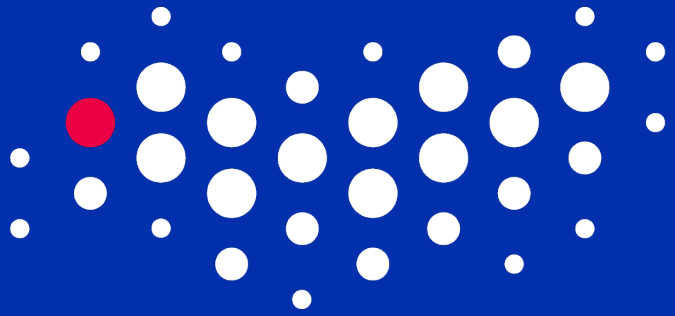


ITMO UNIVERSITY

NLP4IS

Unit: Word Embeddings – Research Topics (Part 1)

Oct 12th, 2018

Outline of this unit

- Finish slides unit 3 – Use Case 2
- After the first couple of units were very practical and applied, this unit is more **research-focused .. and provides foundations for (part of) the projects**
  - Word embeddings / Digital humanties
- NLP tasks: **Named entity recognition** and **coreference resolution**
- Spacy

Research Use Case: Digital humanities (more details)

- You already saw the basic idea in Use Case 1 (last unit)
  - **Research Background:** Word embeddings have been shown to provide good accuracy for word similarity and analogies if **trained on huge corpora**
- **Research Question:**
  - How well do current NLP methods, esp. word embeddings, perform on verifying fine-grained relations and knowledge **from book series corpora?**

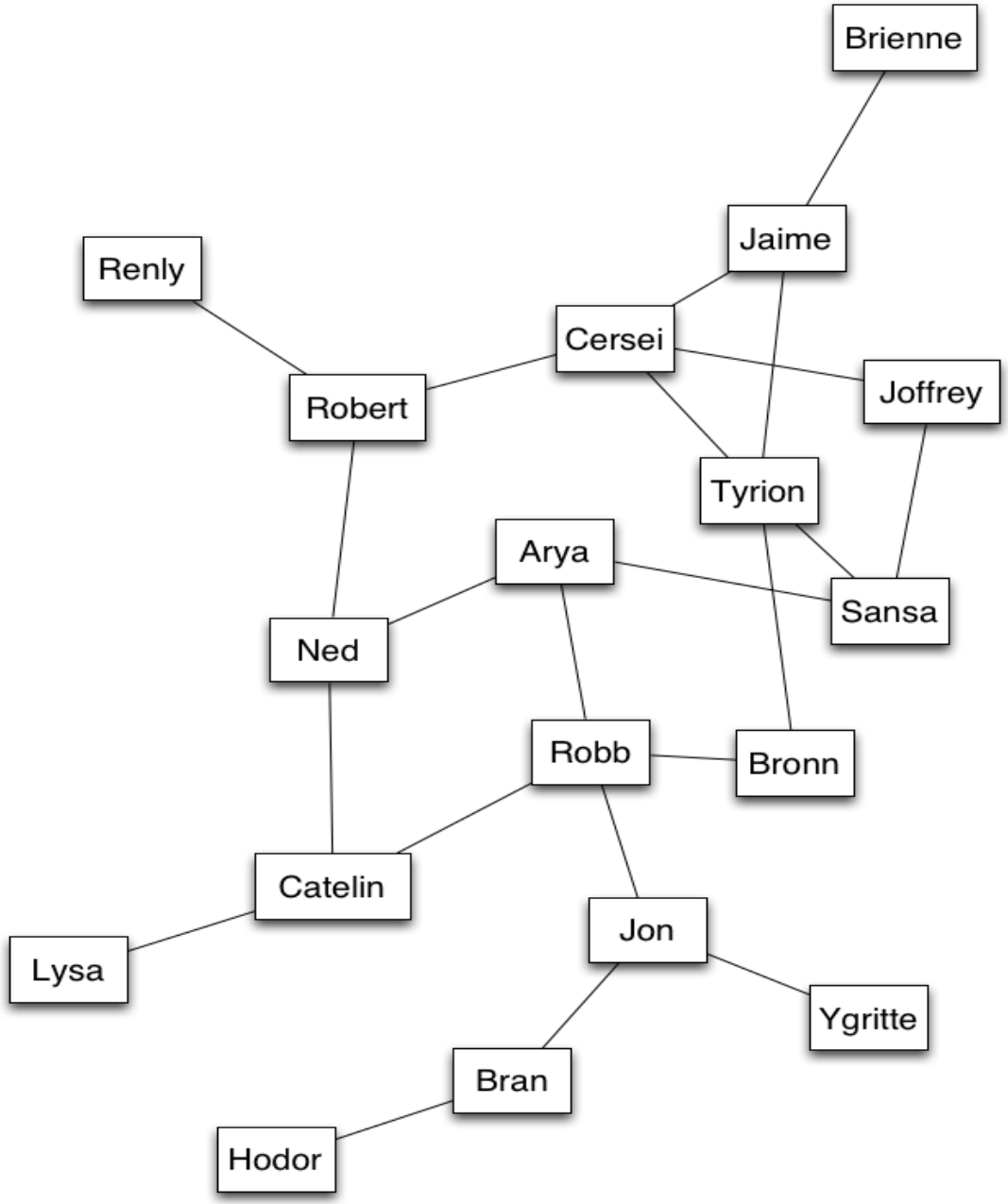
Research UC: Digital humanities (2)

- **Circumstances:**
  - Small Corpus – little training data
  - Domain-specific language and relations
- **Research goals:**
  - See how well word embeddings work for **social network construction** from books.
  - We aim to provide **baseline** benchmarks and **datasets** for **word intrusion and analogies**.
  - Evaluate those datasets with **different WE models** and types.

Research UC: Digital humanities (3) – Social Network extraction

- This was the first idea on how to apply word embeddings
- Scenario:
  - **Given** a book series like “A song of ice and fire” or Harry Potter - in plain text.
  - **Given** a list of eg. 30 most important characters in the books – from an external source.
- How can we create social networks, eg. the 2 strongest relations of every character? Ideas? (see example on next slide)

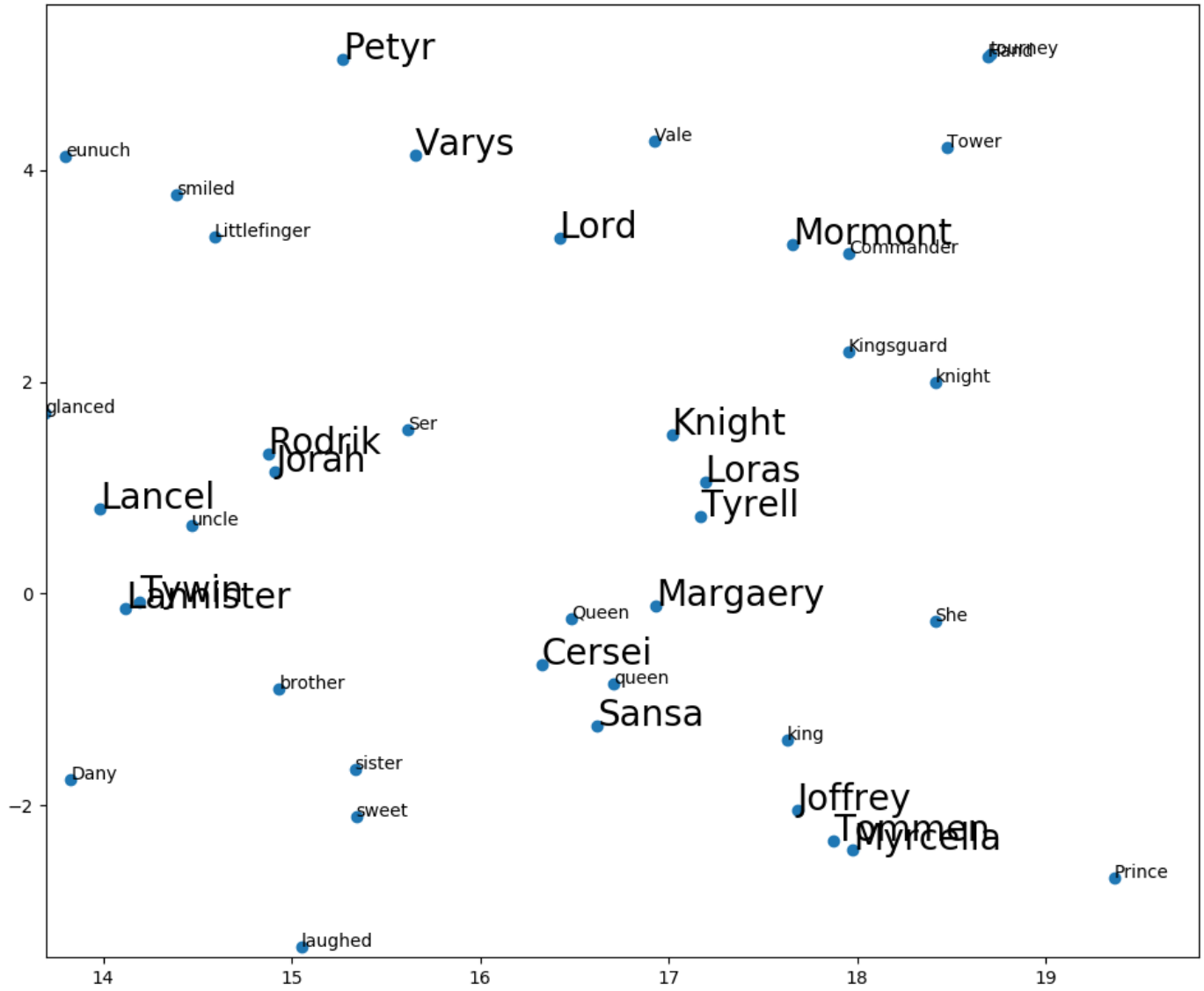
Social network extraction – example graph



Research UC: Digital humanities (4) – Social network extraction

- **Basic (simple idea)**
  - Learn word embeddings on the book corpora
  - For every character, use the embedding model to find the (n=2) closest characters (via cosine similarity)
- **Baseline methods:**
  - ***What is a baseline method?***
  - ***Co-occurrence:*** *counts of co-occurring characters in same sentence, same paragraph, same chapter.*
  - *Similarity using PPMI (we discussed it)*

Social network extraction - Intuition



Social network extraction - Evaluation

- Let's say the systems has the following output, for closest relations to "Jon Snow":
  - Word2vec: Robb, Sansa
  - GloVe: Robb, Ned
  - Co-occ (sentence): Robb, Arya
- Let's say the systems has the following output, for closest relations to "Harry Potter":
  - Word2vec: Ron, Dumbledore
  - Co-occ: Hermine, Ron
- How can we judge? What is a **gold standard?** What to do to create a gold dataset?

Social network extraction – Evaluation (2)

- Two important ways to create a gold dataset:
  - Domain experts
  - Crowdsourcing. What is crowdsourcing?
- Domain experts:
  - In highly technical domains often the only option
  - **Advantage:** usually high quality results, can handle complex problems
  - **Disadvantage:** expensive, not scalable

Social network extraction – Evaluation (3)

Crowdsourcing

- Ask crowdworkers (typically paid) for their judgments
  Often “microtask crowdsourcing”
- Typically more than one judgment per unit, and then use **aggregation rules** (eg. majority vote)
- **Paid crowdsourcing** vs. **games with a purpose**
- Has become very popular for dataset creation in the last couple of years – eg. to create big datasets in order to apply ML methods

Sample question to crowdworker

A Song Of Ice And Fire: Relate Relations Between Characters

Instructions ▲

Please only choose this tasks if you have read the **"Song of Ice and Fire (SOIAF)"** book series by George RR Martin.
The task **requires detailed knowledge** about the characters of the book series.

For a given character of the books, please choose two characters which have the **strongest connection** to the given character. It doesn't matter if a character has already died in the books -- choose the strongest relations throughout the books.

For example:
Catelyn (Stark):

- strong connections to: Ned (Husband), Lysa (Sister), ...
- No strong connections to: Oberyn, Loras, Dany, ...

You choose the 2 candidates where you think the relation is strongest.

***Stannis* has the strongest relation to the following characters (choose two):**

- ☐ Joffrey
- ☐ Robert
- ☐ Cersei
- ☐ Varys
- ☐ Hoster
- ☐ Loras
- ☐ Davos
- ☐ Renly
- ☐ Jon

Social network extraction – Evaluation setup

- 2 book series (ASOIF and HP)
- **30 / 25 most frequent characters** (character list from external source)
- Crowdsourcing setup: **15 votes per unit**, only level 3 (best) workers
- 2 characters with strongest relations, vs characters with more than 4 votes

Social Network Extraction – Results

Table 4 Book series: "Harry Potter": Agreement between the sets of suggested character relations from CF workers and the LT methods, according to the score in Eq. 2.

LT Method	Agreement Top-2	Agreement 4 Votes
(i) Chapter Co-occ	52.5%	65.0%
(ii) Paragraph Co-occ	**70.0%**	**80.0%**
(iii) Sentence Co-occ	**70.0%**	**80.0%**
(iv) w2v-default	45.0%	55.0%
(v) w2v-ww12-i15	65.0%	75.0%
(vi) w2v-ww12-i15-ns	67.5%	**77.5%**
(vii) w2v-CBOW	27.5%	35.0%
(viii) GloVe	35.0%	42.5%
(ix) FastText	**70.0%**	**80.0%**
(x) LexVec	12.5%	17.5%
(xi) Baseline: PPMI	57.5%	70.0%
(xiii) Baseline: Wikipedia FastText	17.5%	22.5%
(xiv) Baseline: Text8 w2v updated	62.5%	72.5%

Word Intrusion and Analogies

- Already discussed, reminder:
  - **Word intrusion:** 4 terms (a b c d), detect intruder
  - **Analogies:** eg. king to queen like prince to X, find X.
- Again models trained on book corpora
- Different categories, see next slide
- Baselines: embeddings trained on Wikipedia, PPMI, others

Word Intrusion and Analogies – Dataset sizes

Table 5 Overview of datasets used for the Analogies and Word Intrusion tasks. Cells contain the number of questions (task units) and number of task categories in parentheses.

Book Series	Dataset-Type	Uni-gram	N-Gram
ASOIF	Analogies	2848 (8)	192 (2)
	Word Intrusion	11180 (13)	2000 (7)
HP	Analogies	4790 (17)	92 (7)
	Word Intrusion	8340 (19)	1920 (7)

Word Intrusion - Results

Table 8 ASOIF Word Intrusion dataset: Accuracy of various word embedding models on selected word intrusion task sections, and total accuracy.

Task Section	family-siblings	names-of-houses	Stark clan	free cities	Total
Number of tasks:	160	7280	1120	700	11180
w2v-default	86.25	75.14	95.98	85.86	**80.33**
w2v-w12-i15	86.25	62.03	92.14	94.57	71.91
w2v-w12-i15-ns	88.12	60.85	91.88	94.29	71.47
w2v-w12-cbow	85.0	57.95	75.27	90.57	66.59
FastText	90.0	58.74	89.64	92.57	70.23
GloVe	80.62	73.19	90.62	88.14	76.25
LexVec	82.5	62.36	90.36	82.43	70.93
Text8 w2v updated	83.75	57.73	89.29	93.57	67.93
Baseline: PPMI	82.8	67.95	75.0	99.47	70.37
Baseline: WP FastText	54.37	51.73	46.07	36.71	49.69

Analogies - Results

Table 6 ASOIF Analogies dataset (uni-grams): Accuracy of various word embedding mod on selected analogy task groups, and total accuracy.

Task Section	first-last-name	child-father	husband-wife	name-location	houses-seats	Total
Number of tasks:	2368	180	30	168	30	2848
w2v-default	11.99	7.22	0.0	1.19	36.67	11.10
w2v-w12-i15	36.74	2.22	6.67	10.71	33.33	32.27
w2v-w12-i15-ns	41.09	4.44	0.0	9.52	30.0	**35.81**
w2v-w12-cbow	0.08	3.89	3.33	9.52	3.33	1.02
GloVe	36.23	3.89	0.0	0.0	30.0	30.86
FastText	14.32	0.56	0.0	0.0	0.0	11.97
LexVec	0.04	8.89	0.0	0.0	0.0	0.6
Text8 w2v updated	36.05	2.63	6.67	5.41	43.33	32.29
Baseline: ONLY-B	31.12	0.0	0.0	0.0	0.0	25.81

Planned student projects

- Project 1: Add **co-reference resolution** to ASIOF and HP
  - Co-ref resolution: Eg. **Robb** fights, and **he** wins.
- Project 2: Study the effect of dataset size on word similarity and analogy quality
  - Using general domain Wikipedia text
- Project 3: Relation extraction with deep learning frameworks and distant supervision (via DBpedia)
- Maybe other projects (dep. on interest and number of projects by Liubov), also I am open to ideas ..

Other projects 1: (will be discussed in more detail later)

- Project: Evaluate the accuracy of WE models trained on different text sizes
  - Inspired by: **Sahlgren and Lenci (2016)**: https://www.aclweb.org/anthology/D16-1099
  - Text Dataset: **Wikipedia**
  - Sizes: eg. 1M, 5M, 10M, 50M, 100M, 500M
  - **Evaluation Datasets:** Word similarity (MEN, WordSim-353, SimLex-999), Analogy datasets (Mikolov, BATS)
  - Here we specifically look at the **impact of term frequencies** (using frequency bins like in Sahlgren and Lenci) on task accuracy

Other projects 2: (will be discussed in more detail later)

- Project: **(Neural) Relation extraction** with OpenNRE and a dataset constructed from DBpedia
  - OpenNRE is an opensource framework for neural relation extraction (supports CNN, PCNN, RNN, BiRNN, etc) – some previous experience helpful with this project
  - Students should extract a dataset from DBpedia (Gerhard can help with this) and then try to do relation extraction (with distant supervision)
  - 1-3 students, depending on depth of investigation, an maybe application of other relation extraction frameworks

NLP Basics: Part-of-speech tagging with NLTK (and Spacy)

- **POS-tagging** has **already been discussed with Liubov in Unit 1**
  - POS-tagging is "also called grammatical tagging or word-category disambiguation, is the process of marking up a word in a text as corresponding to a particular part of speech, based on the context. A simplified form of this is commonly taught to school-age children, in the identification of words as nouns, verbs, adjectives, adverbs, etc."
- POS-tags depend on context. 'to **fight**' vs 'the **fight**'
- Upcoming slides on POS-tagging from
  - See http://web.media.mit.edu/~havasi/MAS.S60/PNLP6.pdf for more details

Tagsets

- What is a tagset?
- Standards and tagging
- Tags for parts of speech:
  - Nouns, verbs, adverbs, adjectives, articles, etc
  - Subtagging
    - nouns can be singular or plural
    - verbs have tenses
  - Different tagsets have different focuses

Tags are cryptic

```
>>> text = nltk.word_tokenize("And now for something completely different")
>>> nltk.pos_tag(text)
[('And', 'CC'), ('now', 'RB'), ('for', 'IN'), ('something', 'NN'),
('completely', 'RB'), ('different', 'JJ')]
```

- Brown and Treebank established some cryptic tags; everyone tends to use Treebank's
  - CC = coordinating conjunction
  - RB = adverb
  - IN = preposition
  - NN = noun
  - JJ = adjective

NLTK can help

- We don’t even remember what all the tags mean sometimes
- but nltk.help.upenn_tagset(*tag*) does!

Homographs

```
>>> text = nltk.word_tokenize("They
refuse to permit us to obtain the refuse
permit")

>>> nltk.pos_tag(text)
[('They', 'PRP'), ('refuse', 'VBP'),
('to', 'TO'), ('permit', 'VB'), ('us',
'PRP'), ('to', 'TO'), ('obtain', 'VB'),
('the', 'DT'), ('refuse', 'NN'),
('permit', 'NN')]
```

Tags in NLTK

```
>>> tagged_token = nltk.tag.str2tuple('fly/NN')
>>> tagged_token
('fly', 'NN')
>>> tagged_token[0]
'fly'
>>> tagged_token[1]
'NN'
```

- Tags are tuples
- Tags can be converted by NLTK between tagsets

Making tuples from a corpus

```
>>> sent = '''
... The/AT grand/JJ jury/NN commented/VBD on/IN a/AT
number/NN of/IN
... other/AP topics/NNS ,/, AMONG/IN them/PPO the/AT
Atlanta/NP and/CC
... Fulton/NP-tl County/NN-tl purchasing/VBG departments/
NNS which/WDT it/PPS
... said/VBD ``/`` ARE/BER well/QL operated/VBN and/CC
follow/VB generally/RB
... accepted/VBN practices/NNS which/WDT inure/VB to/IN
the/AT best/JJT
... interest/NN of/IN both/ABX governments/NNS ''/'' ./.
... '''
>>> [nltk.tag.str2tuple(t) for t in sent.split()]
[('The', 'AT'), ('grand', 'JJ'), ('jury', 'NN'),
('commented', 'VBD'),
('on', 'IN'), ('a', 'AT'), ('number', 'NN'), ... ('.', '.')
```

Many Tag Sets

- Different corpora have different conventions for tagging.
- There are ISO standards for tagging…
- There were many boring meetings
- NLTK made a simplified, unified tagset
- … which no one uses.

Simplified Tagset of NLTK

Tag	Meaning	Examples
ADJ	adjective	*new, good, high, special, big, local*
ADV	adverb	*really, already, still, early, now*
CNJ	conjunction	*and, or, but, if, while, although*
DET	determiner	*the, a, some, most, every, no*
EX	existential	*there, there's*
FW	foreign word	*dolce, ersatz, esprit, quo, maitre*
MOD	modal verb	*will, can, would, may, must, should*
N	noun	*year, home, costs, time, education*
NP	proper noun	*Alison, Africa, April, Washington*
NUM	number	*twenty-four, fourth, 1991, 14:24*
PRO	pronoun	*he, their, her, its, my, I, us*
P	preposition	*on, of, at, with, by, into, under*
TO	the word *to*	*to*
UH	interjection	*ah, bang, ha, whee, hmpf, oops*
V	verb	*is, has, get, do, make, see, run*
VD	past tense	*said, took, told, made, asked*
VG	present participle	*making, going, playing, working*
VN	past participle	*given, taken, begun, sung*
WH	*wh* determiner	*who, which, when, what, where, how*

Tagging in Other Languages

```
>>> nltk.corpus.sinica_treebank.tagged_words()
[('\xe4\xb8\x80', 'Neu'), ('\xe5\x8f\x8b\xe6\x83\x85', 'Nad'), ...]
>>> nltk.corpus.indian.tagged_words()
[('\xe0\xa6\xae\xe0\xa6\xb9\xe0\xa6\xbf\xe0\xa6\xb7\xe0\xa7\x87\xe0\xa6\xb0', 'NN'),
('\xe0\xa6\xb8\xe0\xa6\xa8\xe0\xa7\x8d\xe0\xa6\xa4\xe0\xa6\xbe\xe0\xa6\xa8', 'NN'),
...]
>>> nltk.corpus.mac_morpho.tagged_words()
[('Jersei', 'N'), ('atinge', 'V'), ('m\xe9dia', 'N'), ...]
>>> nltk.corpus.conll2002.tagged_words()
[('Sao', 'NC'), ('Paulo', 'VMI'), ('(', 'Fpa'), ...]
>>> nltk.corpus.cess_cat.tagged_words()
[('El', 'da0ms0'), ('Tribunal_Suprem', 'np0000o'), ...]
```

Bangla: ক ুঁড়েঘরগ ুল রি/'NN' আক ার/'NN' ব াংল ার/'NNP' ব া/'CC' ভ ারতব র্ষে/'NNP' ?/None
ন য/'JJ' ?/None এঞ চল রে/'NN' প রচল তি/'JJ' ক ুঁড়ে/'NN' ঘর/'NN' নয়/'VM' ক/'SYM'
Hindi: प ाकिस्त ान/'NNP' की/'PREP' पू र्व/'JJ' प्रध ानम ंत्री/'NN' बेनज ीर/'NNPC' भ ुट्टो/'NNP'
पर/'PREP' लगे/'VFM' भ्रष्ट ाच ार/'NN' के/'PREP' आर ोपों/'NN' के/'PREP' खिल ाफ/'PREP' भ ुट्टो/'NNP'
द्व ार ा/'PREP' द ायर/'NVB' की/'VFM' गई/'VAUX' य ाचिक ा/'NN' की/'PREP' स ुनव ाई/'NN'
म ंगलव ार/'NN' को/'PREP' वकीलों/'NN' की/'PREP' हड़त ाल/'NN' के/'PREP' क ारण/'PREP'
स्थ गित/'JVB' कर/'VFM' दी/'VAUX' गई/'VAUX' ।/'PUNC'
Marathi: ग्र ामीण/'JJ' जि ल्ह ाध्यक्ष/'NN' ब ाळ स ाहेब/'NNPC' भ ोसले/'NNP' य ांच्य ा/'PRP' ?/None
ध यक्षतेख ाली/'NN' पक्ष ाची/'NN' आज/'NN' ब ै?/None क/'NN' झ ाली/'VM' ./'SYM'
Telugu: ఖాతాదారుల/'NN' నుంచి/'PREP' వచ్చిన/'VJJ' ప త్రాల/'NN' ను/'PREP' సాక్ష్య ా/'NN'

Exercise 1 (simple, 10min)

- Take any book nltk.books or nltk.gutenberg, and part of speech tag it.
  - Start from .raw() text
  - Segment into sentences, then into words.
  - POS-tag the words
- Plot a frequency distribution of the POS-tags
- Plot a frequency distribution of all the nouns in the document

Named Entity Recognition

- See next slides, by Stephan Lesch

  For full slides see:
  http://courses.ischool.berkeley.edu/i290-2/f04/lectures/ner2.ppt

- Example:

  sentence = "**Coca-Cola** spokesman **Mike Tyson** said they will buy **Microsof**t today."
  → Coca-Cola: ORG, Mike Tyson: PERSON, Microsoft: ORG

The who, where, when & how much in a sentence

The task: identify atomic elements of information in text

- person names
- company/organization names
- locations
- dates&times
- percentages
- monetary amounts

example from MUC-7

Delimit the named entities in a text and tag them with NE categores:

<ENAMEX TYPE=„LOCATION“>Italy</ENAMEX>‘s business world was rocked by the announcement <TIMEX TYPE=„DATE“>last Thursday</TIMEX> that Mr. <ENAMEX TYPE=„PERSON“>Verdi</ENAMEX> would leave his job as vice-president of <ENAMEX TYPE=„ORGANIZATION“>Music Masters of Milan, Inc</ENAMEX> to become operations director of <ENAMEX TYPE=„ORGANIZATION“>Arthur Andersen</ENAMEX>.

- „Milan“ is part of organization name
- „Arthur Andersen“ is a company
- „Italy“ is sentence-initial => capitalization useless

difficulties

- too numerous to include in dictionaries
- changing constantly
- appear in many variant forms
- subsequent occurrences might be abbreviated

⇒ list search/matching doesn‘t perform well

Whether a phrase is a proper name, and what name class it has, depends on

- Internal structure:
  „Mr. Brandon“
- Context:
  „The new company, SafeTek, will make air bags.“

Applications

- Information Extraction
- Summary generation
- Machine Translation
- document organization/classification
- automatic indexing of books
- increase accuracy of Internet search results (location Clinton/South Carolina vs. President Clinton)

The hand-crafted approach

uses hand-written context-sensitive reduction rules:

1) title capitalized word => title person_name
   compare „Mr. Jones“ vs. „Mr. Ten-Percent“
   => no rule without exceptions

2) person_name, „the“ adj* „CEO of“ organization
   „Fred Smith, the young dynamic CEO of BlubbCo“
   => ability to grasp non-local patterns

plus help from databases of known named entities

Word features

- Easily determinable token properties:

Feature	Example	Intuition
fourDigitNum	1990	four digit year
containsDigitAndAlpha	A123-456	product code
containsCommaAndPeriod	1.00	monetary amount, percentage
otherNum	34567	other number
allCaps	BBN	Organisation
capPeriod	M.	Person name initial
firstWord	first word of sentence	ignore capitalization
initCap	Sally	capitalized word
lowerCase	can	uncapitalized word
other	,	punctuation, all other words

Histories, bin. features & futures

- History h_t: information derivable from the corpus relative to a token t:
  - text window around token w_i, e.g. $w_{i-2},...,w_{i+2}$
  - word features of these tokens
  - POS, other complex features
- Binary features: yes/no-questions on history used by models to determine probabilities of
- Futures: name classes

Next step: Co-reference Resolution

- "**Joe** walks with **his girlfriend**. **She** likes tea."
- Co-ref resolution important to understand longer texts.
- Step 1: find mentions
- Step 2: cluster connected mentions (which refer to a real-world "thing")

- "**Mike** walks with **his girlfriend**. **She** likes tea."

Co-reference Resolution – **Step 1: Extract Mentions**

- Mentions refer to something concrete in the world
- Mentions are (for example)
  - **Named Entities**
  - **Pronouns:** He, she, they, his, herself, etc.
  - **Common noun** mentions: the tree, her son, man, the park, the naughty child, …
  - **Embedded mentions:** her son
  - **Other nested mentions:** the CFO of IBM

Co-reference Resolution - **Step 2: Create Mention Chains / Clustering**

- Create mention chains
  - → connect mentions which refer to the some real world thing
- Special case: split antecedent.
  "**Mike** and **Ann** go to **Moscow**. **They** take **a plane**."
- Often not clear how to annotate text, if things point to something in the real world.
- **"Co-referent"** if pointing to the some real-world entity

Coreference Resolution: **Applications**

- Text understanding
  - Eg. understanding a discourse structure, eg in our book series: In conversations often back-and-forth with pronouns, etc.
- Machine translation: eg. some languages have no gender in pronouns
- Information / Relation Extraction, question answering, …
  - ”**Mike** and **Ann** go to **Moscow**. **They** take **a plane**.”
  - Q: **Who** flew to Moscow?
  - A: They.

Coref Demo: Neural Coref (intergrated in Spacy)

- https://huggingface.co/coref/
  - Let's try it out
- Demo contains info on how it works and how to train a model (if you are curious)
- Source code is at:
  - https://github.com/huggingface/neuralcoref

Systems 1: Neural Coref

- https://github.com/huggingface/neuralcoref
  - Integrated with spaCy (v2)
  - Easy to use and good accuracy
- Installation:
  - Download spaCy model (for coref, see github)
  - `pip install MODEL_URL`
- Usage:
  - nlp = spacy.load('en_coref_md') # we discuss spacy later
- Misc:
  - Visualisation code: https://github.com/huggingface/neuralcoref-viz

Standford CoreNLP

- Overview: https://nlp.stanford.edu/projects/coref.shtml
- Software: https://stanfordnlp.github.io/CoreNLP/coref.html
- Java-based
- See website for details

- Can also be loaded from within NLTK

Introduction to spaCy

- URL: https://spacy.io/
- Features (they claim):
  - Fast
  - Easy-to-use
  - Mature, gets things done
  - Also for industrial usage
  - Easy to integrate with deep learning
- Python-based

spaCy Installation

- Basics:
  - `bash$ pip install -U spacy`
  - `spacy download en`
- Or with a virtual env (see spacy website)

- First steps:

```
import spacy
nlp = spacy.load('en') # load model

# analyse a document with the model
doc = nlp(u'This is a sentence.')
```

spaCy: First Steps

```
text = open('war_and_peace.txt').read()
doc = nlp(text)

# Find named entities, phrases and concepts
for entity in doc.ents:
    print(entity.text, entity.label_)


# Determine semantic similarities
doc1 = nlp(u'the fries were gross')
doc2 = nlp(u'worst fries ever')
doc1.similarity(doc2)


# Hook in your own deep learning models - or Coref
nlp.add_pipe(load_my_model(), before='parser')
```

spaCy: visual display

```
from spacy import displacy

doc_ent = nlp(u'When Sebastian Thrun started working on self-driving cars at Google '
              u'in 2007, few people outside of the company took him seriously.')

displacy.serve(doc_ent, style='ent')
```

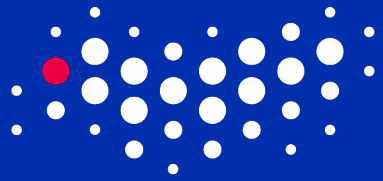
Examples at: https://explosion.ai/demos/

SpaCy: more functions

```
doc = nlp(u"Apple and banana are similar. Pasta and hippo
aren't.")

apple = doc[0]
banana = doc[2]
pasta = doc[6]
hippo = doc[8]

assert apple.similarity(banana) > pasta.similarity(hippo)
assert apple.has_vector, banana.has_vector,
pasta.has_vector, hippo.has_vector
```



ITMO UNIVERSITY

Thank you!

Questions?

Nov 2017