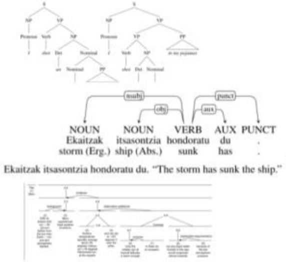
香港中文大學（深圳）
The Chinese University of Hong Kong, Shenzhen

CSC5051/MDS5110/CSC6052 : Natural Language Processing

Lecture 2: Linguistics Basics and Word Representations

Spring 2026
Benyou Wang
School of Data Science

Before LLMs/DL



Pipeline related to **linguistic** structure

Now

End2end without pipelines

Word sequence

Downstream tasks

Today Plan

- **Introduction to Linguistics**
- **Linguistic Structure in NLP**
  - Character: Chinese Characters, Character Encoding
  - Word: Morphemes, Lemmatization, Tokenization
  - Sentence: Phrase Parsing, Dependency
  - Discourse: Discourse Structure and Processing
- **Linguistic structure for today's NLP**
  - Implicit Mastery: Evolving Beyond Explicit Linguistic Rules in Modern LLMs
  - Pendulum Swing: The Future Convergence of Linguistics and Deep Learning

- **Word Representations**
  - Understanding Words from the perspective of Information retrieval
  - How do we represent words in NLP models?
  - Distributional hypothesis
  - Sparse vs dense vectors
- **Word2vec and other variants**
  - Word2vec
  - How to train this model?
  - Skip-gram with negative sampling (SGNS) and other variants
  - Evaluating word embeddings

What is linguistics?

What is linguistics?

Linguistics is the scientific study of language. Linguistics is based on a theoretical as well as a descriptive study of language and is also interlinked with the applied fields of language studies and language learning, which entails the study of specific languages.

Today's course will help us understand two significant aspects in linguistics:

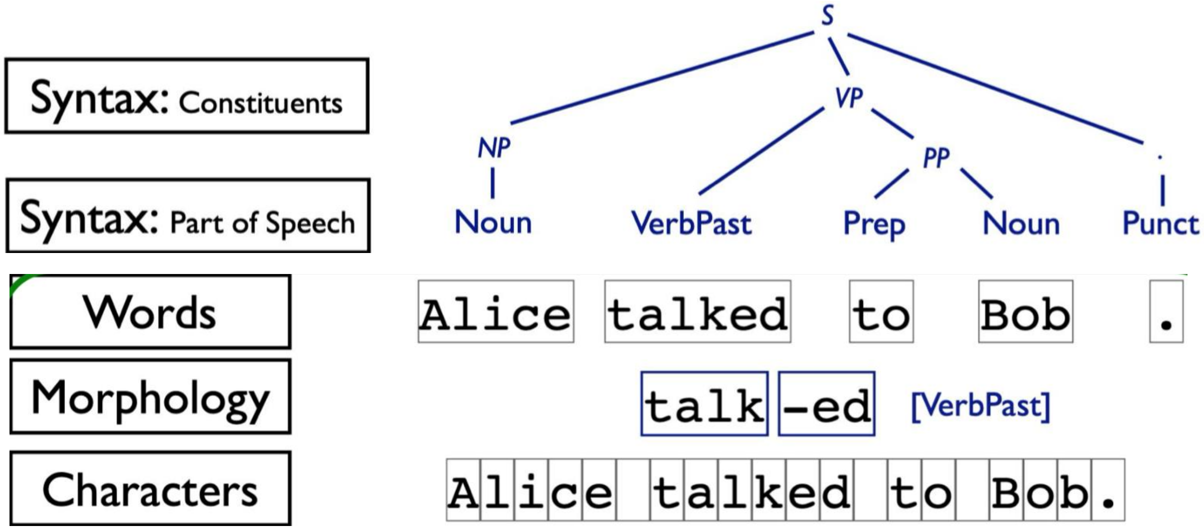
- **How do we understand language?**
- **How can computers process language?**

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

How do we understand language?

An important insight

Language is structured



There's structure underlying language

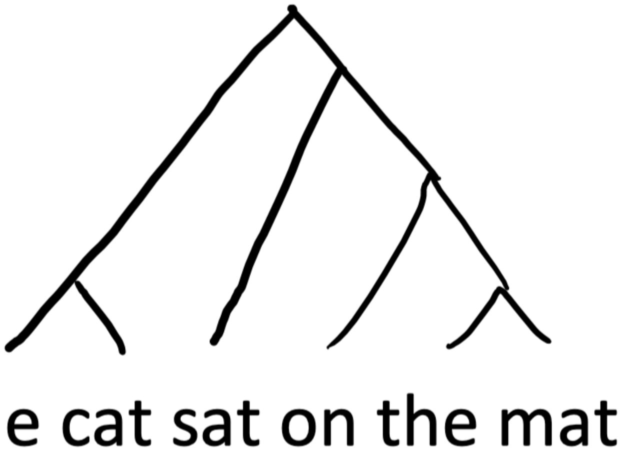
Isabel broke the window
The window was broken by Isabel

The cat is batting the toy
The toy is being batted by the cat

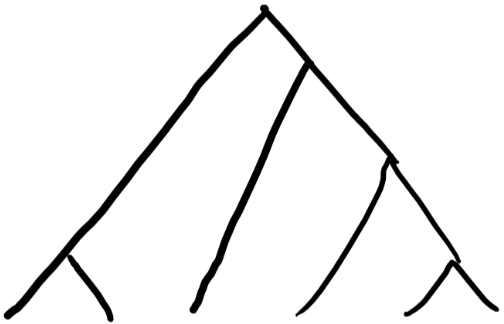
The plid yorbed the plof
The plof was yorbed by the plid

- We have some knowledge of structure that's separate from **the words we use** and **the things we say**.

Structure dictates how we can use language



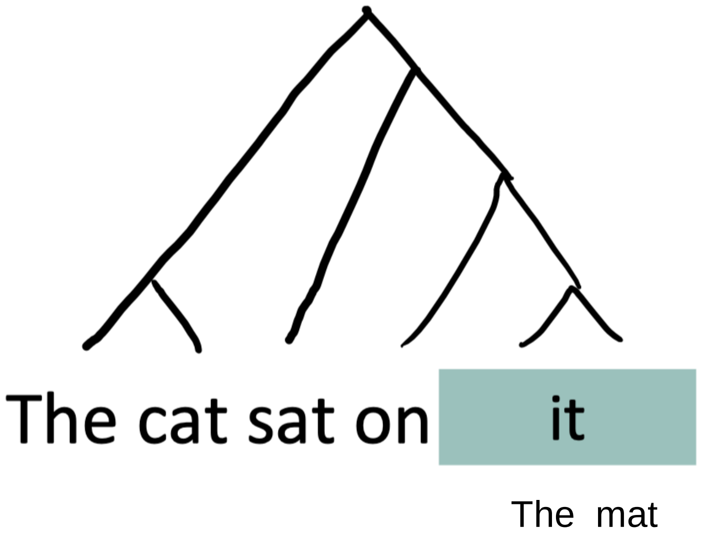
Structure dictates how we can use language



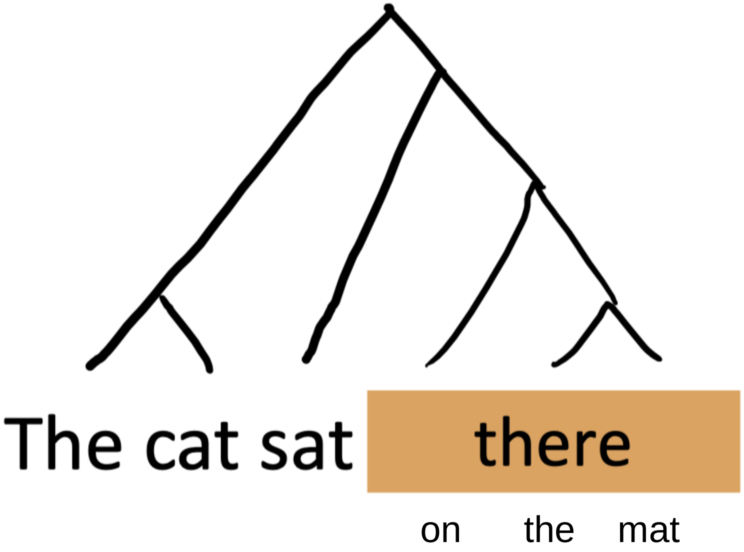
He sat on the mat

The cat

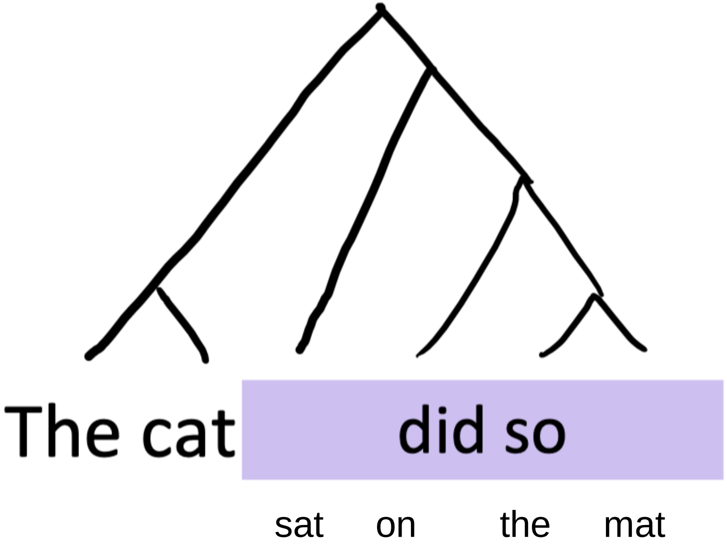
Structure dictates how we can use language



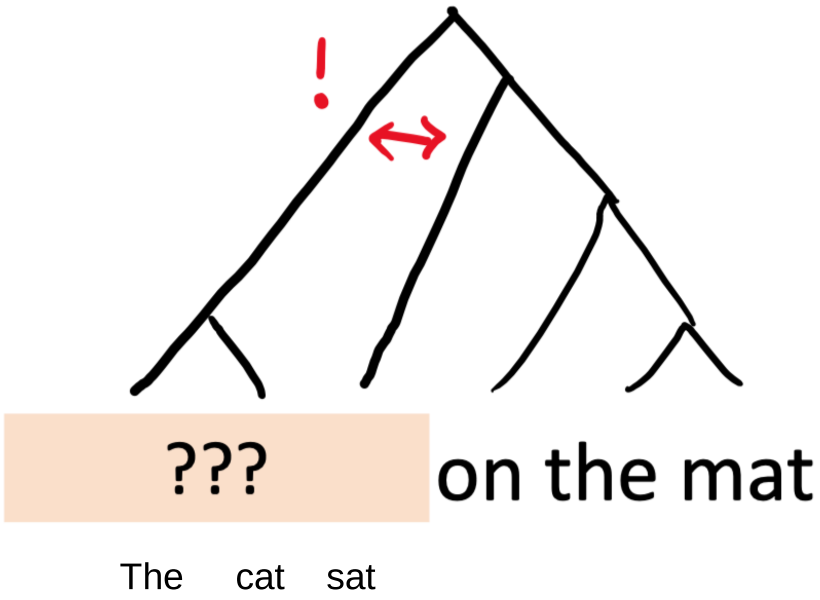
Structure dictates how we can use language



Structure dictates how we can use language

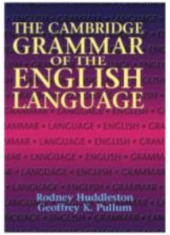


Structure dictates how we can use language



Recall the rules you learned in school

- A community of speakers (eg, Standard American English speakers) share a rough consensus of their implicit rules.
- **A grammar**: an attempt to describe all these rules
- What we are taught as “rules of grammar” often have other purposes than describing the English language



- When they say…
  - Never start a sentence with ‘And’
  - It’s incorrect to say “I don’t want nothing”
  - “Focus your thoughts and sound formal for this high school essay”
  - “The dialect with the most power in the US does not do negation in this way”

Grammaticality

- A community of speakers (eg, Standard American English speakers) share a rough consensus of their implicit rules.
- All the utterances we can generate from these rules are **grammatical**.
  - If we cannot produce an utterance using these rules, it's **ungrammatical**

Example

- Subject, Verb, and Object appear in **SVO order**
- **Subject pronouns** (I/she/he/they) have to be subjects, object pronouns (me/her/him/them) have to be objects
  - ✔ "I love her"   ✘ "Me love she"
  - ✘ "Me a cupcake ate"
    - The meaning is clear
    - But our rules of grammaticality **don't seem to cut us much slack**

Who speak English?

Linguistics differs across people

Local English or Standard English

https://www.youtube.com/watch?v=tQmNIqdwVMw

Vocabulary, pronunciation and grammar matters during communication?

Problem-solving is more important than ~~vocabulary, pronunciation and grammar~~

When speaking a new language, what matters most is your attitude — not your accuracy (including vocabulary and grammar)

https://www.youtube.com/watch?v=Ge7c7otG2mk

Linguistics is not **an art must to be mastered,**
instead, it is (was) **a tool to analyze.**

Today Plan

- **Introduction to Linguistics**
- **Linguistic Structure in NLP**
  - Character: Chinese Characters, Character Encoding
  - Word: Morphemes, Lemmatization, Tokenization
  - Sentence: Phrase Parsing, Dependency
  - Discourse: Discourse Structure and Processing
- **Linguistic structure for today’s NLP**
  - Implicit Mastery: Evolving Beyond Explicit Linguistic Rules in Modern LLMs
  - Pendulum Swing: The Future Convergence of Linguistics and Deep Learning

Beginning

Character

↓

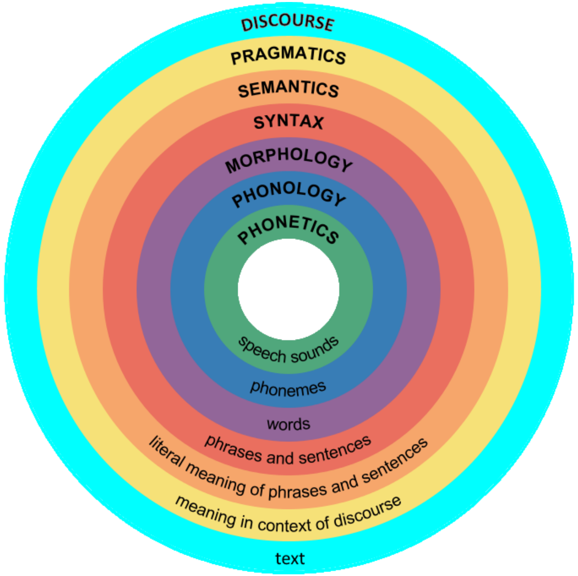
Word

↓

Sentence

↓

Discourse

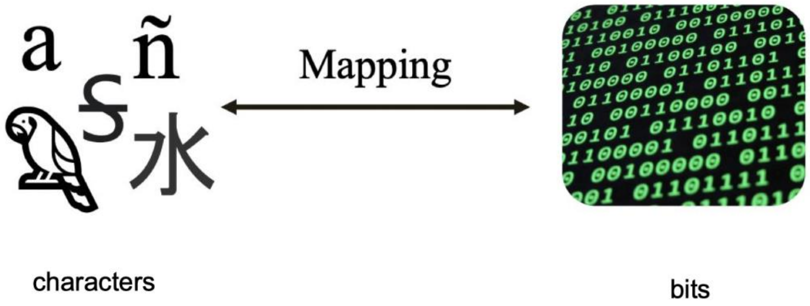


This section will explore linguistic structure by examining the different hierarchical levels of language.

Linguistic Structure in NLP

Character

Character



A character is the **smallest unit** of language and also the smallest unit for computer encoding. The **English language consists of only 26 characters**, which can be easily encoded using a Byte.

However, **Chinese characters are totally different** and much more complex.

Pictograph 象形文字

"Looks like in terms of **shapes**"

Ideographic characters （表意文字）

家　尖　休

“Looks like in terms of **concepts**”

Chinese Characters 汉字

- Chinese character information

Type	Example	Amount
Pictograph 象形	日月火人	Few
Ideographic 指示	上下	Very few
Compound indicative 会意	仁、信	Very few
Semantic-phonetic compounds 形声	请、情、清、晴 钾、钠、钙、镁	> 90%

Chinese character is morphemes syllable words in ideographic writing system.

Large Number of Characters

- Much larger number of characters as compared with the number of letters
- The exact number of existent Chinese characters cannot be precisely ascertained
    - 康熙字典(Kangxi dictionary 1716) 47, 035
    - 中华字海(Zhonghua Zihai dictionary 1994) 87, 019
    - **1000 characters may cover 92% written materials, and 3000 characters cover more than 99%**
- For computer processing purpose, Chinese characters are encoded in 16+ bits.

  2^8 = 256

  2^**16** = 65536

Traditional and Simplified Characters

- Traditional/simplified characters
  - Simp. Chinese: China, Singapore, Malaysia, United Nations.
  - Trad. Chinese: Taiwan, Hong Kong and Macau

- No one-one corresponding between
  - 乾‘dry’ and 幹‘to do’ -> Simplified 干
  - 遊‘travel’ and 游‘swim’ -> Simplified 游
  - simplified -> Traditional is relatively more difficult

Variant Characters

- Characters that have the same meaning and sounds but different shapes 异体字 （鬪 / 鬥 / 斗; 隻 / 只）
- Most of the characters in the Kangxi dictionary are variant character
  - Four variant characters of 回(囬囘田廻)
- Often share the same components as their standard counterparts
  - 裹/裡; 膀/髈; 杯/盃; 秘/祕;
- Becomes hot in Internet
  - 囧(窘)

Dialect characters and Dialectal Use of Standard Characters

- The existence of dialectal characters 方言字

Cantonese	Meaning	Mandarin
而家	Now	现在
同埋	And	和
边个	Who	哪位
边度	Where	哪儿

Shanghai	Meaning	Mandarin
侬	You	你
伊	He/she	他/她
伐	Not	不
白相	Play	玩

Southern Min	Meaning	Mandarin
阮	I/We	我(们)
暗	Late	晚
郎	Person	人
呷	Eat	吃

Character Encoding Standards

- GB “National Standard” in Chinese
  - 7445 characters. It includes 6, 763 simplified characters. Class-1/Class-2 characters
  - BG2312-80, contained only one code point for each character.
  - MSB. Bit-8 of each byte, is set to 1, and therefore becomes a 8-bit character. Otherwise, the byte is interpreted as ASCII. Every Chinese character is represented by a two-byte code. The MSB of both the first and second bytes are set.

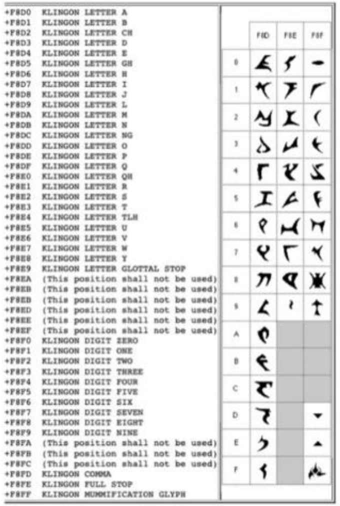
Character Encoding Standards

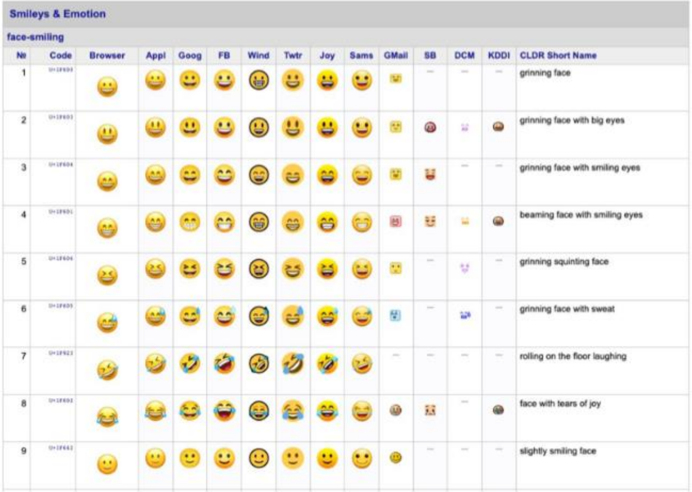
- GBK "National Standard Extension" in Chinese
  - An extension of GB2312
  - Includes 14, 240 traditional characters
  - The scheme is used by Simplified Microsoft Windows 95 and 98
- GB18030
  - Released by the China Standard Press, 2000
  - BG18030 supersedes all previous versions of GB
  - Officially mandatory for all software products sold in the PRC
  - Supports both simplified and traditional Chinese characters

Unicode

- One standard to unify all encoding standards
- ~150K characters, each mapped to a Unicode “codepoint” (e.g. U+0041)

Latin Alphabet: Uppercase		
	U+0041	A
	U+0042	B
	U+0043	C
	U+0044	D
	U+0045	E
	U+0046	F
	U+0047	G
	U+0048	H
	U+0049	I
	U+004A	J
	U+004B	K
	U+004C	L
	U+004D	M
	U+004E	N
	U+004F	O
	U+0050	P
	U+0051	Q
	U+0052	R
	U+0053	S
	U+0054	T
	U+0055	U
	U+0056	V





Unicode

- Industry standard, Universal Character Set
  - More than 100, 000 characters
  - Originates from East Asia
- Implemented by different character encodings
  - **UTF-8:** uses 1 byte for all ASCII characters, up to 4 bytes for other characters
  - **UCS-2:** uses 2 bytes for all characters, but does not include every character in the Unicode
  - **UTF-16:** using 4 bytes to encode characters missing from UCS-2
- Simplified and traditional characters as part of the project of Han unification

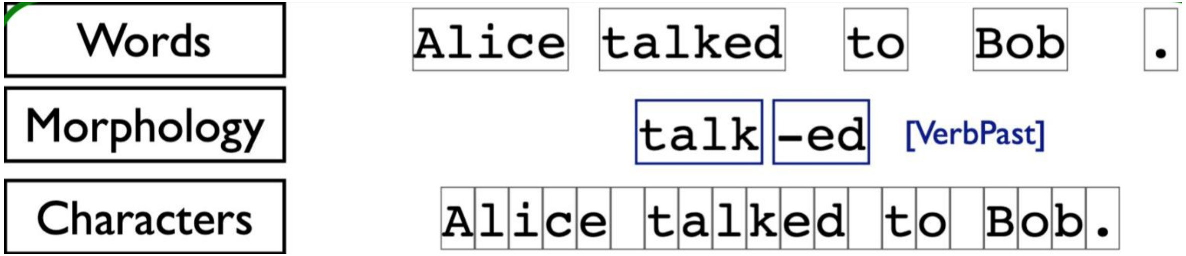
Example

	自	然	语	言	处	理
	B0–F7 lowerA1–EF					
GB2312	D7 D4	C8 BB	D3 EF	D1 D4	B4 A6	C0 ED
	三位 1110（E） 两位 110 一位0 四位1111					
UTF-8	E8 87 AA	E7 84 B6	E8 AF AD	E8 A8 80	E5 A4 84	E7 90 86
	According to UCS					
UTF-16	EA 81	36 71	ED 8B	00 8A	04 59	06 74

Linguistic Structure in NLP

Word

Character to Word



How do we identify the words in a text?

For a language like English, this seems like a really easy problem:
A word is any sequence of alphabetical characters between **whitespaces that's not a punctuation mark**?

That works to a first approximation, but…
… what about abbreviations like *D.C.*?
… what about complex names like *New York*?
… what about contractions like *doesn't* or *couldn't've*?
… what about *New York-based* ?
… what about names like *SARS-Cov-2*, or R2-D2?
… what about languages like Chinese that have no whitespace, or languages like Turkish where one such "word" may express as much information as an entire English sentence?

Words aren't just defined by blanks

Problem 1: Compounding

"ice cream", "website", "web site", "New York-based"

Problem 2: Other writing systems have no blanks

Chinese: 我开始写小说 = 我 开始 写 小说
I start(ed) writing novel(s)

Problem 3: Contractions and Clitics

English: "doesn't" , "I'm" ,

Italian: "dirglielo" = dir + gli(e) + lo
tell + him + it

Tokenization

Any actual NLP system will assume a particular tokenization standard.

- Because so much NLP is based on systems that are trained on particular corpora (text datasets) that everybody uses, these corpora often define a de facto standard.

Penn Treebank 3 standard:

Input:

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

Output:

“_ The _ San _ Francisco-based _ restaurant _ , _” _
they_ said_ ,_ "_ does _ n’t _ charge_ $_ 10 _ " _ . _

https://pytorchnlp.readthedocs.io/en/latest/_modules/torchnlp/datasets/penn_treebank.html

Spelling variants, typos, etc.

The same word can be written in different ways:

— with different **capitalizations**:
lowercase "cat" (in standard running text)
capitalized "Cat" (as first word in a sentence, or in titles/headlines),
all-caps "CAT" (e.g. in headlines)

— with different **abbreviation** or **hyphenation** styles:
US-based, US based, U.S.-based, U.S. based
US-EU relations, U.S./E.U. relations, ...

— with **spelling variants** (e.g. regional variants of English):
labor vs labour, materialize vs materialise,

— with **typos** (teh)

Good practice: Be aware of (and/or document) any normalization (lowercasing, spell-checking, …) your system uses!

How many different words are there in English?

How large is the **vocabulary** of English
(or any other language)?

Vocabulary size = the number of distinct word types

Google N-gram corpus: 1 trillion tokens,
13 million word types that appear 40+ times

If you count words in text, you will find that…

…a few words (mostly closed-class) are very frequent
(the, be, to, of, and, a, in, that,…)

… most words (all open class) are very rare.

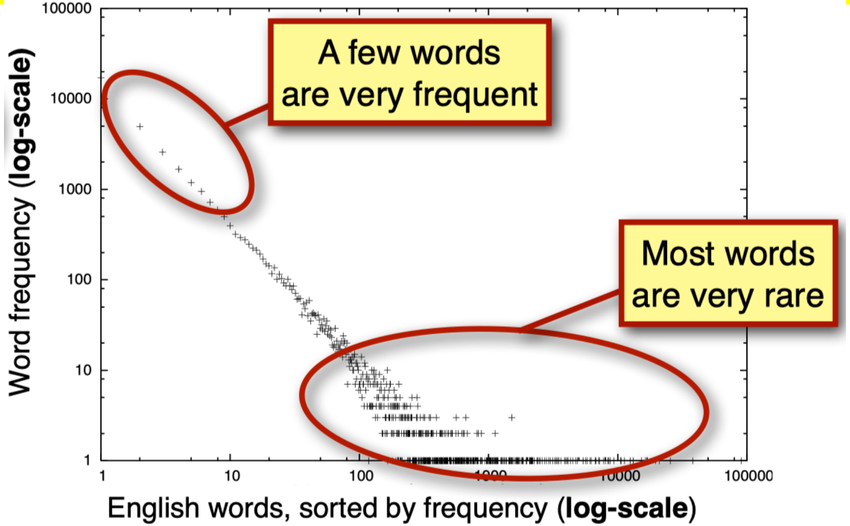
… even if you've read a lot of text,
you will keep finding words you haven't seen before.

Word frequency: the number of occurrences of a word type
in a text (or in a collection of texts)

Zipf's law: the long tail

the r-th most common word w_r has $P(w_r) \propto 1/r$

How many words occur once, twice, 100 times, 1000 times?



w_1 = *the*, w_2 = *to*, …., w_{5346} = *computer*, ...

In natural language:

- A small number of events (e.g. words) occur with **high frequency**
- A large number of events occur with **very low frequency**

How many different words are there in English?

How large is the **vocabulary** of English
(or any other language)?

Vocabulary size = the number of distinct word types

Google N-gram corpus: 1 trillion tokens,
13 million word types that appear 40+ times

[here, we're treating inflected forms (took, taking) as distinct]

You may have heard statements such as
"adults know about 30,000 words"
"you need to know at least 5,000 words to be fluent"

Such statements do not refer to inflected word forms (take/takes/taking/take/takes/took) but to lemmas or dictionary forms (take), and assume if you know a lemma, you know all its inflected forms too.

Word Mapping

Add some generalization by **mapping** different forms of a word to the same symbol:

- **Normalization:** map all variants of the same word (form) to the same canonical variant (e.g. lowercase everything, normalize spellings, perhaps spell-check)
- **Lemmatization**: map each word to its lemma (esp. in English, the lemma is still a word in the language, but lemmatized text is no longer grammatical)
- **Stemming**: remove endings that differ among word forms (no guarantee that the resulting symbol is an actual word)

Motivating challenge: out-of-vocabulary (OOV) words

Many NLP systems assume a fixed vocabulary, but still have to handle **out-of-vocabulary (OOV)** words.

Example

So if our test sentence was

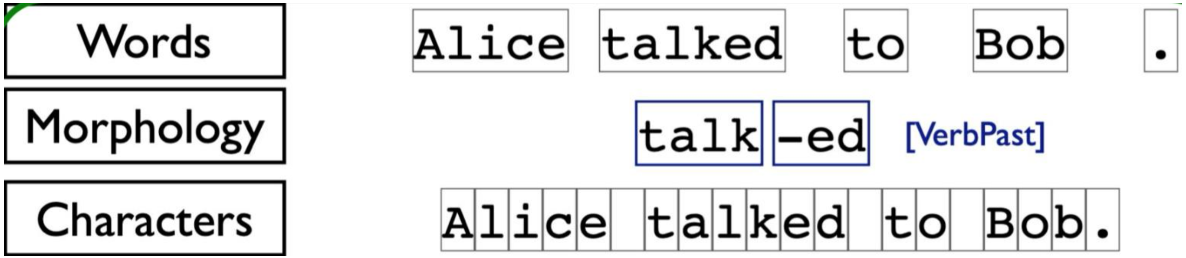
the dogs love the cats

The input would look like:

The **<OOV>** **<OOV>** the **<OOV>**

Where **<OOV>** is the “out of vocabulary” symbol

Review: Linguistic structure



Represent the structure of each word

“books” => “book N pl” (or “book V 3rd sg”)

This requires a **morphological analyzer**

The output is often **a lemma (“book”) plus morphological information (“N pl” i.e. plural noun)**

This is particularly useful for highly inflected languages, e.g. Czech, Finnish, Turkish, etc. (less so for English or Chinese):

Morphemes (语素): stems, affixes

dis-grace-ful-ly
prefix-stem-suffix-suffix

Many word forms consist of a *stem*
plus a number of *affixes* (*prefixes* or *suffixes*)

Exceptions: *Infixes* are inserted inside the stem
Circumfixes (German *ge*seh*en*) surround the stem

Morphemes: the smallest (meaningful/grammatical) parts of words.

Stems (grace) are often free morphemes.

Free morphemes can occur by themselves as words.

Affixes (dis-, -ful, -ly) are usually bound morphemes.

Bound morphemes *have* to combine with others to form words.

So far, we’ve been talking about English Words

Let's see how Chinese words are different.

Chinese Word

- Distinct from both morphemes at a lower level and from phrases at a higher level

- Distributional restriction
  - Can occur freely by itself vs. bound morphemes
  - Grammatical morphemes 的、地、得
  - Content morphemes 饮 冷饮 饮食

- Integrity of word meanings
  - Have meanings that are not predictable from the meanings of their component morphemes
  - 大人 vs. 小人 打人 vs. 打手

Chinese Word Formation

- Disyllabic compounds （双音节复合词）
  - 小人 热心 报告 声音
- Tri-syllabic compounds （三音节复合词）
  - 口香糖 大学生 吹牛皮
- Quad-syllabic compounds （四音节复合词）
  - 花言巧语 口是心非
- **Affixation**
  - Prefix: 第一/第二 Suffix: 儿子/ 化学
- **Reduplication**
  - 商量商量 高高兴兴

Challenges in Chinese - Few formal morphological markings

- No verbal inflections:
  - No tense: a verb will have the same form
    - 我过去 是 学生。 I **was** a student.
    - 我现在 是 学生。 I **am** a student.
    - 我将来 是 学生。 I **will be** student.
  - No personal and number agreements
    - 我 去。 I **go**.
    - 她 去。 She **goes**.
- No nominal endings
  - No number marking
    - 我的书 My book(s)
  - No gender marking
  - No case marking
    - **I** love **her**. Vs. **She** loves **me**.
    - (I=subject case; me = object case; she=subject case; her=object case)
    - 我爱她 她爱我。
    - (我=both subject and object case; 她=both subject and object case)

Challenges in Chinese - Ambiguities in Words

- Lexical Ambiguities:
  - 他很好吃
  - 炸鸡很好吃
- Structural Ambiguities
  - Overlapping (crossing) ambiguity 交集型歧义
    - 网球场 美国会
  - Combinatorial ambiguity 组合型歧义
    - 才能 学生会
  - Mixed type 混合型歧义
    - 太平淡 (too dull), 太平 (peaceful), 平淡

How do state-of-the-art models represent word?

Tokenization for GPT-3, GPT-2... most likely ChatGPT

Tokens 97 **Characters** 248

A helpful rule of thumb is that one token generally corresponds to ~4 characters of text for common English text. This translates to roughly ¾ of a word (so 100 tokens ~= 75 words).

一个有用的经验法则是，对于通常英文文本而言，一个标记对应约4个字符的文本。这大约相当于一个字的¾（因此100个标记~=75 个）。

TEXT TOKEN IDS

https://platform.openai.com/tokenizer

Tokenization for GPT-3, GPT-2… most likely ChatGPT

Byte Pair Encoding (BPE) (Sennrich et al., 2015) is a practical middle ground between character and word level language modeling which effectively interpolates between word level inputs for frequent symbol sequences and character level inputs for infrequent symbol sequences. Despite

Radford A, Wu J, Child R, et al. Language models are unsupervised multitask learners[J]. OpenAI blog, 2019, 1(8): 9.

Example：Byte-Pair Encoding (BPE) for Tokenization

- Use our *data* to automatically tell us what tokens should be
- **Aim:** induce tokens that are **subwords** that
  - Are smaller than words
  - Can be morphemes (e.g., *-est* or *-er)*
  - Or just arbitrary substrings
- Useful for dealing with words never been seen before at training time (out-of-vocabulary words)

Token learner
Input: Raw training corpus
Output: Induces vocabulary via BPE

Token segmenter
Input: Learned vocabulary, test sentence
Output: Segmented token list

Byte-Pair Encoding (BPE) deals OOV words

1. Convert each character into **unicode** bytes

2. Use these **bytes as the base vocabulary** for the BPE algorithm.

3. Tokenize via BPE.

4. Big picture: <u>Now we can deal with any word or character!</u>

Parts of Speech

限定词 determiner	Verb. 3rd person singular present	determiner	adjective	noun	
DT	VBZ	DT	JJ	NN	PART OF SPEECH
This	is	a	simple	sentence	WORDS
	be 3sg present		SIMPLE1	SENTENCE1	MORPHOLOGY

"Parts of speech" are categories of words based on **their function within a sentence**. Understanding them is fundamental to grasping the rules of grammar and syntax in a language.

Parts of Speech

Tag	Description
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural
PDT	Predeterminer
POS	Possessive ending
PRP	Personal pronoun

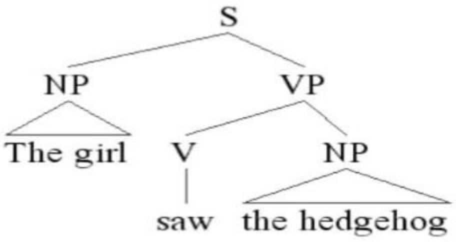
Tag	Description
PRP$	Possessive pronoun
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol
TO	to
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Whdeterminer
WP	Whpronoun
WP$	Possessive whpronoun
WRB	Whadverb

Linguistic Structure in NLP

Sentence

Phrase

- Words are organized into phrases, then comes to sentence



- Syntax studies the regularities and constraints of word order and phrase structure

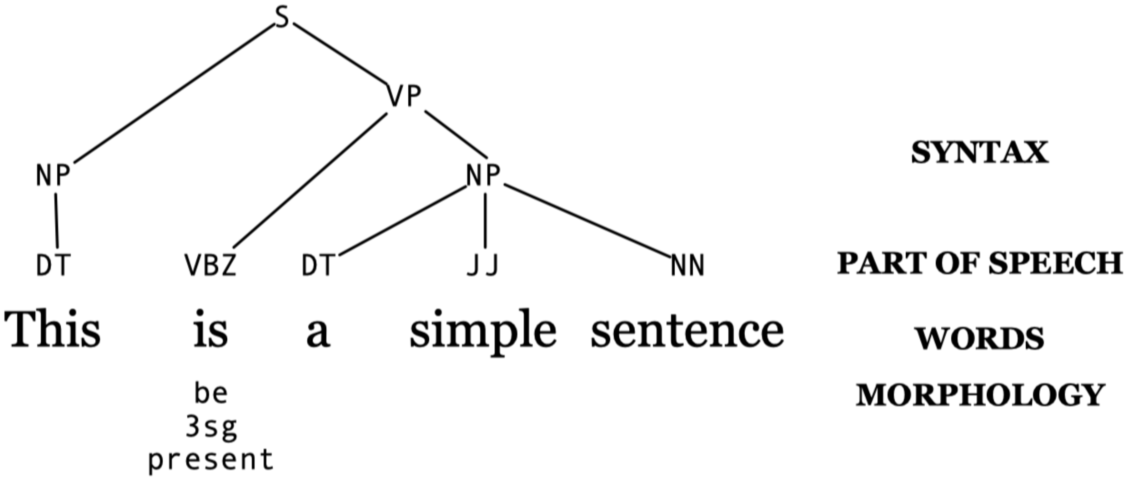
- Major phrase categories in English
  - Noun phrase. Prepositional phrases
  - Verb phrases. Adjective phrases

English Phrase

Category	Description	Examples
Noun Phrase (NP)	A noun and all its modifiers	**The bewildered tourist** was lost.
Verb Phrase (VP)	A verb and all its modifiers	He **was waiting for the rain to stop**.
Gerund Phrase (GP)	A noun phrase that starts with a gerund.	**Taking my dog for a walk** is fun.
Infinitive Phrase (IP)	A noun phrase that begins with an infinitive verb.	**To make lemonade**, you have **to start with lemons**.
Appositive Phrase (AP)	It restates and defines a noun. It consists of one or more words.	**My favorite pastime**, needlepoint, surprises some people.
Participial Phrase (PP)	Begins with a past or present participle.	**Washed with my clothes**, my cell phone no longer worked.
Absolute Phrase	It modifies the whole sentence, not just a noun.	**His tail between his legs**, the dog walked out the door.

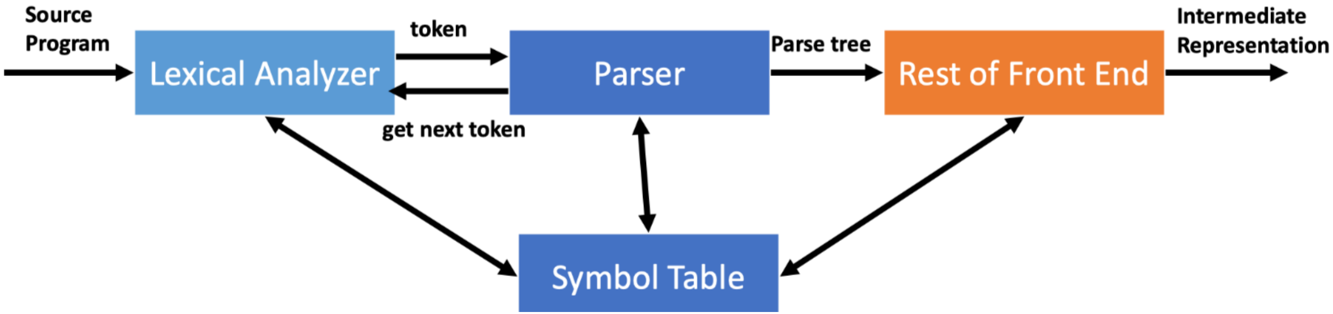
Phrase Structure Grammar

Tells us how to determine the meaning of the sentence from the meaning of the words.



Parsing (Syntax Analysis)

- Syntax
  - Is one of the major components of **grammar**.
  - Is the proper order of words in a phrase or sentence.
  - Is a tool used in writing proper grammatical sentences
  - Native speakers of a language learn correct syntax without realizing it.
  - The complexity of a writer's or speaker's sentences creates a formal or informal level of diction that is presented to its audience.



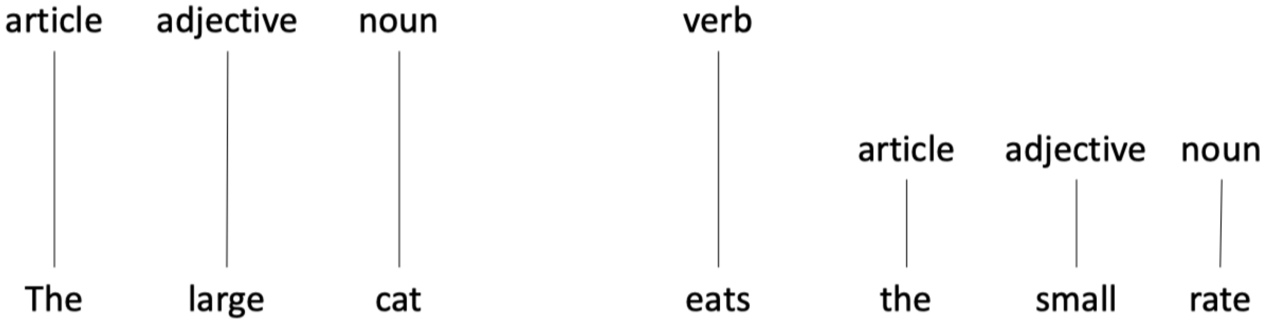
Parse tree (Syntactic Tree)

"The large cat eats the small rat"

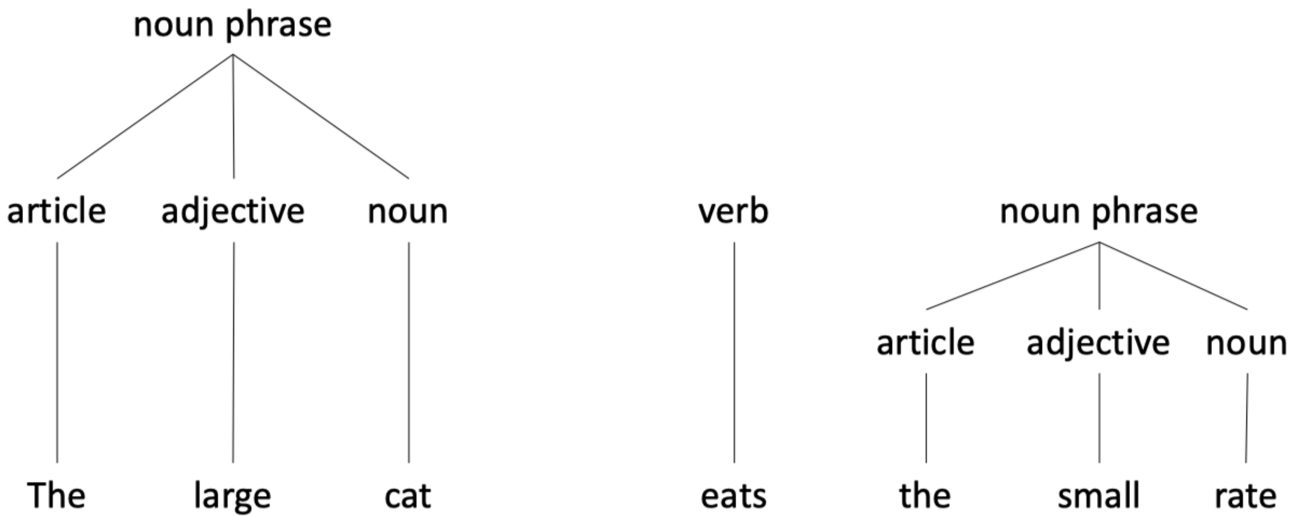
Parse tree (Syntactic Tree)

The large cat eats the small rate

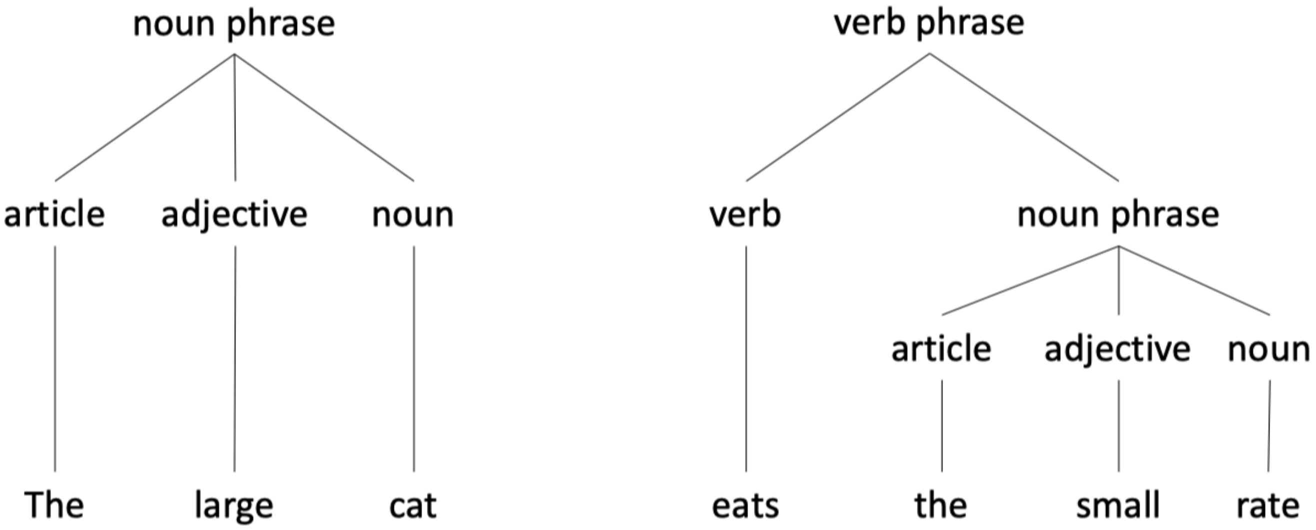
Parse tree (Syntactic Tree)



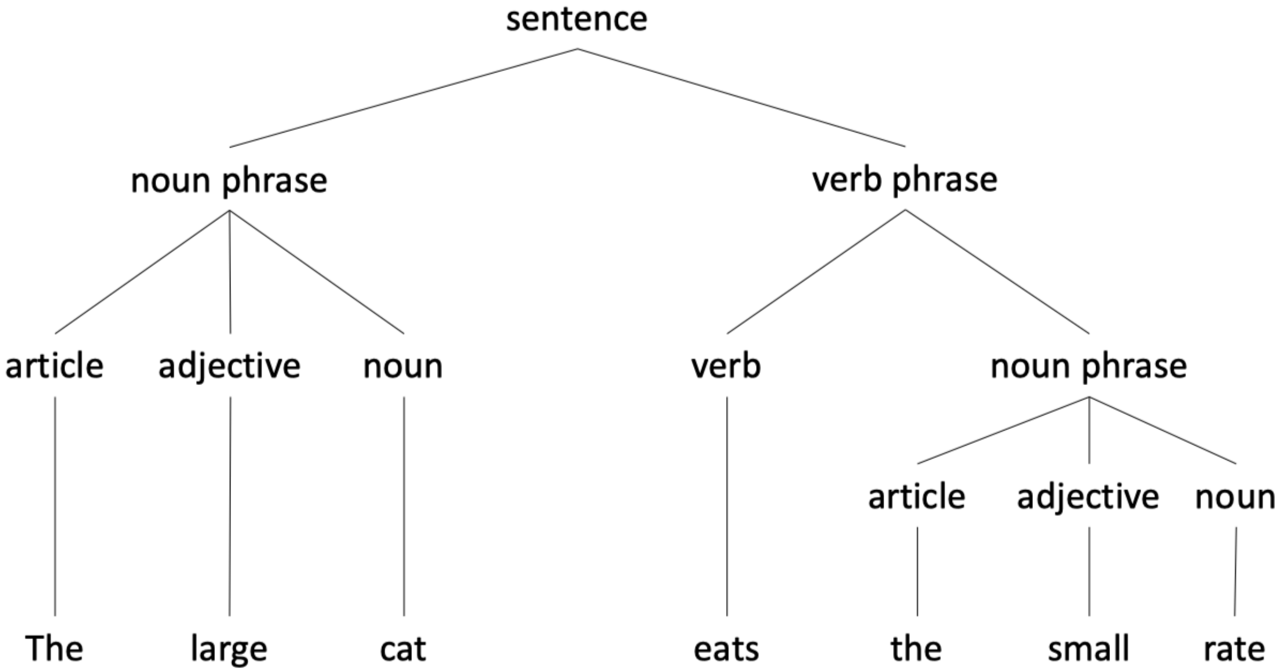
Parse tree (Syntactic Tree)



Parse tree (Syntactic Tree)

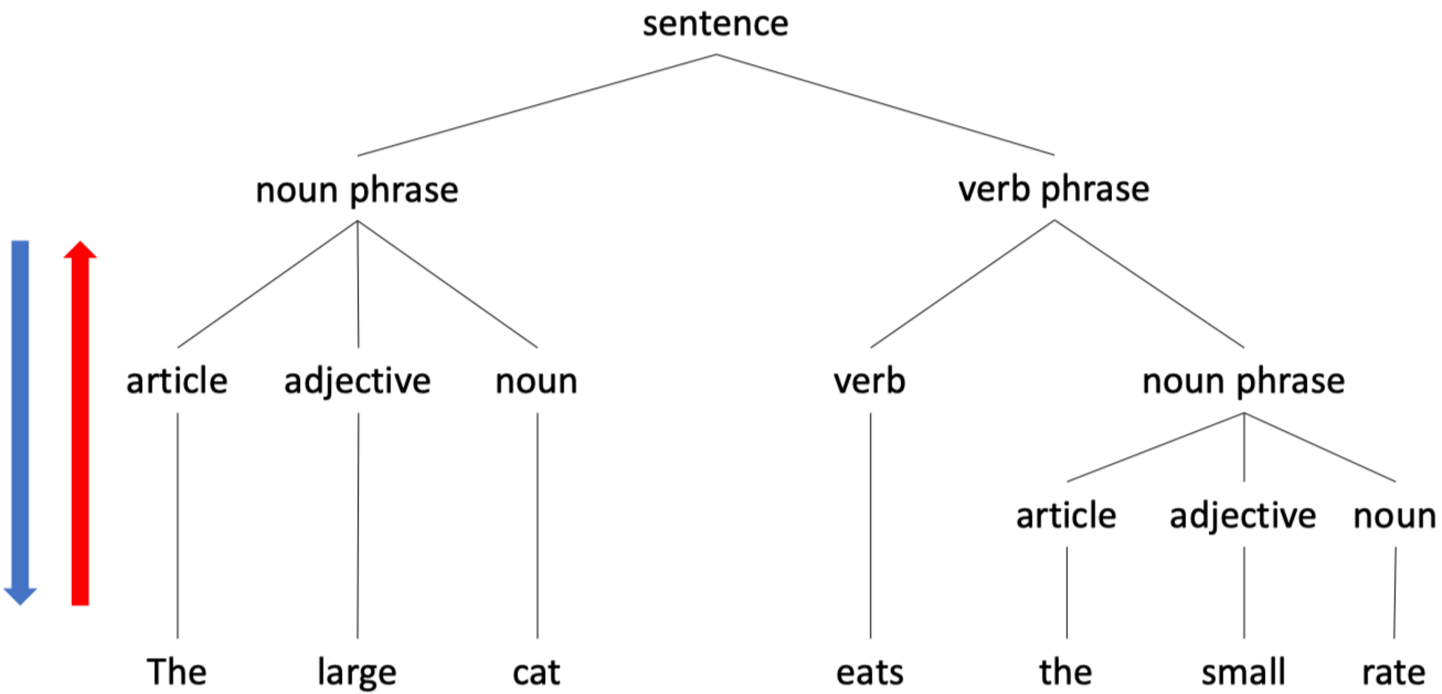


Parse tree (Syntactic Tree)



Parse tree (Syntactic Tree)

- Bottom-up or Top-down



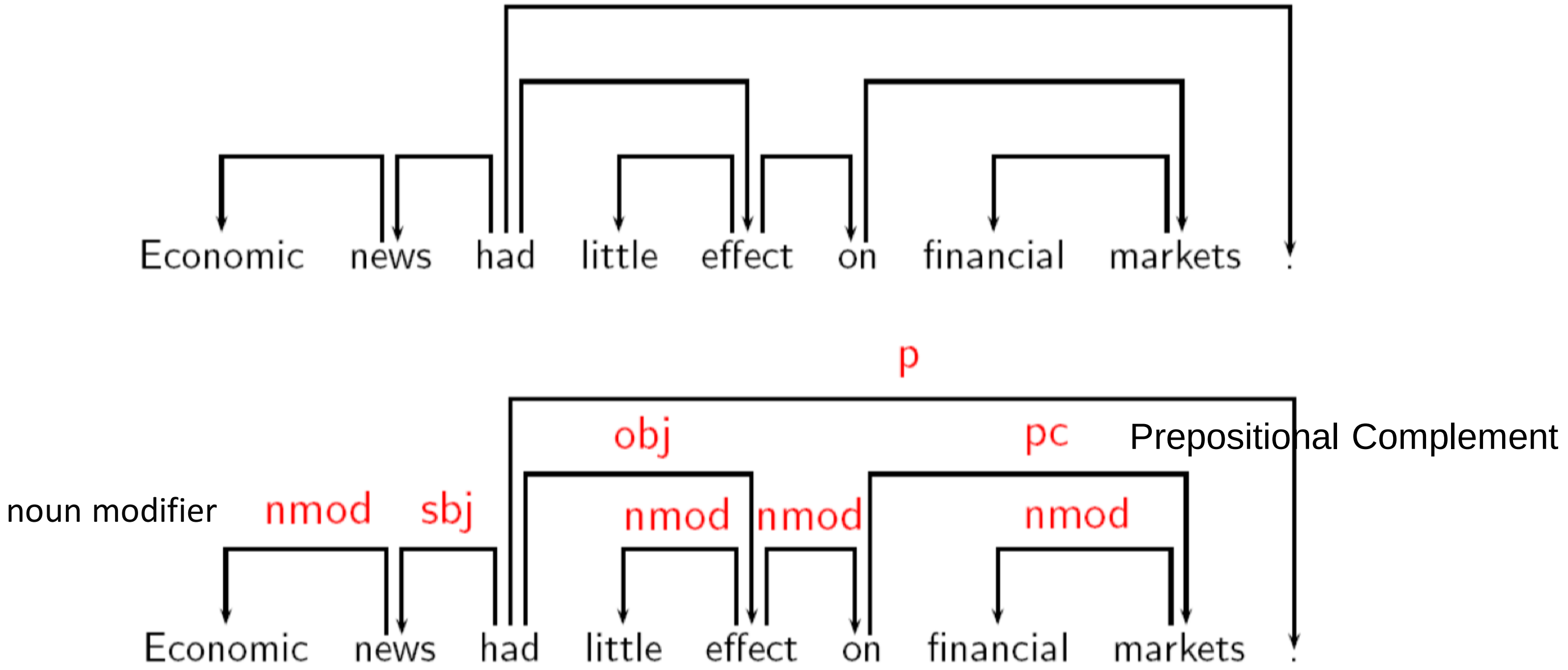
Dependency

Syntactic structure consists of **lexical items**, linked by binary asymmetric relations called **dependencies**.

The sentence is an *organized whole*, the constituent elements of which are *words*. Every word that belongs to a sentence ceases by itself to be isolated as in the dictionary. Between the word and its neighbors, the mind perceives *connections*, the totality of which forms the structure of the sentence. The structural connections establish *dependency* relations between the words. Each connection in principle unites a *superior* term and an *inferior* term. The superior term receives the name *governor*. The inferior term receives the name *subordinate*.

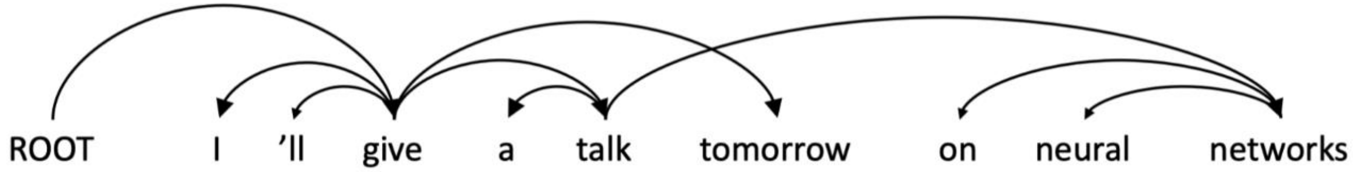
Tesnière, L. (1959). Eléments de Syntaxe Structurale.

Dependency Structure



Dependency Parsing

- A sentence is parsed by choosing for each word what other word (including ROOT) it is a dependent of
- Usually some constraints:
  - Only one word is a dependent of ROOT
  - Don’t want cycles A → B, B → A
- This makes the dependencies a tree
- Final issue is whether arrows can cross (be non-projective) or not

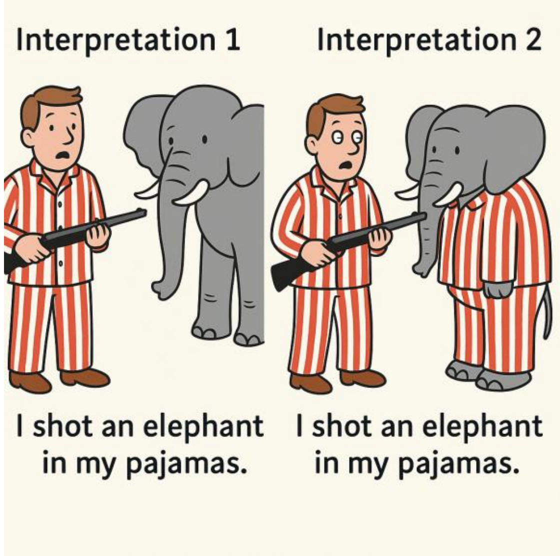


Problems in Parsing - Ambiguity

- "One morning, I shot an elephant in my pajamas. How he got into my pajamas, I don't know."

I shot an elephant in my pajamas

I shot an elephant in my pajamas



Problems in Parsing - Ambiguity

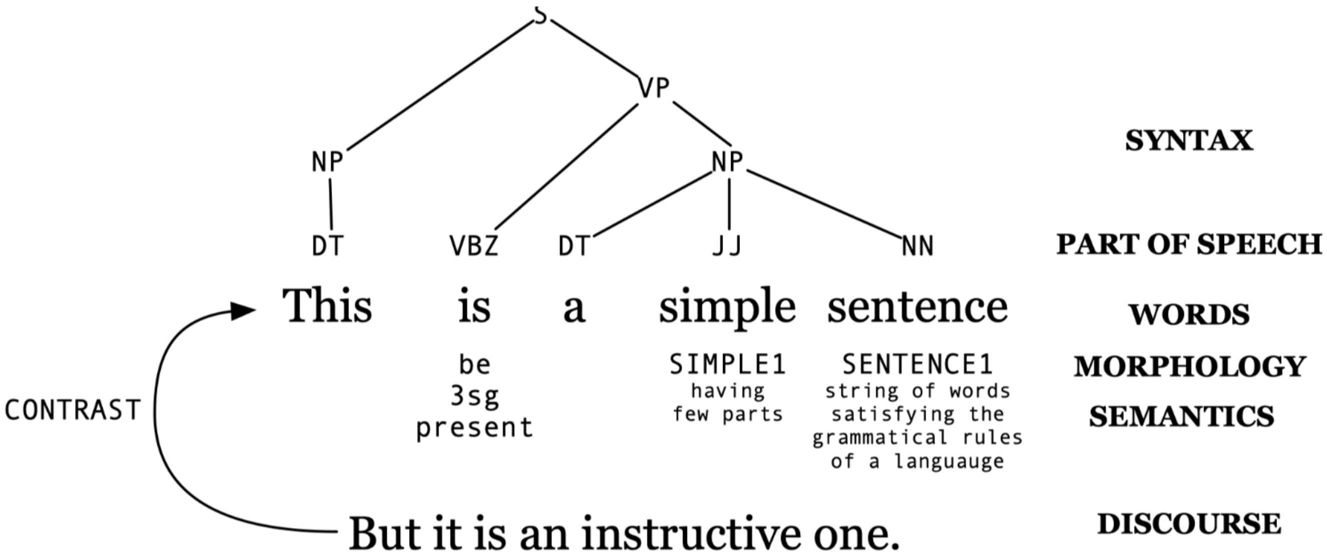
- "One morning, I shot an elephant in my pajamas. How he got into my pajamas, I don't know."
- Syntactical/Structural ambiguity
  - Several parse trees are possible e.g. above sentence
- Semantic/Lexical ambiguity
  - Several word meanings. e.g. bank (银行) or bank (河流)
- Different word categories
  - "He **books** the flight" vs. "The **books** are here."
  - "Fruit flies from the balcony" vs. "Fruit flies are on the balcony"
- Attachment
  - In particular PP (prepositional phrase) binding; often referred to as "binding problem"
  - "One morning, I shot an elephant **in my pajamas**."

Linguistic Structure in NLP

Discourse

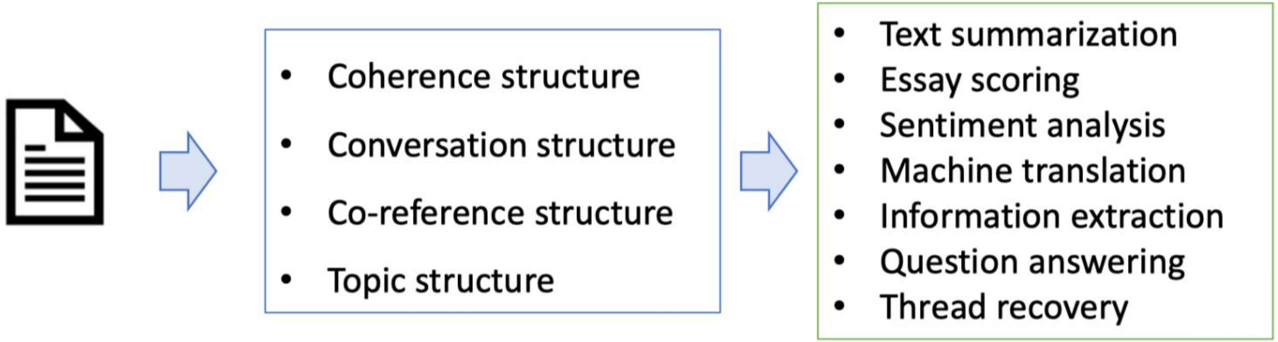
（篇章/语篇）

Discourse Structure



Discourse Processing

- **Discourse** is a group of collocated and coherent sentences
- **Discourse theory** deals with language phenomena that operate beyond the single sentence
- **Discourse analysis/processing** is a suite of **Natural Language Processing (NLP) tasks** to uncover linguistic structures from multi-sentential texts at several levels, which can support many “downstream” NLP applications.



different discourse!

Discourse/Coherence Relations

- Specify the relations between **sentences** or **clauses**.
- Due to the relations, two adjacent sentences can look coherent.

What is the discourse relation between the following two sentences?

John hid Bill's car keys. He was drunk

"Explanation" relation

John did sth. **because** he was drunk

vs.

John hid Bill's car keys. He likes spinach

"Non-coherent"

More Discourse Relations

- Elaboration 细化
  - Dorothy was from Kansas. She lived on the Kansas prairies.
- Result 因果
  - The tin woodman was caught in the rain. His joints rusted.
- Parallel 并列
  - The scarecrow wanted some brains. The tin woodsman wanted a heart.

- How many relations? Which ones?
- What kind of structures? Flat? Trees? Graphs?

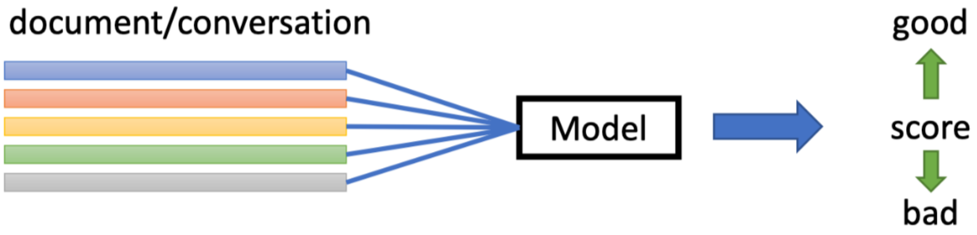
Discourse Parse

- Either two clauses / sentences connected by an explicit connective
  - The federal government suspended sales of U.S. savings bonds because Congress hasn't lifted the ceiling on government debt.
    cause-reason
  - The subject will be written into the plots of prime-time shows, and viewers will be given a 900 number to call. Conjunction

- Or two adjacent sentences connected by an implicit connective
  - Some have raised their cash positions to record levels. High cash positions help buffer a fund when the market falls.
    Implicit=because(cause-reason)

Webber B L, Joshi A K. Anchoring a lexicalized tree-adjoining grammar for discourse[J]. arXiv preprint cmplg/9806017, 1998.

Coherence Models



- Helps predict which sentences are pragmatically appropriate
- Tells us which sentences are closely related

- Applications
  - Essay scoring
  - Summarization (sentence selection and ordering)
  - Generation (including MT)
  - ...

Linguistic Structure for today's NLP

Implicit Mastery: Evolving Beyond Explicit Linguistic Rules in Modern LLMs

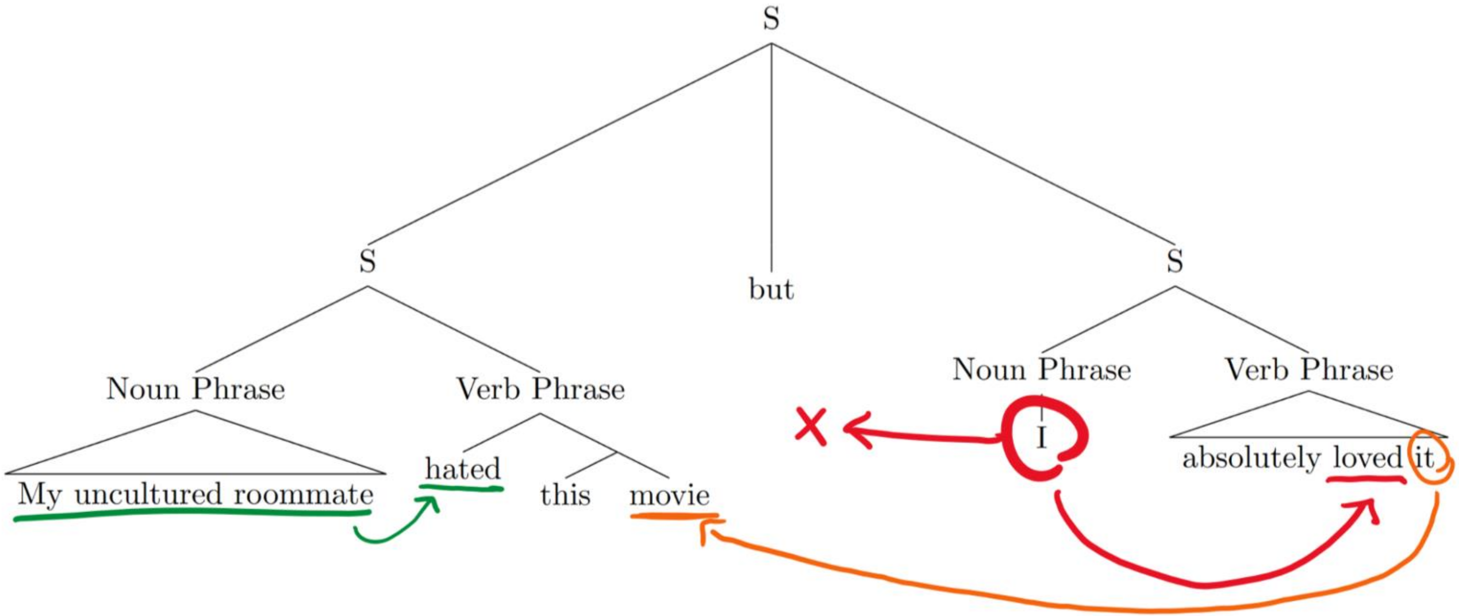
Before Self-supervised Learning

Before self-supervised learning Like GPT, the way to approach doing NLP was through understanding the human language system, and trying to imitate it.

- **Example:** Parsing
  - I want my sentiment analysis system to classify this movie review correctly
    - "My uncultured roommate hated this movie, but I absolutely loved it"
  - How would we do this?
  - We might have some semantic representation of some key words like "hate" and "uncultured", but how does everything relate?

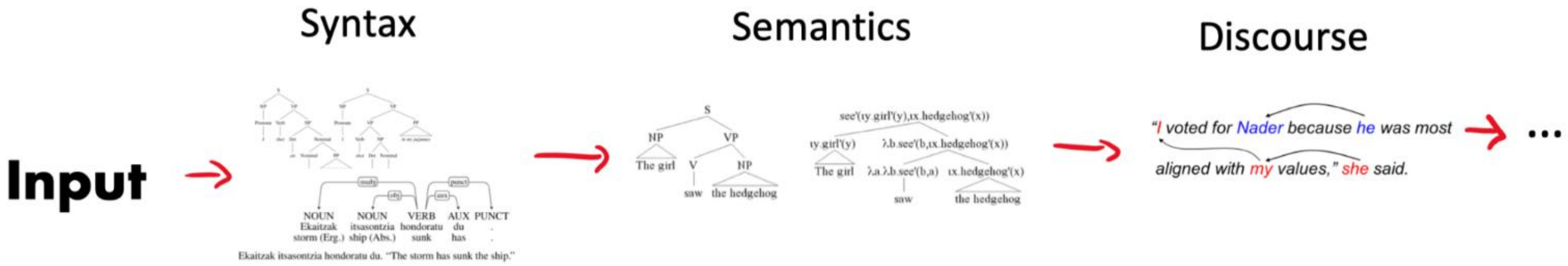
Before Self-supervised Learning

Many linguists might tell you something like this:



Before Self-supervised Learning

They built **Pipelines** for language processing based on language structures.



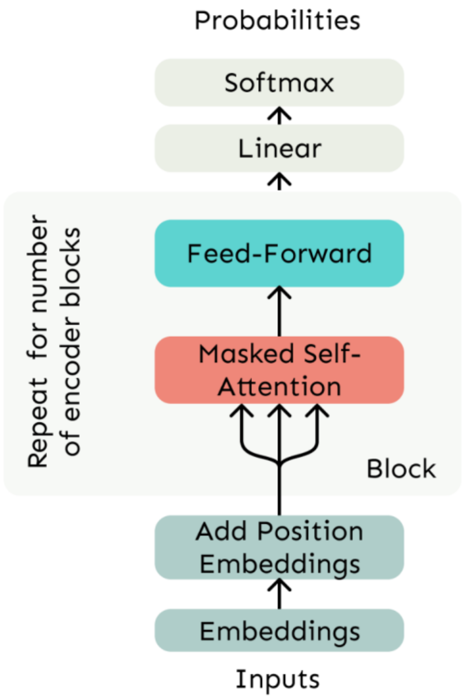
Now, language models easily solve these NLP problems!

IS

Hi! Whaddya think of this movie review? "My uncultured roommate hated this movie, but I absolutely loved it"

As an AI language model, I do not have the ability to form opinions or beliefs. However, based on the given movie review, it seems that the person who wrote it had a positive experience watching the movie, while their roommate did not enjoy it. It's important to remember that everyone has their own tastes and preferences when it comes to movies and other forms of entertainment, and it's okay to have differing opinions. It's always good to approach a movie with an open mind and form your own opinion rather than solely relying on someone else's viewpoint.

How we train our models these days!



Directly learn from unstructured corpora through self-supervised learning, without concerning with **Linguistic Structure**.

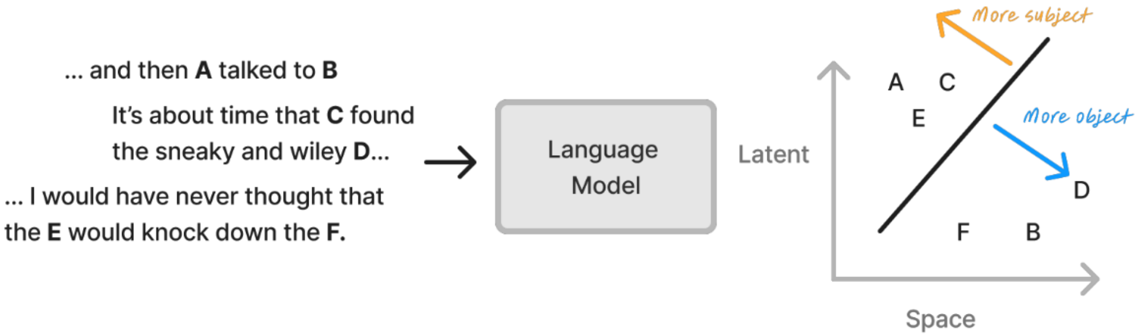
Structured Learning to non-structured Learning

Contemporary Large Language Models like GPT-4 employ a non-structured learning approach, directly learning from raw corpora. This contrasts with earlier methods that depended heavily on explicit, rule-based learning of linguistic structure.

This approach is favored for several reasons:

- It allows for **scalability**, facilitating the learning of vast, unstructured natural language corpora without the need for explicit programming of linguistic rules.
- Language structures are inherently complex and vary depending on context and word usage, making them difficult to effectively encode and learn.

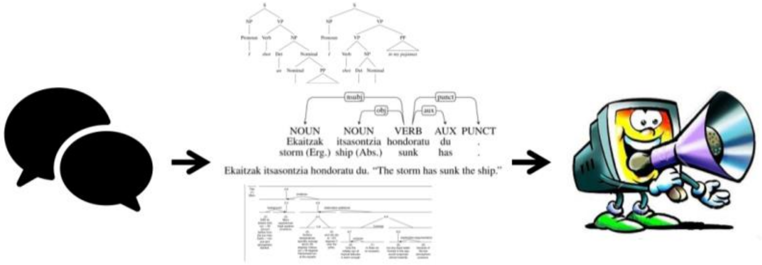
Language models are also aware of linguistic structure even they are not particularly trained for.



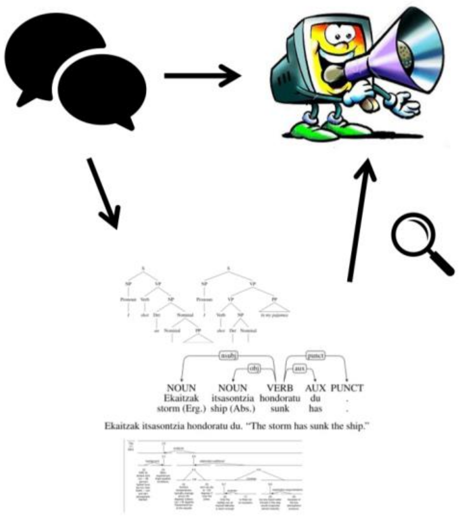
Papadimitriou, Isabel, et al. "Deep subjecthood: Higher-order grammatical features in multilingual BERT." *arXiv preprint arXiv:2101.11043* (2021).

Large language models: a paradigm to implicitly learn linguistics

Before:



Now:



(hidden comprehension)

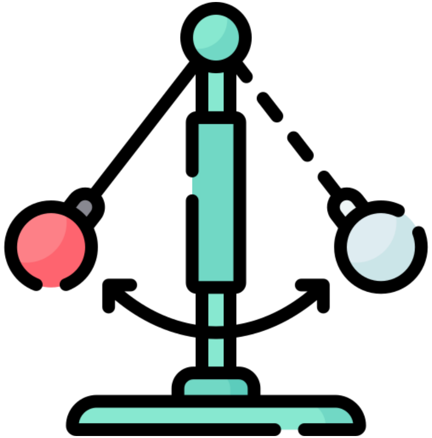
Linguistic Structure for today's NLP

Pendulum Swing: The Future Convergence of Linguistics and Deep Learning

Pendulum Swing

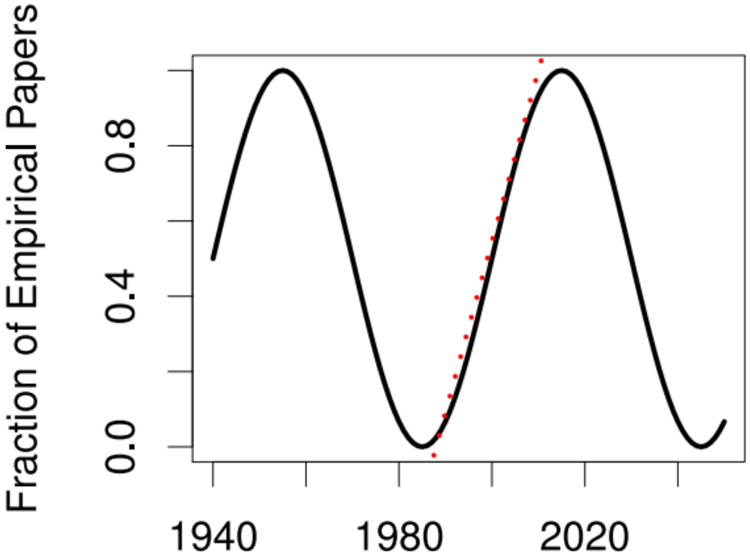
- **Empiricism**: Empiricism in NLP, exemplified by deep learning, relies on vast amounts of data and statistical models to understand and generate human language.
- **Rationalism**: Emphasizes linguistic structure, advocates for a more rule-based approach, rooted in the understanding of grammatical and syntactic principles of language.

Empiricism
(deep learning in NLP)



Rationalism
(Linguistic Structure)

Kenneth Church. A Pendulum Swung Too Far. https://journals.colorado.edu/index.php/lilt/article/view/1245/1081, 2007



- 1950s: Empiricism (Shannon, Skinner, Firth, Harris)
- 1970s: Rationalism (Chomsky, Minsky)
- 1990s: Empiricism (IBM Speech Group, AT&T Bell Labs)
- 2010s: A Return to Rationalism?

https://languagelog.ldc.upenn.edu/myl/ldc/swung-too-far.pdf

The Empirical Era: Deep Learning's Triumph in NLP

- We are currently witnessing a pronounced **swing towards empiricism**, primarily fueled by the advancements in deep learning.
- This era is marked by the emergence of sophisticated language models like GPT-4 and BERT, which have revolutionized our ability to process and generate human language.
- These models, built on the backbone of massive datasets, excel in a variety of tasks such as text generation, translation, and semantic understanding. The reliance on extensive data has not only improved accuracy but also expanded the scope of applications, making NLP more versatile and accessible than ever before.

As the pendulum inevitably swings

However, the dominance of empirical methods in NLP is not without its challenges. While these models excel in processing vast amounts of data, they often grapple with issues like hallucinations, lack of control, and opacity in their decision-making processes.These problems, seemingly intractable within the current framework of deep learning, point to inherent limitations.

As the pendulum inevitably swings, the future of Large Language Models (LLMs) might see a greater integration of rationalism. Such a combination could lead to more robust, controllable, and interpretable AI systems. As LLMs gradually encounter developmental bottlenecks, the incorporation of linguistic structure and theory promises to unlock new avenues and breakthroughs in the field.

Do you think

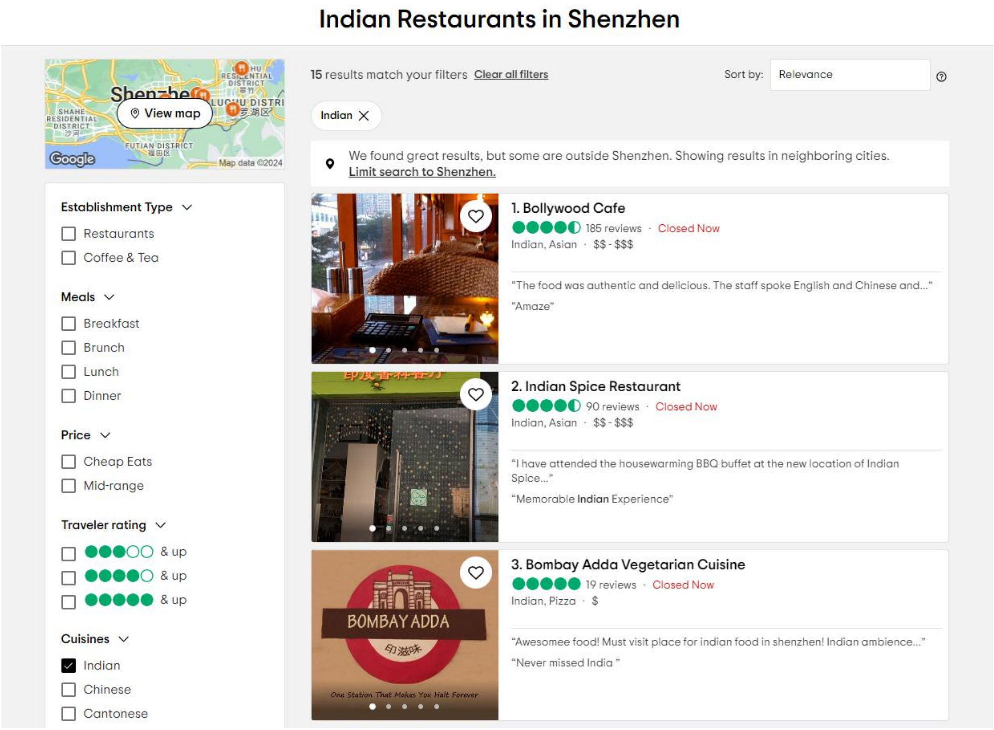
Rationalism (Linguistic Structure) will be back？

Break

Part II: word representation

Application 1: How to find a book in library?

Application 2: How to search?



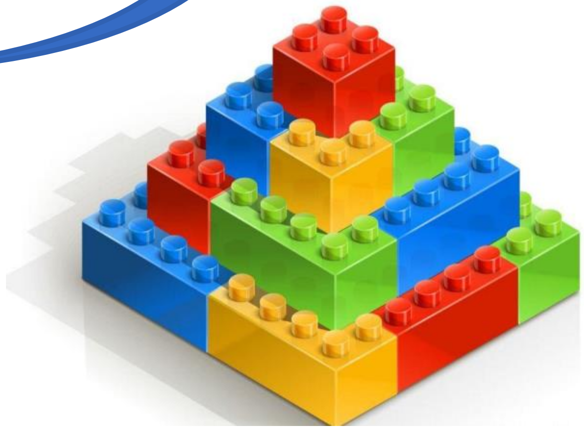
https://www.tripadvisor.com/Restaurants-g297415-c24-Shenzhen_Guangdong.html

Application 3：Talk to your personal assistant

For NLP

- Building blocks
  - Words/subwords/tokens
- Building strategies
  - Neural network
  - Pre-trained objective
    - Language modeling

Side product



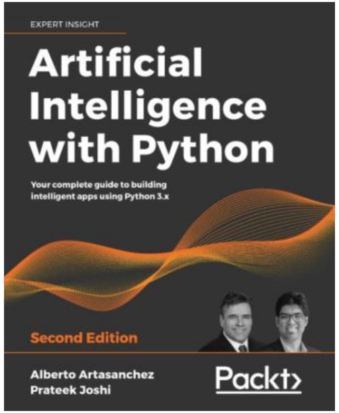
Contents

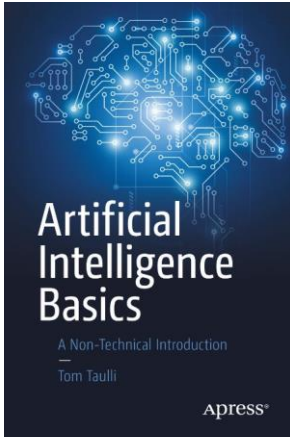
- Word Representations
  - Understanding Words from the perspective of Information retrieval
  - How do we represent words in NLP models?
  - Distributional hypothesis
  - Sparse vs dense vectors
- Word2vec and other variants
  - Word2vec
  - How to train this model?
  - Skip-gram with negative sampling (SGNS) and other variants
  - Evaluating word embeddings

First, we need to know how to represent words

A simple practice: term matching

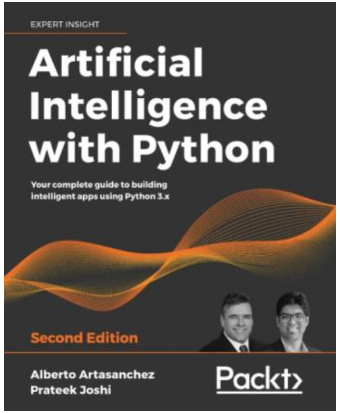
- Find books with **AI** in the title using term matching

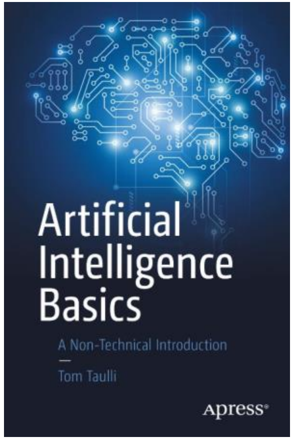




Have the word or not?

2nd Question: Which book is more relevant?



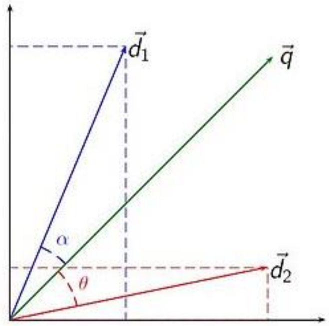


Which one do you prefer if all book have “AI” in their titles

Vector Space Model:

Representation:
Documents and queries are represented as vectors in some space.
-- What are the dimensions?
-- How to map documents and queries to this space?

Scoring Function:
Rank documents by a measure of “closeness” of query and document vectors
-- Distance, Cosine of the angle etc.



$$score(d, q) = \frac{\vec{d} \cdot \vec{q}}{||\vec{d}|| \ ||\vec{q}||}$$

Image from: http://en.wikipedia.org/wiki/Vector_space_model

Vector Space Model: TF-IDF Model

Dimensions: Every word becomes a dimension.

$$D = \langle x_1, x_2, \ldots, x_{|V|} \rangle$$

Term Frequency:

A term that occurs many times in a document is important.

$$tf(t, D) = \log\left(1 + \#(t, D)\right)$$

Inverse Document Frequency:

A term that occurs in many documents is unimportant.

$$idf(t) = \log\left(\frac{N}{|D_t|}\right)$$

TF-IDF:

Term-frequency weighted by inverse document frequency.

$$tfidf(t, D) = tf(t, D) \times idf(t)$$

Vector Space Models: Enhancements

- Efficient scoring
  - Inverted indices – compression, skip lists, etc.
  - Top-k documents without scoring all documents entirely.
- Scoring methods
  - Document length normalizations (e.g., pivoted)
  - BM25 and various other heuristics
- Dimensionality reduction
  - Stemming, phrase representations etc.
  - LSI

Latent Semantic Analysis

Premise

Using one-dimension per word is problematic.

Doesn't scale.
(millions of words in large collections)

Doesn't handle synonymy
(dog vs. canine)

Idea

Project documents and queries into a **lower-dimensional space**.
Instead of a |V| dimensional vector each word will be represented by a **k << |V|** dimensional vector.

Latent Semantic Analysis

Original term-by-document matrix

	D1	D2	D3	D4	D5	D6	Q1
rock	2	1	0	2	0	1	1
granite	1	0	1	0	0	0	0
marble	1	2	0	0	0	0	1
music	0	0	0	1	2	0	0
song	0	0	0	1	0	2	0
band	0	0	0	0	1	0	0

Documents projected into 2D semantic space

	D1	D2	D3	D4	D5	D6	Q1
Dim. 1	-0.888	-0.759	-0.615	-0.961	-0.388	-0.851	-0.845
Dim. 2	0.460	0.652	0.789	-0.276	-0.922	-0.525	0.534

1) Represent co-occurrences as a term-document TD matrix (X).

2) Use singular-value decomposition to factorize TD.

$$X = U \Sigma V^T$$

3) Sigma is a diagonal matrix. Take the top k *singular values.*

4) Project query and documents using the top-k *singular* values.

$$\hat{d} = \Sigma_k^{-1} U_k^T d \quad \hat{q} = \Sigma_k^{-1} U_k^T q$$

5) Cosine similarity to score documents.

Figures stolen from: http://www.cs.cmu.edu/~nasmith/LS2/gimpel.06.pdf

A bigger difference

$$X = U\Sigma V^T$$

Original term-by-document matrix

	D1	D2	D3	D4	D5	D6	Q1
rock	2	1	0	2	0	1	1
granite	1	0	1	0	0	0	0
marble	1	2	0	0	0	0	1
music	0	0	0	1	2	0	0
song	0	0	0	1	0	2	0
band	0	0	0	0	1	0	0

Documents projected into 2D semantic space

	D1	D2	D3	D4	D5	D6	Q1
Dim. 1	-0.888	-0.759	-0.615	-0.961	-0.388	-0.851	-0.845
Dim. 2	0.460	0.652	0.789	-0.276	-0.922	-0.525	0.534

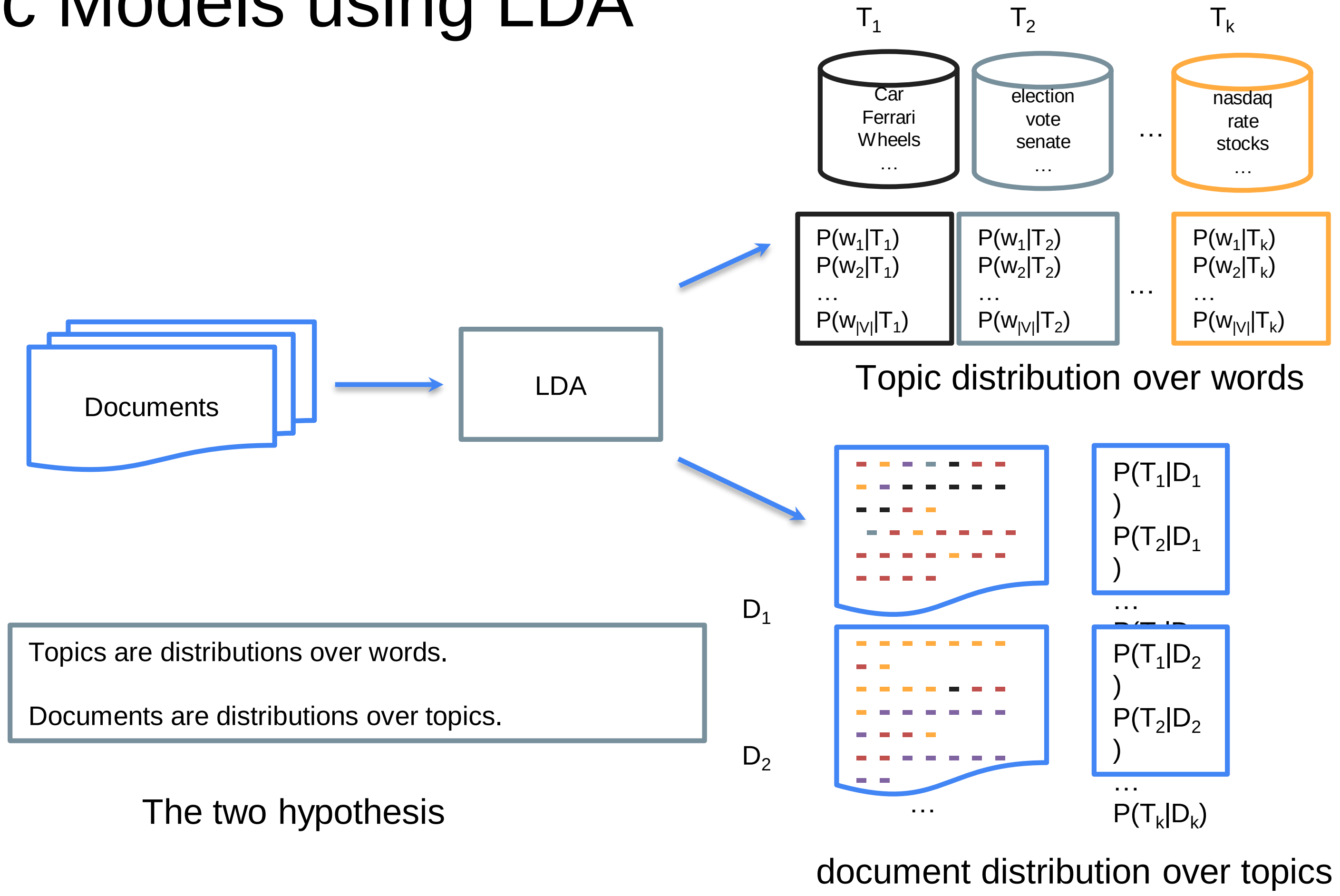
It does not rely on term match;
Even two documents could **be relevant even they do not share any common words**

V^T *implies* a set word vectors, later we will explain it.

Latent Semantic Analysis

- Benefits
  - Addresses synonymy
  - Dimensionality reduction
- Issues
  - Efficient SVD implementations necessary.
    - Many implementations available.
  - New documents need to be handled in a special way.
  - Disconnect w/ retrieval performance.
  - Rely on **direct counting**
- Extensions
  - Probabilistic Latent Semantic Analysis (pre-cursor to LDA).
  - Hierarchical (H-PLSA)

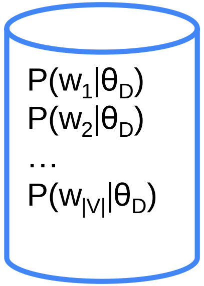
Topic Models using LDA



Probabilistic Retrieval Models: Language Modeling

Generative Assumption

Assume that each document is generated by a probabilistic process specified by a **multinomial distribution.**



Document model parameterized by θ

Query-likelihood Model

If a document D is relevant to a query, then the query should be a high probability sample from the document multinomial.

Later termed as
“uni-gram” **language model.**

Probabilistic Retrieval Models: Query-likelihood Model

rank each document by the probability of specific documents given a query

$$P(d|q) = \frac{P(q|d)P(d)}{P(q)}$$

Since the probability of the query P(q) is the same for all documents, this can be ignored. Further, it is typical to assume that the probability of documents is uniform. Thus, P(d) is also ignored.

$$P(d|q) \propto P(q|d)$$

Probabilistic Retrieval Models: Query-likelihood Model

Rank by the probability that the document model generated the query.

$P(q|Q)$

$$P(Q \mid D) = \prod_{q \in Q} P(q \mid \theta_D)$$

Query model

Document model

Probabilistic Retrieval Models: Estimation

Estimate document model via maximum-likelihood

-- Assume document is a sample of the underlying distribution.

$$\theta_D(w) = \frac{\#(w, D)}{\sum_{w_i \in D} \#(w_i, D)}$$

Zero probability issue! Smoothing could help!

Today's lecture

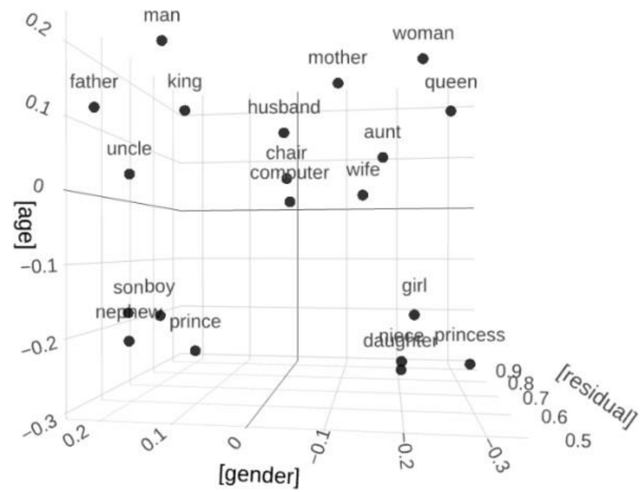
- Word Representations
  - Understanding Words from the perspective of Information retrieval
  - How do we represent words in NLP models?
  - Distributional hypothesis
  - Sparse vs dense vectors
- Word2vec and other variants
  - Word2vec
  - How to train this model?
  - Skip-gram with negative sampling (SGNS) and other variants
  - Evaluating word embeddings

The big idea: model of meaning focusing on similarity

$$v_{\text{cat}} = \begin{pmatrix} -0.224 \\ 0.130 \\ -0.290 \\ 0.276 \end{pmatrix} \qquad v_{\text{dog}} = \begin{pmatrix} -0.124 \\ 0.430 \\ -0.200 \\ 0.329 \end{pmatrix}$$

$$v_{\text{the}} = \begin{pmatrix} 0.234 \\ 0.266 \\ 0.239 \\ -0.199 \end{pmatrix} \qquad v_{\text{language}} = \begin{pmatrix} 0.290 \\ -0.441 \\ 0.762 \\ 0.982 \end{pmatrix}$$

Similar words are “**nearby in the vector space**”



(Bandyopadhyay et al. 2022)

How do we represent words in NLP models?

- Logistic regression

string match

Var	Definition	Value in Fig. 5.2
x_1	count(positive lexicon) $\in$ doc)	3
x_2	count(negative lexicon) $\in$ doc)	2
x_3	$\begin{cases} 1 & \text{if "no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns $\in$ doc)	3
x_5	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	log(word count of doc)	$\ln(64) = 4.15$

What do words mean?

- **Synonyms**: couch/sofa, car/automobile, filbert/hazelnut
- **Antonyms**: dark/light, rise/fall, up/down
- Some words are not synonyms but they share some element of meaning
  - cat/dog, car/bicycle, cow/horse
- Some words are not similar but they are **related**
  - coffee/cup, house/door, chef/menu
- **Affective meanings** or **connotations**:

vanish	disappear	9.8
belief	impression	5.95
muscle	bone	3.65
modest	flexible	0.98
hole	agreement	0.3

SimLex-999

	Valence	Arousal	Dominance
courageous	8.05	5.5	7.38
music	7.67	5.57	6.5
heartbreak	2.45	5.65	3.58
cub	6.71	3.95	4.24

(Osgood et al., 1957)

valence: the pleasantness of the stimulus

arousal: the intensity of emotion provoked by the stimulus

dominance: the degree of control exerted by the stimulus

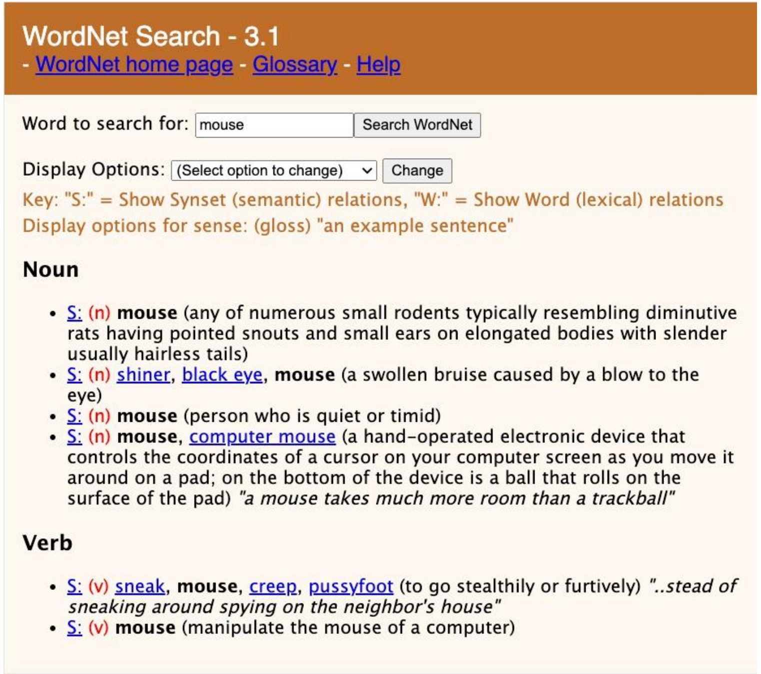
Why word meaning in NLP models?

- With words, a feature is a word identity (= string)
  - Feature 5: `The previous word was "terrible"'
  - Requires **exact same word** to be in the training and testing set

"terrible" ≠ "horrible"

- If we can represent word meaning in vectors:
  - The previous word was vector [35, 22, 17, …]
  - Now in the test set we might see a similar vector [34, 21, 14, …]
  - We can generalize to **similar but unseen** words!!!

Lexical resources



WordNet Search - 3.1
- WordNet home page - Glossary - Help

Word to search for: mouse [Search WordNet]

Display Options: (Select option to change) [Change]

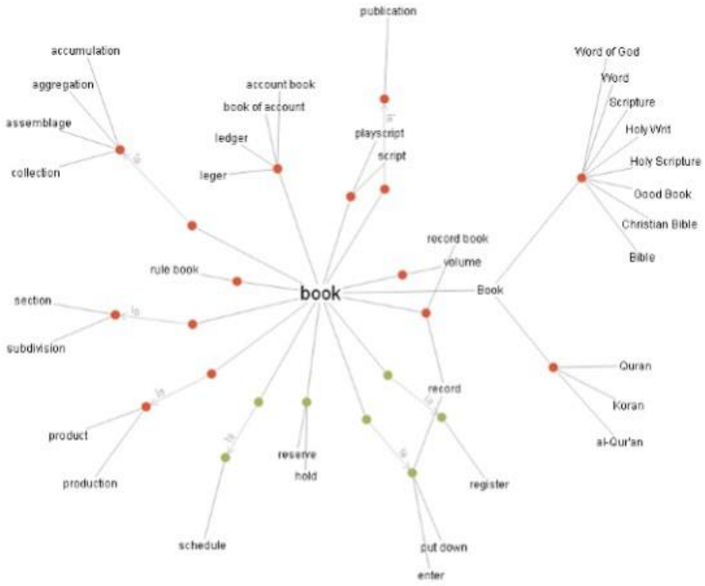
Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations
Display options for sense: (gloss) "an example sentence"

Noun

- S: (n) **mouse** (any of numerous small rodents typically resembling diminutive rats having pointed snouts and small ears on elongated bodies with slender usually hairless tails)
- S: (n) shiner, black eye, **mouse** (a swollen bruise caused by a blow to the eye)
- S: (n) **mouse** (person who is quiet or timid)
- S: (n) **mouse**, computer mouse (a hand-operated electronic device that controls the coordinates of a cursor on your computer screen as you move it around on a pad; on the bottom of the device is a ball that rolls on the surface of the pad) *"a mouse takes much more room than a trackball"*

Verb

- S: (v) sneak, **mouse**, creep, pussyfoot (to go stealthily or furtively) *"..stead of sneaking around spying on the neighbor's house"*
- S: (v) **mouse** (manipulate the mouse of a computer)



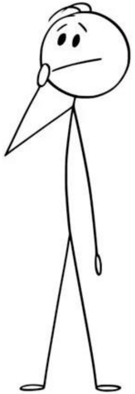
http://wordnetweb.princeton.edu/

(-) Huge amounts of human labor to create and maintain

Contents

- Word Representations
    - Understanding Words from the perspective of Information retrieval
    - How do we represent words in NLP models?
    - Distributional hypothesis
    - Sparse vs dense vectors
- Word2vec and other variants
    - Word2vec
    - How to train this model?
    - Skip-gram with negative sampling (SGNS) and other variants
    - Evaluating word embeddings

Distributional hypothesis



- “The meaning of a word is its use in the language”
- “If A and B have almost identical environments we say that they are synonyms.”
- “You shall know a word by the company it keeps”

[Wittgenstein PI 43]

[Harris 1954]

[Firth 1957]

Distributional hypothesis

Distributional hypothesis: words that occur in similar **contexts** tend to have similar meanings

J.R.Firth 1957

- “You shall know a word by the company it keeps”
- One of the most successful ideas of modern statistical NLP!

When a word *w* appears in a text, its context is the set of words that appear nearby (within a fixed-size window).

…government debt problems turning into	***banking***	*crises as happened in 2009…*
…saying that Europe needs unified	***banking***	*regulation to replace the hodgepodge…*
…India has just given its	***banking***	*system a shot in the arm…*

These context words will represent “*banking”.*

Distributional hypothesis

“Ongchoi”

Ongchoi is delicious sautéed with garlic

Ongchoi is superb over rice

Ongchoi leaves with salty sauces

Distributional hypothesis

“Ongchoi”

Ongchoi is delicious sautéed with garlic

Ongchoi is superb over rice

Ongchoi leaves with salty sauces

Q: What do you think ‘Ongchoi’ means?

A) a savory snack
B) a green vegetable
C) an alcoholic beverage
D) a cooking sauce

Distributional hypothesis

“Ongchoi”

Ongchoi is delicious sautéed with garlic

Ongchoi is superb over rice

Ongchoi leaves with salty sauces

You may have seen these sentences before:

spinach **sautéed with garlic over rice** chard stem

and **leaves** are **delicious** collard greens and other

salty leafty greens

Distributional hypothesis

“Ongchoi”

Ongchoi is a leafty green like spinach, chard or collard greens

空心菜
kangkong
rau muống
…

How can do the same thing computationally?

- Count the words in the context of ongchoi
- See what other words occur in those contexts

We can represent a word’s context using vectors!

Contents

- Word Representations
  - Understanding Words from the perspective of Information retrieval
  - How do we represent words in NLP models?
  - Distributional hypothesis
  - Sparse vs dense vectors
- Word2vec and other variants
  - Word2vec
  - How to train this model?
  - Skip-gram with negative sampling (SGNS) and other variants
  - Evaluating word embeddings

Words and vectors

Q: What is the dimension of each such vector?

A: |V|

First solution: Let's use **word-word co-occurrence counts** to represent the meaning of words!

Each word is represented by the corresponding **row vector**

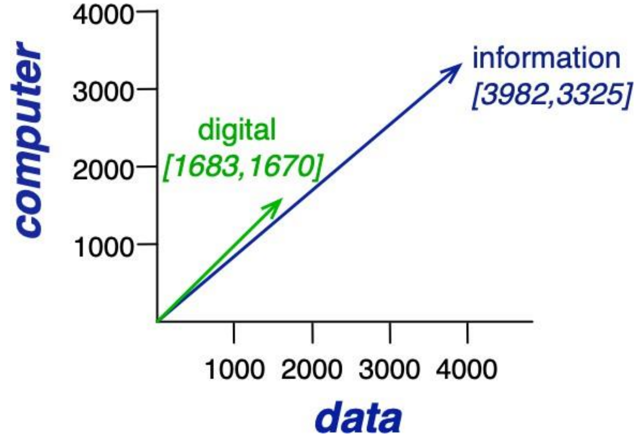
context words:
4 words to the left +
4 words to the right

is traditionally followed by	**cherry**	pie, a traditional dessert
often mixed, such as	**strawberry**	rhubarb pie. Apple pie
computer peripherals and personal	**digital**	assistants. These devices usually
a computer. This includes	**information**	available on the internet

	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...

Most entries are 0s $\Longrightarrow$ sparse vectors

Measuring similarity



A common similarity metric: **cosine** of the angle between the two vectors (the larger, the more similar the two vectors

$$\cos(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \|\mathbf{v}\|}$$

$$\cos(\mathbf{u}, \mathbf{v}) = \frac{\sum_{i=1}^{|V|} u_i v_i}{\sqrt{\sum_{i=1}^{|V|} u_i^2} \sqrt{\sum_{i=1}^{|V|} v_i^2}}$$

Q: Why cosine similarity instead of dot product $\mathbf{u} \cdot \mathbf{v}$?

Any issues with this model?

Raw frequency count is a bad representation!

- Frequency is clearly useful; if “pie” appears a lot near “cherry”, that's useful information.
- But overly frequent words like “the”, “it", or “they” also appear a lot near “cherry”. They are not very informative about the context.

Sparse vs dense vectors

- The vectors in the word-word occurrence matrix are
  - **Long**: vocabulary size
  - **Sparse**: most are 0's
- Alternative: we want to represent words as **short** (50-300 dimensional) & **dense** (real-valued) vectors
  - The basis for modern NLP systems

$$v_{\text{cat}} = \begin{pmatrix} -0.224 \\ 0.130 \\ -0.290 \\ 0.276 \end{pmatrix} \quad v_{\text{dog}} = \begin{pmatrix} -0.124 \\ 0.430 \\ -0.200 \\ 0.329 \end{pmatrix}$$

$$v_{\text{the}} = \begin{pmatrix} 0.234 \\ 0.266 \\ 0.239 \\ -0.199 \end{pmatrix} \quad v_{\text{language}} = \begin{pmatrix} 0.290 \\ -0.441 \\ 0.762 \\ 0.982 \end{pmatrix}$$

Why dense vectors?

- Short vectors are easier to use as **features** in ML systems
- Dense vectors generalize better than explicit counts (points in real space vs points in integer space)
- Sparse vectors can't capture higher-order co-occurrence
  - w_1 co-occurs with "car", w_2 co-occurs with "automobile"
  - They should be similar but they aren't because "car" and "automobile" are distinct dimensions
- In practice, they work better!

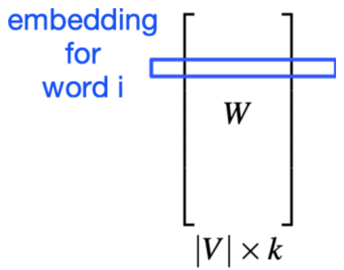
How to get short dense vectors?

- **Count-based methods**: Singular value decomposition (SVD) of count matrix

Singular value decomposition (SVD) of PPMI weighted co-occurrence matrix

$$\underset{|V|\times|V|}{\begin{bmatrix} \\ X \\ \\ \end{bmatrix}} = \underset{|V|\times|V|}{\begin{bmatrix} \\ W \\ \\ \end{bmatrix}} \underset{|V|\times|V|}{\begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_V \end{bmatrix}} \underset{|V|\times|V|}{\begin{bmatrix} \\ C \\ \\ \end{bmatrix}}$$

$$\underset{|V|\times|V|}{\begin{bmatrix} \\ X \\ \\ \end{bmatrix}} = \underset{|V|\times k}{\begin{bmatrix} \\ W \\ \\ \end{bmatrix}} \underset{k\times k}{\begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_k \end{bmatrix}} \begin{bmatrix} C \\ k\times|V| \end{bmatrix}$$



We can approximate the full matrix by only keeping the top k (e.g., 100) singular values!

How to get short dense vectors?

- **Count-based methods**: Singular value decomposition (SVD) of count matrix

- **Prediction-based methods**:
  - Vectors are created by training a classifier to predict whether a word c ("pie") is likely to appear in the context of a word w ("cherry")
  - Examples: word2vec (Mikolov et al., 2013), Glove (Pennington et al., 2014), FastText (Bojanowski et al., 2017)

***Don't count, predict!* A systematic comparison of context-counting vs. context-predicting semantic vectors**

Marco Baroni and **Georgiana Dinu** and **Germán Kruszewski**
Center for Mind/Brain Sciences (University of Trento, Italy)

(Baroni et al., 2014)

Also called word **embeddings!**

Contents

- Word Representations
  - Understanding Words from the perspective of Information retrieval
  - How do we represent words in NLP models?
  - Distributional hypothesis
  - Sparse vs dense vectors
- Word2vec and other variants
  - Word2vec
  - How to train this model?
  - Skip-gram with negative sampling (SGNS) and other variants
  - Evaluating word embeddings

Word embeddings

Goal: represent words as **short** (50-300 dimensional) & **dense** (real-valued) vectors

Count-based approaches

- Used since the 90s
- Sparse word-word co-occurrence PPMI matrix
- Decomposed with SVD

Prediction-based approaches

- Formulated as a machine learning problem
- Word2vec (Mikolov et al., 2013)
- GloVe (Pennington et al., 2014)

Underlying theory: Distributional Hypothesis *(Firth, '57)*

"Similar words occur in similar contexts"

Word embeddings: the learning problem

Learning vectors from text for representing words

- **Input**: a large text corpus, vocabulary *V*, vector dimension d (e.g., 300)
- **Output**: $f : V \to \mathbb{R}^d$

Each coordinate/dimension of the vector doesn't have a particular interpretation

$$v_{\text{cat}} = \begin{pmatrix} -0.224 \\ 0.130 \\ -0.290 \\ 0.276 \end{pmatrix} \qquad v_{\text{dog}} = \begin{pmatrix} -0.124 \\ 0.430 \\ -0.200 \\ 0.329 \end{pmatrix}$$

$$v_{\text{the}} = \begin{pmatrix} 0.234 \\ 0.266 \\ 0.239 \\ -0.199 \end{pmatrix} \qquad v_{\text{language}} = \begin{pmatrix} 0.290 \\ -0.441 \\ 0.762 \\ 0.982 \end{pmatrix}$$

Word embeddings

- Basic property: similar words have similar vectors

word w*= "sweden"

$$\arg\max_{w \in V} \cos(e(w), e(w^*))$$

Word	Cosine distance
norway	0.760124
denmark	0.715460
finland	0.620022
switzerland	0.588132
belgium	0.585835
netherlands	0.574631
iceland	0.562368
estonia	0.547621
slovenia	0.531408

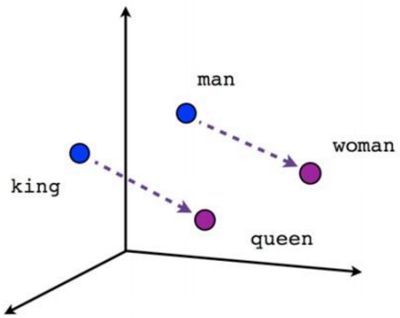
$\cos(u, v)$ ranges between -1 and 1

Word embeddings

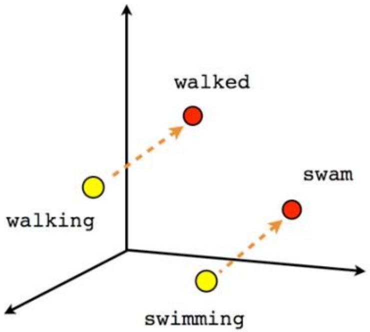
ACL'19 **Towards Understanding Linear Word Analogies**

Kawin Ethayarajh, David Duvenaud[†], Graeme Hirst
University of Toronto
[†]Vector Institute
{kawin, duvenaud, gh}@cs.toronto.edu

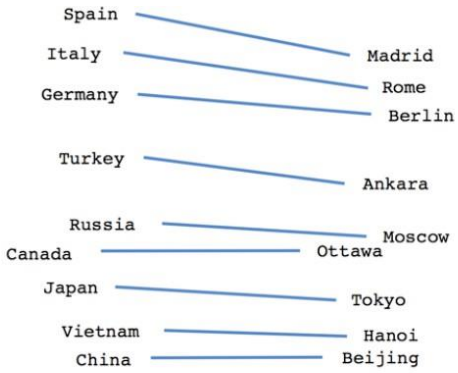
- They have some other nice properties too!



Male-Female



Verb tense



Country-Capital

$$v_{\text{man}} - v_{\text{woman}} \approx v_{\text{king}} - v_{\text{queen}}$$

$$v_{\text{Paris}} - v_{\text{France}} \approx v_{\text{Rome}} - v_{\text{Italy}}$$

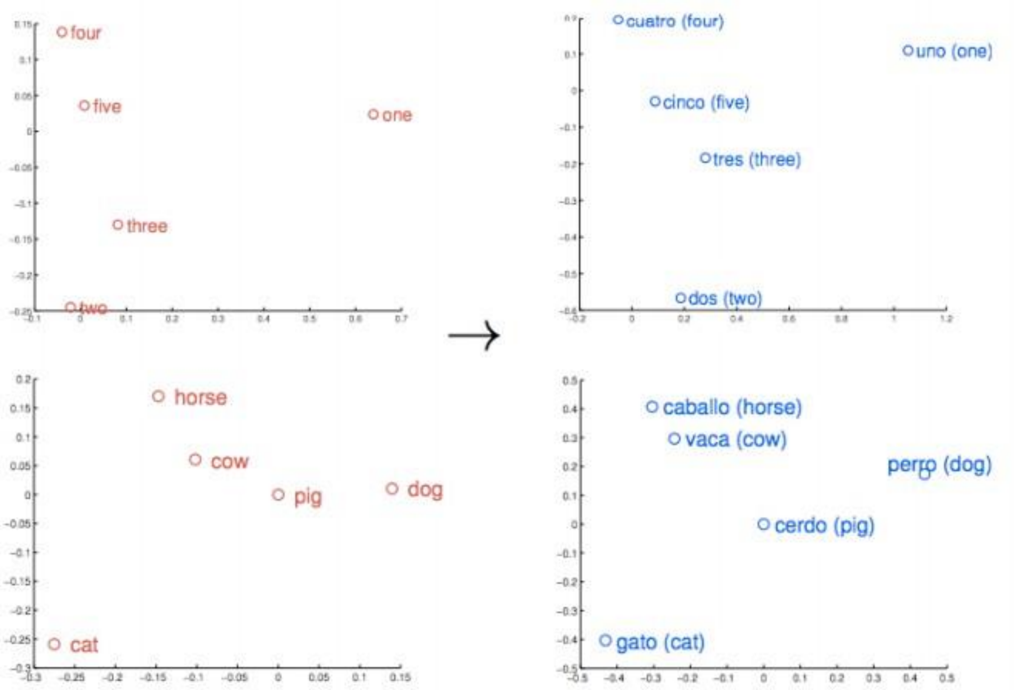
Word analogy test: $a : a^* :: b : b^*$

$$b^* = \arg\max_{w \in V} \cos(e(w), e(a^*) - e(a) + e(b))$$

Word embeddings

- They have some other nice properties too!

$$v(\text{cuatro}) \approx W v(\text{four})$$



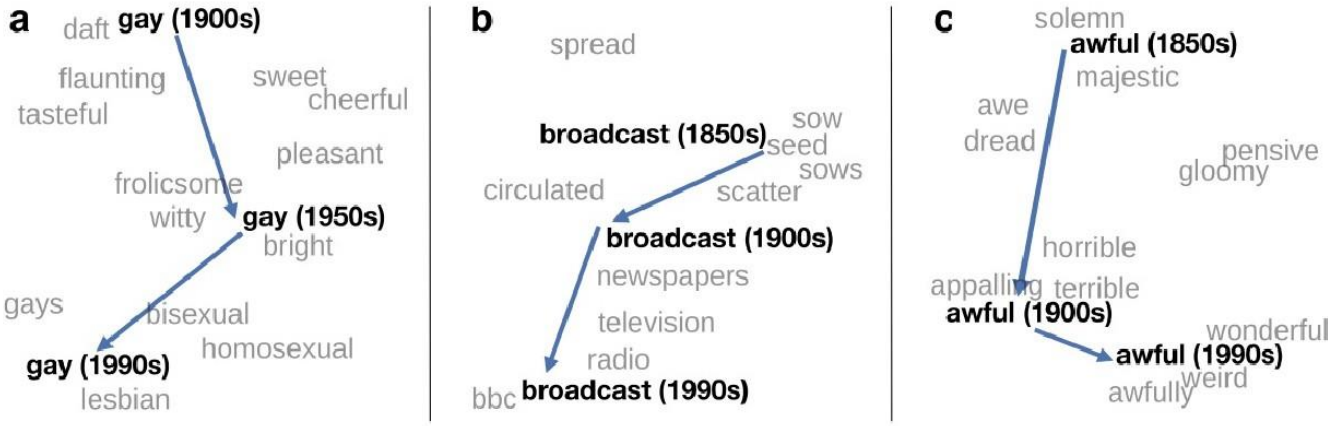
(Mikolov et al, 2013): Exploiting Similarities among Languages for Machine Translation

The word "cuatro" comes from the Latin word *quattuor*, which means "four"

Embeddings as a window onto historical semantics

Train embeddings on different decades of historical text to see meanings shift

~30 million books, 1850-1990, Google Books data



William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change. Proceedings of ACL.

Embeddings reflect cultural bias!

Bolukbasi, Tolga, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam T. Kalai. "Man is to computer programmer as woman is to homemaker? debiasing word embeddings." In *NeurIPS*, pp. 4349-4357. 2016.

Ask "Paris : France :: Tokyo : x"

- x = Japan

Ask "father : doctor :: mother : x"

- x = nurse

Ask "man : computer programmer :: woman : x"

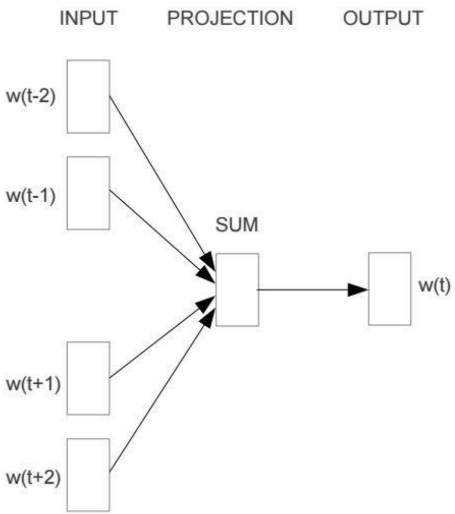
- x = homemaker

Algorithms that use embeddings as part of e.g., hiring searches for programmers, might lead to bias in hiring

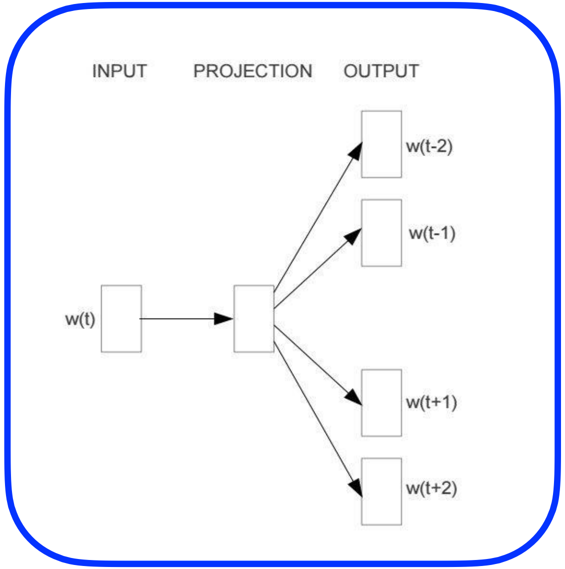
word2vec

- (Mikolov et al 2013a): Efficient Estimation of Word Representations in Vector Space
- (Mikolov et al 2013b): Distributed Representations of Words and Phrases and their Compositionality

Thomas
Mikolov



Continuous Bag of Words
(CBOW)

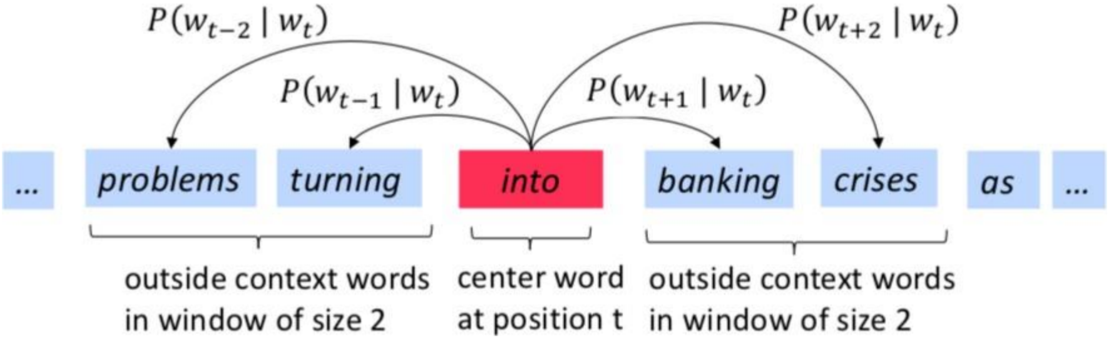


Skip-gram

Skip-gram

- Assume that we have a large corpus $w_1, w_2, \ldots, w_T \in V$
- **Key idea:** Use each word to **predict** other words in its context
- Context: a fixed window of size 2*m* (m = 2 in the example)

A classification problem!

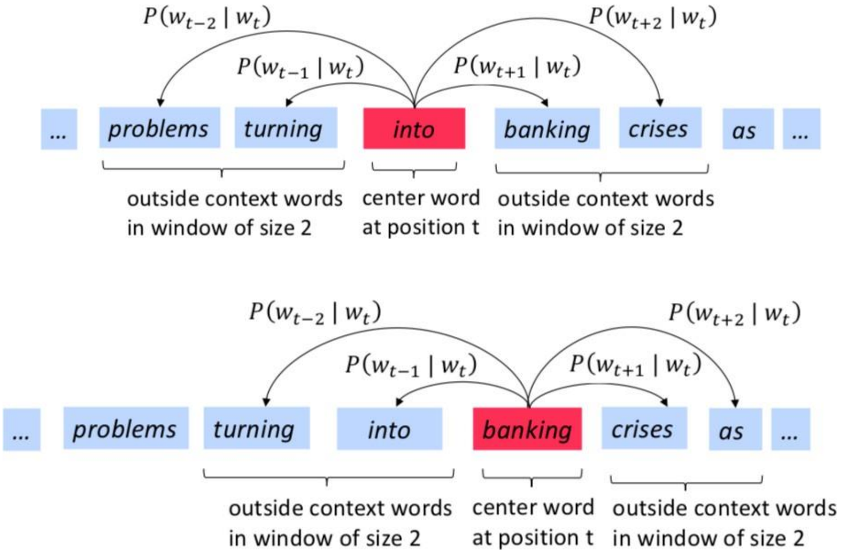


$P(b \mid a)$ = given the center word is a, what is the probability that b is a context word?

$P(\cdot \mid a)$ is a probability distribution defined over V: $\sum_{w \in V} P(w \mid a) = 1$

We are going to define this distribution soon!

Skip-gram



Convert the training data into:
(into, problems)
(into, turning)
(into, banking)
(into, crises)
(banking, turning)
(banking, into)
(banking, crises)
(banking, as)
…

Our goal is to find parameters that can maximize

P(problems | into) × P(turning | into) × P(banking | into) × P(crises | into) × P(turning | banking) × P(into | banking) × P(crises | banking) × P(as | banking)...

Skip-gram: objective function

- For each position $t = 1,2,\ldots T$, predict context words within context size m, given center word w_t:

all the parameters to be optimized

$$\mathcal{L}(\theta) = \prod_{t=1}^{T} \prod_{-m \leq j \leq m, j \neq 0} P(w_{t+j} \mid w_t; \theta)$$

- It is equivalent as minimizing the (average) negative log likelihood:

$$J(\theta) = -\frac{1}{T} \log \mathcal{L}(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{-m \leq j \leq m, j \neq 0} \log P(w_{t+j} \mid w_t; \theta)$$

How to define $P(w_{t+j} \mid w_t; \theta)$?

- Use two sets of vectors for each word in the vocabulary

$\mathbf{u}_a \in \mathbb{R}^d$: vector for center word a , $\forall a \in V$

$\mathbf{v}_b \in \mathbb{R}^d$: vector for context word b , $\forall b \in V$

- Use inner product $\mathbf{u}_a \cdot \mathbf{v}_b$ to measure how likely word *a* appears with context word *b*

Normalize it as a probability distribution.

$$P(w_{t+j} \mid w_t) = \frac{\exp(\mathbf{u}_{w_t} \cdot \mathbf{v}_{w_{t+j}})}{\sum_{k \in V} \exp(\mathbf{u}_{w_t} \cdot \mathbf{v}_k)}$$

Recall that $P(\cdot \mid a)$ is a probability distribution defined over V…

Important note

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0}\log\frac{\exp(\mathbf{u}_{w_t}\cdot\mathbf{v}_{w_{t+j}})}{\sum_{k\in V}\exp(\mathbf{u}_{w_t}\cdot\mathbf{v}_k)}$$

- The key point is that the *parameters* used to optimize this training objective—when the training corpus is large enough—can give us very good representations of words (following the principle of distributional hypothesis)!

How many parameters in this model?

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0}\log\frac{\exp(\mathbf{u}_{w_t}\cdot\mathbf{v}_{w_{t+j}})}{\sum_{k\in V}\exp(\mathbf{u}_{w_t}\cdot\mathbf{v}_k)}$$

How many parameters does this model have (i.e. what is size of)?

(a) $d|V|$

(b) $2d|V|$

(c) $2m|V|$

(d) $2md|V|$

d = dimension of each vector

How many parameters in this model?

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0}\log\frac{\exp(\mathbf{u}_{w_t}\cdot\mathbf{v}_{w_{t+j}})}{\sum_{k\in V}\exp(\mathbf{u}_{w_t}\cdot\mathbf{v}_k)}$$

How many parameters does this model have (i.e. what is size of)?

(a) $d|V|$

(b) $2d|V|$

(c) $2m|V|$

(d) $2md|V|$

d = dimension of each vector

The answer is (b).

Each word has two d-dimensional vectors, so it is $2 \times |V| \times d$.

word2vec formulation

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0}\log\frac{\exp(\mathbf{u}_{w_t}\cdot\mathbf{v}_{w_{t+j}})}{\sum_{k\in V}\exp(\mathbf{u}_{w_t}\cdot\mathbf{v}_k)}$$

Q: Why do we need two vectors for each word instead of one?

A: because one word is not likely to appear in its own context window, e.g., $P(\text{dog} \mid \text{dog})$ should be low. If we use one set of vectors only, it essentially needs to minimize $\mathbf{u}_{\text{dog}} \cdot \mathbf{u}_{\text{dog}}$..

Q: Which set of vectors are used as word embeddings?

A: This is an empirical question. Typically just $\mathbf{u}_w$ but you can also concatenate the two vectors..

Contents

Contents

- Word Representations
    - Understanding Words from the perspective of Information retrieval
    - How do we represent words in NLP models?
    - Distributional hypothesis
    - Sparse vs dense vectors
- Word2vec and other variants
    - Word2vec
    - How to train this model?
    - Skip-gram with negative sampling (SGNS) and other variants
    - Evaluating word embeddings

Skip-gram with negative sampling (SGNS)

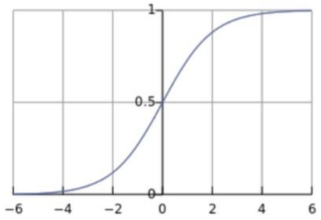
Problem: every time you get one pair of (t, c), you need to update $\mathbf{v}_k$ with all the words in the vocabulary! This is very expensive computationally.

$$\frac{\partial y}{\partial \mathbf{u}_t} = -\mathbf{v}_c + \sum_{k \in V} P(k \mid t)\mathbf{v}_k$$

$$\frac{\partial y}{\partial \mathbf{v}_k} = \begin{cases} (P(k \mid t) - 1)\,\mathbf{u}_t & k = c \\ P(k \mid t)\mathbf{u}_t & k \neq c \end{cases}$$

Negative sampling: instead of considering all the words in V, let's randomly sample K (5-20) negative examples.

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$



softmax: $$y = -\log\left(\frac{\exp(\mathbf{u}_t \cdot \mathbf{v}_c)}{\sum_{k \in V} \exp(\mathbf{u}_t \cdot \mathbf{v}_k)}\right)$$

Negative sampling: $$y = -\log(\sigma(\mathbf{u}_t \cdot \mathbf{v}_c)) - \sum_{i=1}^{K} \mathbb{E}_{j \sim P(w)} \log(\sigma(-\mathbf{u}_t \cdot \mathbf{v}_j))$$

Skip-gram with negative sampling (SGNS)

Key idea: Convert the $|V|$ -way classification into a set of binary classification tasks.

Every time we get a pair of words (*t*, *c*), we don't predict *c* among all the words in the vocabulary. Instead, we predict (*t*, *c*) is a positive pair, and (*t*, *c'*) is a negative pair for a small number of sampled *c'*.

positive examples +

t	c
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

negative examples -

t	c	t	c
apricot	aardvark	apricot	seven
apricot	my	apricot	forever
apricot	where	apricot	dear
apricot	coaxial	apricot	if

$$y = -\log(\sigma(\mathbf{u}_t \cdot \mathbf{v}_c)) - \sum_{i=1}^{K} \mathbb{E}_{j \sim P(w)} \log(\sigma(-\mathbf{u}_t \cdot \mathbf{v}_j))$$

P(w): sampling according to the frequency of words

Similar to **binary logistic regression**, but we need to optimize and together.

$$P(y = 1 \mid t, c) = \sigma(\mathbf{u}_t \cdot \mathbf{v}_c) \qquad p(y = 0 \mid t, c') = 1 - \sigma(\mathbf{u}_t \cdot \mathbf{v}_{c'}) = \sigma(-\mathbf{u}_t \cdot \mathbf{v}_{c'})$$

Understanding SGNS

$$y = -\log(\sigma(\mathbf{u}_t \cdot \mathbf{v}_c)) - \sum_{i=1}^{K} \mathbb{E}_{j \sim P(w)} \log(\sigma(-\mathbf{u}_t \cdot \mathbf{v}_j))$$

In skip-gram with negative sampling (SGNS), how many parameters need to be updated in θ for every (t, c) pair?

(a) Kd

(b) $2Kd$

(c) $(K+1)d$

(d) $(K+2)d$

Understanding SGNS

$$y = -\log(\sigma(\mathbf{u}_t \cdot \mathbf{v}_c)) - \sum_{i=1}^{K} \mathbb{E}_{j \sim P(w)} \log(\sigma(-\mathbf{u}_t \cdot \mathbf{v}_j))$$

In skip-gram with negative sampling (SGNS), how many parameters need to be updated in θ for every (t, c) pair?

(a) Kd

(b) $2Kd$

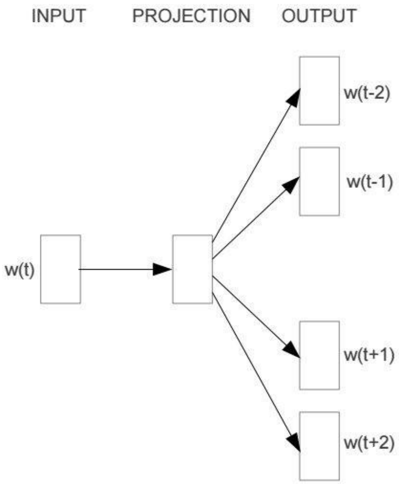
(c) $(K+1)d$

(d) $(K+2)d$

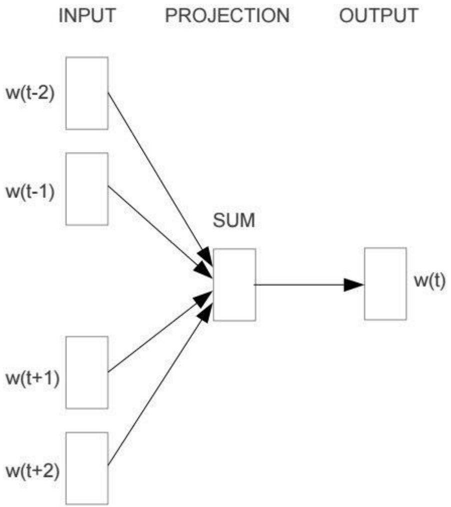
The answer is (d).

We need to calculate gradients with respect to $\mathbf{u}_t$ and (K + 1) $\mathbf{v}_i$ (one positive and K negatives).

Continuous Bag of Words (CBOW)



Skip-gram



Continuous Bag of Words (CBOW)

$$L(\theta) = \prod_{t=1}^{T} P\left(w_t \mid \{w_{t+j}\}, -m \leq j \leq m, j \neq 0\right)$$

$$\bar{\mathbf{v}}_t = \frac{1}{2m} \sum_{-m \leq j \leq m, j \neq 0} \mathbf{v}_{t+j}$$

$$P(w_t \mid \{w_{t+j}\}) = \frac{\exp(\mathbf{u}_{w_t} \cdot \bar{\mathbf{v}}_t)}{\sum_{k \in V} \exp(\mathbf{u}_k \cdot \bar{\mathbf{v}}_t)}$$

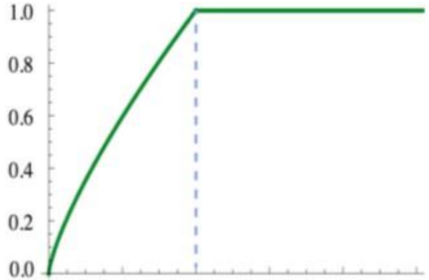
GloVe: Global Vectors

- Key idea: let's approximate $\mathbf{u}_i \cdot \mathbf{v}_j$ using their co-occurrence counts directly
- Take the global co-occurrence statistics: $X_{i,j}$

$$J(\theta) = \sum_{i,j \in V} f(X_{i,j}) \left(\mathbf{u}_i \cdot \mathbf{v}_j + b_i + \tilde{b}_j - \log X_{i,j} \right)^2$$

- Training faster
- Scalable to very large corpora

$f \sim$



(Pennington et al, 2014): GloVe: Global Vectors for Word Representation

Trained word embeddings available

- word2vec: https://code.google.com/archive/p/word2vec/
- GloVe: https://nlp.stanford.edu/projects/glove/
- FastText: https://fasttext.cc/

Download pre-trained word vectors

- Pre-trained word vectors. This data is made available under the Public Domain Dedication and License v1.0 whose full text can be found at: http://www.opendatacommons.org/licenses/pddl/1.0/.
  - Wikipedia 2014 + Gigaword 5 (6B tokens, 400K vocab, uncased, 50d, 100d, 200d, & 300d vectors, 822 MB download): glove.6B.zip
  - Common Crawl (42B tokens, 1.9M vocab, uncased, 300d vectors, 1.75 GB download): glove.42B.300d.zip
  - Common Crawl (840B tokens, 2.2M vocab, cased, 300d vectors, 2.03 GB download): glove.840B.300d.zip
  - Twitter (2B tweets, 27B tokens, 1.2M vocab, uncased, 25d, 50d, 100d, & 200d vectors, 1.42 GB download): glove.twitter.27B.zip
- Ruby script for preprocessing Twitter data

Differ in algorithms, text corpora, dimensions, cased/uncased…

Applied to many other languages

Easy to use!

```
from gensim.models import KeyedVectors
# Load vectors directly from the file
model = KeyedVectors.load_word2vec_format('data/GoogleGoogleNews-vectors-negative300.bin', binary=True)
# Access vectors for specific words with a keyed lookup:
vector = model['easy']
```

```
In [17]: model.similarity('straightforward', 'easy')
Out[17]: 0.5717043285477517

In [18]: model.similarity('simple', 'impossible')
Out[18]: 0.29156160264633707

In [19]: model.most_similar('simple')
Out[19]: [('straightforward', 0.7460169196128845),
          ('Simple', 0.7108174562454224),
          ('uncomplicated', 0.6297484636306763),
          ('simplest', 0.6171397566795349),
          ('easy', 0.5990299582481384),
          ('fairly_straightforward', 0.5893306732177734),
          ('deceptively_simple', 0.5743066072463989),
          ('simpler', 0.5537199378013611),
          ('simplistic', 0.5516539216041565),
          ('disarmingly_simple', 0.5365327000617981)]
```

Contents

- Word Representations
  - Understanding Words from the perspective of Information retrieval
  - How do we represent words in NLP models?
  - Distributional hypothesis
  - Sparse vs dense vectors
- Word2vec and other variants
  - Word2vec
  - How to train this model?
  - Skip-gram with negative sampling (SGNS) and other variants
  - Evaluating word embeddings

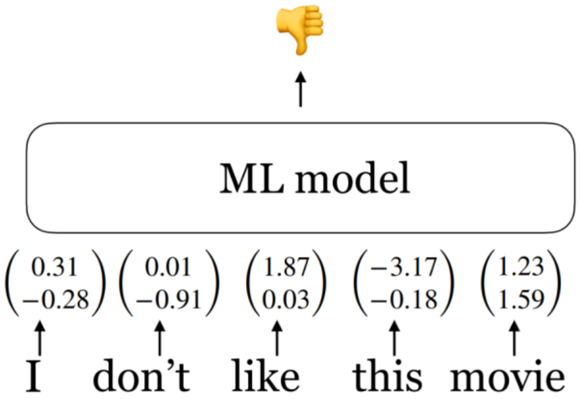
Extrinsic vs intrinsic evaluation

Extrinsic evaluation

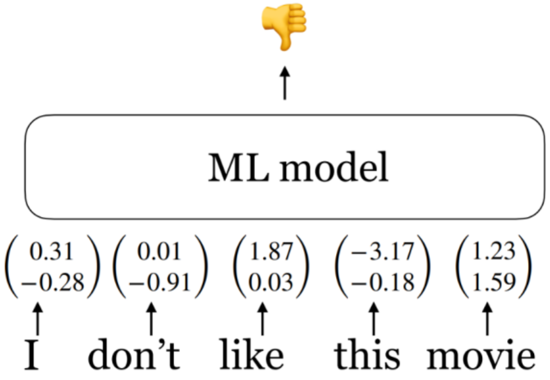
- Let's plug these word embeddings into a real NLP system and see whether this improves performance
- Could take a long time but still the most important evaluation metric

Intrinsic evaluation

- Evaluate on a specific/intermediate subtask
- Fast to compute
- Not clear if it really helps downstream tasks



Extrinsic evaluation



A straightforward solution: given an input sentence $x_1, x_2, \ldots, x_n$

Instead of using a bag-of-words model, we can compute $\text{vec}(x) = \mathbf{e}(x_1) + \mathbf{e}(x_2) + \ldots + \mathbf{e}(x_n)$

And then train a logistic regression classifier on $vec(x)$ as we did before!

There are much better ways to do this e.g., take word embeddings as input of neural networks

Intrinsic evaluation: word similarity

Word similarity

Example dataset: wordsim-353
353 pairs of words with human judgement

http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/

Word 1	Word 2	Human (mean)
tiger	cat	7.35
tiger	tiger	10
book	paper	7.46
computer	internet	7.58
plane	car	5.77
professor	doctor	6.62
stock	phone	1.62
stock	CD	1.31
stock	jaguar	0.92

Cosine similarity:

$$\cos(\boldsymbol{u}_i, \boldsymbol{u}_j) = \frac{\boldsymbol{u}_i \cdot \boldsymbol{u}_j}{||\boldsymbol{u}_i||_2 \times ||\boldsymbol{u}_j||_2}.$$

Metric: Spearman rank correlation

Intrinsic evaluation: word similarity

Model	Size	WS353	MC	RG	SCWS	RW
SVD	6B	35.3	35.1	42.5	38.3	25.6
SVD-S	6B	56.5	71.5	71.0	53.6	34.7
SVD-L	6B	65.7	<u>72.7</u>	75.1	56.5	37.0
CBOW†	6B	57.2	65.6	68.2	57.0	32.5
SG†	6B	62.8	65.2	69.7	<u>58.1</u>	37.2
GloVe	6B	<u>65.8</u>	<u>72.7</u>	<u>77.8</u>	53.9	<u>38.1</u>
SVD-L	42B	74.0	76.4	74.1	58.3	39.9
GloVe	42B	<u>**75.9**</u>	<u>**83.6**</u>	<u>**82.9**</u>	<u>**59.6**</u>	<u>**47.8**</u>
CBOW*	100B	68.4	79.6	75.4	59.4	45.5

SG: Skip-gram

Intrinsic evaluation: word analogy

Word analogy test: $a : a* :: b : b*$

$$b^* = \arg\max_{w \in V} \cos(e(w), e(a^*) - e(a) + e(b))$$

semantic

Chicago:Illinois Philadelphia: ?

syntactic

bad:worst cool: ?

More examples at

http://download.tensorflow.org/data/questions-words.txt

Metric: accuracy

Intrinsic evaluation: word analogy

Model	Dim.	Size	Sem.	Syn.	Tot.
ivLBL	100	1.5B	55.9	50.1	53.2
HPCA	100	1.6B	4.2	16.4	10.8
GloVe	100	1.6B	<u>67.5</u>	<u>54.3</u>	<u>60.3</u>
SG	300	1B	61	61	61
CBOW	300	1.6B	16.1	52.6	36.1
vLBL	300	1.5B	54.2	<u>64.8</u>	60.0
ivLBL	300	1.5B	65.2	63.0	64.0
GloVe	300	1.6B	<u>80.8</u>	61.5	<u>70.3</u>
SVD	300	6B	6.3	8.1	7.3
SVD-S	300	6B	36.7	46.6	42.1
SVD-L	300	6B	56.6	63.0	60.1
CBOW†	300	6B	63.6	<u>67.4</u>	65.7
SG†	300	6B	73.0	66.0	69.1
GloVe	300	6B	<u>77.4</u>	67.0	<u>71.7</u>
CBOW	1000	6B	57.3	68.9	63.7
SG	1000	6B	66.1	65.1	65.6
SVD-L	300	42B	38.4	58.2	49.2
GloVe	300	42B	**<u>81.9</u>**	**<u>69.3</u>**	**<u>75.0</u>**

Acknowledgement

- CS224N/Ling284: Natural Language Processing with Deep Learning, Stanford University
- COMP5021: Natural Language Processing, Xu Ruifeng ,Harbin Institute of Technology, Shenzhen
- CS447: Natural Language Processing, University of Illinois Urbana-Champaign
- CSCI 375: Natural Language Processing(Spring 2023), Williams College
- INFR10078, Foundations of Natural Language Processing, The University of Edinburgh

Thanks

For yourself

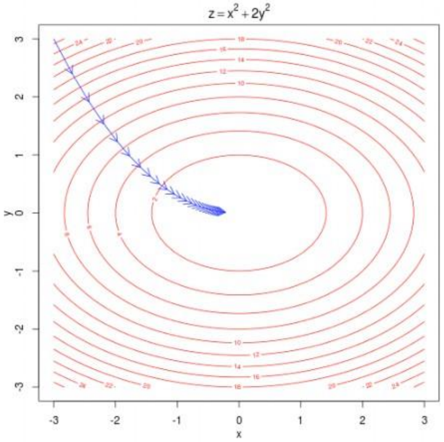
How to train this model?

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0}\log\frac{\exp(\mathbf{u}_{w_t}\cdot\mathbf{v}_{w_{t+j}})}{\sum_{k\in V}\exp(\mathbf{u}_{w_t}\cdot\mathbf{v}_k)}$$

- To train such a model, we need to compute the vector gradient $\nabla_\theta J(\theta) = ?$

- Again, θ represents all $2d\,|\,V\,|$ model parameters, in one vector.

$$\theta = \begin{bmatrix} v_{aardvark} \\ v_a \\ \vdots \\ v_{zebra} \\ u_{aardvark} \\ u_a \\ \vdots \\ u_{zebra} \end{bmatrix}$$



Vectorized gradients

$$f(\mathbf{x}) = \mathbf{x} \cdot \mathbf{a}$$

$$\mathbf{x}, \mathbf{a} \in \mathbb{R}^n$$

$$\frac{\partial f}{\partial \mathbf{x}} = \mathbf{a}$$

$$f = x_1 a_1 + x_2 a_2 + \ldots + x_n a_n$$

$$\frac{\partial f}{\partial \mathbf{x}} = [\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}]$$

Vectorized gradients: exercises

Let $f = \exp(\mathbf{w} \cdot \mathbf{x})$, what is the value of $\frac{\partial f}{\partial \mathbf{x}}$? $\mathbf{w}, \mathbf{x} \in \mathbb{R}^n$

(a) $\mathbf{w}$

(b) $\exp(\mathbf{w} \cdot \mathbf{x})$

(c) $\exp(\mathbf{w} \cdot \mathbf{x})\mathbf{w}$

(d) $\mathbf{x}$

Vectorized gradients: exercises

Let $f = \exp(\mathbf{w} \cdot \mathbf{x})$, what is the value of $\frac{\partial f}{\partial \mathbf{x}}$? $\mathbf{w}, \mathbf{x} \in \mathbb{R}^n$

(a) $\mathbf{w}$

(b) $\exp(\mathbf{w} \cdot \mathbf{x})$

(c) $\exp(\mathbf{w} \cdot \mathbf{x})\mathbf{w}$

(d) $\mathbf{x}$

The answer is (c).

$$\frac{\partial}{\partial x_i} = \frac{\exp(\sum_{k=1}^{n} w_i x_i)}{\partial x_i} = \exp(\sum_{k=1}^{n} w_i x_i) w_i$$

Let's compute gradients for word2vec

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0}\log\frac{\exp(\mathbf{u}_{w_t}\cdot\mathbf{v}_{w_{t+j}})}{\sum_{k\in V}\exp(\mathbf{u}_{w_t}\cdot\mathbf{v}_k)}$$

Consider one pair of center/context words (t, c):

$$y = -\log\left(\frac{\exp(\mathbf{u}_t\cdot\mathbf{v}_c)}{\sum_{k\in V}\exp(\mathbf{u}_t\cdot\mathbf{v}_k)}\right)$$

We need to compute the gradient of with respect to

$\mathbf{u}_t$ and $\mathbf{v}_k, \forall k \in V$

Let's compute gradients for word2vec

$$y = -\log\left(\frac{\exp(\mathbf{u}_t \cdot \mathbf{v}_c)}{\sum_{k \in V} \exp(\mathbf{u}_t \cdot \mathbf{v}_k)}\right)$$

$$\begin{aligned} y &= -\log(\exp(\mathbf{u}_t \cdot \mathbf{v}_c)) + \log(\sum_{k \in V} \exp(\mathbf{u}_t \cdot \mathbf{v}_k)) \\ &= -\mathbf{u}_t \cdot \mathbf{v}_c + \log(\sum_{k \in V} \exp(\mathbf{u}_t \cdot \mathbf{v}_k)) \end{aligned}$$

Let's compute gradients for word2vec

$$y = -\log\left(\frac{\exp(\mathbf{u}_t \cdot \mathbf{v}_c)}{\sum_{k \in V} \exp(\mathbf{u}_t \cdot \mathbf{v}_k)}\right)$$

$$y = -\log(\exp(\mathbf{u}_t \cdot \mathbf{v}_c)) + \log(\sum_{k \in V} \exp(\mathbf{u}_t \cdot \mathbf{v}_k))$$

$$= -\mathbf{u}_t \cdot \mathbf{v}_c + \log(\sum_{k \in V} \exp(\mathbf{u}_t \cdot \mathbf{v}_k))$$

$$\frac{\partial y}{\partial \mathbf{u}_t} = \frac{\partial(-\mathbf{u}_t \cdot \mathbf{v}_c)}{\partial \mathbf{u}_t} + \frac{\partial(\log \sum_{k \in V} \exp(\mathbf{u}_t \cdot \mathbf{v}_k))}{\partial \mathbf{u}_t}$$

$$= -\mathbf{v}_c + \frac{\frac{\partial \sum_{k \in V} \exp(\mathbf{u}_t \cdot \mathbf{v}_k)}{\partial \mathbf{u}_t}}{\sum_{k \in V} \exp(\mathbf{u}_t \cdot \mathbf{v}_k)}$$

$$= -\mathbf{v}_c + \frac{\sum_{k \in V} \exp(\mathbf{u}_t \cdot \mathbf{v}_k) \cdot \mathbf{v}_k}{\sum_{k \in V} \exp(\mathbf{u}_t \cdot \mathbf{v}_k)}$$

Let's compute gradients for word2vec

$$y = -\log\left(\frac{\exp(\mathbf{u}_t \cdot \mathbf{v}_c)}{\sum_{k \in V} \exp(\mathbf{u}_t \cdot \mathbf{v}_k)}\right)$$

$$\begin{aligned} y &= -\log(\exp(\mathbf{u}_t \cdot \mathbf{v}_c)) + \log(\sum_{k \in V} \exp(\mathbf{u}_t \cdot \mathbf{v}_k)) \\ &= -\mathbf{u}_t \cdot \mathbf{v}_c + \log(\sum_{k \in V} \exp(\mathbf{u}_t \cdot \mathbf{v}_k)) \end{aligned}$$

$$\begin{aligned} \frac{\partial y}{\partial \mathbf{u}_t} &= \frac{\partial(-\mathbf{u}_t \cdot \mathbf{v}_c)}{\partial \mathbf{u}_t} + \frac{\partial(\log \sum_{k \in V} \exp(\mathbf{u}_t \cdot \mathbf{v}_k))}{\partial \mathbf{u}_t} \\ &= -\mathbf{v}_c + \frac{\frac{\partial \sum_{k \in V} \exp(\mathbf{u}_t \cdot \mathbf{v}_k)}{\partial \mathbf{u}_t}}{\sum_{k \in V} \exp(\mathbf{u}_t \cdot \mathbf{v}_k)} \\ &= -\mathbf{v}_c + \frac{\sum_{k \in V} \exp(\mathbf{u}_t \cdot \mathbf{v}_k) \cdot \mathbf{v}_k}{\sum_{k \in V} \exp(\mathbf{u}_t \cdot \mathbf{v}_k)} \end{aligned}$$

Recall that

$$P(w_{t+j} \mid w_t) = \frac{\exp(\mathbf{u}_{w_t} \cdot \mathbf{v}_{w_{t+j}})}{\sum_{k \in V} \exp(\mathbf{u}_{w_t} \cdot \mathbf{v}_k)}$$

Let's compute gradients for word2vec

$$y = -\log\left(\frac{\exp(\mathbf{u}_t \cdot \mathbf{v}_c)}{\sum_{k\in V}\exp(\mathbf{u}_t \cdot \mathbf{v}_k)}\right)$$

$$y = -\log(\exp(\mathbf{u}_t \cdot \mathbf{v}_c)) + \log(\sum_{k\in V}\exp(\mathbf{u}_t \cdot \mathbf{v}_k))$$
$$= -\mathbf{u}_t \cdot \mathbf{v}_c + \log(\sum_{k\in V}\exp(\mathbf{u}_t \cdot \mathbf{v}_k))$$

$$\frac{\partial y}{\partial \mathbf{u}_t} = \frac{\partial(-\mathbf{u}_t \cdot \mathbf{v}_c)}{\partial \mathbf{u}_t} + \frac{\partial(\log\sum_{k\in V}\exp(\mathbf{u}_t \cdot \mathbf{v}_k))}{\partial \mathbf{u}_t}$$

$$= -\mathbf{v}_c + \frac{\frac{\partial\sum_{k\in V}\exp(\mathbf{u}_t\cdot\mathbf{v}_k)}{\partial \mathbf{u}_t}}{\sum_{k\in V}\exp(\mathbf{u}_t \cdot \mathbf{v}_k)}$$

$$= -\mathbf{v}_c + \frac{\sum_{k\in V}\exp(\mathbf{u}_t \cdot \mathbf{v}_k)\cdot\mathbf{v}_k}{\sum_{k\in V}\exp(\mathbf{u}_t \cdot \mathbf{v}_k)}$$

$$= -\mathbf{v}_c + \sum_{k\in V}\frac{\exp(\mathbf{u}_t \cdot \mathbf{v}_k)}{\sum_{k'\in V}\exp(\mathbf{u}_t \cdot \mathbf{v}_{k'})}\mathbf{v}_k$$

$$= -\mathbf{v}_c + \sum_{k\in V}P(k \mid t)\mathbf{v}_k$$

Recall that

$$P(w_{t+j} \mid w_t) = \frac{\exp(\mathbf{u}_{w_t} \cdot \mathbf{v}_{w_{t+j}})}{\sum_{k\in V}\exp(\mathbf{u}_{w_t} \cdot \mathbf{v}_k)}$$

Overall algorithm

- Input: text corpus, embedding size d, vocabulary V , **context size m**
- Initialize $\mathbf{u}_i, \mathbf{v}_i$ randomly $\forall i \in V$
- Run through the training corpus and for each training instance (t, c):
  - Update $\mathbf{u}_t \leftarrow \mathbf{u}_t - \eta \frac{\partial y}{\partial \mathbf{u_t}}$ $\quad \frac{\partial y}{\partial \mathbf{u}_t} = -\mathbf{v}_c + \sum_{k \in V} P(k \mid t)\mathbf{v}_k$
  - Update $\mathbf{v}_k \leftarrow \mathbf{v}_k - \eta \frac{\partial y}{\partial \mathbf{v_k}}, \forall k \in V$ $\quad \frac{\partial y}{\partial \mathbf{v}_k} = \begin{cases} (P(k \mid t) - 1)\, \mathbf{u}_t & k = c \\ P(k \mid t)\mathbf{u}_t & k \neq c \end{cases}$

Convert the training data into:
(into, problems)
(into, turning)
(into, banking)
(into, crises)
(banking, turning)
(banking, into)
(banking, crises)
(banking, as)
…

Overall algorithm

- Input: text corpus, embedding size d, vocabulary V , **context size m**
- Initialize $\mathbf{u}_i, \mathbf{v}_i$ randomly $\forall i \in V$
- Run through the training corpus and for each training instance (t, c):
  - Update $\mathbf{u}_t \leftarrow \mathbf{u}_t - \eta \frac{\partial y}{\partial \mathbf{u_t}}$ $\qquad \frac{\partial y}{\partial \mathbf{u}_t} = -\mathbf{v}_c + \sum_{k \in V} P(k \mid t) \mathbf{v}_k$
  - Update $\mathbf{v}_k \leftarrow \mathbf{v}_k - \eta \frac{\partial y}{\partial \mathbf{v_k}}, \forall k \in V$ $\qquad \frac{\partial y}{\partial \mathbf{v}_k} = \begin{cases} (P(k \mid t) - 1) \mathbf{u}_t & k = c \\ P(k \mid t) \mathbf{u}_t & k \neq c \end{cases}$

Convert the training data into:
(into, problems)
(into, turning)
(into, banking)
(into, crises)
(banking, turning)
(banking, into)
(banking, crises)
(banking, as)
…

Q: Can you think of any issues with this algorithm?