Text Analytics

MSiA 414

Instructor: Emilia Apostolova, PhD

Agenda

Introductions

Course preliminaries

A brief history of NLP (part 1 of 2)

Character encodings

Regular expressions

Tokenization, text normalization

POS tagging, chunking, syntax parsing

Homework discussion

Self-Introduction

Emilia Apostolova, PhD

CTO, Language.ai

Academic Experience: PhD in Computer Science, Information Extraction from Radiology reports. 2-years of Research Experience at the National Institutes of Health, adjunct faculty at DePaul University.

Industry Experience: Software Developer of 17 years. Since 2011, building commercial NLP/ML applications.

Contact Information

PLEASE JOIN the #msia414-2019 SLACK CHANNEL!

The main and most reliable source of course-related announcements.

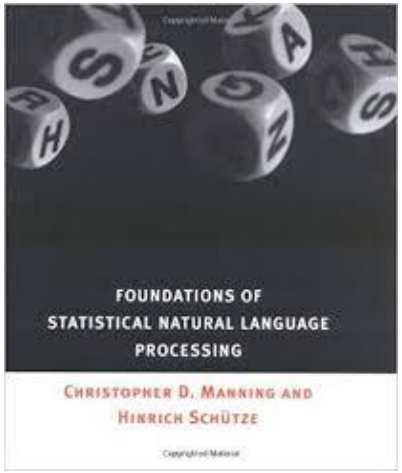
You can email me at emilia@language.ai. I will *try* to respond within 48 hours.

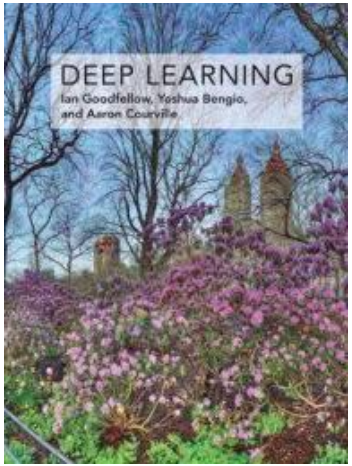
Course Preliminaries

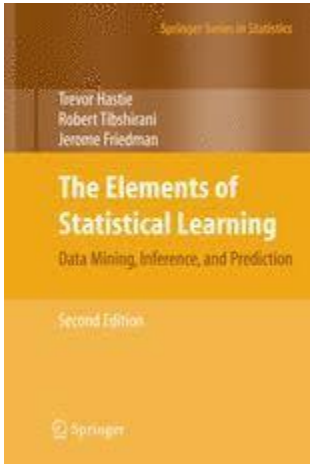
- Course objective
- Homework and independent projects
- Late submissions
- Class and lab participation
- Final exam
- Grading:

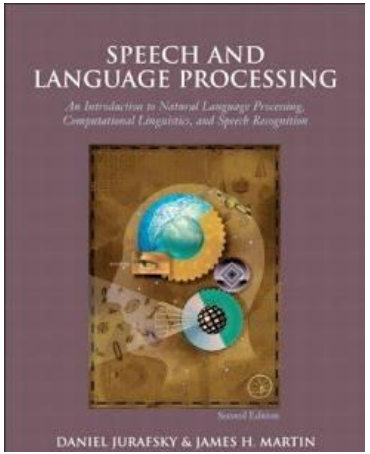
  40% Class and lab participation
  20% Homework
  20% Independent project
  20% Final exam

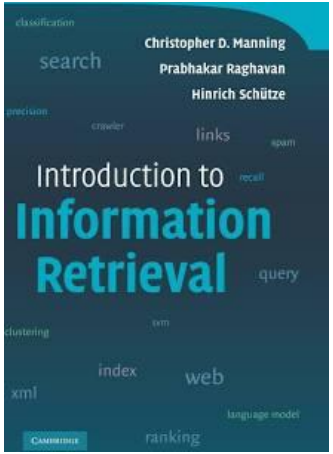
Recommended Reading









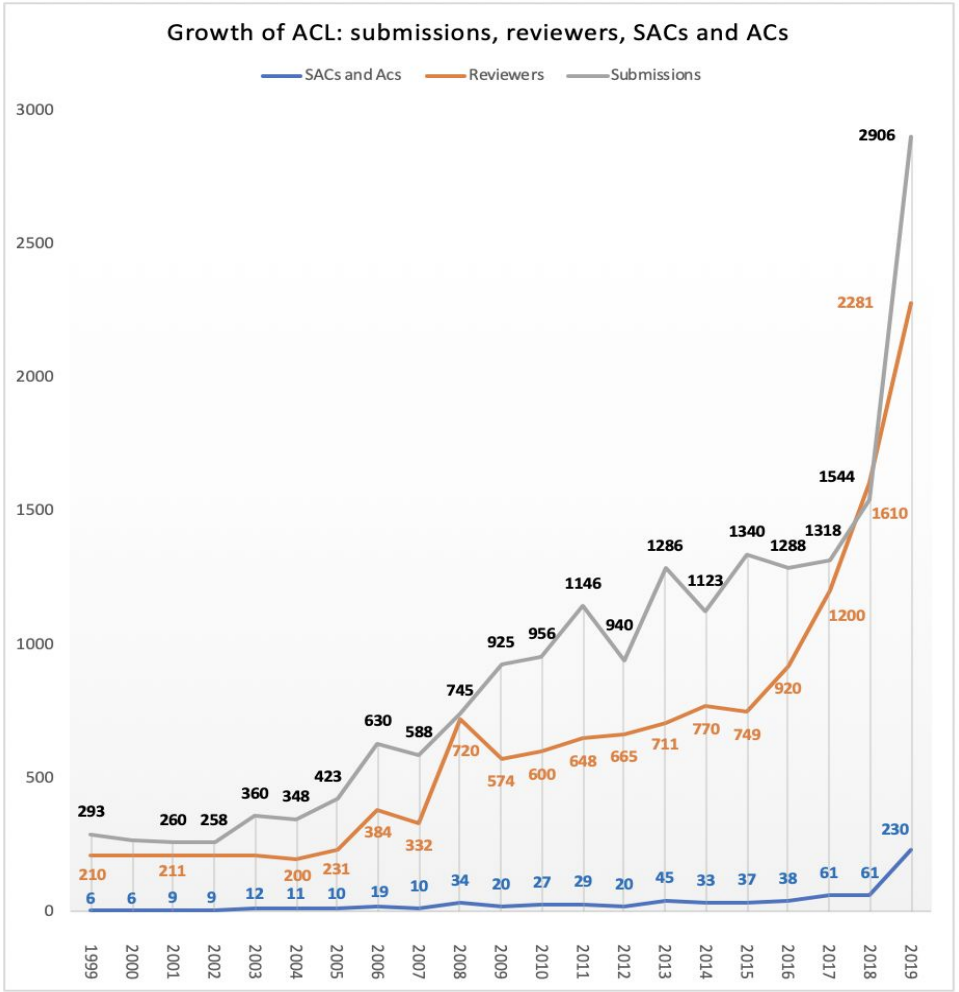


Required Reading

Association for Computational Linguistics (ACL) 

Conference publications: ACL, North American Chapter of ACL (NAACL), Empirical Methods in Natural Language Processing (EMNLP), Int'l Committee on Computational Linguistics Conference (COLING).

https://arxiv.org/ - Owned and operated by Cornell University. Non-exclusive license, non-peer reviewed, fast, open-access scientific research venue. Fast access to cutting edge scientific publications (pre-print), but also some low-quality peer-review-rejected articles.



	Area	Long	Short	Total
1.	Information Extraction, Text Mining	156	93	249
2.	Machine Learning	148	73	221
3.	Machine Translation	102	105	207
4.	Dialogue and Interactive Systems	125	57	182
5.	Generation	97	58	155
6.	Question Answering	99	55	154
7.	Sentiment Analysis, Argument Mining	91	60	151
8.	Word-level Semantics	78	59	137
9.	Applications	65	72	137
10.	Resources and Evaluation	70	60	130
11.	Multidisciplinary, AC COI	70	44	114
12.	Sentence-level Semantics	70	42	112
13.	Tagging, Chunking, Syntax, Parsing	50	49	99
14.	Social Media	51	42	93
15.	Summarization	48	35	83
16.	Document Analysis	48	33	81
17.	Vision, Robotics Multimodal Grounding, Speech	56	23	79
18.	Multilinguality	43	32	75
19.	Textual Inference, Other Areas of Semantics	44	30	74
20.	Linguistic Theories, Cognitive, Psycholinguistics	39	21	60
21.	Discourse and Pragmatics	33	24	57
22.	Phonology, Morphology, Word Segmentation	26	18	44
	Total	**1609**	**1085**	**2694**

Google Scholar

word representations

Articles

About 3,240,000 results (**0.06** sec)

Any time
Since 2019
Since 2018
Since 2015
Custom range...

Sort by relevance
Sort by date

include patents
include citations

Create alert

Efficient estimation of **word representations** in vector space

[PDF] arxiv.org

T Mikolov, K Chen, G Corrado, J Dean - arXiv preprint arXiv:1301.378[illegible] 2013 - arxiv.org

We propose two novel model architectures for computing continuous vector **representations** of words from very large data sets. The quality of these **representations** is measured in a **word** similarity task, and the results are compared to the previously best performing …

Cited by 11913 Related articles All 32 versions

[PDF] Linguistic regularities in continuous space **word representations**

[PDF] aclweb.org

T Mikolov, W Yih, G Zweig - Proceedings of the 2013 Conference of the …, 2013 - aclweb.org

Continuous space language models have recently demonstrated outstanding results across a variety of tasks. In this paper, we examine the vector-space **word representations** that are implicitly learned by the input-layer weights. We find that these **representations** are …

Cited by 2247 Related articles All 12 versions

Word representations: a simple and general method for semi-supervised learning

[PDF] aclweb.org

J Turian, L Ratinov, Y Bengio - Proceedings of the 48th annual meeting …, 2010 - dl.acm.org

If we take an existing supervised NLP system, a simple and general way to improve accuracy is to use unsupervised **word representations** as extra **word** features. We evaluate Brown clusters, Collobert and Weston (2008) embeddings, and HLBL (Mnih & Hinton, 2009) …

Cited by 1835 Related articles All 19 versions

Deep contextualized **word representations**

[PDF] arxiv.org

ME Peters, M Neumann, M Iyyer, M Gardner… - arXiv preprint arXiv …, 2018 - arxiv.org

We introduce a new type of deep contextualized **word** representation that models both (1) complex characteristics of **word** use (eg, syntax and semantics), and (2) how these uses vary across linguistic contexts (ie, to model polysemy). Our **word** vectors are learned functions of …

Cited by 1242 Related articles All 11 versions

A subjective list of essential Data Scientist skills

1. Ability and interest to proactively and constantly learn on their own.
2. Excellent software engineering skills.
3. Ability to understand business problems and find commonsensical and creative solutions.
4. Ability to communicate clearly ideas, “team player”.

Homework: focus on skills 1 and 2.

Independent project: focus on skills 1,2,3,4.

Natural Language Processing (NLP)

Text Analytics, Computational Linguistics.

The science and art of teaching machines to understand / produce natural language. Machine learning for unstructured or semi-structured data (similar to image processing).

Multi-disciplinary: Statistics and Machine Learning, Computer Science and Engineering, Linguistics, Mathematics, Neuroscience, etc.

Extremely empirical scientific field: breakthroughs typically come through experiments and often lack a formal method justification.

As an NLP scientist/engineer keeping up with the latest academic research and the latest NLP tools/frameworks is not optional.

On the importance of software engineering skills in data science.
http://blog.indeed.com/2019/06/28/top-10-ai-jobs-salaries-cities/

An open-ended list of NLP tasks / topics / applications

Speech Recognition and Text-to-speech

Tokenization, stemming, morphological parsing

Part of speech tagging, chunking, syntax parsing

Word-sense disambiguation, semantic parsing

Information Retrieval

Text classification

Sentiment analysis

An open-ended list of NLP tasks / topics / applications cont'd

Information Extraction: Named Entity Recognition, Relation Extraction, Coreference resolution, Knowledge Extraction, Term and ontology extraction.

Machine Translation

Text Mining

Text Summarization

Question Answering

Dialogue and interactive systems

…..

A brief history of NLP: 1950s-1960

First attempts of NLP. A period of excitement and optimism. First use of computers for non-numeric problems.

Marvin Minsky: "Within a generation ... the problem of creating 'artificial intelligence' will substantially be solved."

1954 - The IBM-Georgetown Experiment - Machine Translation of 60 Russian sentences into English; The Journal Mechanical Translation (MT) began publication.

1961 - International Conference on Machine Translation of Languages and Applied Language Analysis

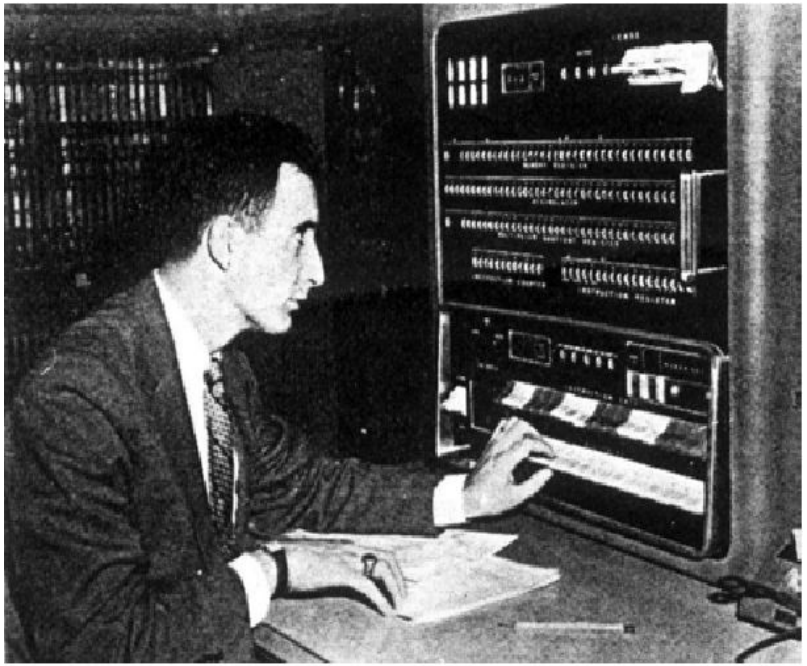
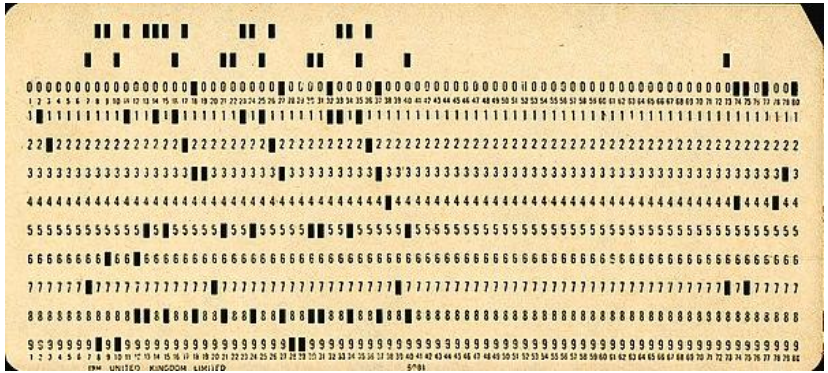


Figure 4: Operating the computer

Hutchins J. The first public demonstration of machine translation: the Georgetown-IBM system, 7th January 1954. Publicación electrónica en: http://www. hutchinsweb. me. uk/GUIBM-2005. pdf. 2005.

A brief history of NLP: 1950s-1960s cont'd

Primitive computing resources. Assembler programming using punched cards. Access to machines was restricted, extremely limited storage, very slow. A lot of programming effort spent on saving space and time.



A punched card from mid 20th century. Source: Wikipedia ©

Main line of work: translation as a look-up, dictionary-based word-for-word processing.

Main line of research: Syntax. Formal definition of language, computational grammar. Linguistic theory: Noam Chomsky's generative grammar.

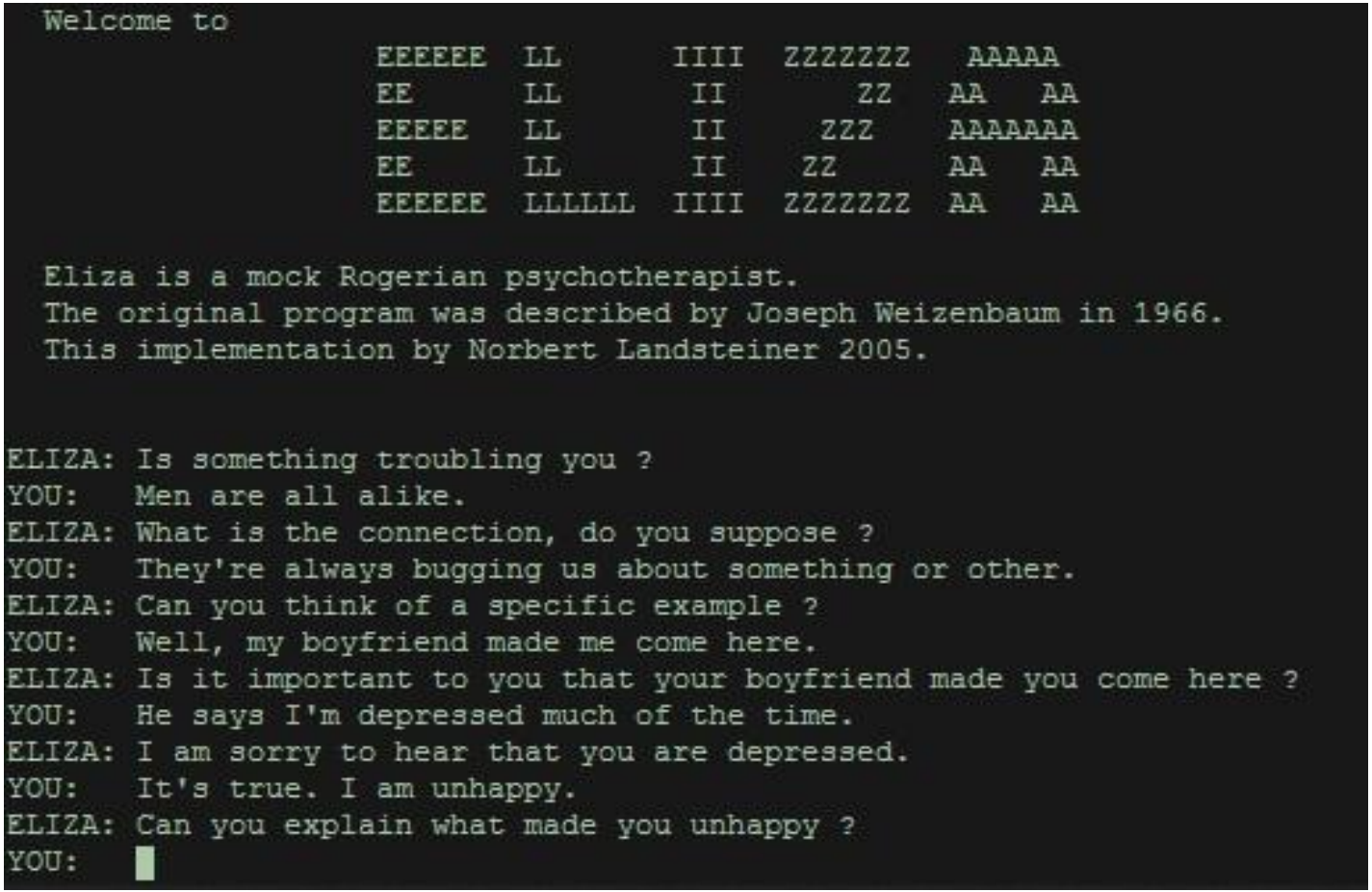
Formal Grammar

A set of rules for transforming strings. Definition of terminal and non-terminal symbols (alphabet), and transformation rules.

For example, S → start symbol; a, b: alphabet. Rules:

1. S → aSb
2. S → ab

What would the language of this grammar look like?



One of the first NLP programs, attempting the Turing Test. Peter Norvig's Lisp implementation of Eliza is ~100 lines of code: pattern matching and substitution.

A brief history of NLP

The 1966 ALPAC (Automatic Language Processing advisory committee) report concluded that MT was a difficult task that is not likely to be achieved any time soon and led to funding cuts. MT remained dormant for a while.

The next phase of NLP focused on knowledge representations, Artificial Intelligence, reasoning and inference.

A brief history of NLP: 1970s-1980s

A period of expanding community, growing confidence, practical resources became available: parsers, grammars, toolkits, etc. New tasks were addressed: Message Understanding (i.e. information extraction). The Message Understanding Conferences (MUC) started in 1987.

MUC was initiated and financed by DARPA (Defense Advanced Research Projects Agency) to encourage the development of new and better methods of information extraction, in the form of competition. One of the first NLP “shared tasks”. Required evaluation standards and metrics.

MUC-3: Extracting Information from News Reports

Chinchor N, Lewis DD, Hirschman L. Evaluating message understanding systems: an analysis of the third message understanding conference (MUC-3). Computational linguistics. 1993 Sep 1;19(3):409-49.

Task: Fill in template slots for events describing Latin American terrorist activity.

15 participating systems.

Train and tests sets with automated evaluation procedures (Precision, Recall) and statistical significance tests.

TST2-MUC3-0069

BOGOTA, 7 SEP 89 (INRAVISION TELEVISION CADENA 1) -- [REPORT] [MARIBEL OSORIO] [TEXT] MEDELLIN CONTINUES TO LIVE THROUGH A WAVE OF TERROR. FOLLOWING LAST NIGHT'S ATTACK ON A BANK, WHICH CAUSED A LOT OF DAMAGE, A LOAD OF DYNAMITE WAS HURLED AGAINST A POLICE STATION. FORTUNATELY NO ONE WAS HURT. HOWEVER, AT APPROXIMATELY 1700 TODAY A BOMB EXPLODED INSIDE A FAST-FOOD RESTAURANT.

A MEDIUM-SIZED BOMB EXPLODED SHORTLY BEFORE 1700 AT THE PRESTO INSTALLATIONS LOCATED ON [WORDS INDISTINCT] AND PLAYA AVENUE. APPROXIMATELY 35 PEOPLE WERE INSIDE THE RESTAURANT AT THE TIME. A WORKER NOTICED A SUSPICIOUS PACKAGE UNDER A TABLE WHERE MINUTES BEFORE TWO MEN HAD BEEN SEATED. AFTER AN INITIAL MINOR EXPLOSION, THE PACKAGE EXPLODED. THE 35 PEOPLE HAD ALREADY BEEN EVACUATED FROM THE BUILDING, AND ONLY 1 POLICEMAN WAS SLIGHTLY INJURED; HE WAS THROWN TO THE GROUND BY THE SHOCK WAVE. THE AREA WAS IMMEDIATELY CORDONED OFF BY THE AUTHORITIES WHILE THE OTHER BUSINESSES CLOSED THEIR DOORS. IT IS NOT KNOWN HOW MUCH DAMAGE WAS CAUSED; HOWEVER, MOST OF THE DAMAGE WAS OCCURRED INSIDE THE RESTAURANT. THE MEN WHO LEFT THE BOMB FLED AND THERE ARE NO CLUES AS TO THEIR WHEREABOUTS.

Template slots: Date and Location of incident, Type of Incident, Physical Target(s), Human Target(s), etc.

A brief history of NLP: 1970s-1980s cont'd

Research on discourse, dialogue, text generation. Related to the development of Expert Systems at the time: AI systems emulating the advice of human "experts" with the use of "knowledge base" and an "inference engine". One of the first successful applications of AI. Prolog - declarative, logic programming language.

MYCIN, an example of an Expert System used for patient diagnosis: identify bacterial infections and prescribe antibiotics. Developed in the 70s at Stanford University. Uses a knowledge base of ~600 rules, and an inference engine. Input: a long series of yes/no questions. Output: a probability-ranked list of possible infection types and recommended course of treatment. Mycin uses "backward reasoning".

Backward Reasoning / Backward Chaining

Knowledge Base:

1. If X croaks and X eats flies – Then X is a frog
2. If X chirps and X sings – Then X is a canary
3. If X is a frog – Then X is green
4. If X is a canary – Then X is yellow

Input: Fritz croaks. Fritz eats flies

Question: Is Fritz green?

Inference: Work backwards towards the goal.

A brief history of NLP: 1970s-1980s cont'd

Work on machine-readable lexicons and ontologies.

Wordnet: A lexical database of the English language developed in the 80s at Princeton University. Contains words, their lexical categories (e.g. adjective, noun, verb, etc.), word definitions, and word relationships, e.g. hypernym (parent), hyponym (child), meronym (part of), etc. Used, at the time, for automatic text analysis and AI applications.

Hugely influential resource (more than 30k citations according to Google Scholar). Used to introduce prior knowledge (context) in a variety of NLP tasks. Similar to the use of word embeddings in the context of deep learning (topic of next lecture).

A semantic graph of words that can be used in a variety of ways to **convert meaning into numbers** (which btw is really what NLP is about).

WordNet Search - 3.1

- WordNet home page - Glossary - Help

Word to search for: dog 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) **dog**, domestic dog, Canis familiaris (a member of the genus Canis (probably descended from the common wolf) that has been domesticated by man since prehistoric times; occurs in many breeds) *"the dog barked all night"*
  - *direct hyponym* / *full hyponym*
  - *part meronym*
  - *member holonym*
  - *direct hypernym* / ***inherited hypernym*** / *sister term*
    - S: (n) canine, canid (any of various fissiped mammals with nonretractile claws and typically long muzzles)
      - S: (n) carnivore (a terrestrial or aquatic flesh-eating mammal) *"terrestrial carnivores have four or five clawed digits on each limb"*
        - S: (n) placental, placental mammal, eutherian, eutherian mammal (mammals having a placenta; all mammals except monotremes and marsupials)
          - S: (n) mammal, mammalian (any warm-blooded vertebrate having the skin more or less covered with hair; young are born alive except for the small subclass of monotremes and nourished with milk)
            - S: (n) vertebrate, craniate (animals having a bony or cartilaginous skeleton with a segmented spinal column and a large brain enclosed in a skull or cranium)
              - S: (n) chordate (any animal of the phylum Chordata having a notochord or spinal column)
                - S: (n) animal, animate being, beast, brute, creature, fauna (a living organism characterized by voluntary movement)

WordNet Search - 3.1

- WordNet home page - Glossary - Help

Word to search for: dog 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) **dog**, domestic dog, Canis familiaris (a member of the genus Canis (probably descended from the common wolf) that has been domesticated by man since prehistoric times; occurs in many breeds) *"the dog barked all night"*

  - *direct hyponym* / ***full hyponym***
    - S: (n) puppy (a young dog)
    - S: (n) pooch, doggie, doggy, barker, bow-wow (informal terms for dogs)
    - S: (n) cur, mongrel, mutt (an inferior dog or one of mixed breed)
      - S: (n) feist, fice (a nervous belligerent little mongrel dog)
      - S: (n) pariah dog, pye-dog, pie-dog (ownerless half-wild mongrel dog common around Asian villages especially India)
    - S: (n) lapdog (a dog small and tame enough to be held in the lap)
    - S: (n) toy dog, toy (any of several breeds of very small dogs kept purely as pets)
      - S: (n) Chihuahua (an old breed of tiny short-haired dog with protruding eyes from Mexico held to antedate Aztec civilization)
      - S: (n) Japanese spaniel (breed of toy dogs originating in Japan having a silky black-and-white or red-and-white coat)
      - S: (n) Maltese dog, Maltese terrier, Maltese (breed of toy dogs having a long straight silky white coat)
      - S: (n) Pekinese, Pekingese, Peke (a Chinese breed of small short-legged dogs with a long silky coat and broad flat muzzle)
      - S: (n) Shih-Tzu (a Chinese breed of small dog similar to a Pekingese)
      - S: (n) toy spaniel (a very small spaniel)
        - S: (n) English toy spaniel (British breed having a long

A brief history of NLP: 1990s-2000s

All about **statistical** language processing. Large amounts of data (the rise of the WWW) + computing resources able to handle it. Machine learning from large corpora.

Significant progress in practical tasks (e.g. Speech Recognition, Information Retrieval, Information Extraction, Machine Translation, etc.).

Growth in NLP corpora and linguistic resources (Linguistic Data Consortium), shared tasks and competitions (US government grants, European Union grants), focus on robustness and portability. Public NLP resources (tokenizers, parsers, etc). Prototype system assembly via **pipeline NLP processing** (e.g., the IBM Watson System and Apache UIMA).

IBM Watson Computer

A question-answering computer system capable of answering questions posed in natural language.

Developed by David Ferrucci and team (~2004-2011) specifically to answer Jeopardy questions.

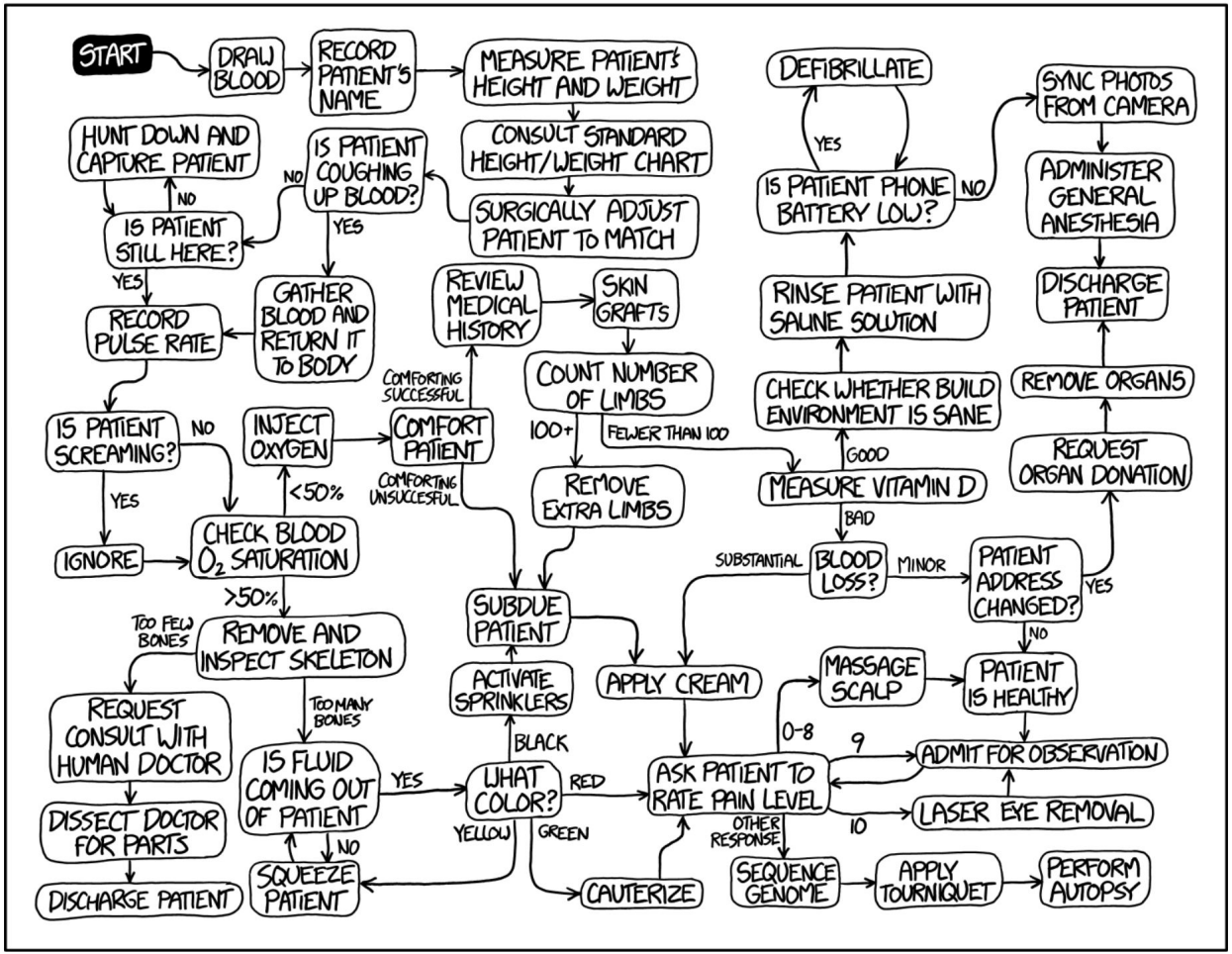
Won first place ($1 million) against 2 human competitors in 2011.

https://www.youtube.com/watch?v=WFR3lOm_xhE

IBM open-sourced the Watson pipeline engine, Apache UIMA (Unstructured Information Management Architecture).

Disclaimer: If you work for IBM, please skip the next slide.

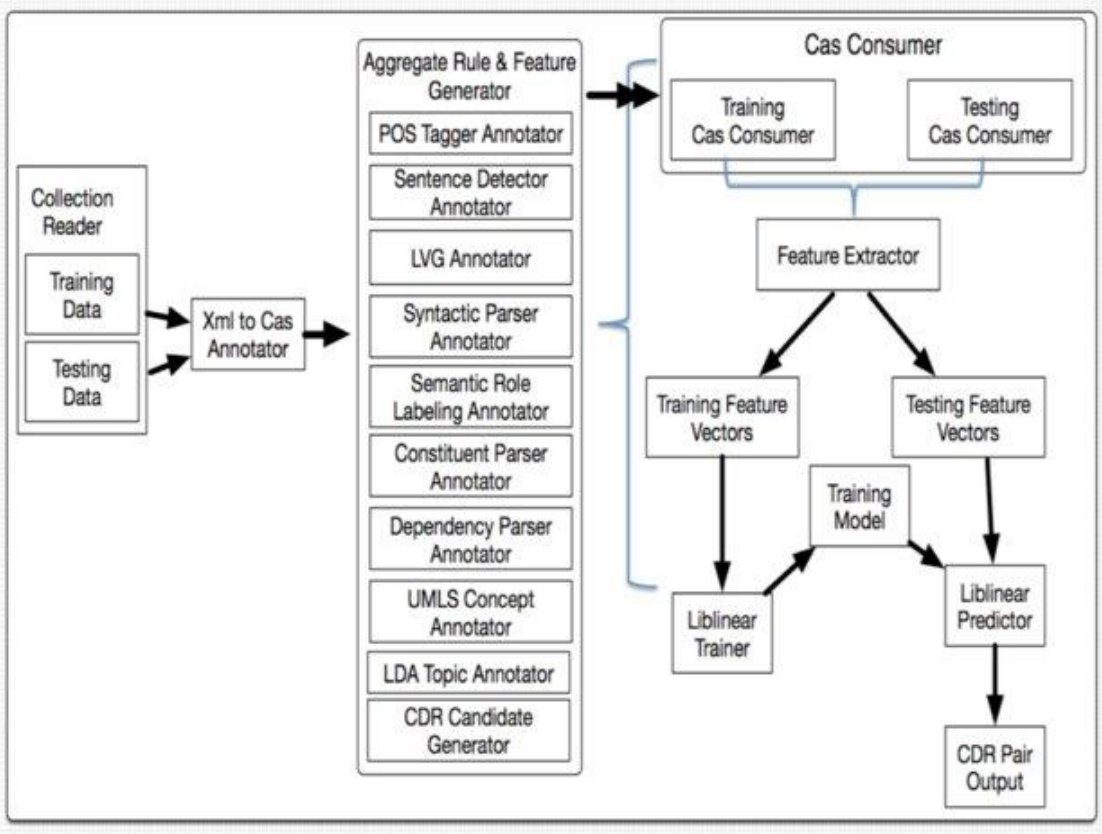
A GUIDE TO THE MEDICAL DIAGNOSTIC AND TREATMENT ALGORITHM USED BY IBM's WATSON COMPUTER SYSTEM



Source:

https://xkcd.com/1619/

Pipe-line NLP, a sample UIMA architecture



Li, Dingcheng, et al. "Resolution of chemical disease relations with diverse features and rules." *The fifth BioCreative challenge evaluation workshop*. 2015.

A brief history of NLP: 2010s - present

All about **deep learning** (focus of the next lecture).

Error-aggregating NLP pipelines replaced by focus on minimal pre-processing, multi-task learning.

Key deep learning boom factors: **large datasets** (not algorithms) and **processing power**.

Many NLP approaches started in the image-processing community (images, similar to texts, are a type of “unstructured” data).

ImageNet: Where Have We Gone? Where Are We Going? with Fei-Fei Li.

Working with text: the bad news on Character Encodings

Main source of bugs and issues for an NLP data scientist / NLP engineer. Ignored by most NLP textbooks / courses.

NLP datasets are plain text = ascii = characters are 8 bits? PDF, various word document formats, HTML, XML and content type tags.

On the overall ignorance of character encodings and some amazingly buggy mainstream software encoding conversions, including NLP libraries.

Character encoding issues can manifest at any stage of an NLP system: document input, pre-processing, ML predictions, post-processing, system output. Difficult to reproduce / debug as they typically manifest for one-off documents, sometimes without a stacktrace.

The good news on Character Encodings

1. They don’t change very often (you need to understand / learn once).
2. It is not that hard.
3. There is an excellent, informative and entertaining, blog post by Joel Spolsky (the creator of Stackoverflow.com and Trello, among other things):

   The Absolute Minimum Every Software Developer Absolutely, Positively Must Know About Unicode and Character Sets (No Excuses!)

4. You will thank me later.

Character Encodings: ASCII

- Published in 1963.
- Represents characters with numbers between 32 and 127 (32=space, A=65). Codes below 32 are unprintable (control chars).
- ASCII can be stored in 7 bits.
- At the time, computers used 8-bit bytes, one bit to spare!

USASCII code chart

Bits b_4	b_3	b_2	b_1	Row ↓ / Column → 0	0	1	2	3	4	5	6	7
					b_7 0	0	0	0	1	1	1	1
					b_6 0	0	1	1	0	0	1	1
					b_5 0	1	0	1	0	1	0	1
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	|
1	1	0	1	13	CR	GS	-	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

US-ASCII Code Chart. From the material delivered with TermiNet 300 impact type printer with Keyboard, February 1972, General Electric Data communication Product Dept., Waynesboro, Virginia.

ASCII: the extra bit and the OEM Babylon of encodings

Codes 128-255 can be used for “custom” encodings for non-English characters.

OEM - IBM character set, accented European chars and drawing chars.

PCs started selling outside the U.S. and the extra bit was used to create all kinds of OEM char sets.

OEM char sets, aka “high ASCII”, “extended ASCII”.

Eventually this OEM free-for-all got codified in the ANSI standard.

Character encodings: the ANSI standard

Codes below 128 are the same as ASCII

Code above 128 use “code pages”. For example, Israel DOS used a code page called 862, while Greek users used 737. The national versions of MS-DOS had dozens of these code pages.

Not possible to use multiple code pages on the same computers, for example to show Hebrew and Greek on the same computer.

Asian alphabets have thousands of letters, not going to fit in 8 bits. This was usually solved by the messy “double byte character set” in which some letters were stored in one byte and others took two.

Then the Internet happened, strings are now moved from one computer to another ...

Character Encodings: Unicode

Unicode is not a 16-bit character encoding. There are more than 2^16 (65,536) unicode chars and they don't all fit in 2 bytes.

In Unicode, a letter maps to something called a **code point** which is still just a theoretical concept. The idea that "A" in a Times New Roman font is the same character as the "A" in a Helvetica font, but different from "a" in lower case.

Code points: every letter in every alphabet is assigned a magic number by the Unicode consortium which is written like this: **U+0639**

The U+ means "Unicode" and the numbers are hexadecimal. **U+0639** is the Arabic letter Ain. The English letter A would be **U+0041**. You can find them all using the the Unicode web site.

Unicode Encodings

“Hello” → U+0048 U+0065 U+006C U+006C U+006F

00 48 00 65 00 6C 00 6C 00 6F

or

48 00 65 00 6C 00 6C 00 6F 00 ?

High-endian and low-endian mode depending on what's faster for the particular CPU. The solution: storing a FE FF at the beginning of every Unicode string, a Unicode Byte Order Mark.

Unicode Encodings

English texts rarely used code points above U+00FF, doubling the amount of storage needed for strings... The solution: **UTF8**

UTF8 stores Unicode characters using 8-bit bytes. In UTF-8, every code point from 0-127 is stored in a single byte. Only code points 128 and above are stored using 2, 3, in fact, up to 6 bytes. English texts look exactly the same in UTF-8 as they did in ASCII. UTF-8 also has the bug-inducing property that it *might* work with ignorant, ASCII processing code.

UTF7 - same as UTF8 but guarantees that the high bit will always be zero, so that it works with old 7-bit processing systems.

UCS-4 - each code point stored in 4 bytes.

Character Encodings

Any encoding can be converted to another, with one catch: some of the letters might not show up, if there's no equivalent for the Unicode code point you're trying to represent in the encoding you're trying to represent it in.

Other popular English-text encodings: **Windows-1252** and ISO-8859-1, aka **Latin-1** (useful for Western European language) **UTF 7, 8, 16, and 32** can represent any code point, unlike Windows-1252 and Latin1.

You cannot work with text without knowing the correct encoding. If a document character encoding tag is missing or wrong (yes, it happens), use character-encoding guessing library, e.g. https://pypi.org/project/chardet/

Python2's default encoding is ASCII, python3's default encoding is UTF8. Please read the python encode and encode documentation.

```
In [29]: s="Това изречение е на български"

In [30]: s.encode('ascii')
---------------------------------------------------------------------------
UnicodeEncodeError                        Traceback (most recent call last)
<ipython-input-30-532f2946fbcf> in <module>
----> 1 s.encode('ascii')

UnicodeEncodeError: 'ascii' codec can't encode characters in position 0-3: ordinal not in range(128)

In [31]: s.encode('latin1')
---------------------------------------------------------------------------
UnicodeEncodeError                        Traceback (most recent call last)
<ipython-input-31-d1ce3d2fd1b8> in <module>
----> 1 s.encode('latin1')

UnicodeEncodeError: 'latin-1' codec can't encode characters in position 0-3: ordinal not in range(256)

In [32]: s.encode('latin1', errors='ignore').decode('latin1')
Out[32]: '    '

In [33]: s.encode('utf7').decode('utf7')
Out[33]: 'Това изречение е на български'

In [34]: s.encode('utf7').decode('utf8')
Out[34]: '+BCIEPgQyBDA +BDgENwRABDUERwQ1BD0EOAQ1 +BDU +BD0EMA +BDEESgQ7BDMEMARABEEEOgQ4-'

In [35]: s.encode('utf8').decode('utf8')
Out[35]: 'Това изречение е на български'
```

Regular Expressions

As an NLP data scientist a knowledge of regular expressions is not optional.

You will use them for text pre-processing, keyword search, rule-based NLP approaches, even search and replace operations in your IDE (integrated development environment) or text editor of choice, in the Unix grep command, etc.

Regular Expressions are more or less standardized across programming languages and operating systems.

Regular Expression (regex) - a notation for characterizing a set of strings.

Created in 1951 by the American mathematician Kleene. Closely related to the computer science subfields of automata and formal languages.

Regular Expression Patterns

https://docs.python.org/3/library/re.html

. matches any character except a newline. If the DOTALL flag has been specified, this matches any character including a newline.

* Causes the resulting RE to match 0 or more repetitions of the preceding RE, as many repetitions as are possible. ab* will match ‘a’, ‘ab’, or ‘a’ followed by any number of ‘b’s.

+ Causes the resulting RE to match 1 or more repetitions of the preceding RE. ab+ will match ‘a’ followed by any non-zero number of ‘b’s; it will not match just ‘a’.

? Causes the resulting RE to match 0 or 1 repetitions of the preceding RE. ab? will match either ‘a’ or ‘ab’.

{m} Specifies that exactly m copies of the previous RE should be matched; fewer matches cause the entire RE not to match. For example, a{6} will match exactly six 'a' characters, but not five."

{m,n} Causes the resulting RE to match from m to n repetitions of the preceding RE, attempting to match as many repetitions as possible. For example, a{3,5} will match from 3 to 5 'a' characters. Omitting m specifies a lower bound of zero, and omitting n specifies an infinite upper bound. As an example, a{4,}b will match 'aaaab' or a thousand 'a' characters followed by a 'b', but not 'aaab'. The comma may not be omitted or the modifier would be confused with the previously described form.

(...) Matches whatever regular expression is inside the parentheses, and indicates the start and end of a group.

\ Either escapes special characters (permitting you to match characters like '*', '?', and so forth), or signals a special sequence; special sequences are discussed below.

^ (Caret.) Matches the start of the string, and in MULTILINE mode also matches immediately after each newline.

$ Matches the end of the string or just before the newline at the end of the string, and in MULTILINE mode also matches before a newline. foo matches both ‘foo’ and ‘foobar’, while the regular expression foo$ matches only ‘foo’. More interestingly, searching for foo.$ in 'foo1\nfoo2\n' matches ‘foo2’ normally, but ‘foo1’ in MULTILINE mode; searching for a single $ in 'foo\n' will find two (empty) matches: one just before the newline, and one at the end of the string.

[] → Used to indicate a set of characters. In a set: Characters can be listed individually, e.g. [amk] will match 'a', 'm', or 'k'. Ranges of characters can be indicated by giving two characters and separating them by a '-', for example [a-z] will match any lowercase ASCII letter, [0-5][0-9] will match all the two-digits numbers from 00 to 59. If the first character of the set is '^', all the characters that are not in the set will be matched. For example, [^5] will match any character except '5', and [^^] will match any character except '^'. ^ has no special meaning if it’s not the first character in the set.

| → A|B, where A and B can be arbitrary REs, creates a regular expression that will match either A or B. An arbitrary number of REs can be separated by the '|' in this way.

Regex Special Sequences

consist of '\' and a character from the list below:

\b → word boundary

\d → digit, same as [0-9] + Unicode digits

\s → white space, ASCII [\t\n\r\f\v] + Unicode white space chars

\w → Matches Unicode word characters; this includes most characters that can be part of a word in any language, as well as numbers and the underscore. If the ASCII flag is used, only [a-zA-Z0-9_] is matched.

Regex Compilation Flags

re.IGNORECASE --> Perform case-insensitive matching;

re.MULTILINE --> When specified, the pattern character '^' matches at the beginning of the string and at the beginning of each line (immediately following each newline); and the pattern character '$' matches at the end of the string and at the end of each line (immediately preceding each newline).

re.DOTALL --> Make the '.' special character match any character at all, including a newline; without this flag, '.' will match anything except a newline. Corresponds to the inline flag (?s).

re.compile(pattern, flags= re.IGNORECASE | re.MULTILINE | re.DOTALL)

Regex Exercises

Write a regular expression to find:

Email addresses in text

Phone numbers

URLs

Text Preprocessing: Tokenization

The task of dividing text into meaningful units. Similar to the task of sentence segmentation and word-segmentation, though token != word.

Why would you need tokenization?

Come up with a tokenizer for English newswire sentences:

Notorious GandCrab hacker group 'returns from retirement'

Researchers at cyber-security company Secureworks say they reached their conclusion after analyzing a new strain of computer virus. Their code had scrambled data on victims' computers and demanded blackmail payments to decrypt it. It is estimated to have affected more than 1.5 million machines, with hospitals among those affected. In May, the group had surprised many in the security industry when it announced it was "retiring" after earning more than $2bn (£1.6bn) from the trade. Source: BBC.com

General Domain Tokenization Considerations

Period: *Dr. , the U.S.,*

Apostrophe: *I'll, isn't, the boys' toys, yesterday's talk*

Hyphenation: *ice-box = ice box = icebox, A-1 paper size, a final "take-it-or-leave-it" offer*

Numbers: *$12.99, +1 888 777 4041*

Tokenization in other languages: [某人]都还给老师了。

Domain-specific Tokenization

*DISCHARGE MEDICATIONS: 1. Lisinopril 5 mg q. Day. 2. Metoprolol 25 mg twice a day. 3. Furosemide 20 mg q. Day. 4. Lovenox 100 mg subcutaneously q. day until INR greater than 2.0. 5. Levofloxacin 250 mg q. day for two more days until [**11-20**].*

*DISCHARGE INSTRUCTIONS: 1. The patient's primary care physician is [**Last Name (NamePattern4) **]. [**First Name8 (NamePattern2) **] [**Last Name (NamePattern1) **] at [**Telephone/Fax (1) 21233**]. The patient does not have a Cardiologist. He needs a Cardiologist for his outpatient follow-up. Dr. [**Last Name (STitle) 295**] will recommend a Cardiologist for his outpatient follow-up. 2. He will also have his INR checked on Wednesday [**3145-11-21**]. The goal of his INR would be 2.0 to 3.0.*

"Model: PSC23MG/PSI23MG\nEXCLUSIVE CustomCool technology\nEXCLUSIVE ClimateKeeper\"temperatur\nmanagement system with TurboCool\" setting\nEXCLUSIVE FrostGuard\" technology\nEXCLUSIVE SmartWater\" Plus\nfiltration system\nEXCLUSIVE LightTouch! dispenser\nChild lock Quick Ice\" setting\nDoor alarm Filtration indicator light\nEXCLUSIVE Ti-level fresh food lighting\nElectronic touch temperature controls\nDigital temperature display\nVIntegrated snelt support System\nAdjustable gallon door storage\nSlide-out shelves\nSpilproof shelves\nQuickSpace\" shelf\nTilt-out freezer door bins\nSlide 'n Store full-extension freezer basket\nSlide-out wire freezer baskets\nContoured doors\nQuiet design (Premium)\nCapacities (cu. ft.) and Dimensions:\nFresh\nFood Freezer Total Dimensions (H\nModel\nPSIZ3MG\n1438\n826\n\"\" \"Heght to top of hinge 1 /4.\n\"**Add 2-716 for door andle\nOG\n"

Tokenization: practical considerations

Tokenization is not a solved problem. Evaluate your tokenizers on each dataset. Familiarize yourself with NLP lib tokenizers' source code and functionality:

https://www.nltk.org/api/nltk.tokenize.html

https://github.com/explosion/spaCy/blob/master/spacy/tokenizer.pyx

https://github.com/apache/opennlp/tree/master/opennlp-tools/src/main/java/opennlp/tools/tokenize

Text Normalization

Ignore case? (convert to lower/upper case)

Ignore numbers? (19.99, 144 → <number>, 19.99, 144 → 00.00, 000)

Remove non-alphanumeric characters? White space normalization?

Unicode equivalence?

```
In [70]: s1="Amélie"

In [71]: s2="Amélie"

In [72]: s1==s2
Out[72]: False

In [73]: print(len(s1), len(s2))
6 7
```

A	m	é		l	i	e
0041	006d	00e9		006c	0069	0065
0041	006d	0065	0301	006c	0069	0065
A	m	e	◌́	l	i	e

Stemming? The Porter Stemmer algorithm.

Part-of-Speech (POS) Tagging

a.k.a grammatical tagging, word-category disambiguation

Less common pre-processing step, POS features do not always improve ML performance, especially in the era of deep-learning. Very few resources available for domain-specific POS tagging.

POS-tagging exercise (English grammar 101):

“I am not young enough to know everything.”

— Oscar Wilde

POS Corpora

The Brown Corpus: 500 texts, ~ 1 million words. Domains: fiction and non-fiction books, news articles.

The Penn Treebank: includes the Brown corpus, plus additional news articles, bulletins, radio transcripts, etc. Close to 5 million tagged words (Includes syntactic parsing).

List of POS tags used in the Penn Treebank Project

https://universaldependencies.org/u/pos/all.html

See https://nlp.stanford.edu/links/statnlp.html See for a list of additional POS corpora and general NLP resources.

Statistical POS Tagging Techniques

HMM: a table of probabilities for certain sequences. For example, once you've seen an article such as 'the', perhaps the next word is a noun 40% of the time, an adjective 40%, and a number 20%. Knowing this, a program can decide that "can" in "the can" is far more likely to be a noun than a verb or a modal. The same method can of course be used to benefit from knowledge about following words.

Viterbi Algorithm - an efficient, dynamic programming algorithm (recursively finding the optimal solutions to the sub-problems and caching them) for finding the most likely sequence of hidden states (POS tags), given a sequence of observed states (words). Given the state transition probabilities, and the initial distribution, compute the most likely sequence of hidden states.

Chunking

A task between POS-tagging and full sentence syntax parsing. Grouping of words into “chunks”, e.g Noun Phrase, Verb Phrase, etc. Shallow parsing.

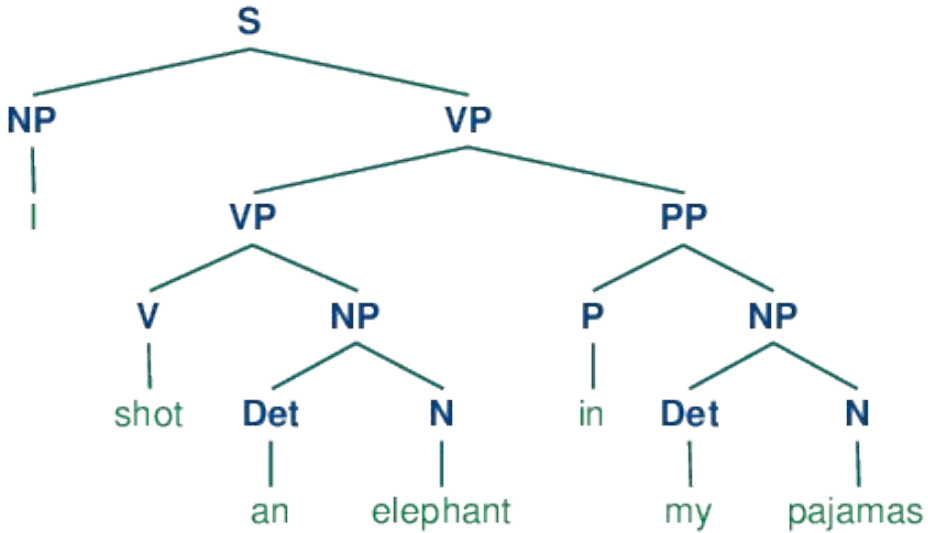
Chunking and Named Entity Recognition are very similar problems: classifying tokens into categories.

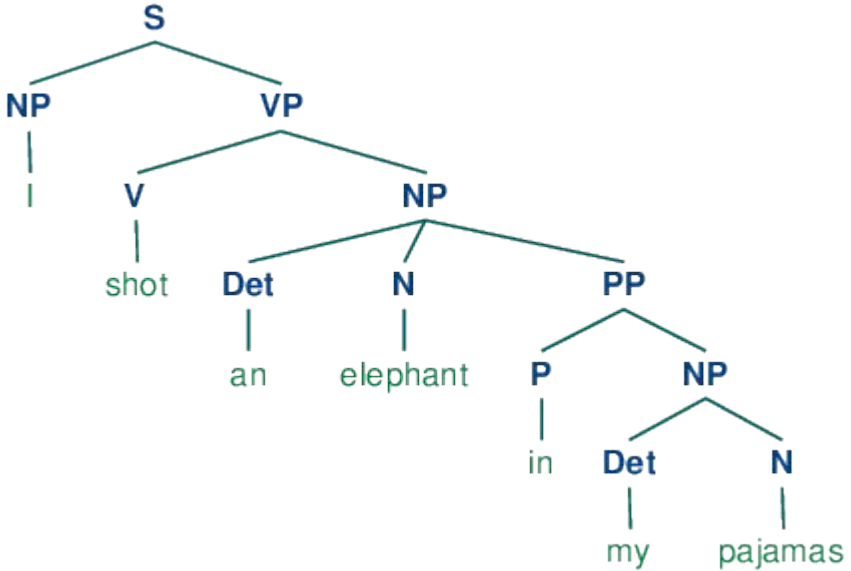
```
(S
  (NP Every/DT day/NN)
  ,/,
  (NP I/PRP)
  (VP buy/VBP)
  (NP something/NN)
  (PP from/IN)
  (NP the/DT corner/NN shop/NN))
```

```
(S
  (NP
    (S
      (NP Every/DT NN day/NN)
      ,/,
      (NP I/PRP)
      (VP
        buy/VBP
        (NP something/NN)
        (PP from/IN (NP the/DT
corner/NN shop/NN))))))
```

Syntax Parsing

Language ambiguity on any level. Context and Grounding.





Probabilistic Context Free Grammars (PCFG)

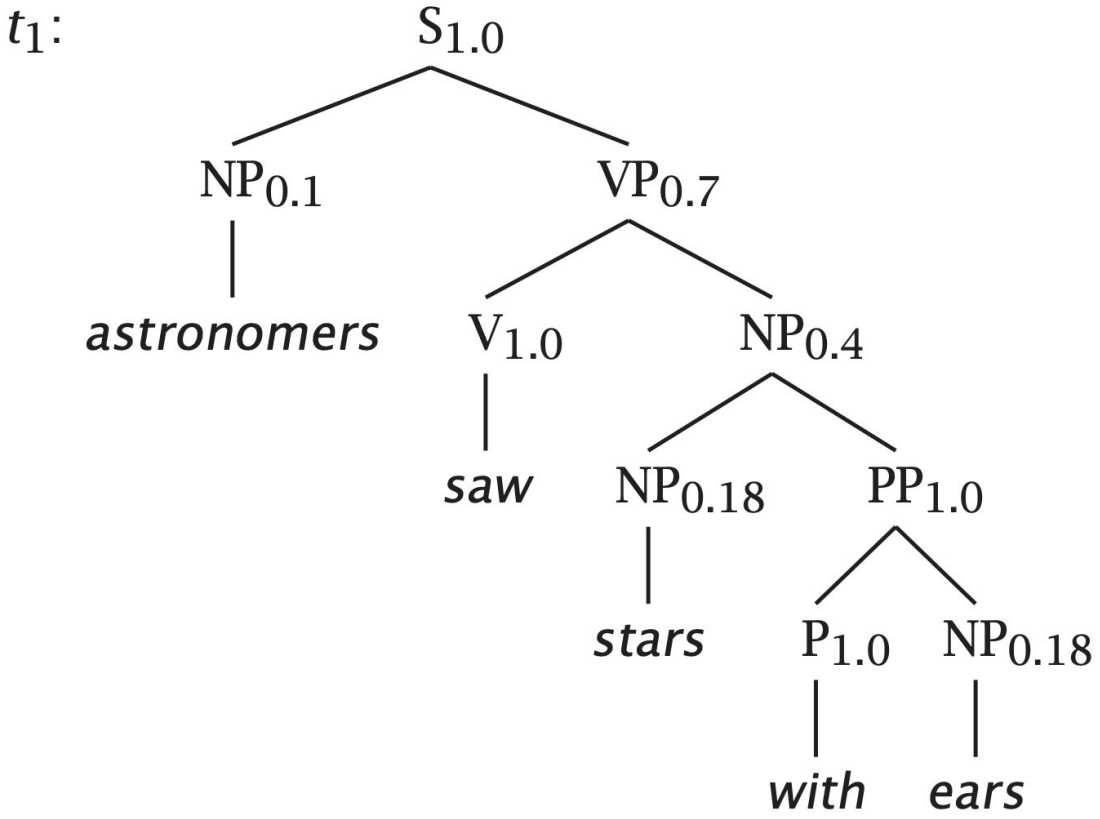
- A start symbol
- A set of terminal symbols
- A set of non-terminal symbols
- A set of transition rules (start symbol → sequences of terminal and non-terminal symbols)
- The corresponding set of probabilities for each transition rule

Context-free: the probability of a subtree doesn’t depend on words outside the subtree or on the order of the words on the subtree.

PCFG Example

S → NP VP	1.0	NP → NP PP	0.4
PP → P NP	1.0	NP → *astronomers*	0.1
VP → V NP	0.7	NP → *ears*	0.18
VP → VP PP	0.3	NP → *saw*	0.04
P → *with*	1.0	NP → *stars*	0.18
V → *saw*	1.0	NP → *telescopes*	0.1

Grammar rules and probabilities. S → start; S, NP, PP, VP, P, V → non-terminals; word in *italics* → terminals.



P(t1) = 1.0 ×

0.1 × 0.7 ×

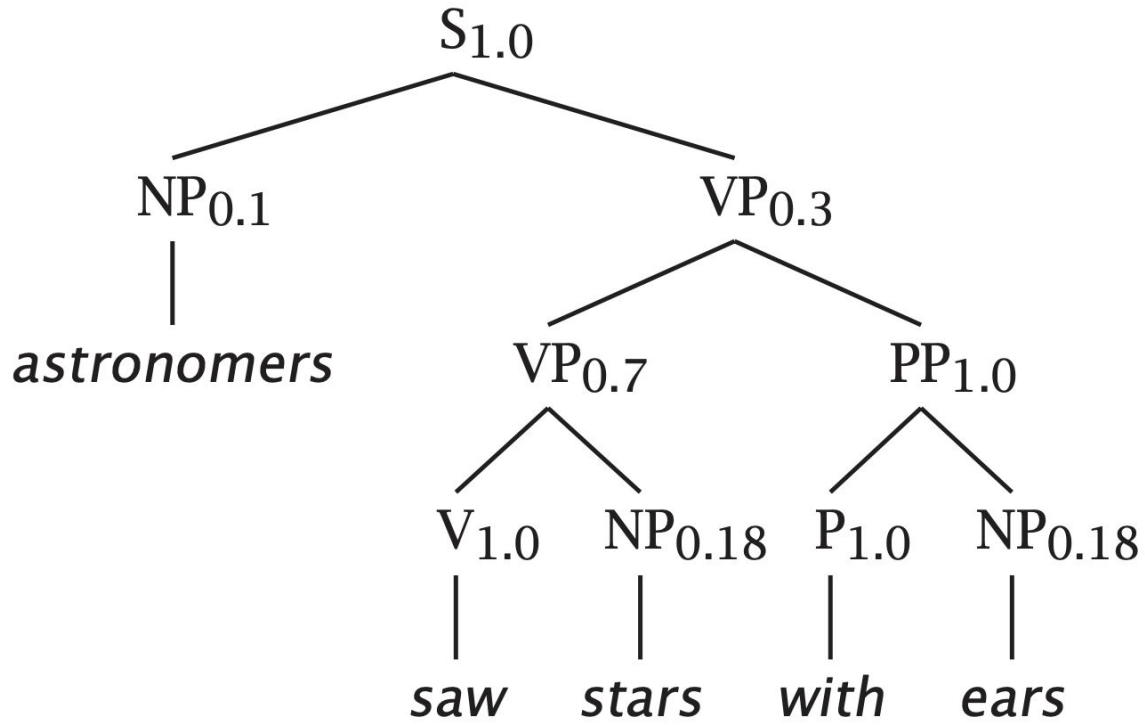
1.0 × 0.4

× 0.18 × 1.0

× 1.0 × 0.18

= 0.0009072

t_2:



P(t2) = 1.0 × 0.1 × 0.3 × 0.7 × 1.0 × 0.18 × 1.0 × 1.0 × 0.18 = 0.0006804

Labs and Homework

The objective of the lab sessions and homework is to prepare you for being a successful NLP data scientist / researcher / engineer. As a result, the class tasks attempt to prepare you for what you can expect working in the field. The focus is on **independent reading** of non-trivial, advanced academic research, **independent software implementations**, including software library installations, and troubleshooting. Your work will also be judged on how well you are **able to communicate your ideas/work**.

License and data usage agreements: Please make sure you understand and comply with the license and usage agreements for any library or dataset you are using as part of the Text Analytics course.

Homework source code links

Create a msia414 git repository under your account in https://github.com/MSIA. Make sure that I and the course TA have permissions to access your repo (my github id is *ema-*, note the dash at the end, the TA's id is *timoderbeste*)

Create a **git branch for each homework** / assignment. For example:

git checkout -b homework1

Edit, add and commit your files. Push your branch to the remote repository:

git push -u origin homework1 (where origin is your github remote)

Provide a link to your branch with the homework submission file. Note that it will reflect the time of your last branch commit.

Lecture 1 Required Reading

Course objective: Independent reading of advanced academic publications. Involves reading some of the cited literature, independent research on mentioned approaches, corpora, performance metrics, source code, etc. Give yourself enough time for each required reading publication! In my experience, sometimes it can take me a full day to read/fully understand a 4-page scientific paper...

Charniak E. A maximum-entropy-inspired parser. in Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference 2000 Apr 29 (pp. 132-139). Association for Computational Linguistics.