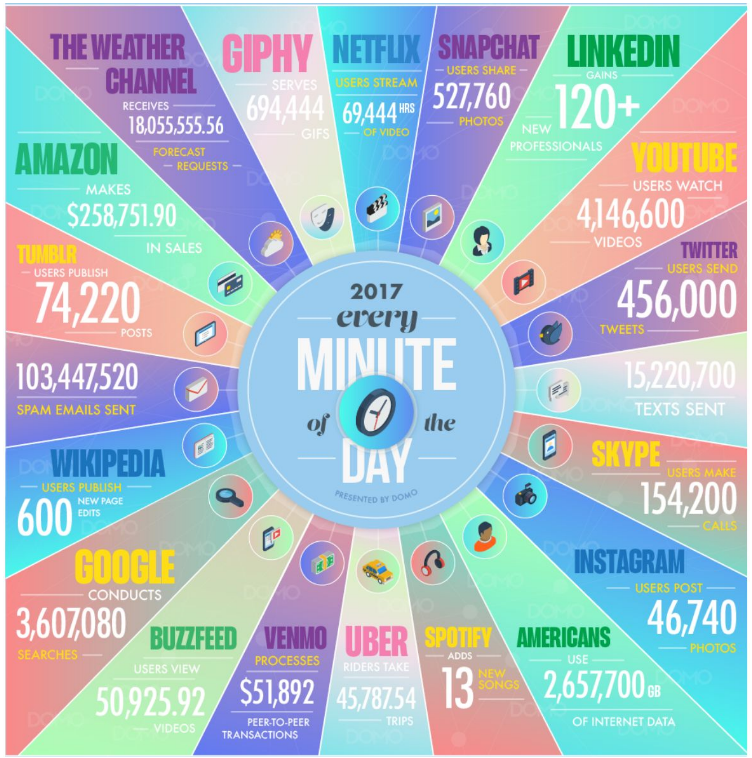
Week 1: Introduction and overview

Text Analytics and Natural Language Processing

Instructor: Benjamin Batorsky

Explosion of data...unstructured data, that is


2017 every MINUTE of the DAY
PRESENTED BY DOMO
THE WEATHER CHANNEL RECEIVES 18,055,555.56 FORECAST REQUESTS
GIPHY SERVES 694,444 GIFS
NETFLIX USERS STREAM 69,444 HRS OF VIDEO
SNAPCHAT USERS SHARE 527,760 PHOTOS
LINKEDIN GAINS 120+ NEW PROFESSIONALS
YOUTUBE USERS WATCH 4,146,600 VIDEOS
TWITTER USERS SEND 456,000 TWEETS
15,220,700 TEXTS SENT
SKYPE USERS MAKE 154,200 CALLS
INSTAGRAM USERS POST 46,740 PHOTOS
AMERICANS USE 2,657,700 GB OF INTERNET DATA
SPOTIFY ADDS 13 NEW SONGS
UBER RIDERS TAKE 45,787.54 TRIPS
VENMO PROCESSES $51,892 PEER-TO-PEER TRANSACTIONS
BUZZFEED USERS VIEW 50,925.92 VIDEOS
GOOGLE CONDUCTS 3,607,080 SEARCHES
WIKIPEDIA USERS PUBLISH 600 NEW PAGE EDITS
103,447,520 SPAM EMAILS SENT
TUMBLR USERS PUBLISH 74,220 POSTS
AMAZON MAKES $258,751.90 IN SALES

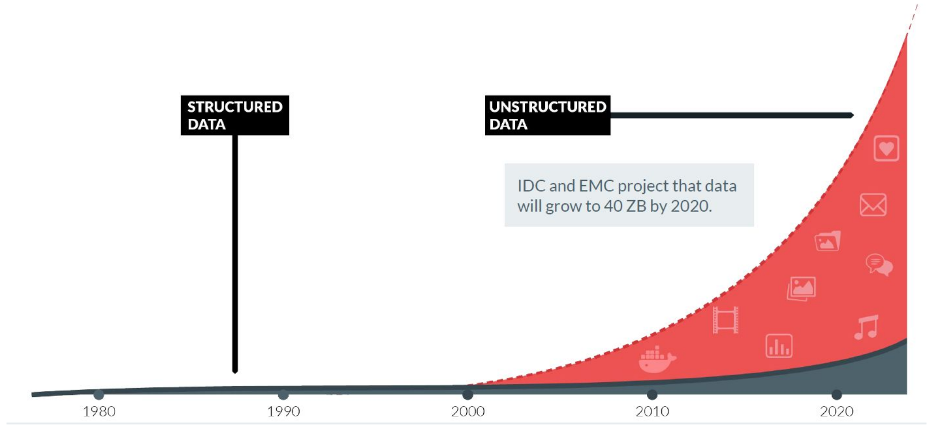
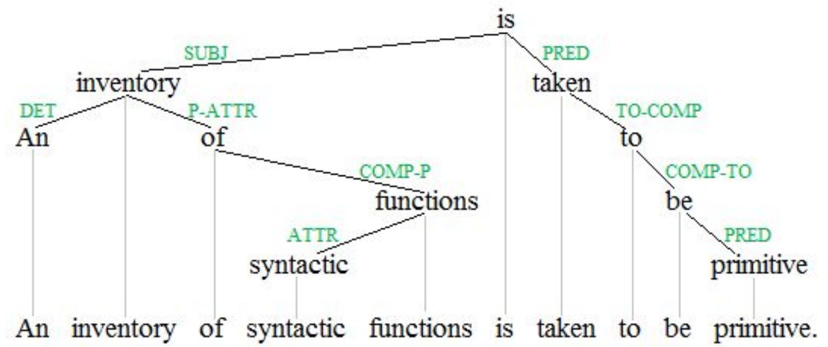


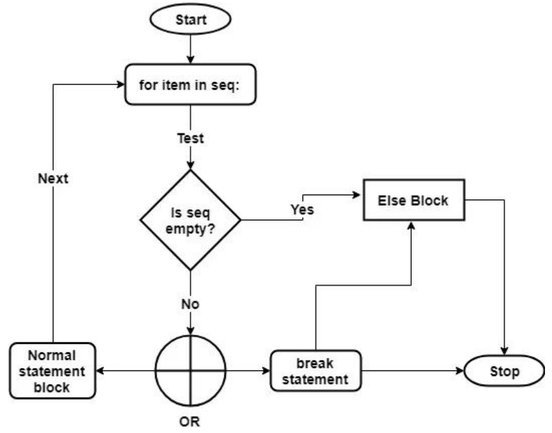
Chart:https://www.datanami.com/2017/02/01/solving-storage-just-beginning-minio-ceo-periasamy/
Data: IDC Structured Versus Unstructured Data: The Balance of Power Continues to Shift, March 2014

What is Natural Language?

"A language that has developed naturally in use (as contrasted with an artificial language or computer code)." (Oxford Dictionary definition)



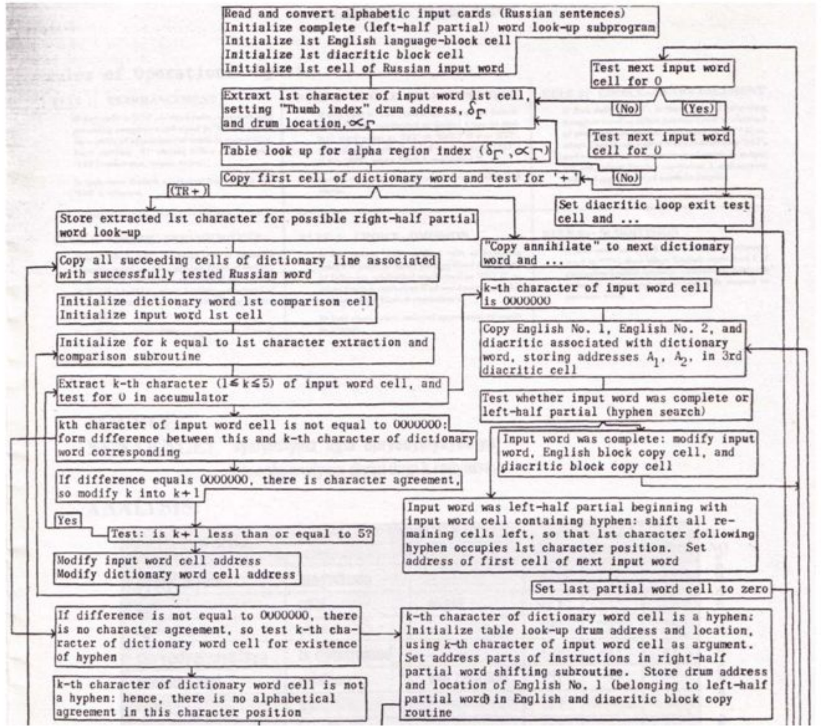
https://en.wikipedia.org/wiki/Dependency_grammar



https://www.techbeamers.com/python-for-loop/

How do we process language into text?

1950

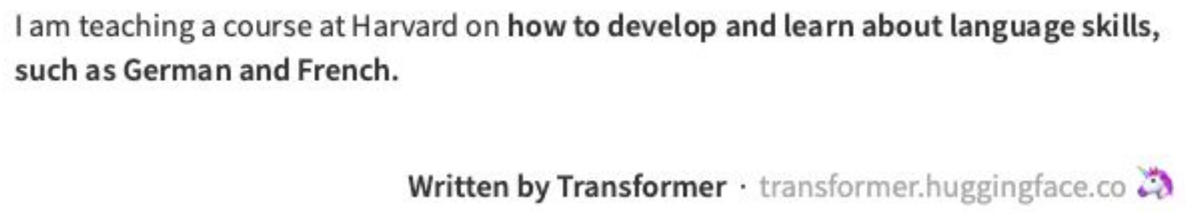

Read and convert alphabetic input cards (Russian sentences)
Initialize complete (left-half partial) word look-up subprogram
Initialize 1st English language-block cell
Initialize 1st diacritic block cell
Initialize 1st cell of Russian input word
Test next input word cell for 0
(No)
(Yes)
Extraxt 1st character of input word 1st cell, setting "Thumb index" drum address, δΓ and drum location, ∝Γ
Test next input word cell for 0
Table look up for alpha region index (δΓ, ∝Γ)
(No)
Copy first cell of dictionary word and test for '+'
(TR+)
Set diacritic loop exit test cell and ...
Store extracted 1st character for possible right-half partial word look-up
"Copy annihilate" to next dictionary word and ...
Copy all succeeding cells of dictionary line associated with successfully tested Russian word
k-th character of input word cell is 0000000
Initialize dictionary word 1st comparison cell
Initialize input word 1st cell
Copy English No. 1, English No. 2, and diacritic associated with dictionary word, storing addresses A1, A2, in 3rd diacritic cell
Initialize for k equal to 1st character extraction and comparison subroutine
Extract k-th character (1≤k≤5) of input word cell, and test for 0 in accumulator
Test whether input word was complete or left-half partial (hyphen search)
kth character of input word cell is not equal to 0000000: form difference between this and k-th character of dictionary word corresponding
Input word was complete: modify input word, English block copy cell, and diacritic block copy cell
If difference equals 0000000, there is character agreement, so modify k into k+1
Input word was left-half partial beginning with input word cell containing hyphen: shift all remaining cells left, so that 1st character following hyphen occupies 1st character position. Set address of first cell of next input word
Yes
Test: is k+1 less than or equal to 5?
Modify input word cell address
Modify dictionary word cell address
Set last partial word cell to zero
If difference is not equal to 0000000, there is no character agreement, so test k-th character of dictionary word cell for existence of hyphen
k-th character of dictionary word cell is a hyphen: Initialize table look-up drum address and location, using k-th character of input word cell as argument. Set address parts of instructions in right-half partial word shifting subroutine. Store drum address and location of English No. 1 (belonging to left-half partial word) in English and diacritic block copy routine
k-th character of dictionary word cell is not a hyphen: hence, there is no alphabetical agreement in this character position

2013

https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Now we can do things like this

Write with Transformer from HuggingFace:
https://transformer.huggingface.co/doc/distil-gpt2

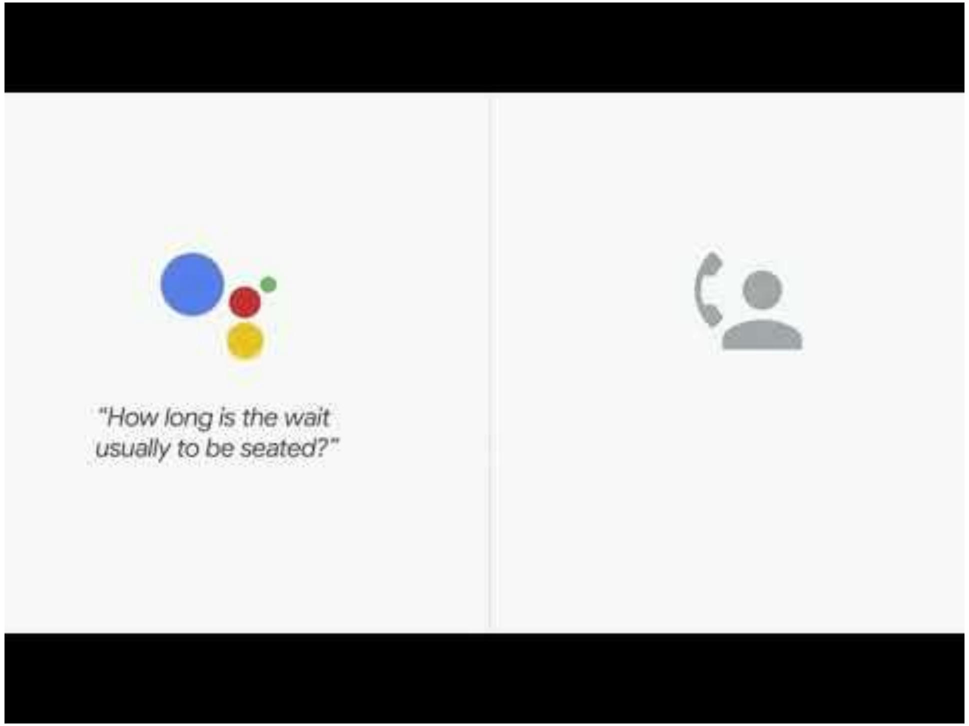


I am teaching a course at Harvard on **how to develop and learn about language skills, such as German and French.**

Written by Transformer · transformer.huggingface.co 🦄

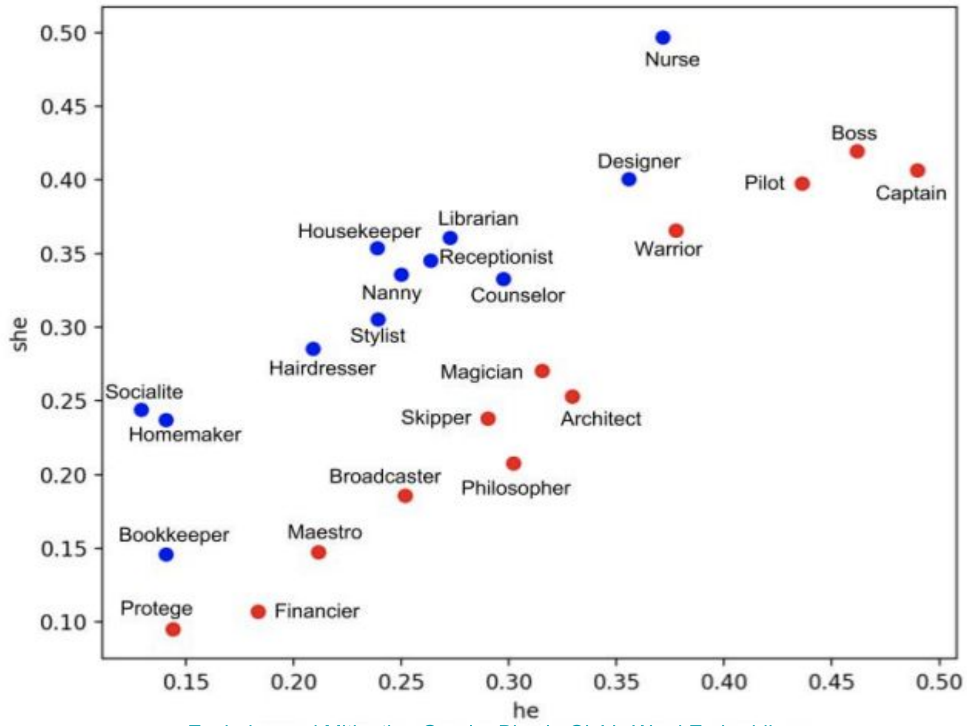
And this:

Google Assistant making a reservation



Though also, this:

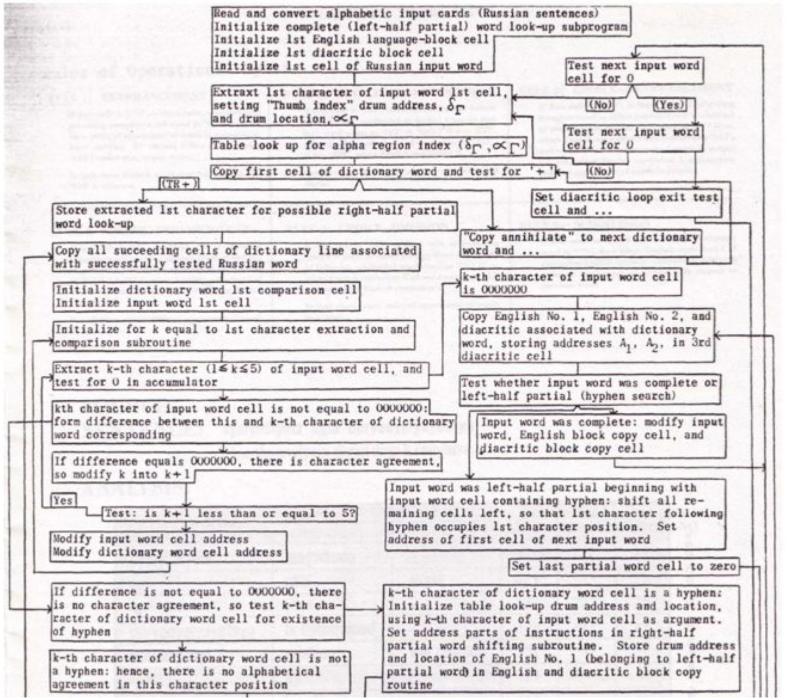
Word vector similarity between gender words and occupations



Exploring and Mitigating Gender Bias in GloVe Word Embeddings

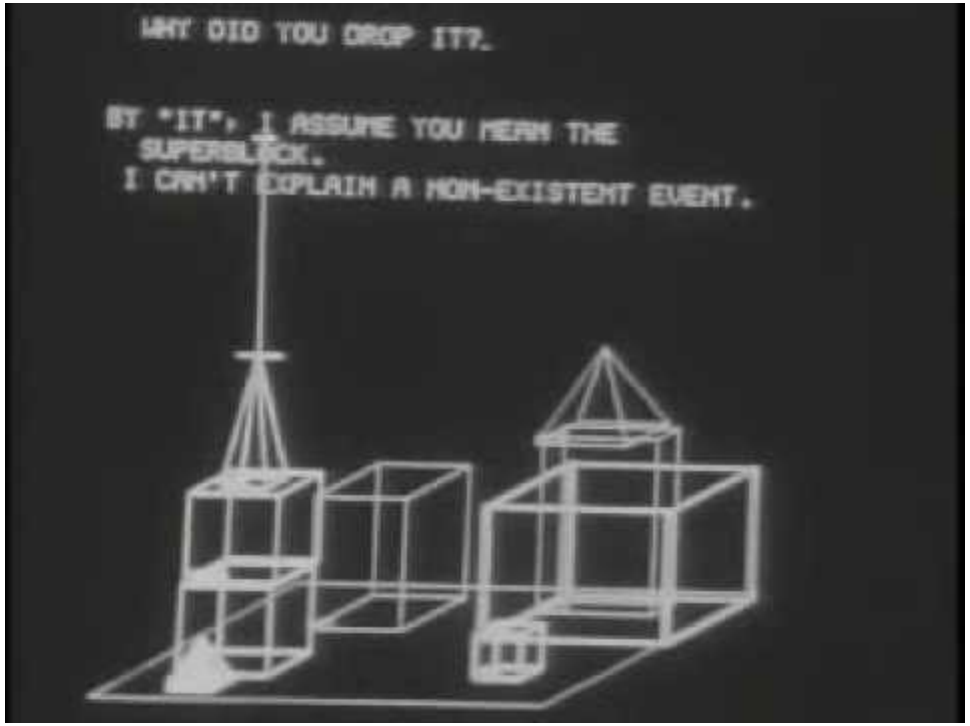
History of NLP: 1940s-1950s

- Focus on machine translation
  - Actually a VERY difficult task
- Approach mainly dictionary-based
- Mostly focus on syntax vs semantics
  - Syntax: Structure
  - Semantics: Meaning
  - “Cows flow supremely”, syntactically correct, semantically meaningless
  - Syntax-driven processing
    - Store information about relationships between different parts of sentence
    - Cows flow, the flow is supreme
  - Semantic-driven processing
    - Incorporate information outside of the language itself, like a cow is an animal
- No application-ready solutions by the end of this era
  - But started the conversation/thinking


Read and convert alphabetic input cards (Russian sentences) Initialize complete (left-half partial) word look-up subprogram Initialize 1st English language-block cell Initialize 1st diacritic block cell Initialize 1st cell of Russian input word
Test next input word cell for 0
(No)
(Yes)
Extraxt 1st character of input word 1st cell, setting "Thumb index" drum address, δ_Γ and drum location, $\propto_\Gamma$
Test next input word cell for 0
Table look up for alpha region index (δ_Γ, $\propto_\Gamma$)
(No)
Copy first cell of dictionary word and test for '+'
(TR+)
Set diacritic loop exit test cell and ...
Store extracted 1st character for possible right-half partial word look-up
"Copy annihilate" to next dictionary word and ...
Copy all succeeding cells of dictionary line associated with successfully tested Russian word
k-th character of input word cell is 0000000
Initialize dictionary word 1st comparison cell Initialize input word 1st cell
Copy English No. 1, English No. 2, and diacritic associated with dictionary word, storing addresses A_1, A_2, in 3rd diacritic cell
Initialize for k equal to 1st character extraction and comparison subroutine
Extract k-th character ($1 \leq k \leq 5$) of input word cell, and test for 0 in accumulator
Test whether input word was complete or left-half partial (hyphen search)
kth character of input word cell is not equal to 0000000: form difference between this and k-th character of dictionary word corresponding
Input word was complete: modify input word, English block copy cell, and diacritic block copy cell
If difference equals 0000000, there is character agreement, so modify k into k+1
Input word was left-half partial beginning with input word cell containing hyphen: shift all remaining cells left, so that 1st character following hyphen occupies 1st character position. Set address of first cell of next input word
Yes
Test: is k+1 less than or equal to 5?
Modify input word cell address Modify dictionary word cell address
Set last partial word cell to zero
If difference is not equal to 0000000, there is no character agreement, so test k-th character of dictionary word cell for existence of hyphen
k-th character of dictionary word cell is a hyphen: Initialize table look-up drum address and location, using k-th character of input word cell as argument. Set address parts of instructions in right-half partial word shifting subroutine. Store drum address and location of English No. 1 (belonging to left-half partial word) in English and diacritic block copy routine
k-th character of dictionary word cell is not a hyphen: hence, there is no alphabetical agreement in this character position

1960s-1970s: Shift to semantic-driven processing

- Semantic-driven processing
  - Focus on meaning/relationships
  - Incorporate information outside of the language itself
- SHRLDU
  - Map query to stored relationship map of the machine state
- Focus on pulling in different data sources
  - Lots of focus on tools for data integration
- Still not enough
  - Uninteresting results on actual dealing with natural language
  - User-machine exchanges difficult to handle “naturally”

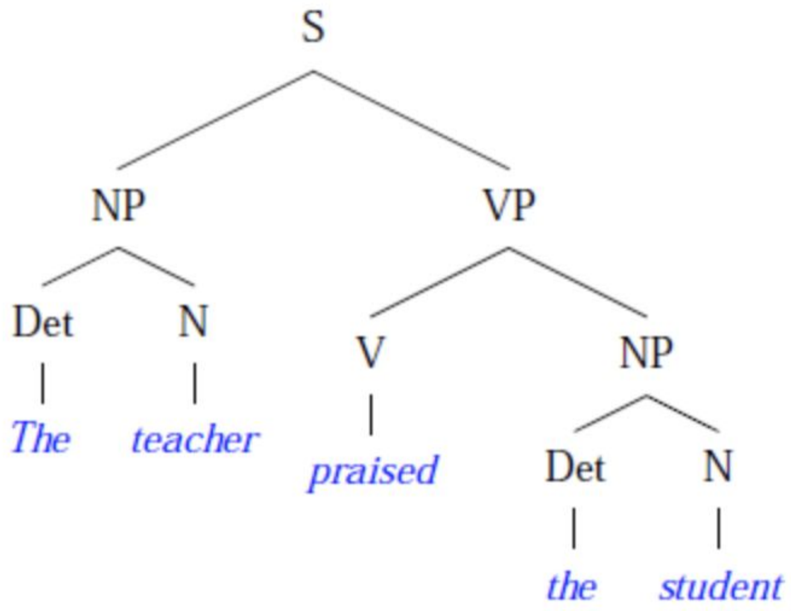


WHY DID YOU DROP IT?.

BY "IT", I ASSUME YOU MEAN THE SUPERBLOCK.
I CAN'T EXPLAIN A NON-EXISTENT EVENT.

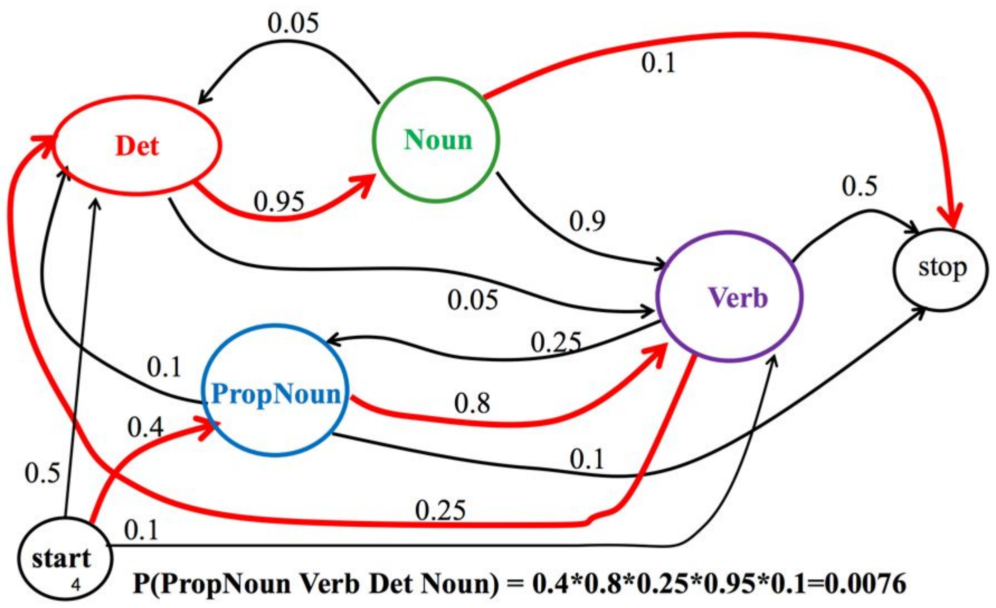
1970s-1980s: Joining of linguistics and AI

- Linguistics (Study of language)
  - Noam Chomsky's ideas of "generative grammar"
  - Focus on categorization/computability
- AI
  - Increasing focus on representations of meanings/intention
- Joining together, building of community
  - New conferences
  - New standardized tasks to measure performance



1990s-2000s: Shift towards statistical NLP

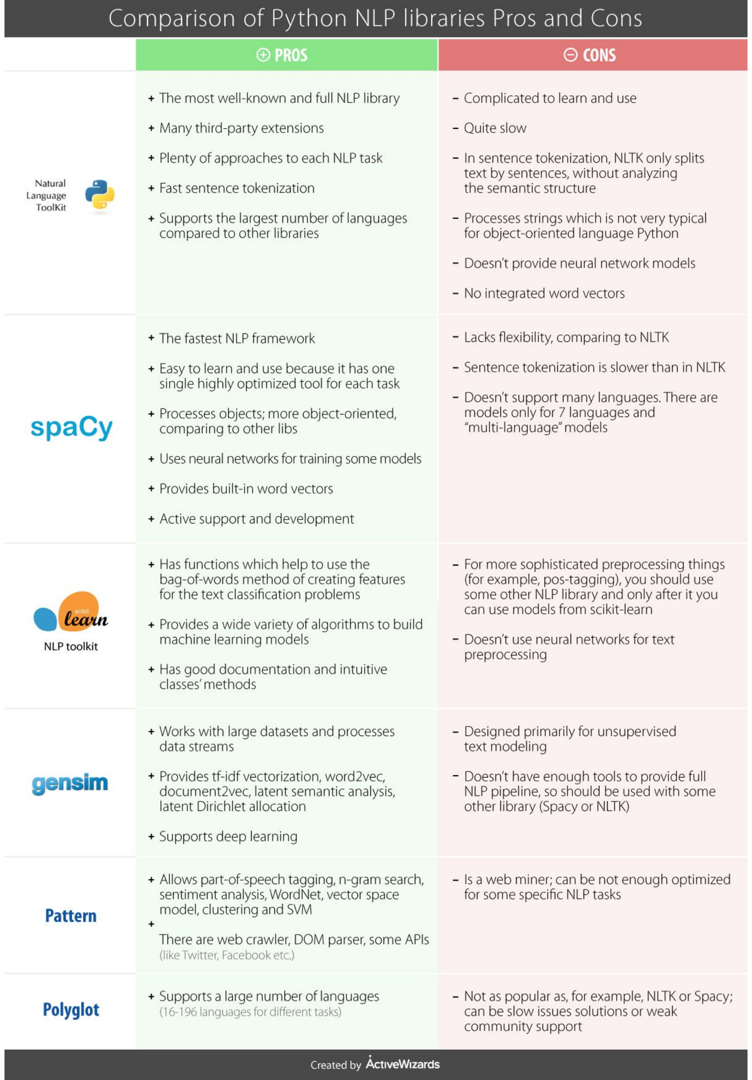
- Increasing attention on probabilistic models of language
  - Visual of parsing trees with probabilities, Markov Models
  - Probabilities of certain words appearing, the precursor to modern language models
- Also newly available data sources for developing/testing
  - WordNet (https://wordnet.princeton.edu/)
    - Inventory of English words and their relationships
  - Penn TreeBank (https://catalog.ldc.upenn.edu/LDC99T42)
    - Text parsed for part of speech, dependencies, etc
  - Ontonotes (https://catalog.ldc.upenn.edu/LDC2013T19)
    - Text labelled with entities (e.g. places and persons)
  - The internet!



To the notebooks!

https://github.com/bpben/nlp_lessons/blob/master/notebooks_instructor/week_1_intro.ipynb

NLP libraries



Comparison of Python NLP libraries Pros and Cons

	⊕ PROS	⊖ CONS
Natural Language ToolKit	+ The most well-known and full NLP library + Many third-party extensions + Plenty of approaches to each NLP task + Fast sentence tokenization + Supports the largest number of languages compared to other libraries	- Complicated to learn and use - Quite slow - In sentence tokenization, NLTK only splits text by sentences, without analyzing the semantic structure - Processes strings which is not very typical for object-oriented language Python - Doesn't provide neural network models - No integrated word vectors
spaCy	+ The fastest NLP framework + Easy to learn and use because it has one single highly optimized tool for each task + Processes objects; more object-oriented, comparing to other libs + Uses neural networks for training some models + Provides built-in word vectors + Active support and development	- Lacks flexibility, comparing to NLTK - Sentence tokenization is slower than in NLTK - Doesn't support many languages. There are models only for 7 languages and "multi-language" models
scikit learn NLP toolkit	+ Has functions which help to use the bag-of-words method of creating features for the text classification problems + Provides a wide variety of algorithms to build machine learning models + Has good documentation and intuitive classes' methods	- For more sophisticated preprocessing things (for example, pos-tagging), you should use some other NLP library and only after it you can use models from scikit-learn - Doesn't use neural networks for text preprocessing
gensim	+ Works with large datasets and processes data streams + Provides tf-idf vectorization, word2vec, document2vec, latent semantic analysis, latent Dirichlet allocation + Supports deep learning	- Designed primarily for unsupervised text modeling - Doesn't have enough tools to provide full NLP pipeline, so should be used with some other library (Spacy or NLTK)
Pattern	+ Allows part-of-speech tagging, n-gram search, sentiment analysis, WordNet, vector space model, clustering and SVM + There are web crawler, DOM parser, some APIs (like Twitter, Facebook etc.)	- Is a web miner; can be not enough optimized for some specific NLP tasks
Polyglot	+ Supports a large number of languages (16-196 languages for different tasks)	- Not as popular as, for example, NLTK or Spacy; can be slow issues solutions or weak community support

Created by ActiveWizards

[Comparison of Top 6 Python NLP Libraries - ActiveWizards — your AI partner]

Note on Stanza (formerly StanfordNLP)

- https://github.com/stanfordnlp/stanza/
- Similar “pipeline” structure to spaCy
  - stanza.download('en')
  - nlp = stanza.Pipeline('en')
- Large amount of languages models (POS/NER)
  - https://stanfordnlp.github.io/stanza/available_models.html
- Spacy-Stanza
  - https://spacy.io/universe/project/spacy-stanza
  - Most of the same attributes available
- If looking for a language that’s not available consider Stanza or http://ufal.mff.cuni.cz/udpipe

Tokenization

- Definition: “A useful semantic unit”
- What is a “useful semantic unit”?

“I am learning Natural Language Processing (NLP)”

What are the useful units of meaning here?

I, am, learning (split on whitespace)

Natural Language Processing (entity)

NLP (entity as acronym)

(NLP) (special treatment of text in parentheses?

N-grams

- N-gram: Continuous sequence of N tokens
  - Tokens != words (see previous slides)
- Important considerations
  - “not”, “good” vs “not good”
  - Named-entities (e.g. Cities, people)
  - Exploring co-occurrence

count(natural language) / count (language)

count(english language) / count (language)

“I am learning Natural Language Processing (NLP)”

<split on whitespace>

Unigrams

I, am, learning, Natural, Language, Processing, (NLP)

Bigrams

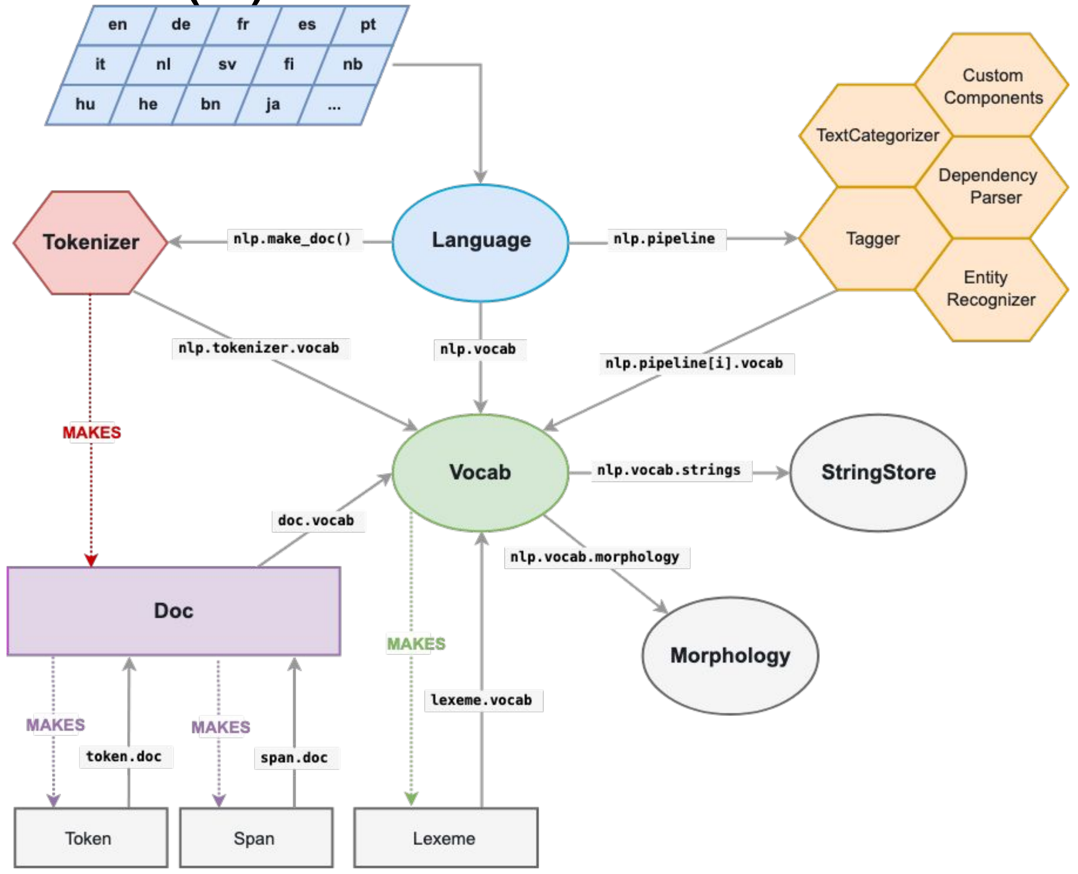
I am, am learning, learning Natural...

7-grams

I am learning Natural Language Processing (NLP)

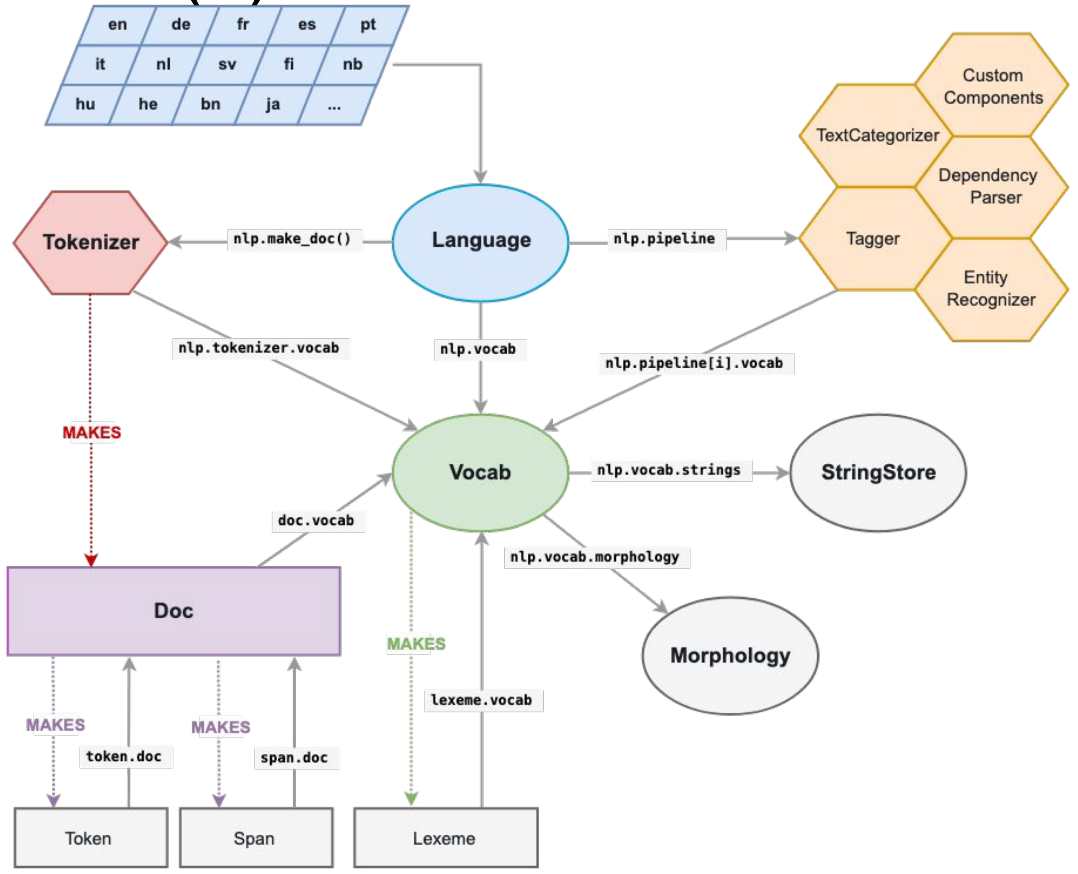
SpaCy’s Language models (1)

- Language model: Tokenization/processing pipeline
- Base: Tokenizer + Vocab
  - Tokenizer
    - Processing of text
  - Vocab
    - Contains all “lexemes”
- Document -> Span -> Token
  - .text = raw text
  - Document/Span
  - Token
    - .shape_
      - We = Xx
      - we = xx
    - .is_alpha (boolean)
    - .is_stop (boolean)



SpaCy's Language models (2)

- Can load "trained" language models
- Additional pipeline components
    - Entity recognition
    - Part of speech tagging
    - Dependency parsing
    - Categorization
    - Anything else (custom attributes)
- Products of these
    - Document/Span
        - .ents (named entities)
        - .sents
        - .noun_chunks
    - Token
        - .dep_ (e.g. nsubj)
        - .lemma_ (dictionary form)
        - .pos_ (part of speech)



What is the point of NLP?

What is the main challenge of working with text data?

How is it different from structured data (e.g. stock ticker, height/weight)?

What are we looking for when we’re trying to find these “useful semantic units”?

Main goal: Create an informative representation of text

Stemming vs Lemmatization

Goal: Reduce related words to a common “base form”

Stemming

Set of heuristics to transform suffixes. Most of the word stays intact.

“He does natural language processing”

↓

“He doe natural language process”

Implemented in: NLTK, PyStemmer

Lemmatization

Transforms word to their “lemma”, or dictionary form. This may change the word entirely.

“He is doing natural language processing”

↓

“He be do natural language process”

Implemented in: NLTK, spaCy

Stop words

- Extremely common words that have little information
  - “the”, “at”, “from”, “to”, “in”
- No single list of these
  - Consequences to choosing overly strict/broad

In June 2020, I took a course at Harvard Extension School in Cambridge.

<Remove stop words (SpaCy)>

June 2020 took course Harvard Extension School Cambridge

Issues with stop words

Scikit-learn

Example: “hasn’t” is removed using “hasnt”

https://github.com/scikit-learn/scikit-learn/blob/fd237278e895b42abe8d8d09105cbb82dc2cbba7/sklearn/feature_extraction/_stop_words.py

SpaCy

Example: “hasn’t” is removed using “has” and contraction “n’t”

https://github.com/explosion/spaCy/blob/master/spacy/lang/en/stop_words.py

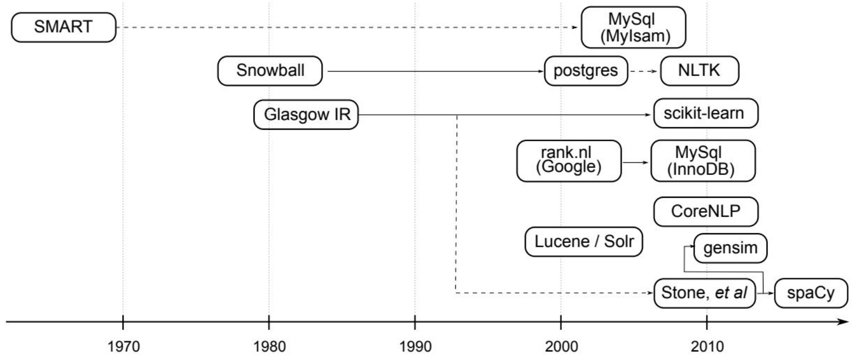


Figure 1: Family tree of popular stop word lists.

https://www.aclweb.org/anthology/W18-2502.pdf

Named-Entities

- Named-entity: A real-world named object (e.g. person, place, organization)
  - New York City is different than just an assembly of three words "new", "york" and "city"
- To identify these
  - Dictionaries
  - Pattern-matching
  - Models

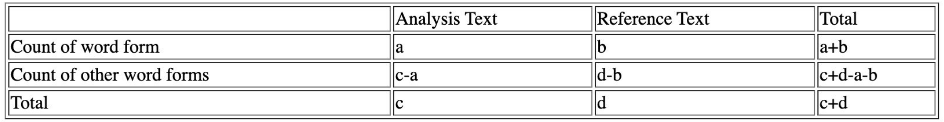
In June 2020 [DATE], I took a course at Harvard Extension School [ORG] in Cambridge [LOC].

Exercise: Create a tokenization pipeline

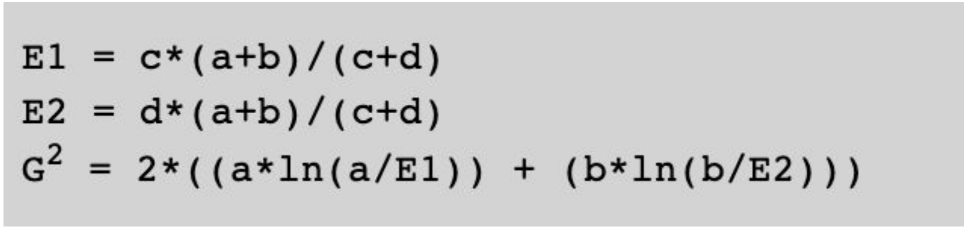
https://github.com/bpben/nlp_lessons/blob/master/notebooks_instructor/week_1_intro.ipynb

Log-likelihood ratio with word counts

- Enables comparison of word-level counts between two documents/corpora
- Test of hypothesis: Frequency of word is equivalent between analysis and reference
  - Expected count = number of words in document i * the pooled frequency across both documents
- Provides a test of significance for differences in frequency
- Accessible via simple word counts!



	Analysis Text	Reference Text	Total
Count of word form	a	b	a+b
Count of other word forms	c-a	d-b	c+d-a-b
Total	c	d	c+d

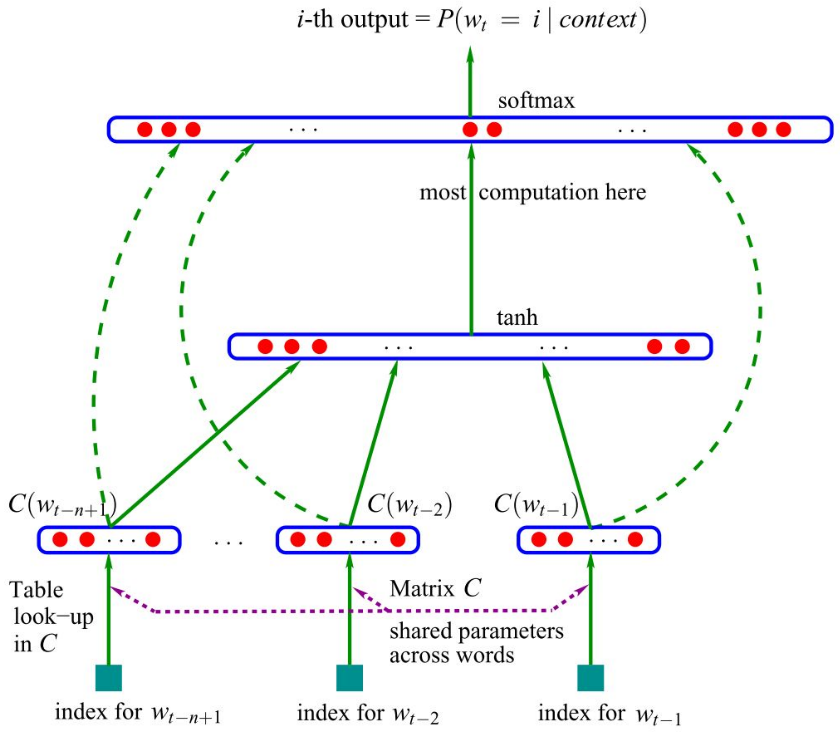


$$E1 = c*(a+b)/(c+d)$$
$$E2 = d*(a+b)/(c+d)$$
$$G^2 = 2*((a*\ln(a/E1)) + (b*\ln(b/E2)))$$

https://wordhoard.northwestern.edu/userman/analysis-comparewords.html

2000s: Advent of Neural Models for NLP

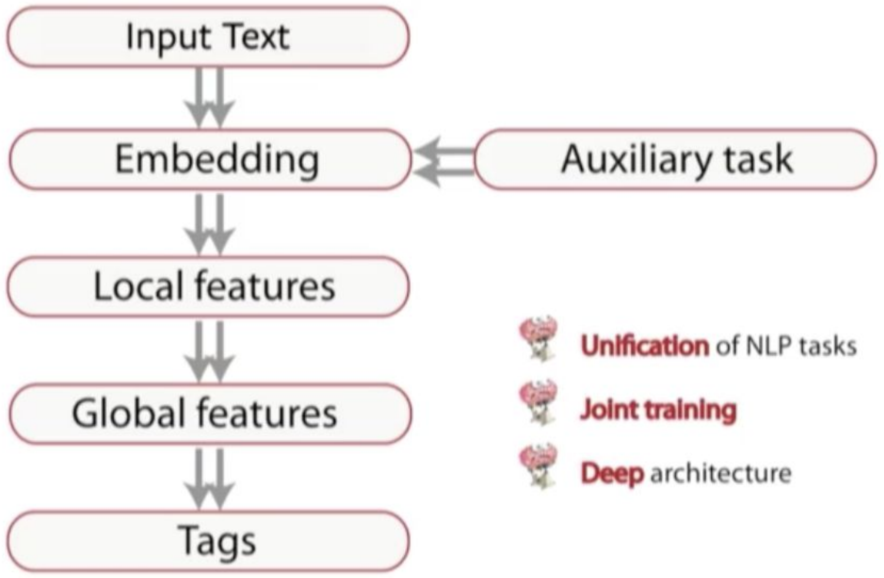
- Focused on language modelling task
    - Given context words, predict next word
    - “Do you want to go out for <mask>”
    - “I ate so much I am so <mask>”
    - “I’m so tired, I think I’ll take a <mask>”
- Apply Neural Net architectures to language modelling task
    - Neural Net: Model that connects inputs to outputs through sets of computations
- Bengio et. al. (2001) A Neural Probabilistic Language Model
    - Context words fed into a matrix that “represents” the information
    - These representations then fed into a computation layer
    - Output: Prediction of the target word
        - “Full”: 70% likely, “tired” 20% likely, etc



A Neural Probabilistic Language Model
(https://papers.nips.cc/paper/1839-a-neural-probabilistic-language-model.pdf)

2008: Multi-task learning

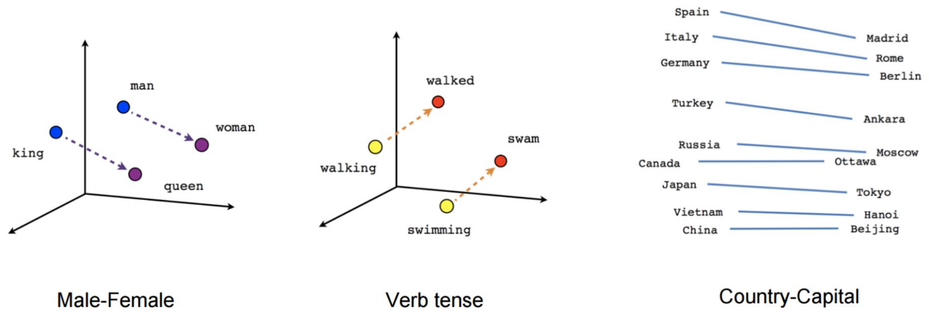
- Multi-task learning: One model, multiple related tasks
  - Trains representations (e.g. lookup tables) jointly so that it performs well on both
- Collobert and Weston (2008)
  - Word lookup table trained jointly from two different tasks
  - Basically the precursor to the idea of word embeddings



https://www.facebook.com/722677142/posts/10155393881507143/

2013: Word embeddings

- Mikolov paper
  - Efficient computation of embedding technique used in the 2000s
  - Word2Vec implementation
- Huge amount of adaptation/modifications
  - GloVe: Co-occurance based
  - Sub-word/character-level
  - Use of different corpora for training
  - Out-of-vocabulary handling
  - Training implementation in open-source libraries (e.g. gensim)
- Major revolution: It's fairly easy to make more informed representations of words
- Caveat: Widespread use raises issues of bias
  - Extensive research finds training on most large datasets introduces gender/racial bias
  - Not an issue with the method particularly, more with irresponsible use



[1301.3781] Efficient Estimation of Word Representations in Vector Space