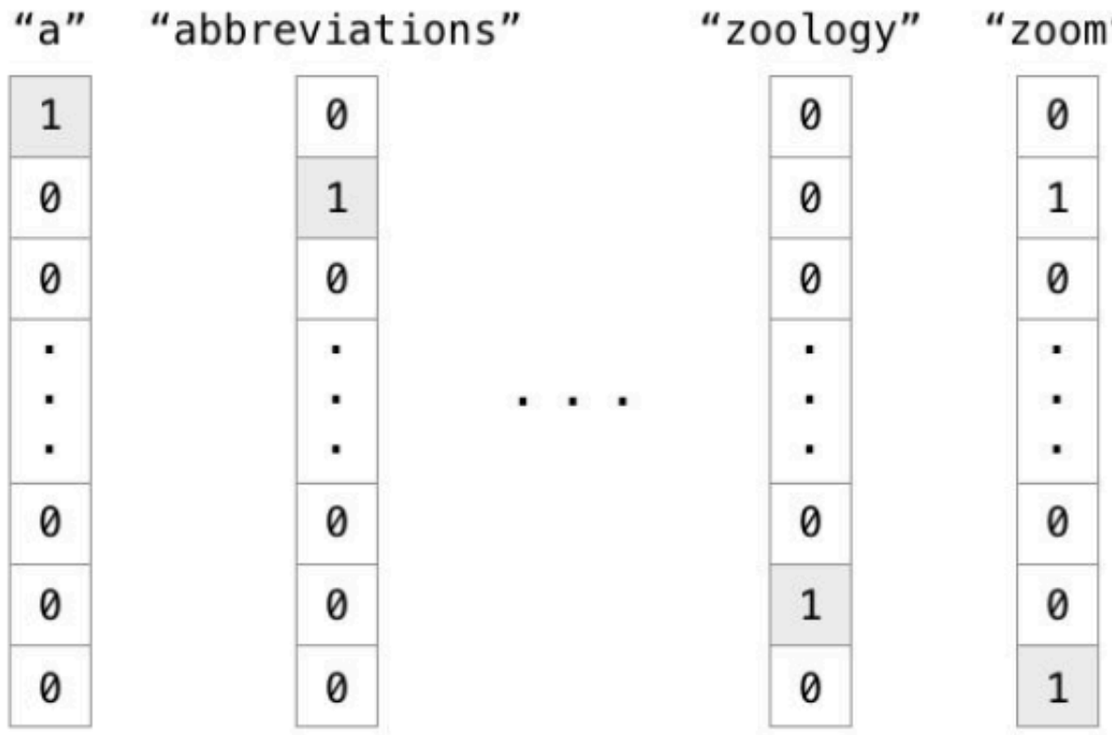
- Typical vocabulary sizes will vary between 10k and 250k.

One hot encoding

- Use word ID, to get a **basic representation** of word via one-hot encoding of the ID

- one-hot vector of an ID is a vector filled with **0**s, except for a **1** at the position associated with the ID. e.g., for vocabulary size D=10, the one-hot vector of word (w) ID=4 is
e(w) = [0 0 0 1 0 0 0 0 0 0]

- a one-hot encoding makes **no assumption about word similarity** and all words are equally similar/different from each other



One hot encoding

- **Pros**: Quick and simple

- **Cons**: Size of input vector scales with size of vocabulary. Must pre-determine vocabulary size.
  - Cannot scale to large or infinite vocabularies (Zipf's law!)
  - Computationally expensive - large input vector results in far too many parameters to learn.
  - "**Out-of-Vocabulary**" (OOV) problem: How would you handle unseen words in the test set? (One solution is to have an "UNKNOWN" symbol that represents low-frequency or unseen words)
  - No relationship between words: Each word is an independent unit vector

D("cat", "refrigerator") = D("cat", "cats")
D("spoon", "knife") = D("spoon", "dog")

Bag of words

- **Bag of words (BOW):** is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity.

- **Vocabulary:** set of all the words in corpus
  Documents: Words in document w.r.t vocab with multiplicity

Sentence 1: "The cat sat on the hat"
Sentence 2: "The dog ate the cat and the hat"

Vocab = { the, cat, sat, on, hat, dog, ate, and }

Sentence 1: { 2, 1, 1, 1, 1, 0, 0, 0 }
Sentence 2 : { 3, 1, 0, 0, 1, 1, 1, 1}

Bag of words

- **Pros**:
  - Quick and simple
- **Cons**:
  - Bag of words model is very **high-dimensional** and **sparse**
  - **Cannot** capture **semantics** or **morphology**
  - **Orderless**

TF-IDF

- Raw counts are problematic: frequent words will characterize most words -> not informative

- Importance increases proportionally to the number of times a word appears in the document; but is inversely proportional to the frequency of the word in the corpus.

- **TF-IDF** (for term (t) – document (d)):

$$TF - IDF(t,d) = \frac{count(t,d)}{|d|} \cdot \log \frac{|D|}{|\{d \in D : t \in d\}|}$$

D – set of all documents

TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).
IDF(t) = log (Total number of documents / Number of documents with term t in it).

TF-IDF

- Document D1 contains 100 words.
- cat appears 3 times in D1
- **TF(cat) = 3 / 100 = 0.3**
- Corpus contains 10 million documents
- cat appears in 1000 documents
- **IDF(cat) = log (10,000,000 / 1,000) = 4**
- **TF-IDF (cat) = 0.3 * 4**

TF-IDF

- **Pros**:
  - Easy to compute
  - Has some basic metric to extract the most descriptive terms in a document
  - Thus, can easily compute the similarity between 2 documents using it, e.g., using cosine similarity
- **Cons**:
  - Based on the bag-of-words (BoW) model, therefore it does not capture position in text, semantics, co-occurrences in different documents, etc. TF-IDF is only useful as a lexical level feature.
  - Cannot capture semantics (unlike word embeddings)
  - Orderless

N-gram

- Vocab = set of all n-grams in corpus
  Document = n-grams in document w.r.t vocab with multiplicity

For bigram:

Sentence 1: "The cat sat on the hat"

Sentence 2: "The dog ate the cat and the hat”

Vocab = { the cat, cat sat, sat on, on the, the hat, the dog, dog ate, ate the, cat and, and the}

Sentence 1: { 1, 1, 1, 1, 1, 0, 0, 0, 0, 0}
Sentence 2 : { 1, 0, 0, 0, 0, 1, 1, 1, 1, 1}

N-gram

- **Pros**:
  - Incorporates order of words
  - Simple and quick
- **Cons**:
  - Very large vocabulary
  - Data sparsity

Data sparsity

Data sparsity:# of all possible n-grams: $|V|^n$, where $|V|$ is the size of the vocabulary. Most of them never occur.

Training Set:

- ... denied the allegations
- ... denied the reports
- ... denied the claims
- ... denied the request

Test Set:

- ... denied the offer
- ... denied the loan

$$P(\text{offer}|\text{denied the}) = 0$$

N-gram

- **Pros**:
  - Incorporates order of words
  - Simple and quick
- **Cons**:
  - Very large vocabulary
  - Data sparsity
  - False independence assumption

False independence assumption

False independence assumption: Because in an n-gram language model we assume that each word is only conditioned on the previous n-1 words

False conditional independence assumption

"The dogs chasing the cat bark". The tri-gram probability $P(\text{bark}|\text{the cat})$ is very low (not observed in the corpus by the model, because the cat never barks and the plural verb "bark" has appeared after singular noun "cat"), but the whole sentence totally makes sentence.

N-gram

- **Pros**:
  - Incorporates order of words
  - Simple and quick
- **Cons**:
  - Very large vocabulary
  - Data sparsity
  - False independence assumption
  - **Cannot** capture **syntactic/semantic similarity**

The Distributional Hypothesis

- The **Distributional Hypothesis**: words that occur in the same contexts tend to have similar meanings (Harris, 1954)

- “You shall know a word by the company it keeps” (Firth, 1957)

The Distributional Hypothesis

Distributional Semantics (*Count*)

- Used since the 90's
- Sparse word-context PMI/PPMI matrix
- Decomposed with SVD

Word Embeddings (*Predict*)

- Inspired by deep learning
- `word2vec` *(Mikolov et al., 2013)*
- GloVe *(Pennington et al., 2014)*

Underlying Theory: **The Distributional Hypothesis** *(Harris, '54; Firth, '57)*

"Similar words occur in similar contexts"

Learning Dense embeddings

Matrix Factorization

Factorize word-context matrix.

	$Context_1$	$Context_1$	....	Context k
$Word_1$				
$Word_2$				
⋮				
$Word_n$				

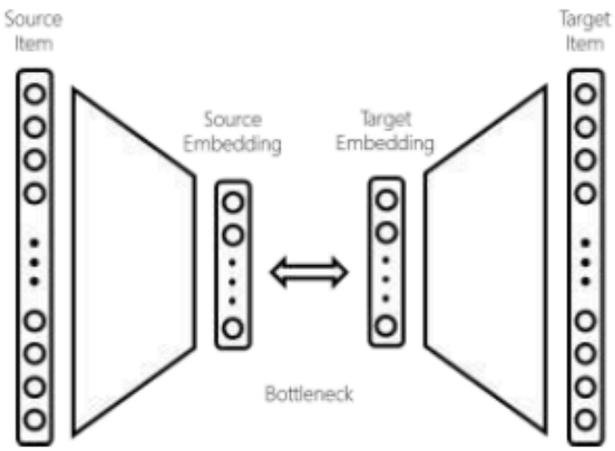
E.g.,

LDA (Word-Document),

GloVe (Word-NeighboringWord)

Neural Networks

A neural network with a bottleneck, word and context as input and output respectively.



E.g.,

Word2vec (Word-NeighboringWord)

Deerwester, Dumais, Landauer, Furnas, and Harshman, Indexing by latent semantic analysis, JASIS, 1990.
Pennington, Socher, and Manning, GloVe: Global Vectors for Word Representation, EMNLP, 2014.
Mikolov, Sutskever, Chen, Corrado, and Dean, Distributed representations of words and phrases and their compositionality, NIPS, 2013.