Deep Neural Networks for Natural Language Processing (AI6127)

JUNG-JAE KIM

INTRODUCTION

Contents

- Course logistics
- What is Natural Language Processing (NLP)?
- Why deep learning for NLP?
- Course outline
- TF-IDF

Instructors

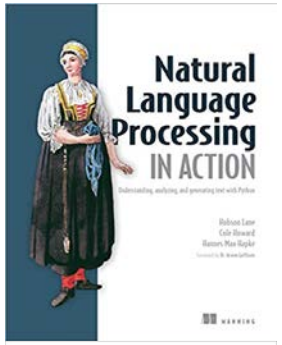
- Jung-jae Kim
  - Leader of Explainable & Automatic AI Group, Semantic Text Analytics lab
  - Institute for Infocomm Research, A-STAR
  - jjkim@i2r.a-star.edu.sg
- Shafiq Joty
  - Assistant Professor @ SCSE
  - Senior Research Manager @ Salesforce AI Research
  - srjoty@ntu.edu.sg

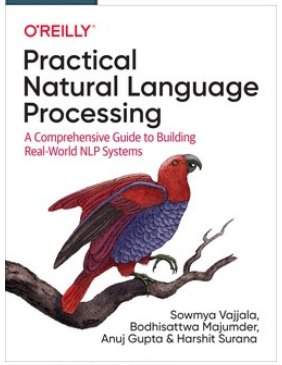
Optional textbook/resource

- Course “Natural Language Processing with Deep Learning” by Chris Manning, Stanford University
  - http://web.stanford.edu/class/cs224n/ (slides, video, notes, etc)
    - Acknowledge using many slides from this course
  - https://www.youtube.com/playlist?list=PLoROMvodv4rOhcuXMZkNm7j3fVwBBY42z
    - Complete videos from the 2019 edition
- Textbook “Speech and Language Processing” by D. Jurafsky and J.H. Martin (3rd edition, draft)
  - https://web.stanford.edu/~jurafsky/slp3/

NLP textbooks with github sites

- Natural Language Processing in Action: Understanding, analyzing, and generating text with Python
  - Written by Hobson Lane, Hannes Hapke, Cole Howard
  - https://github.com/totalgood/nlpia
- Practical Natural Language Processing
  - Written by Sowmya Vajjala, Bodhisattwa Majumder, Anuj Gupta, Harshit Surana
  - https://github.com/practical-nlp/practical-nlp
- NLP course by HuggingFace
  - https://huggingface.co/course/chapter1/1





Prerequisites

- Fundamentals of machine learning and deep learning
- Basics of linear algebra and probability
  - Linear algebra with Python:
    - https://github.com/MacroAnalyst/Linear_Algebra_With_Python
- Proficiency in Python and PyTorch
  - All course assignments/project will be in Python and PyTorch
  - Supplementary materials (IPython notebooks)
    - Python: https://colab.research.google.com/drive/1bQG32CFoMZ-jBk02uaFon60tER3yFx4c
    - Numpy: https://drive.google.com/file/d/1cUzRzQGURrCKes8XynvTTA4Zvl_gUJdc/view
    - PyTorch: https://drive.google.com/file/d/18cgPOj2QKQN0WR9_vXoz6BoravvS9mTm/view, https://colab.research.google.com/drive/1c33y8bkdr7SJ_I8-wmqTAhld-y7KcspA

Course objectives

- Understanding of deep learning methods for NLP
  - NLP task/application definition
  - NLP methods before deep learning
  - From representation learning (word vector) to deep learning methods (feed-forward network, convolutional neural network (CNN), recurrent neural network (RNN), attention)
  - Even latest developments (transformer, BERT)
- Capability to build deep learning systems for NLP applications
  - Data pre-processing, training, evaluation

Lecture/Tutorial

- Lecture
  - Including hands-on slides w/o actual hands-on during lecture, for practice after lecture
- Tutorial
  - Mostly, ‘programming’ questions
  - Relatively simple programming compared to assignments

Grading policy

- 3 Assignments: 3 x 15% = 45%
  - Deadlines: 5th, 8th, 11th weeks (To be announced 2 weeks before the deadlines)
- Project (1-3 people): 55%
  - Proposal: 5% (deadline: 6th week)
  - Update: 10% (deadline: 9th week)
  - Presentation: 10% (deadline: 13th week)
  - Report: 30% (deadline: 14th week; no late submission is allowed)
- Late submission policy
  - 10% off per day late
  - Not accepted after 5 late days

Course project

- An end-to-end system
  - E.g. Machine reading comprehension (MRC; Q&A), Machine translation (MT), Dialogue, Summarization, Named entity recognition (NER).
- A new component
  - E.g. New attentions, new layer, new architecture
- Project evaluation criteria will be explained in Week 4
- Starting thinking about project
  - Find your partner(s)

Collaboration policy

- We should follow NTU's Student Code of Conduct
- Rule 1: For assignments, everything you submit should be your own work. Don't share your solution code with others; however discussing ideas or general strategies is fine and encouraged.
- Rule 2: For final project, you can work in a group. But, make sure you acknowledge the source if you are using other's code/material.

Contents

- Course logistics
- What is NLP?
- Why deep learning for NLP?
- Course outline
- TF-IDF

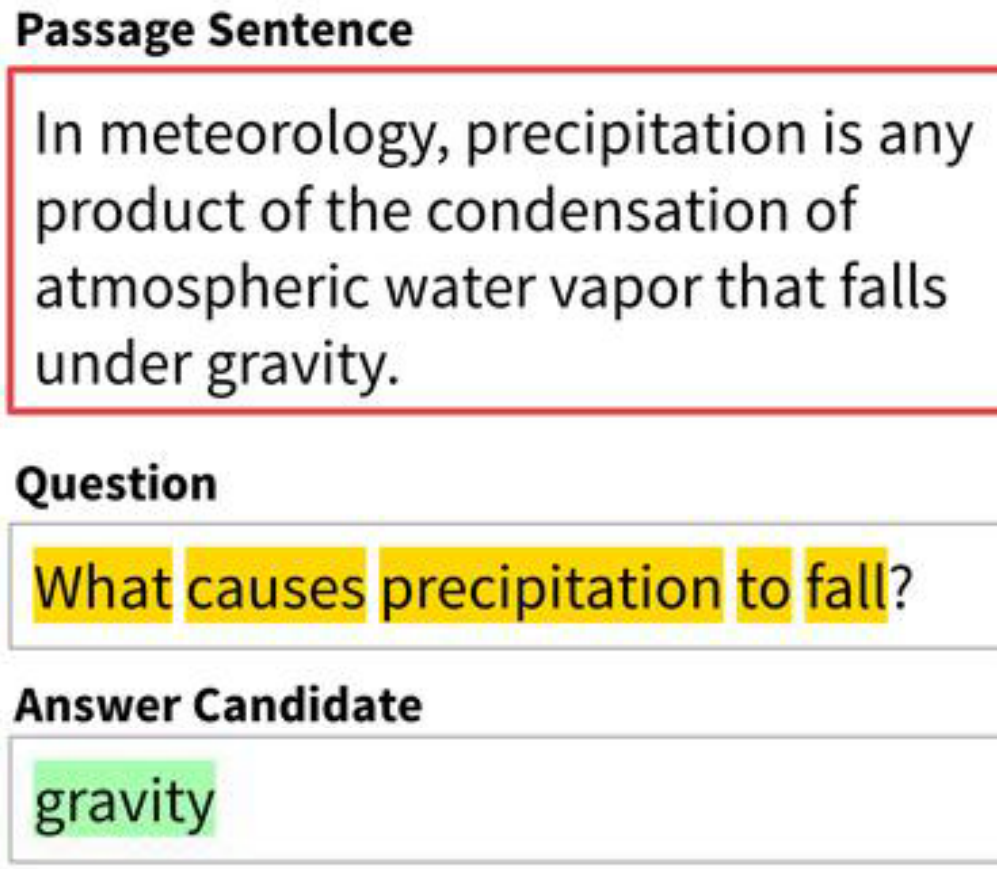
Brief introduction to NLP

- NLP is NOT
  - Linguistics, but computational methods for processing natural languages
  - Processing audio data of natural languages (automatic speech recognition), but text data
- Goals can be very far reaching…
  - True text understanding
  - Real-time participation in spoken dialogs about any topics
- Or very down-to-earth…
  - Classifying a text into one of pre-defined categories
  - Extracting names (e.g. person, location, organization) from documents
  - Task-oriented dialogue system (e.g. flight booking)

Example NLP applications: Machine translation

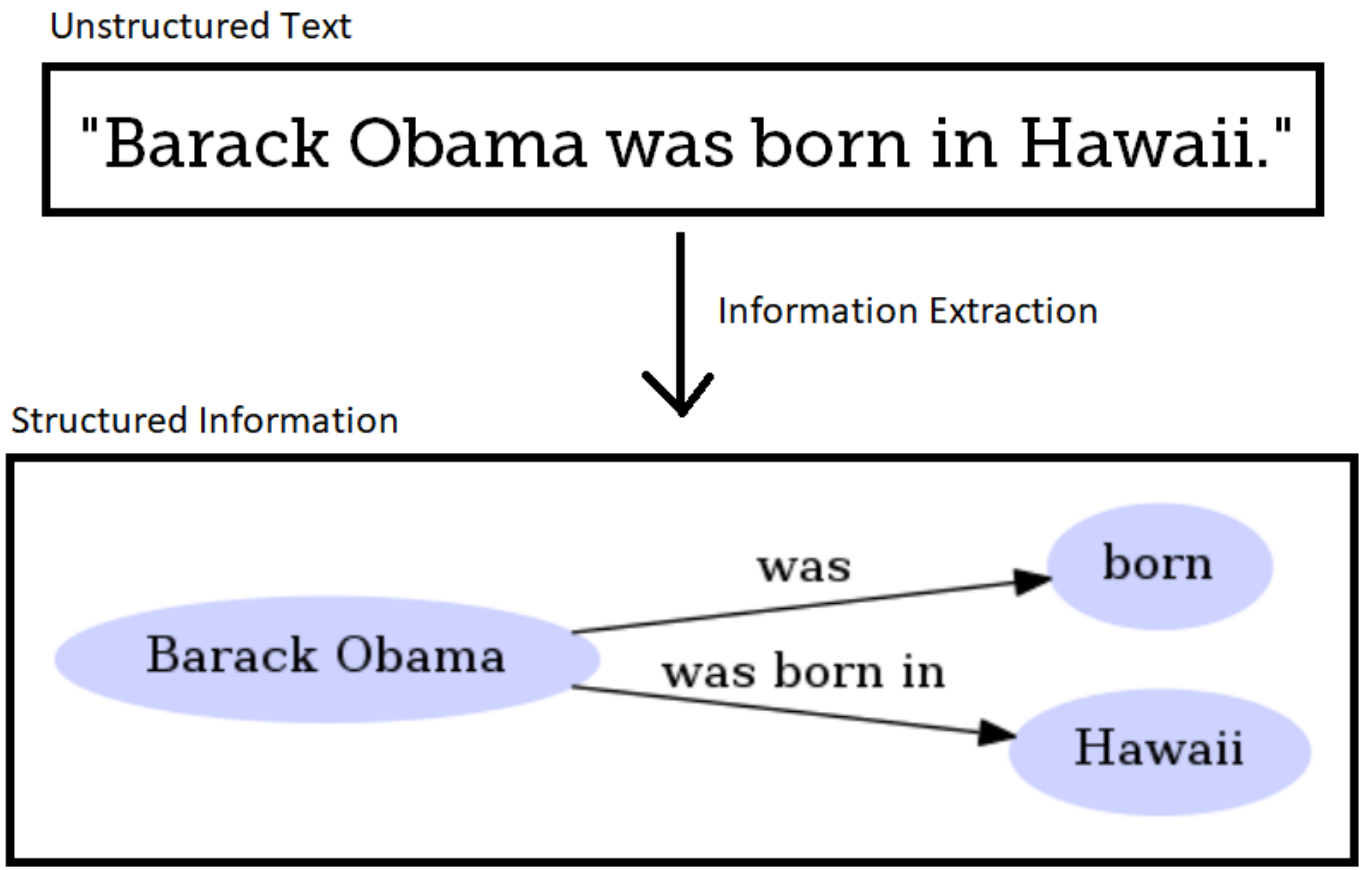
https://www.theverge.com/2018/12/6/18129203/google-translate-gender-specific-translations-languages

Example NLP applications: Question answering (machine reading comprehension)



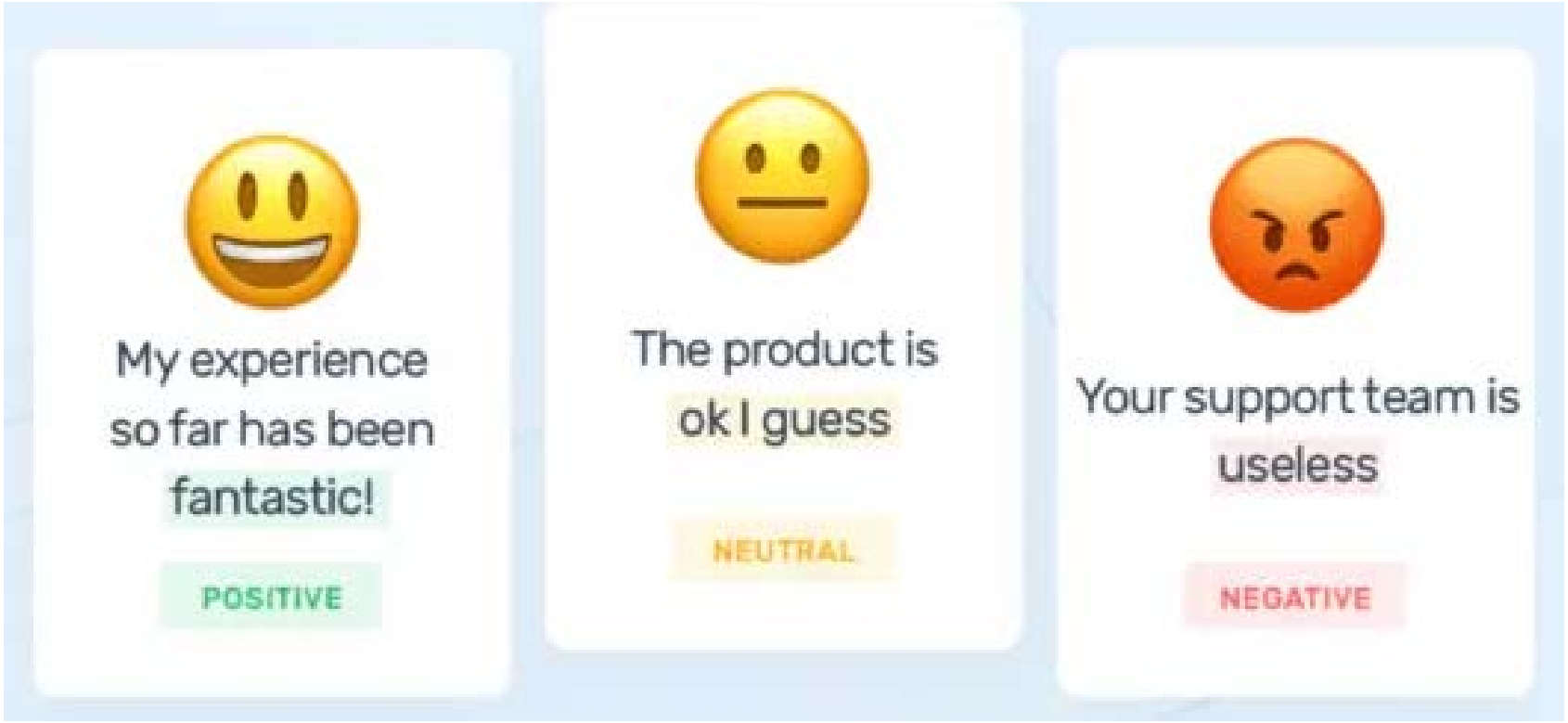
https://paperswithcode.com/task/question-answering

Example NLP applications: Information extraction



https://medium.com/analytics-vidhya/entity-linking-a-primary-nlp-task-for-information-extraction-22f9d4b90aa8

Example NLP applications: Sentiment analysis



https://monkeylearn.com/sentiment-analysis/

Example NLP applications: Others

- Conversational agents: e.g. Apple SIRI, Amazon's alexa, Microsoft's Cortana
- Summarization: "Please summarize my discussion with Sue about NLP", "What people say about the new Nikon 5000?"
- Text Generation: Data2Text, Table2Text, Video/Image2Text
- Document Classification: e.g. spam detection, news filtering
- Multimodal: Image-to-text (image/video captioning, visual Q&A)

Source: NTUNLP

Language-related topics

- Phonetics and phonology (sounds)
- Morphology (structure of words)
- Syntax (structure of sentences)
- Semantics (meaning)
- Pragmatics (language usage)
- Discourse and dialogue (multiple utterances/sentences)

Source: NTUNLP

Traditional approach to language processing

- Language as symbol
  - Each word is considered as a discrete symbol
  - Relationship between words are represented manually (e.g. WordNet) or learnt from a large corpus (collection of texts)
  - Syntactic and other linguistic relationship are represented/learnt separately
- Issues
  - Complexity in representing, learning and using linguistic/situational/contextual/world/visual knowledge
    - But interpretation depends on these
  - Difficult to optimize from upstream tasks to downstream applications

Source: CS224n

Why is NLP hard?

- Human languages are ambiguous, e.g.
  - I made her duck
  - Girl hit by car in hospital
  - Singapore court to try shooting defendant
  - Farmer Bill dies in house

Source: CS224n

Contents

- Course logistics
- What is NLP?
- Why deep learning for NLP?
- Course outline
- TF-IDF

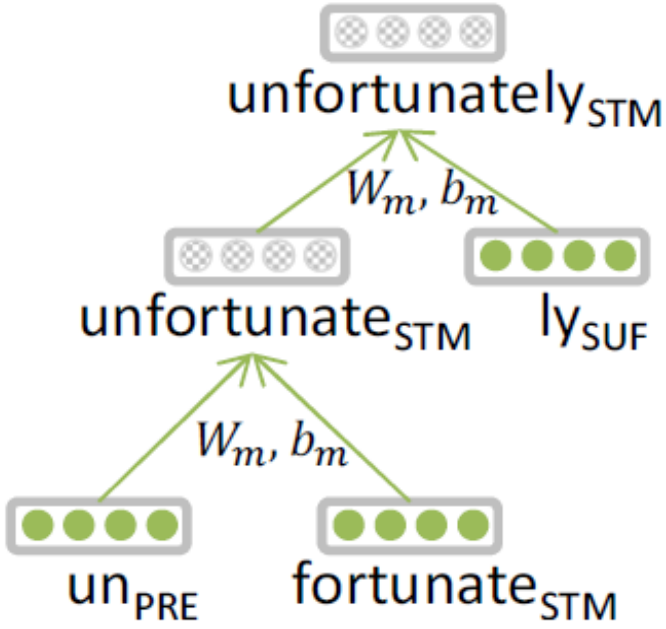
Deep learning (DL) for NLP

- Combine NLP knowledge with representation learning and deep learning
- Improvements in recent years in all levels
  - Linguistic levels: speech, words, syntax, semantics, discourse
  - Full applications: sentiment analysis, question answering, dialogue agents, machine translation

Source: CS224n

Deep learning for NLP: Morphology

- Words are made of morphemes
  - E.g. un interest ed
    prefix stem suffix
- DL
  - Every morpheme is a vector
  - A neural network combines two vectors into one vector
    - (Luong et al., 2013)
  - Application: Word similarity

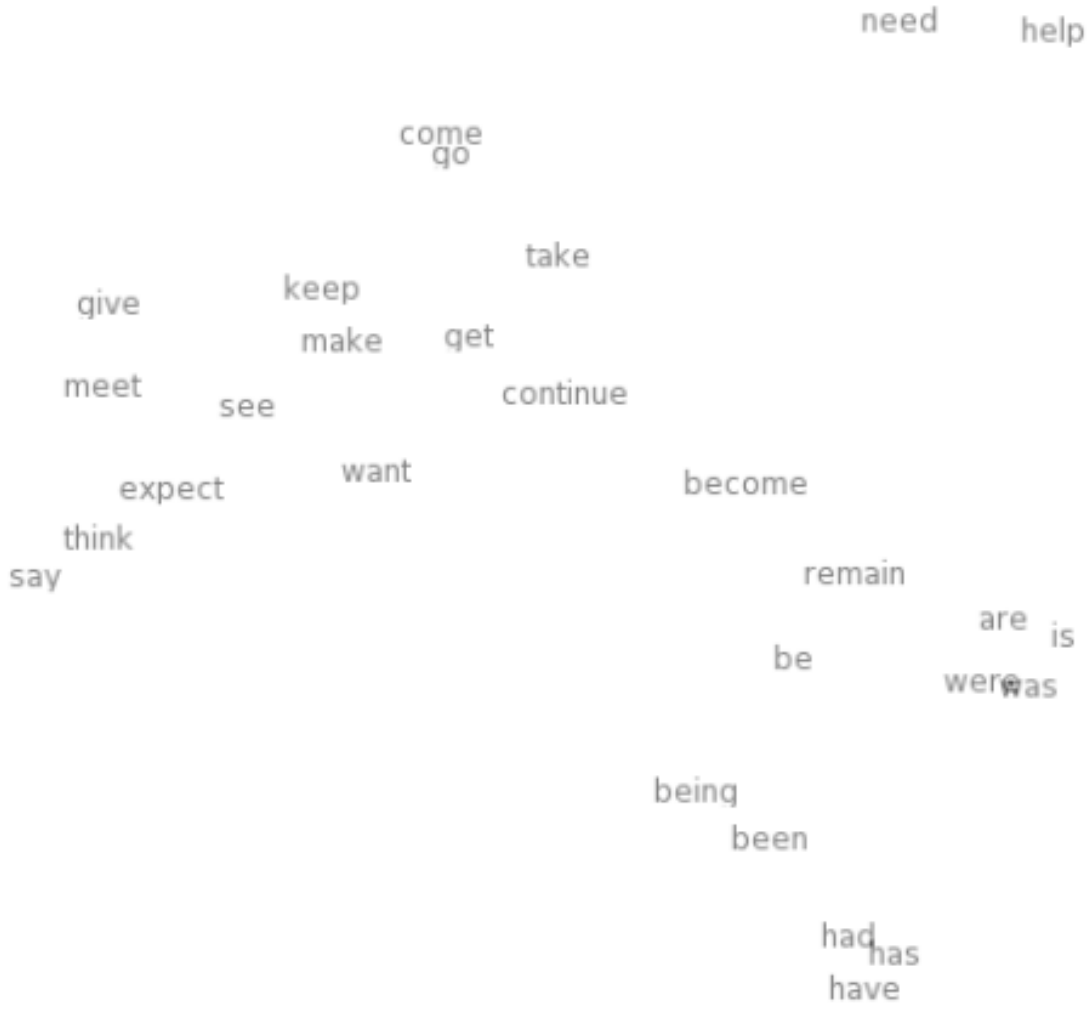


https://nlp.stanford.edu/~lmthang/data/papers/conll13_morpho.pdf

Source: CS224n

Deep learning for NLP: Word meaning

$$expect = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \\ 0.487 \end{pmatrix}$$



Source: CS224n

Deep learning for NLP: Word similarities

- Nearest words to ‘frog’
  1. frogs
  2. toad
  3. litoria
  4. leptodactylidae
  5. rana
  6. lizard
  7. eleutherodactylus

litoria

leptodactylidae

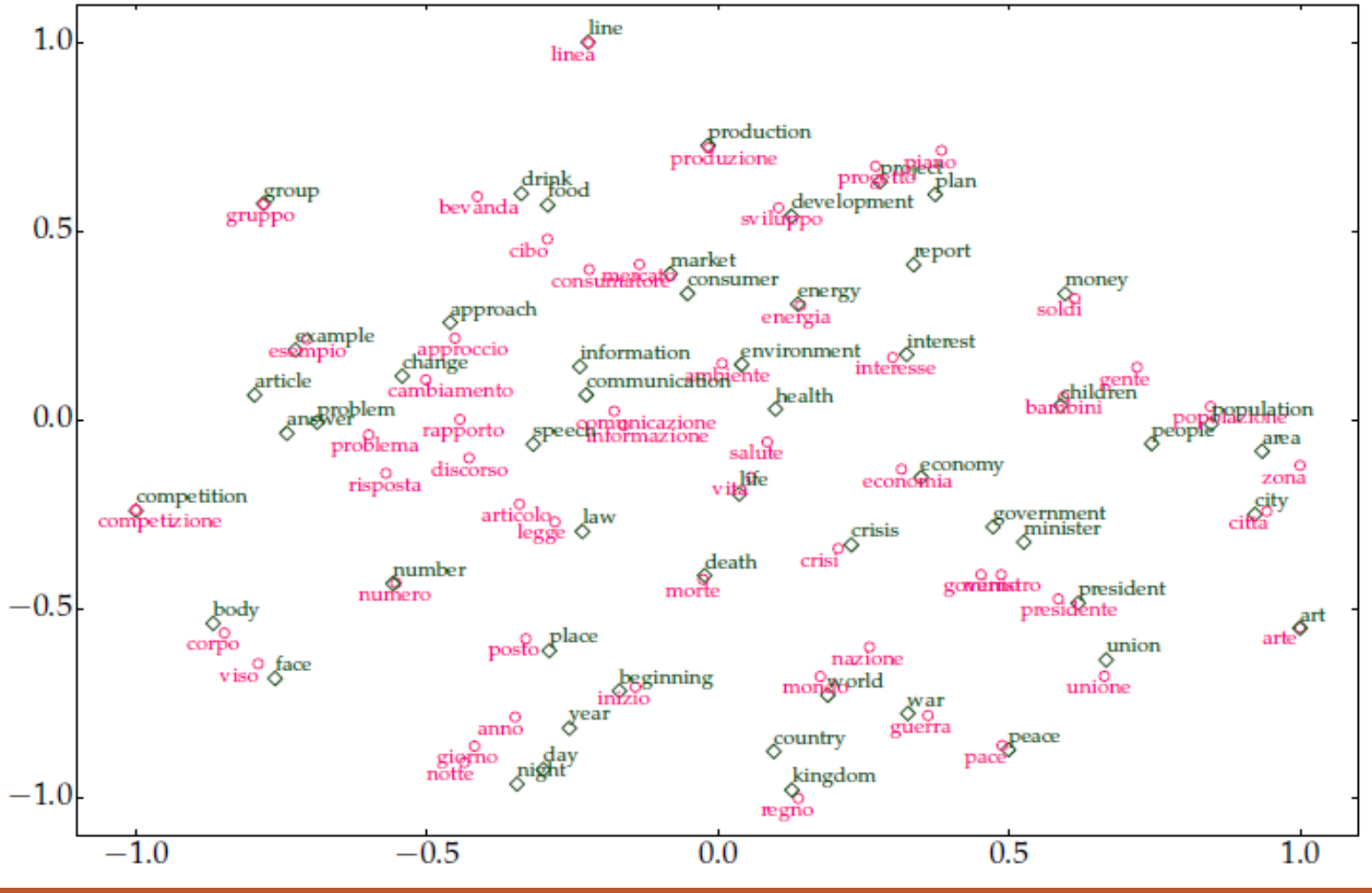
rana

eleutherodactylus

https://nlp.stanford.edu/projects/glove/

Source: CS224n

Deep learning for NLP: Word meaning (cross-lingual)

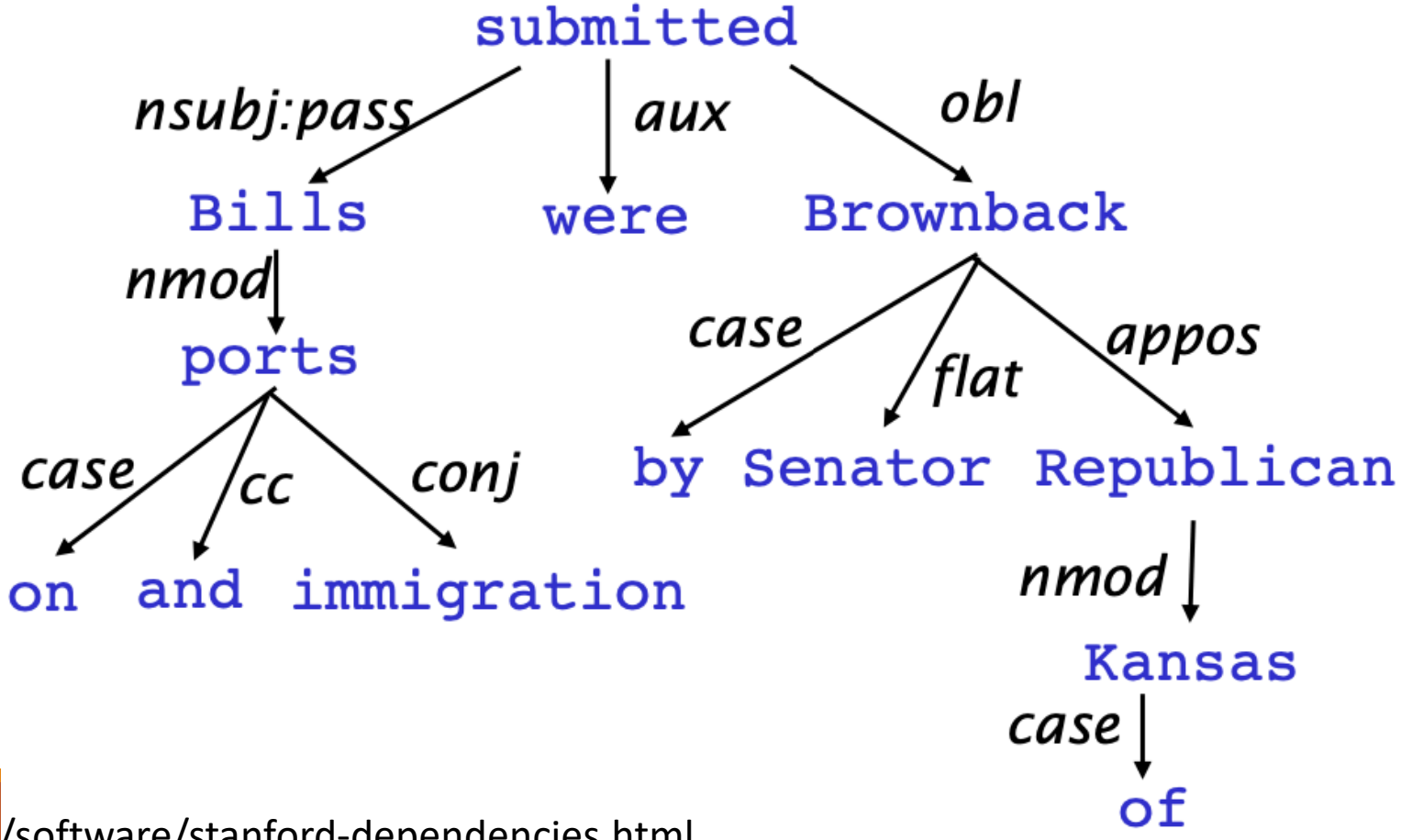


(Ruder et al, 2019)

https://arxiv.org/pdf/1706.04902.pdf

Deep learning for NLP: Syntax (parsing)

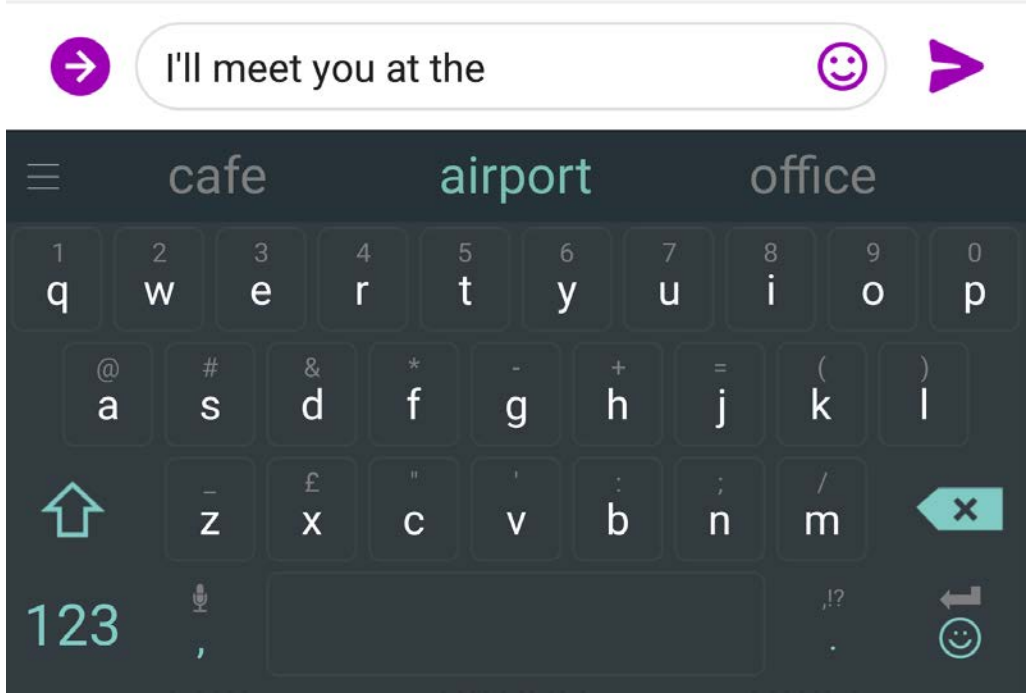
- Relationship between words in a sentence

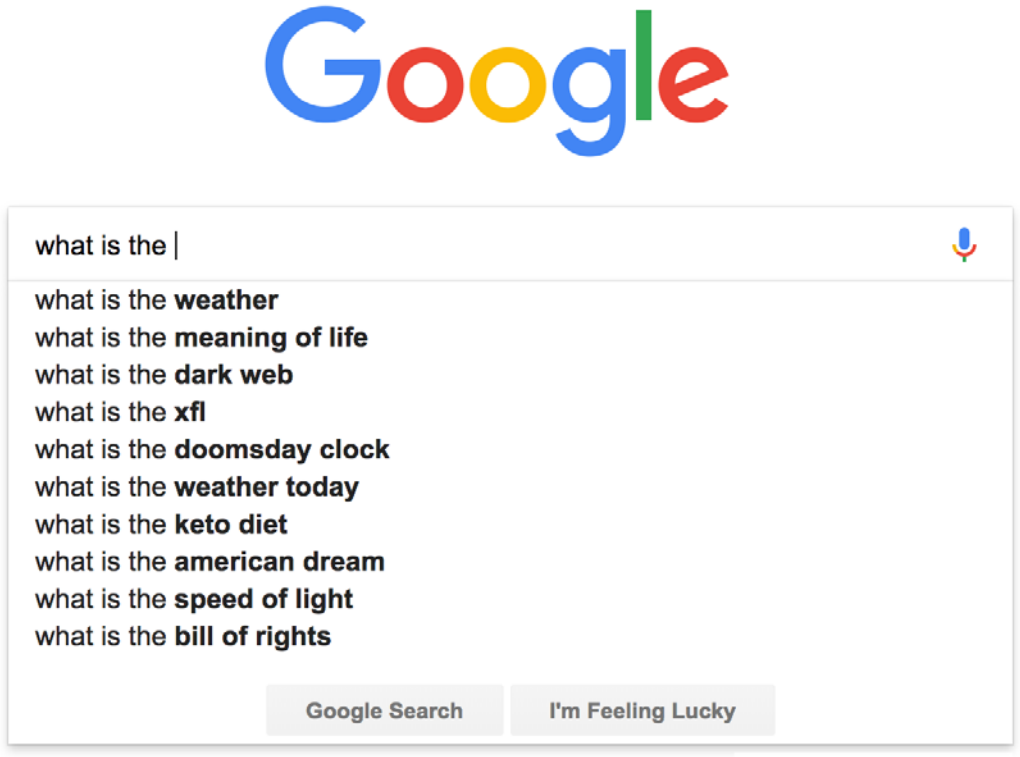


https://nlp.stanford.edu/software/stanford-dependencies.html

Deep learning for NLP: Language modeling

- Takes a list of words (history/context), and attempts to predict the word that follows them

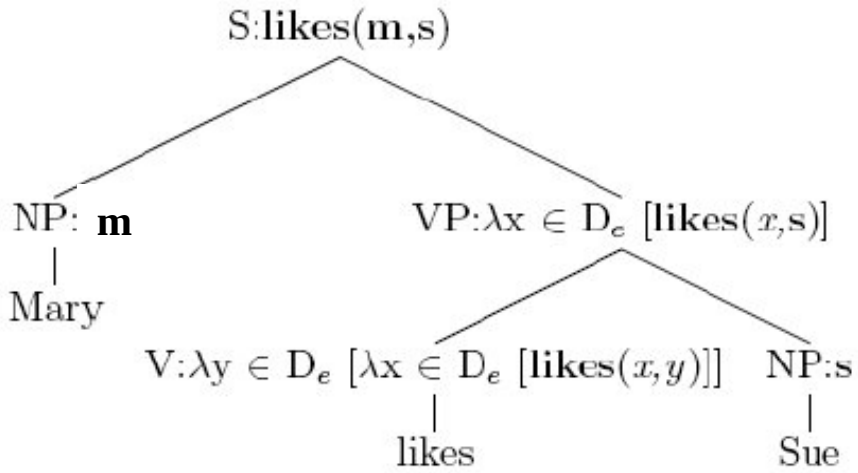




Source: NTUNLP

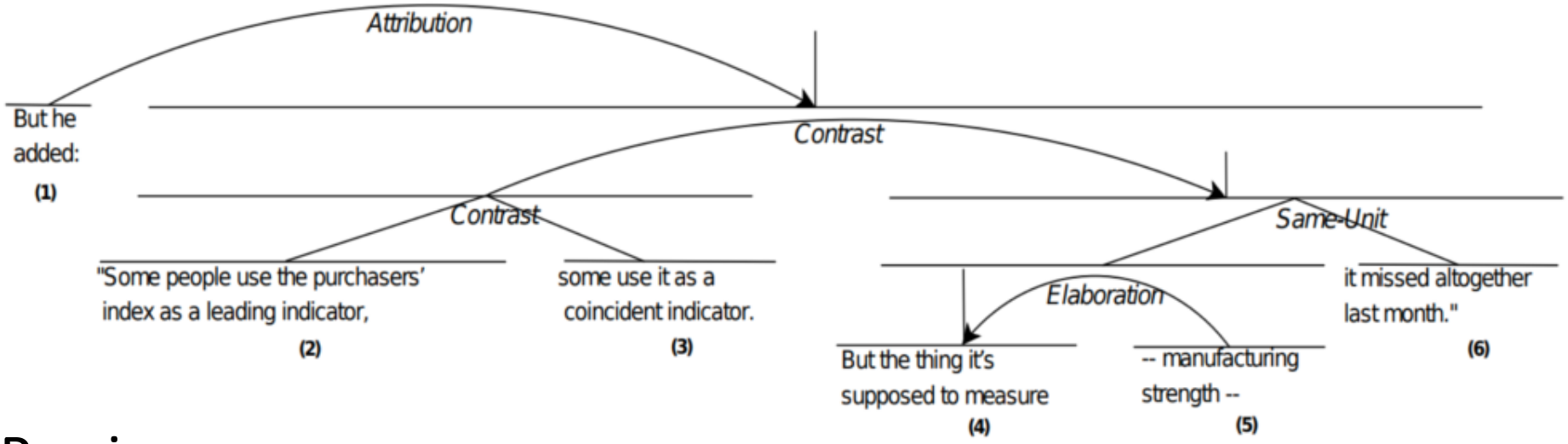
Deep learning for NLP: Semantics

- Traditional: Lambda calculus
  - Take as inputs carefully engineered functions
  - No notion of similarity or fuzziness of language
    - E.g. Mary loves/may like Sue
- DL for semantic parsing:
  - End-to-end encoder-decoder model
    - Input: Sentence (e.g. Mary likes Sue)
    - Output: Semantic representation (e.g. likes(Mary,Sue))

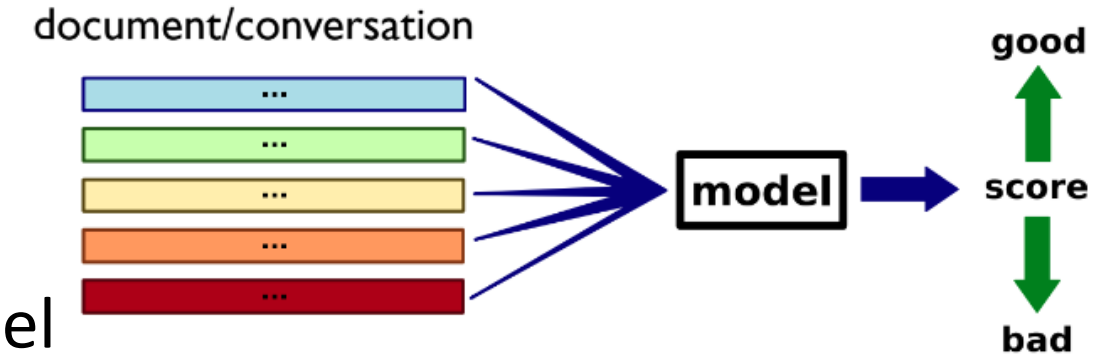


Source: CS224n

Deep learning for NLP: Discourse



Parsing



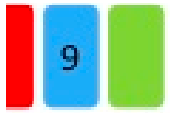
Coherence model

https://www.aclweb.org/anthology/P13-1048/ https://www.aclweb.org/anthology/P17-1121/

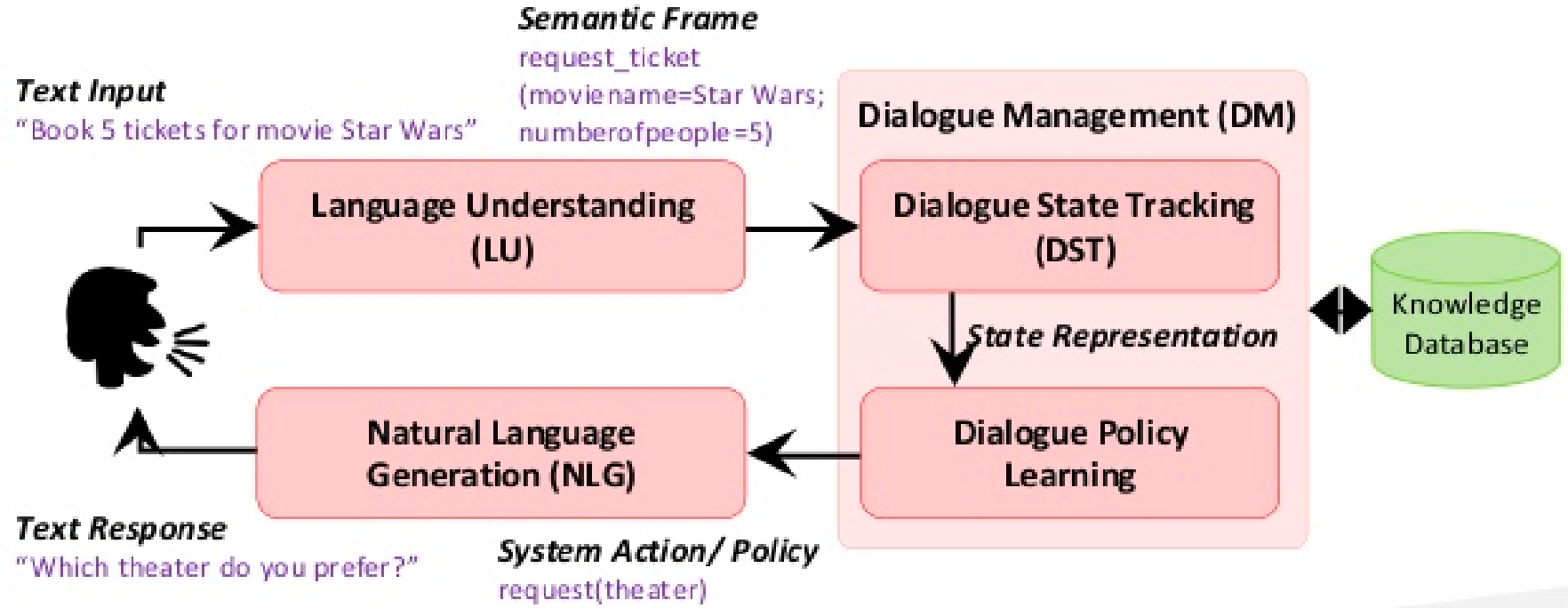
Source: NTUNLP

Deep learning for NLP: Dialogue systems

- Microsoft Research Asia



Task-Oriented Dialogue System Framework



Source: MSRA

Contents

- Course logistics
- What is NLP?
- Why deep learning for NLP?
- Course outline
- TF-IDF

Course outline: Weeks 2-3

[Week 2: Word vectors]

- Before representation learning: TF-IDF
- Word vectors
- Word2vec: Learning word vectors
  - Stochastic gradient descent (SGD)
  - Negative sampling
- GloVe
- Word vector evaluation

[Week 3: Word-window classification]

- NLP application: Named entity recognition (NER)
  - Before deep neural network: HMM, CRF
- Word-window classification
  - Softmax classifier
  - Multilayer Perceptron: Gradient, Backpropagation

Course outline: Weeks 4-5

[Week 4: Convolutional neural network]

- NLP application: Dependency parsing
  - Phrase structure, dependency structure, part-of-speech (POS) tagging
  - Before deep neural network: MaltParser
  - Neural approach
- NLP application: Text classification
- Convolutional Neural Network (CNN)
  - CNN for text classification

[Week 5: Recurrent neural network]

- NLP application: Language modeling
  - Before deep learning: N-gram language model
- Recurrent Neural Network (RNN)
  - Vanishing gradient problem
  - LSTM, GRU
  - Bidirectional RNNs, multi-layer RNNs

Course outline: Weeks 6-7

[Week 6: seq-2-seq with attention]

- NLP application: Machine Translation
  - Pre-neural Machine Translation
- Neural Machine Translation (NMT) using sequence-to-sequence (seq2seq)
- Neural Machine Translation using sequence-to-sequence with attention

[Week 7: Subword models]

- Motivation of subword models
- Purely character-level models
- Byte Pair Encoding
- Hybrid NMT

Course outline: Weeks 8-9

[Week 8: Recursive Neural Nets & Parsing]

- Syntactic structures: Constituency & Dependency structures
- Need for syntactic structures
- Are languages hierarchical?
- Parsing with Recursive Neural Networks
- Backpropagation through tree
- Semantic composition with Tree RNNs
- Recent Constituency Parsers

[Week 9: Seq2Seq variants& Transformers]

- Recap: sequence-to-sequence with attention
- Seq2Seq Variants
- Transformer Seq2Seq
- NLP applications: Machine translation, summarization, parsing, dialogue generation

Course outline: Weeks 10-11

[Week 10: Self-supervised Models]

- Self-supervised pretraining: Cove, ELMO, GPT, BERT, XLNet, Roberta, Albert, Bart, XLM
- NLP tasks/applications: NLI (+XNLI), GLUE (+SuperGlue), QA (SQUAD), machine translation, summarization

[Week 11: Adversarial NLP]

- Adversarial Nets: Generative adversarial nets (GANs), domain adversarial nets (DANs)
- NLP tasks/applications: Machine translation, domain transfer
- Adversarial examples & training: Adversarial attacks, defence, robust NLP

Course outline: Week 12

[Week 12: Multilingual NLP]

- Why multilingual NLP?
- Cross-lingual models

Contents

- Course logistics
- What is NLP?
- Why deep learning for NLP?
- Course outline
- TF-IDF

Unstructured text data

- How can we represent a text as features for machine learning? E.g. sentiment analysis
- Manually write a keyword set
  - Time-consuming, costly, no guarantee of high coverage
- Consider all words (in training data) as features
  - What is a word?
  - How to represent a word?
  - How many words are required as features?
  - How to deal with ‘unseen’ words in test data? (not covered in this week)

The script is not only incredibly tight but precise as well and Poésy delivers not just one captivating plot to follow but two: the exciting drama we get to see played out through the lens and the equally intriguing subtext underneath it all. A heated encounter between ex-lovers that is only escalated by the introduction of live ammunition might have sufficed for some filmmakers, but there is also so much of the story Poésy wants you to put together yourself.

https://www.ukfilmreview.co.uk/post/the-tear-s-thing-short-film-review

Tokenization

- Identifying the tokens (words and symbols) in a text that we may want to deal with
  - Or, word segmentation, word tokenization
- For English, why not just use white-space?
  - E.g. two: all. filmmakers,
- Word-internal punctuation: e.g. Ph.D. A*Star 03/06/20 google.com
- Clitics: e.g. What’re, I’m
- Multi-token words (named entity recognition)
  - E.g. New York; Rock ‘n’ roll

Source: IR

Binary term-document incidence matrix

- Assume we have a large collection of documents (called corpus)
- Each document is represented by a binary vector $\in \{0,1\}^{|V|}$
- This is a bag of words model: Doesn't consider the ordering of words in a document
  - *E.g. "John is quicker than Mary"* and *"Mary is quicker than John"* have the same vectors

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

Shakespeare Corpus

Source: IR

Term-document count matrices

- Consider the number of occurrences of a word in a document:
  - Each document is a count vector in $\mathbb{N}^{|V|}$: a column below

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

Source: IR

Term frequency (tf)

- The term frequency (count) $tf_{t,d}$ of term (word) *t* in document *d* is defined as the number of times that *t* occurs in *d*.
- The log frequency weight of term t in d is

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

  - 0 → 0, 1 → 1, 2 → 1.3, 10 → 2, 1000 → 4, etc.

Source: IR

Inverse document frequency (idf)

- Rare terms are more informative than frequent terms
  - Stop words (e.g. the, of)
- df_t is the document frequency of *t*: the number of documents in training data that contain *t*
  - df_t is an inverse measure of the informativeness of *t*
  - $df_t \leq N$
- We define the idf (inverse document frequency) of *t* by

$$idf_t = \log_{10}(N/df_t)$$

  - We use log (N/df_t) instead of N/df_t to "dampen" the effect of idf.

The base of the log is not important.

Source: IR

tf-idf vector

- The tf-idf weight of a term is the product of its tf weight and its idf weight.

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

- Increases with the number of occurrences within a document
- Increases with the rarity of the term in the collection

Source: IR

Binary → count → weight matrix

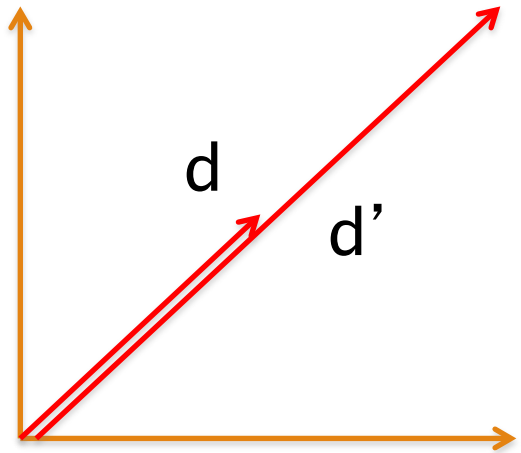
- Each document is now represented by a real-valued vector of tf-idf weights $\in \mathbb{R}^{|V|}$

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	5.25	3.18	0	0	0	0.35
Brutus	1.21	6.1	0	1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95

Source: IR

Motivation of Length normalization

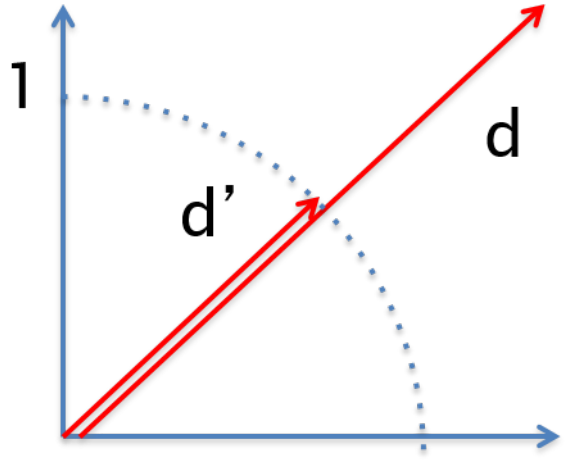
- Thought experiment: take a document *d* and append it to itself. Call this document *d'*.
- "Semantically" *d* and *d'* have the same content
- The Euclidean distance between the two document vectors can be quite large
- But the angle between the two document vectors is ~0, corresponding to maximal similarity



Source: IR

Length normalization

- A vector can be (length-) normalized by dividing each of its components by its length – for this we use the L_2 norm: $\|\vec{x}\|_2 = \sqrt{\sum_i x_i^2}$
- Dividing a vector by its L_2 norm makes it a unit (length) vector (on surface of unit hypersphere)
- Unit vector has length 1



- Effect on the two documents *d* and *d'* (*d* appended to itself) from earlier slide: they have identical vectors after length-normalization.
- Long and short documents now have comparable weights

Source: IR

Documents as vectors

- So we have a |V|-dimensional vector space
- Terms are axes of the space
- Documents are points or vectors in this space
- Very high-dimensional: tens of millions of dimensions when you apply this to a big corpus (text collection)
  - Select top-k frequent terms (|V| = k)
  - Select terms whose frequencies are higher than k
- These are very sparse vectors - most entries are zero.

Source: IR

Document similarity: Cosine similarity

- E.g. Find the document (d) in given collection that is the closest to a new document (q)
  - $\vec{d}$ and $\vec{q}$ are TF-IDF vectors

Dot product

$$\cos(\vec{q},\vec{d}) = \frac{\vec{q}\cdot\vec{d}}{|\vec{q}||\vec{d}|} = \frac{\vec{q}}{|\vec{q}|}\cdot\frac{\vec{d}}{|\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i \times d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2}\sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

Unit vectors