Note for Lecture 1 and 2

CS224N Natural Language Processing with Deep Learning

Intersection of computer science, artificial intelligence and linguistics

Goal: for computers to process or “understand” natural language in order to perform tasks

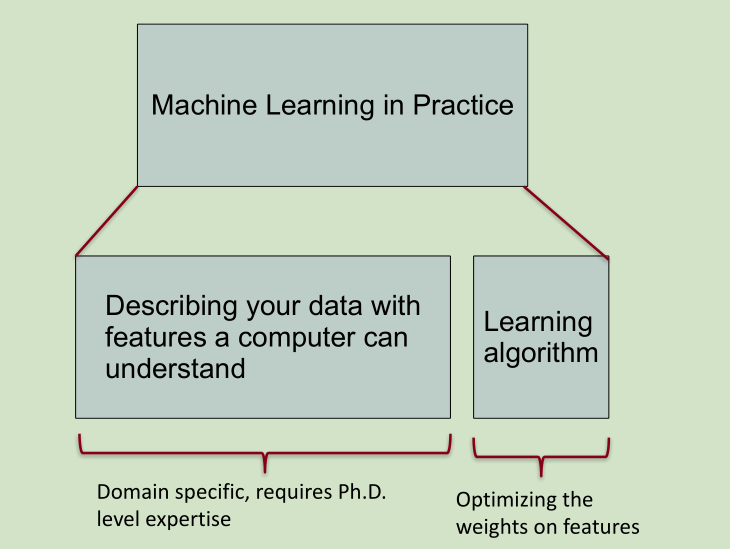
Fully u**nderstanding and representing** the **meaning** of language

Special about human language:

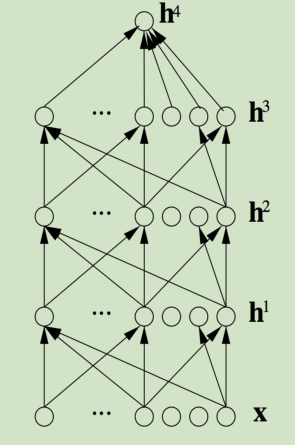
* A human language is a system specifically constructed to convey the speaker/writer’s meaning
* **discrete/symbolic/categorical signaling system**
* categorical symbols of language can be encoded as signal for communication in several ways: sound, gesture, writing/image
* brain encoding appears to be a **continuous pattern of activation**, and the symbols are transmitted via **continuous signals** of sound/vision
* **sparsity**: problem for machine learning due to the large vocabulary symbolic encoding of words

Deep Learning:

* subfield of machine learning
  + Most machine learning methods work well because of **human-designed representations** and **input features**



* Deep learning algorithms attempt to learn (multiple levels of) representations (h1, h2, h3) and an output (h4) from “row” input x



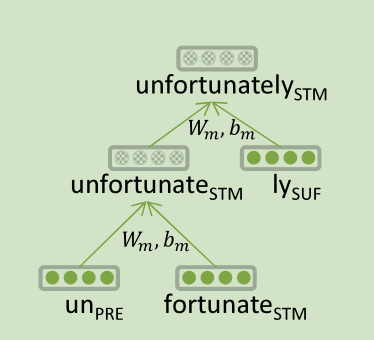
* Reasons for exploring deep learning:
  + Manually designed features are often over-specified incomplete and take a long time to design and validate
  + Learned features are easy to adapt and fast to learn
  + Deep learning provides a very flexible universal learnable framework for representing world, visual and linguistic information
  + Can learn unsupervised and supervised
  + Improved performance from 2010

Deep NLP: Deep Learning + NLP

* Combine the ideas and goals of NLP with using representation learning and deep learning methods to solve them

Representations of NLP levels: Morphology

* Traditional: words made of morphemes ---- prefix + stem + suffix
* DL:
  + Every morpheme is a vector
  + A neural network combines two vectors into one vector



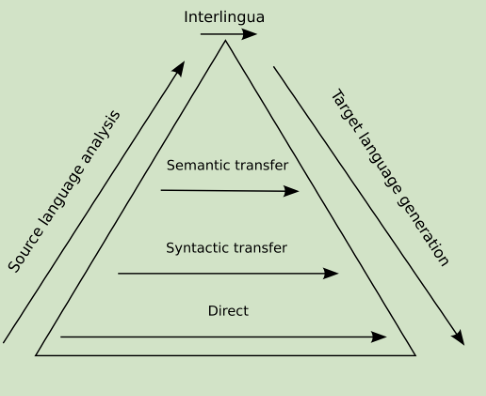
* Neural networks can accurately determine the grammatical structure of sentences

Representations of NLP levels: Semantics

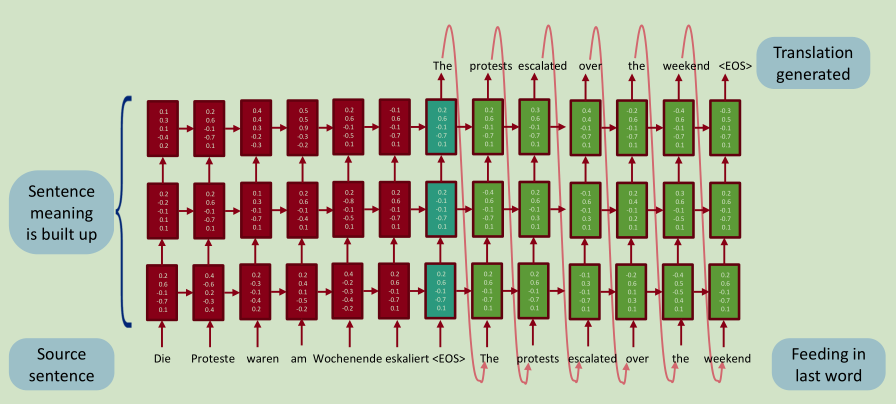
* Traditional: Lambda calculus
  + Carefully engineered functions
  + Take as inputs specific other functions
  + No notion of similarity or fuzziness of language
  + Every word and every phrase and every logical expression is a vector
  + A neural network combines two vectors into one vector

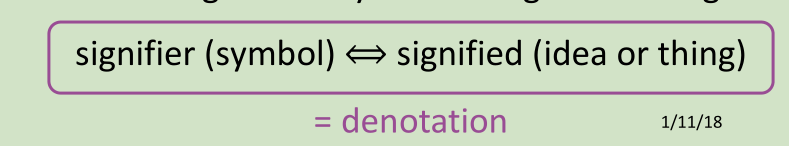
NLP Applications:

* Sentiment Analysis
  + Traditional: treat sentence as a bag-of-words (ignore word order); consult a curated list of “positive” and “negative” words to determine sentiment of sentence; need hand-designed features to capture negation
  + Same deep learning model that could be used for morphology, syntax and logical semantics. -> Recursive Neural Network (aka TreeRNNs)
* Question Answering:
  + Traditional: a lot of feature engineering to capture world and other knowledge
  + DL: facts are stored in vectors
* Dialogue agent
  + Recurrent Neural Networks
* Machine Translation:
  + Traditional:



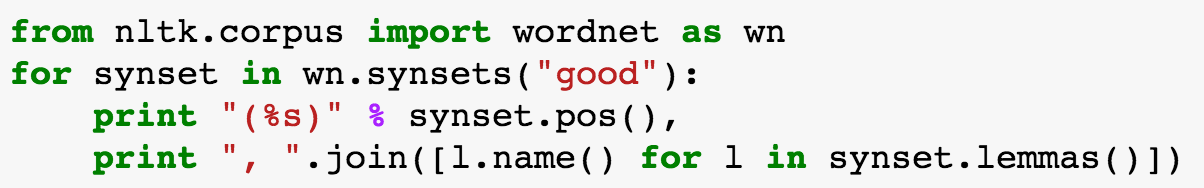
* + DL approach: source sentence is mapped to vector, then output sentence generated

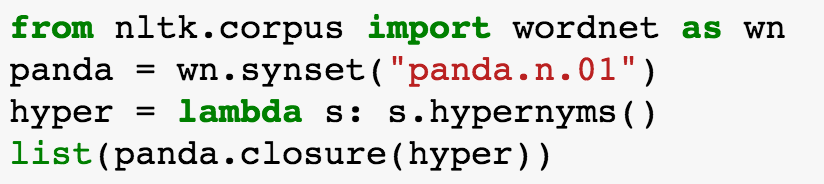




**WordNet**: a resource containing lists of synonym sets and hypernyms (“is a” relationships)

* Problems:
  + Great as a resource but missing nuance
  + Missing new meaning of words (impossible to keep up-to-date)
  + Subjective
  + Requires human labor to create and adapt
  + Hard to compute accurate word similarity





Traditional representation of words:

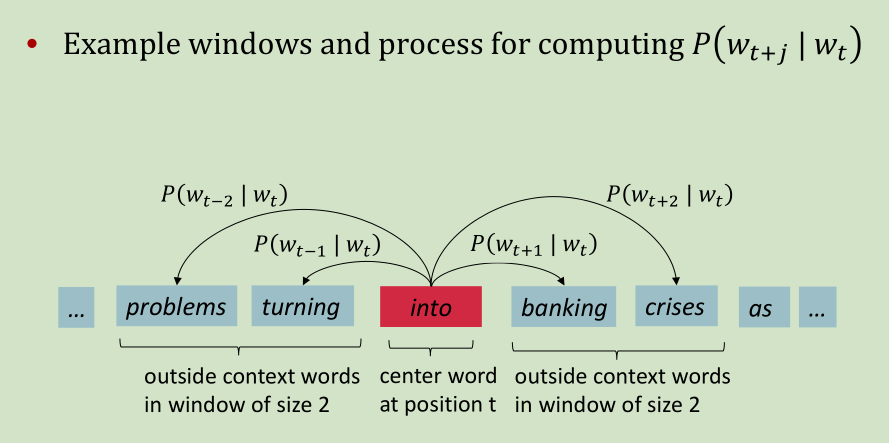
* Regard words as discrete symbols represented by one-hot vectors, vector dimension = number of words in vocabulary
  + Problem: no natural notion of similarity for one-hot vectors

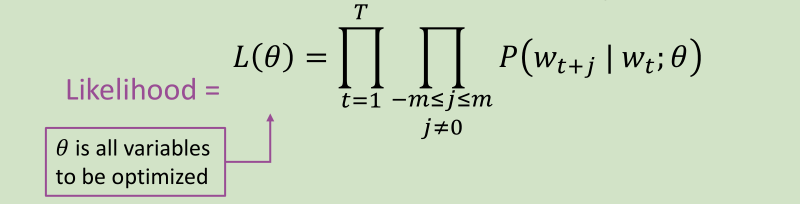
Solution: rely on WordNet’s list of synonyms to get similarity

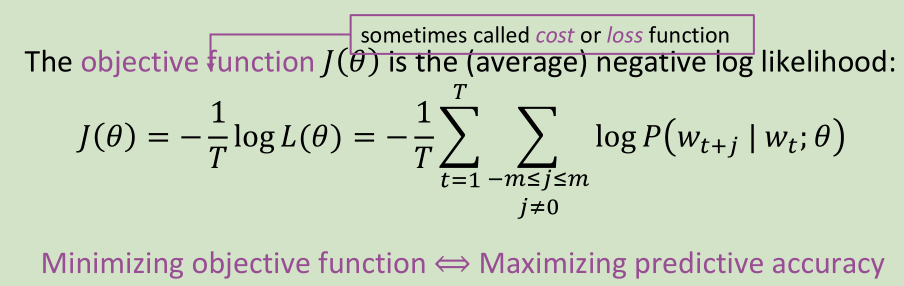
* Represent words by their context: a word’s meaning is given by the words that frequently appear close-by
  + When a word w appears in a text, its context is the set of words that appear nearby (within a fixed-size window).
  + Use the many contexts of w to build up a representation of w
  + Build a dense vector for each word chosen so that it is similar to vectors of words that appear in similar contexts
  + Word vectors/ word embedding/ word representations\

Word2vec: a framework for learning word vectors

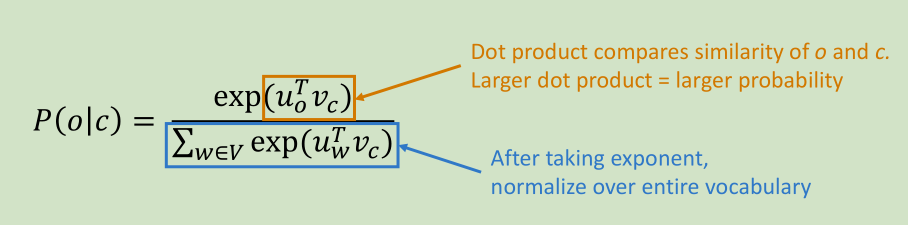
* We have a large corpus of text
* Every word in a fixed vocabulary is represented by a vector
* Go through each position t in the text, which has a center word c and context (“outside”) words o
* Use the similarity of the word vectors for c and o to calculate the probability of o given c (or vice versa)
* Keep adjusting the word vectors to maximize this probability

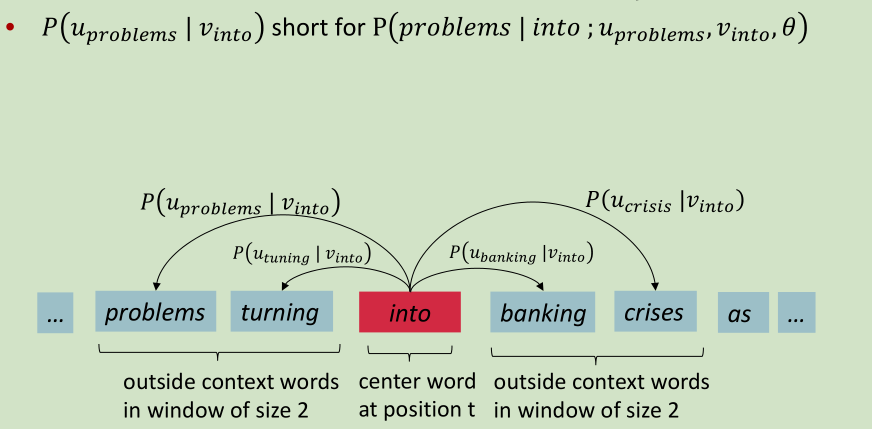


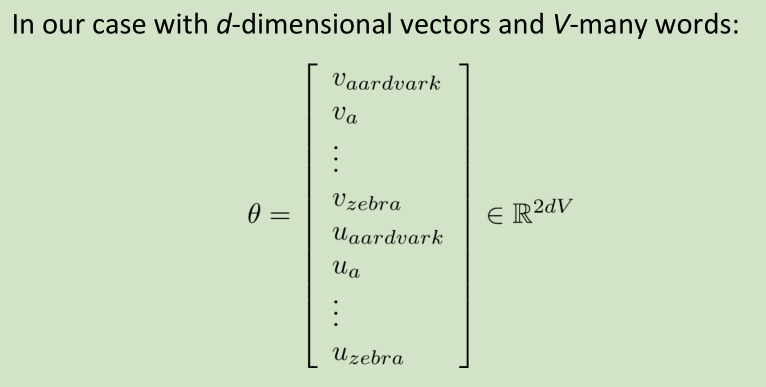
* For each position t = 1, ..., T, predict context words within a window of fixed size m, given center word wj 



* Use two vectors per word “w” to calculate the likelihood
  + “vw” when “w” is a center word
  + “u­w” when “w” is a context word
  + For a center word “c” and a context word “o”

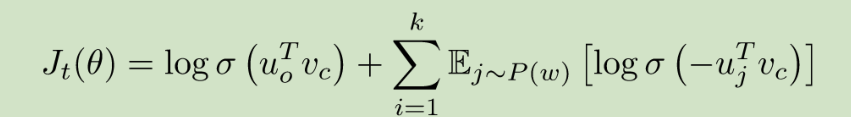


* 
* ­­­The prediction function is an example of the softmax function,
  + Softmax function maps arbitrary values xi to a probability distribution pi
  + “max” because amplifies probability of largest xi
  + “soft” because still assigns some probability to smaller xi
* Train the model: compute all vector gradients

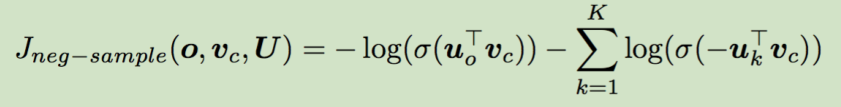


Two model variants:

* Skip-grams (SG)
  + Predict context (“outside”) words (position independent) given center word
  + Maximize:



* + Negative sampling:
    - Take k negative samples



* + - Maximize probability that real outside word appears, minimize probability that random words appear around center word
    - The unigram distribution U(w) raised to 3/4 power

C:\Users\dell\AppData\Local\Temp\1545661194(1).png

* Continuous Bag of Words (CBOW)
  + Predict center word from (bag of) context words (sum of surrounding word vectors) instead of predicting surrounding single words from center word as in skip-gram model