Natural Language processing in Python

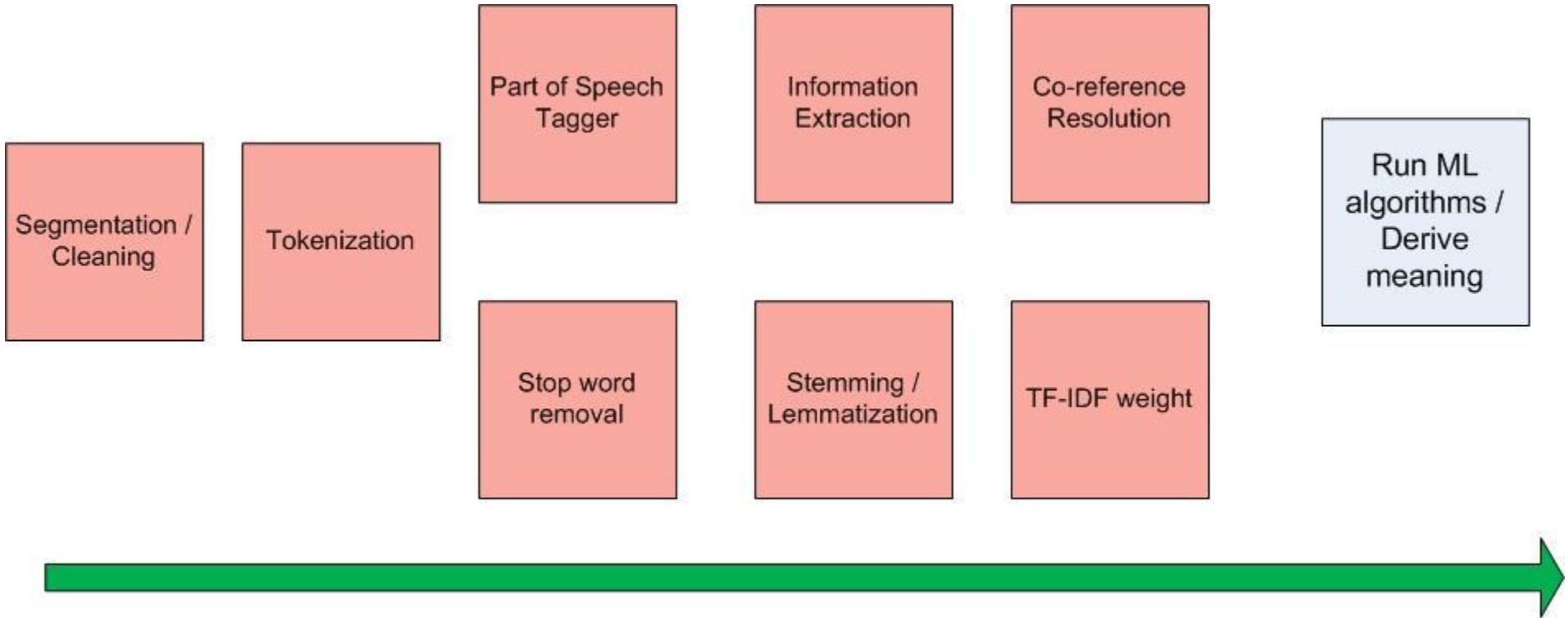
Chuck Chan

May 18, 2016

What is it used for

- Text classification
  - Document classification
  - information extraction
  - Sentiment analysis
- Automatic summarization
- Question answering

Pipeline


Segmentation / Cleaning
Tokenization
Part of Speech Tagger
Information Extraction
Co-reference Resolution
Run ML algorithms / Derive meaning
Stop word removal
Stemming / Lemmatization
TF-IDF weight

Python Libraries

- NLTK
  - Most comprehensive python library for NLP
  - Links to Stanford NER and POS Tagger
  - Apache 2.0 License
- Spacy
  - Cython based with OpenMP for multi-threaded generators
  - Used for speed and production
  - MIT License
- Gensim
  - Topic modeling, word2vec
  - Cython based
  - GNU Lesser General Public License

Cleaning text

- Remove html tags
  - Libraries: BeautifulSoup
- Deal with formatting issues
- Deal with encoding issues (Python 2.7)
  - Libraries: unicodedata, codecs
- No case insensitive

Pre-processing text

- Steps are to clean the data from the documents
  - Get rid of text that will bias classification algorithms
- Correctly weight words based on document other than by frequency
- Refine term counting so things are not double counted

Problems

- Word-sense disambiguation
  - Set[n]:a group or collection of things that belong together
  - Set[v] put, lay, or stand
  - Solved by relabeling with POS tagger and collocation dictionaries
- Metaphors and Similes
  - No reference point for comparisons
- Accuracy and speed

Tokenization

- Breaking a text body up into terms, symbols, and phrases.
- Tokenization can be used to separate out paragraphs, sentences, or terms.
- Usually separated by whitespace, sometimes include punctuation

Filtering Stop Words

- **Stop Word** a word that has the same likelihood of occurring in those documents not relevant to a query as in those documents relevant to the query.

Word	Count
the	3332
and	2972
a	1775
to	1725
of	1440
was	1161
it	1027
in	906
that	877
he	877

Top ten frequent words from Tom Sawyer

Types of words in stop words

- Determiners (the, a, an, another)
- Coordinating conjunctions (for, an, nor, but, or, yet, so)
- Prepositions(in, under, towards, before)
- Punctuation

List of stop words (Rank.nl)

- a
- about
- above
- after
- again
- against
- all
- am
- an
- and
- any
- are
- aren't
- as
- at
- be
- because
- been
- before
- being
- below
- between
- both
- but
- by
- can't
- cannot
- could
- couldn't
- did
- didn't
- do
- does
- doesn't
- doing
- don't
- down
- during
- each
- few
- for
- from
- further
- had
- hadn't
- has
- hasn't
- have
- haven't
- having
- he
- he'd
- he'll
- he's
- her
- here
- here's
- hers
- herself
- him
- himself
- his
- how
- how's
- I
- I'd
- I'll
- I'm
- I've
- if
- in
- into
- is
- isn't
- it
- it's
- its
- itself
- let's
- me
- more
- most
- mustn't
- my
- myself
- no
- nor
- not
- of
- off
- on
- once
- only
- or
- other
- ought
- our
- ours
- ourselves
- out
- over
- own
- same
- shan't
- she
- she'd
- she'll
- she's
- should
- shouldn't
- so
- some
- such
- than
- that
- that's
- the
- their
- theirs
- them
- themselves
- then
- there
- there's
- these
- they
- they'd
- they'll
- they're
- they've
- this
- those
- through
- to
- too
- under
- until
- up
- very
- was
- wasn't
- we
- we'd
- we'll
- we're
- we've
- were
- weren't
- what
- what's
- when
- when's
- where
- where's
- which
- while
- who
- who's
- whom
- why
- why's
- with
- won't
- would
- wouldn't
- you
- you'd
- you'll
- you're
- you've
- your
- yours
- yourself
- yourselves

Contractions & Compound words

- Problems
  - Examples: co-operation / cooperation may be counted as one or two tokens
  - Words such as San Francisco are counted twice for San and for Francisco
  - Possessives and contractions such as “Tom’s” and “you’re” may also be counted as two or more tokens

Compound terms

- Term types
  - Contractions: you're, we'll
  - Compound words: Los Altos, San Francisco, Facebook Inc.
  - Possessives: Tom's, Bill's
  - Collocations: crystal clear, middle management
- Identification
  - Using Named Entity Recognition(NER) to identify entities and tokenize them as one object
  - Chunking
  - Use bigrams to identify compound words manually
  - Have a dictionary or Regex
    - Use join function to combine with underscore
    - Los Altos → "Los_Altos" , you're → "you" "are"

Regular Expressions

- Search pattern mainly used for pattern matching
  - Library re
  - Cheat sheet: http://regexlib.com/CheatSheet.aspx
- Optimization tips
  - Consider speed vs. readability tradeoff.
  - Sparingly use and limit characters for greedy qualifiers [* or +]
  - Specify lengths `(hello){2}`
  - Use non capture group `(?:pattern)`
  - Extract common characters from alterations: use `th(is|at)` vs. `this|that`
  - Order alternations from most common to least `a(?:most|common|to|least)`
  - Use anchors for beginning[^] and end[$] of patterns

Special cases to consider

- Unicode characters
- Math equations, signs, numerical value
  - Fractions, decimals, and equations
- Twitter usernames, hashes, Emails addresses
  - #Hashtag, @Cruz
- Dates and Time
  - Previous dates, future dates, project time range
- Words with apostrophe or dashes
  - Contractions, hyphenated words
- Currency denominations
  - Large values, small values
- Chemical names, Genetic/protein sequences
- Account numbers
  - SSN, telephone, pins, userID

Reducing the number of distinct words

- Lemmatization
  - use of a vocabulary and morphological analysis of words
  - Slow and memory intensive but accurate

car, cars, car's, cars' => car

"better" has "good"

- Stemming
  - Use crude heuristic process that chops off the ends of words
  - Quick, but has false positives

; "universal", "university", and "universe" to "univers".

Part of speech tag

- Labels text to word classes
- Based on stochastic or rules based algorithms

Tag	Meaning	English Examples
ADJ	adjective	*new, good, high, special, big, local*
ADP	adposition	*on, of, at, with, by, into, under*
ADV	adverb	*really, already, still, early, now*
CONJ	conjunction	*and, or, but, if, while, although*
DET	determiner, article	*the, a, some, most, every, no, which*
NOUN	noun	*year, home, costs, time, Africa*
NUM	numeral	*twenty-four, fourth, 1991, 14:24*
PRT	particle	*at, on, out, over per, that, up, with*
PRON	pronoun	*he, their, her, its, my, I, us*
VERB	verb	*is, say, told, given, playing, would*
.	punctuation marks	*. , ; !*
X	other	*ersatz, esprit, dunno, gr8, univeristy*

Information Extraction

- Chunking & Chinking: Extraction of short phrases
  - Based on Part of speech patterns
  - Not very accurate
- NER: Task to extract and classify names of persons, organizations, locations, date time
  - Algorithm Conditional Random fields, Gazetteers, and rules

TF-IDF weights

- Reflect how important a word is to a document in a collection
- TF and IDF weights can have different weighing schemes depending on usage
- Weight increase with number of occurrence in a document and the rarity of the term in the collection

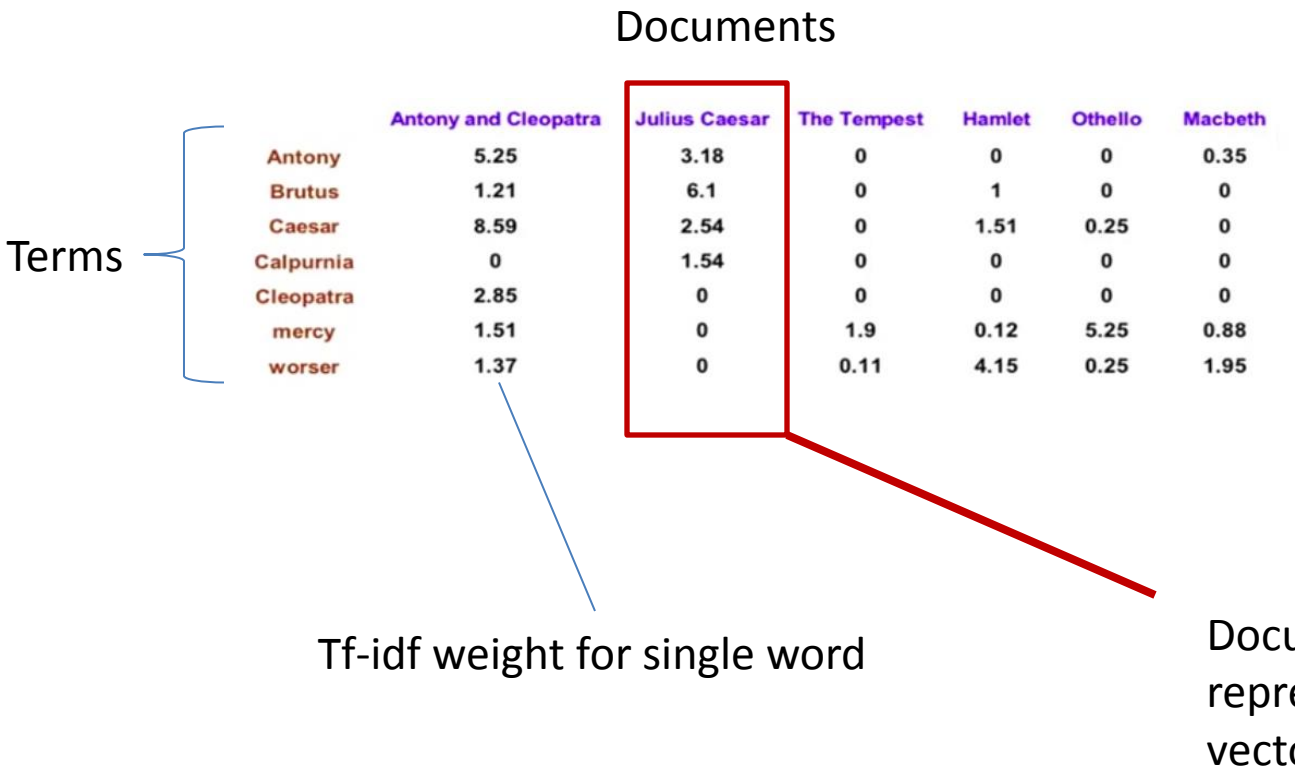
$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_t.$$

$$w_{t,d} = (1 + \log \text{tf}_{t,d}) \times \log_{10}(N / \text{df}_t)$$

Represent document as a weight vector of terms in vector space R

Term Occurrence	tf weight
0	0
1	1
2	1.3
10	2
1000	4

TF-IDF to vector space



Term Frequency

- **Term frequency** measures how frequently a term occurs in a document
  - Generally represented as:
    - $Tf(t) = \frac{(Number\ of\ times\ term\ t\ appears\ in\ a\ document)}{(Total\ number\ of\ terms\ in\ the\ document)}$

We use log scale because relevance does not increase proportionally with term frequency(count)

$$w_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{tf}_{t,d}, & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases}$$

This offsets the case where a term occurs once ($\log_{10}(1) = 0$)

Inverse document Frequency

- **Inverse Document frequency** measure of how much information the word provides across all documents
  - measures how important a term is
  - $IDF(t) = \log \frac{Total\ collection\ of\ documents}{Number\ of\ documents\ with\ term\ t\ in\ it}$
  - IDF (value of each word)
    - rare terms are more informative (should be high weight)
    - Frequent terms are less informative

$$\text{idf}_t = \log_{10}(N/\text{df}_t)$$

Log dampens score

Total
Number of documents

df = document frequency score: number of documents that contain term t, df<N

There is 1 IDF value for each term in a collection

Demonstration

Latent Dirichlet allocation using Gensim