Text Analytics

MSiA 414
Lecture 2

Lecture 1 Recap

History of NLP 1950 - 2000s

What is the difference between UTF8 and Unicode?

What do you think the following regex captures: <([\w]+).*>(.*?)<\/\1>

Why is text normalization performed? Examples of text normalization.

What is the difference between POS-tagging, chunking, and syntax parsing?

Homework 1 Discussion

Agenda

A short history of NLP (part 2 of 2)

Deep learning

Backpropagation, mini-batch gradient descent, softmax, log loss

The state-of-the-art in NLP

One-hot encodings and word embeddings

Neural language models

word2vec

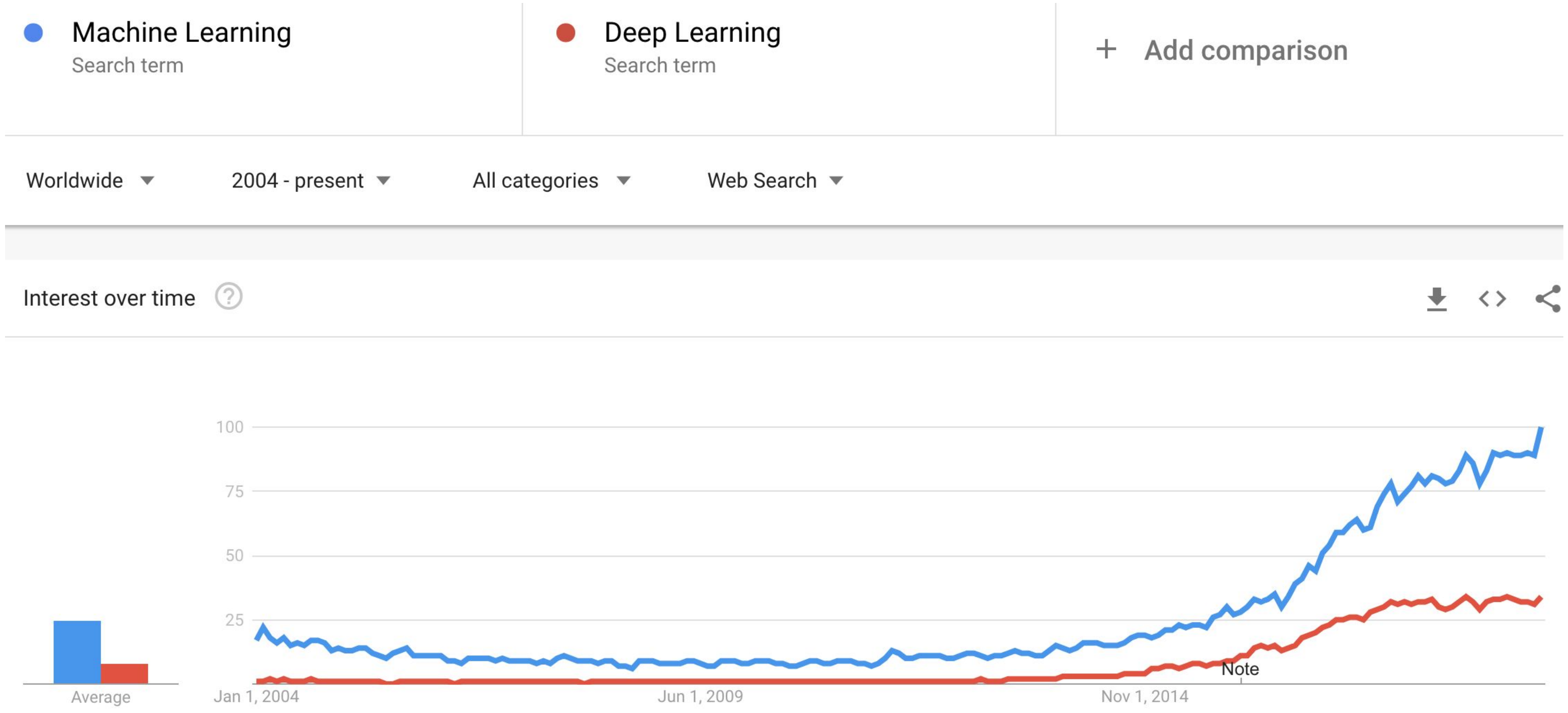
A short history of NLP (part 2 of 2)

~2010 - present: **Deep Learning in mainstream NLP**

Key deep learning boom factors:

1) **processing power** and

2) **large datasets**.



● Machine Learning
Search term

● Deep Learning
Search term

+ Add comparison

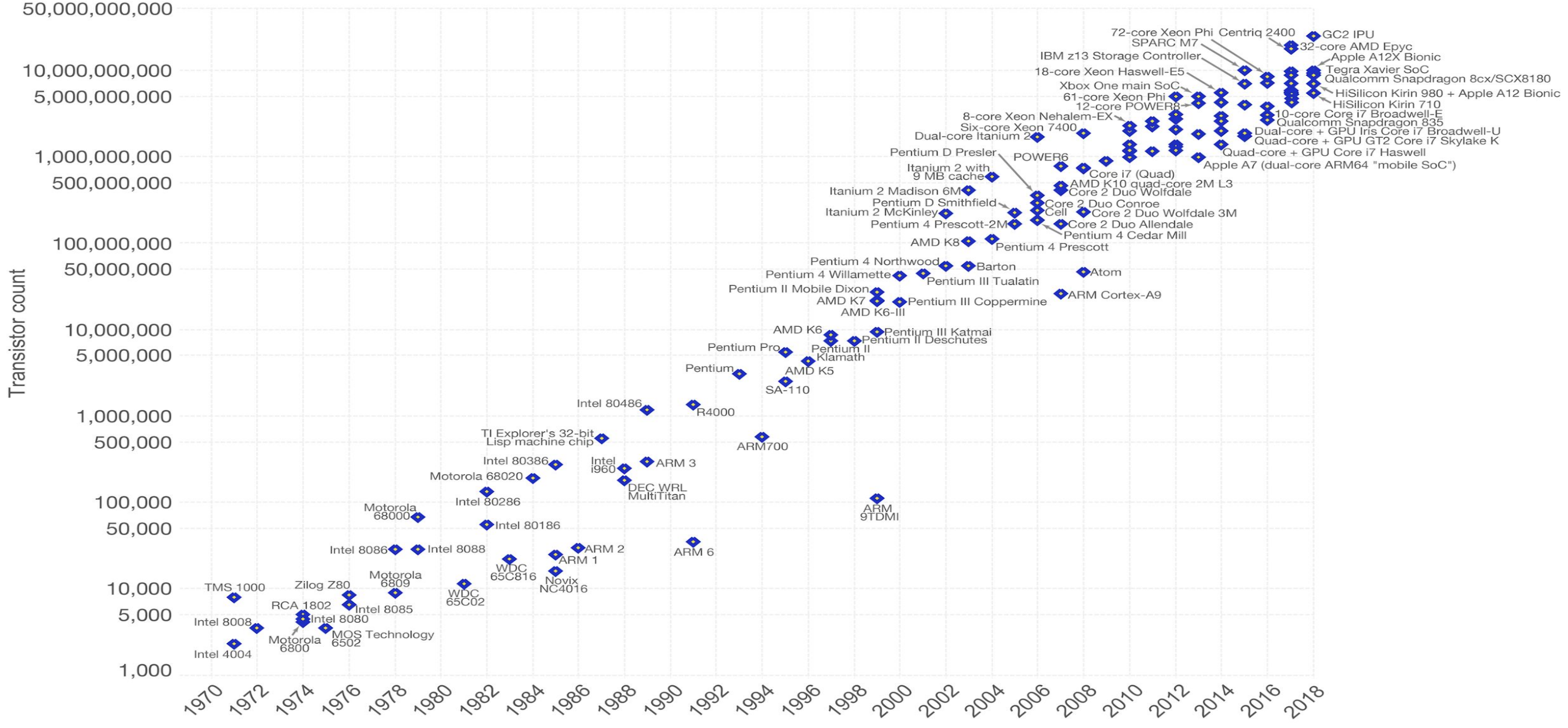
Worldwide ▼ 2004 - present ▼ All categories ▼ Web Search ▼

Interest over time

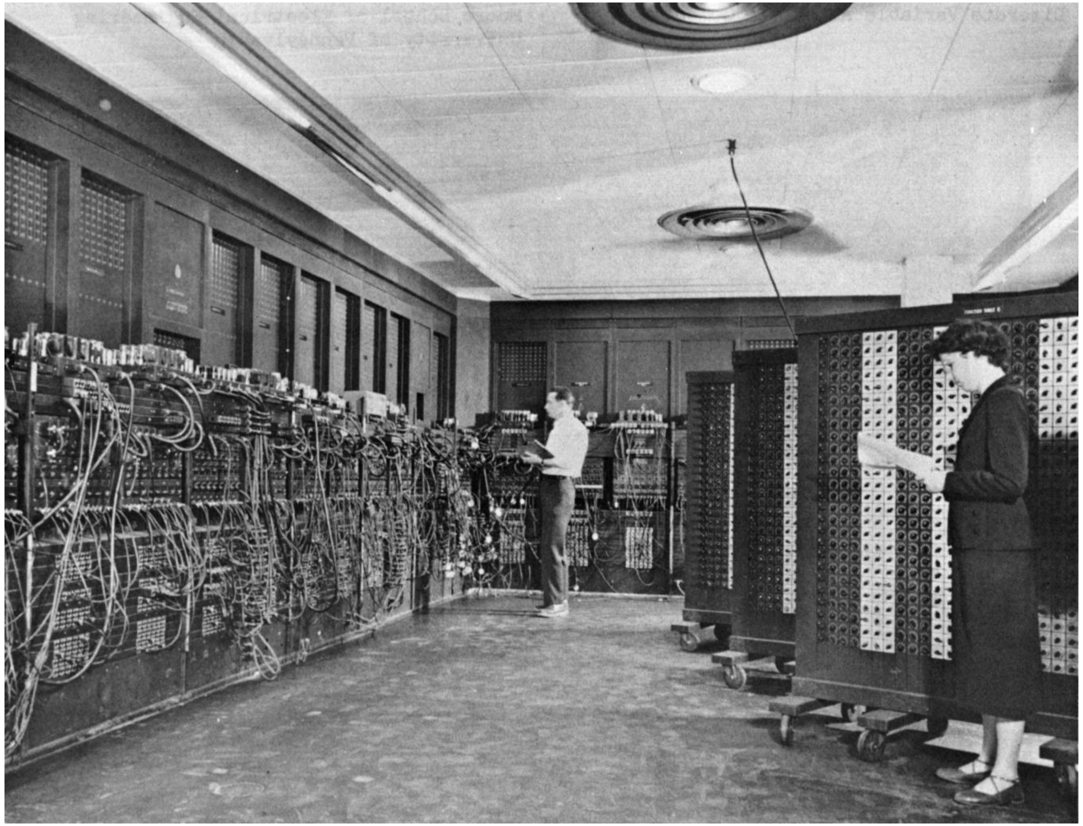
Moore's Law – The number of transistors on integrated circuit chips (1971-2018)

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are linked to Moore's law.

Our World in Data



Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)
The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.



ENIAC: 1946, the first electronic general-purpose computer.

2018 iPhone.

1 Zetabyte = 10^21 bytes = 1000000000000000000000 bytes

33 Zettabytes of data produced in 2018

175 Zettabytes estimated for 2025

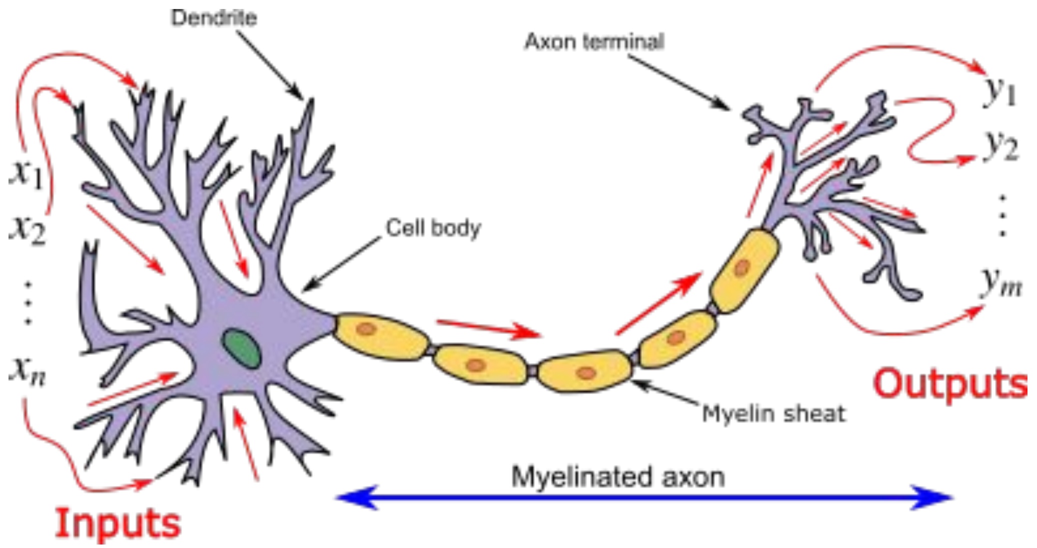
Source: Source: IDC White Paper, November 2018

Deep neural networks trained on large datasets:

For example, ResNet: trained on million ImageNet images; Bert: pre-trained on all of Wikipedia + 11,000 books

History of Deep Learning

1943: First mathematical model of neural networks, developed by a logician and neuroscientist (McCulloch-Pitts neurons).



1947: *What we want is a machine that can learn from experience.* Alan Turing

History of Deep Learning Cont’d

1957: Rosenblatt, a psychologist, wrote a paper entitled “**The Perceptron**: A Perceiving and Recognizing Automaton”. [He will] “construct an electronic or electromechanical system which would learn to recognize similarities or identities between patterns of optical, electrical, or tonal information, in a manner which may be closely analogous to the perceptual processes of a biological brain.”

The perceptron is an algorithm for learning a binary classifier with a **threshold function**: a function that maps its input (a real-valued vector) to an output value of 0 or 1 (a single binary value).

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0, \\ 0 & \text{otherwise} \end{cases}$$

The Perceptron

The simplest feed-forward neural network. Linear classifier.

Training procedure:

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0, \\ 0 & \text{otherwise} \end{cases}$$

- Iterate over training examples
- If the network output is correct leave the weights alone.
- If the output is 1 when it should have been 0, subtract the input vector from the weight vector.
- If the out is 0 when it should have been 1, add the input vector to the weight vector.

Please watch Geoffrey Hinton's 8-minute video on Perceptrons.

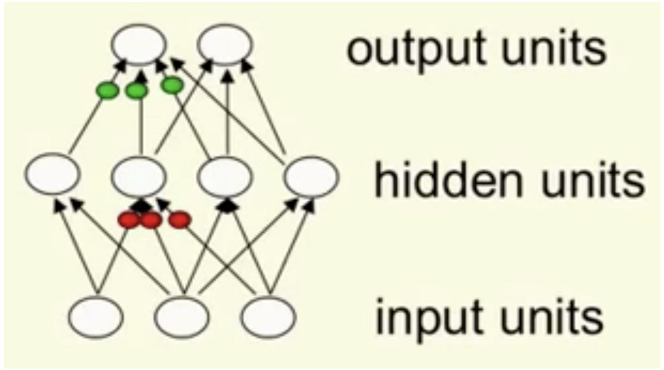
History of Deep Learning Cont’d

1969: Marvin Minsky and Seymour A. Papert publish a book ("Perceptrons") pointing out the limitations of perceptron.

1970s: The first "AI winter" (i.e. lack of funding).

1980s: The backpropagation algorithm and multi-layer neural network learning.

Backpropagation



Single-layer networks require feature engineering. **Multi-layer networks** (with hidden units) can learn without feature engineering (the computer learns the features from raw input).

Computes **error derivatives** of the hidden units efficiently, all at the same time. In other words, computes how fast the error changes, as we change the hidden activity, and then compute the error derivatives for the weights going into the hidden units.

Backpropagation

Forward pass to compute the output of the network.

Compute the error: $E_{total} = \sum \frac{1}{2}(target - output)^2$

Update each of the weights in the network so that they cause the actual output to be closer the target output, thereby minimizing the error for each output neuron and the network as a whole.

Geoffrey Hinton's 11-minute video on backpropagation

Mini-batch gradient descent

Stochastic gradient descent (“online learning”) - update network weights one training example at a time.

Full gradient descent - Calculates the error for each example in the training dataset, but only updates the model after all training examples have been evaluated (i.e. at the end of each training epoch).

Mini-batch gradient descent - less computation is used (weighs updated less often), parallel computation for the gradients of the examples (matrix multiplication is very efficient, especially on GPUs).

Geoffrey Hinton on mini-batch gradient descent (8 minutes)

Softmax function

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_i}}$$

- The most common output function in neural NLP. Forces the network output (Y) to represent categories, i.e. probabilities that sum up to 1. Outputs a vector that represents the probability distributions of a list of potential outcomes.
- transfer function ~ activation function ~ output function ~ squashing function
- transfer function = activation function + output function

```
import numpy as np

def softmax(x):
  return np.exp(x) / float(sum(np.exp(x)))
scores = [3.0, 1.0, 0.2]
print(softmax(scores))
print(sum(softmax(scores)))
```

```
[0.8360188  0.11314284 0.05083836]
0.9999999999999999
```

Cross-entropy, aka log-loss

Cost function = objective function = loss function = error function (the thing we want to minimize).

Cross Entropy: the negative log probability of the right answer. Measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label. So predicting a probability of .0001 when the actual observation label is 1 would be bad and result in a high loss value.

```
import math
def binary_cross_entory(predicted_y, actual_y):
  if actual_y == 1:
     return -math.log(predicted_y)
  else:
     return -math.log(1 - predicted_y)
print(binary_cross_entory(0.1, 1))
print(binary_cross_entory(0.0001, 1))
print(binary_cross_entory(0.1, 0))
print(binary_cross_entory(0.0001, 0))
```

```
2.3025850929940455
9.210340371976182
0.10536051565782628
0.00010000500033334732
```

History of Deep Learning Cont’d

1990s: the second “AI winter”, also affected Deep Learning

Geoffrey Hinton - University of Toronto, Google Brain

Yann LeCun - Facebook, New York University

Yoshua Bengio - University of Montreal

Recipients of the 2018 ACM Turing award: the “Nobel prize” of Computer Science.

Please watch: Geoffrey Hinton and Yann LeCun, 2018 ACM Turing Award Lecture "The Deep Learning Revolution"

The State-of-the-Art (SOTA) in NLP

Deep Learning and the Use **Context (prior knowledge)**

- Word embeddings: Word2vec, Glove
- Embeddings with context: Elmo, Bert
- Multi-task learning: MT DNN

Improve SOTA formula:

Unsupervised pre-training on a large dataset(s)

+

Training on a much smaller labelled dataset (or multiple datasets / tasks)

=

Beat the previous SOTA

Why Word Embeddings?

Training Data:

Text	**Label**
The food was horrible	Negative review
The restaurant was great	Positive review

Test Data:

Text	**Label**
The restaurant was atrocious	?

“One-hot” encodings

Training texts: The food was horrible The restaurant was great
Dictionary: The, food, was, horrible, restaurant, great, <UNKNOWN>

Index	**Token**	**One-hot encoding**
0	<UNKNOWN>	1 0 0 0 0 0 0
1	The	0 1 0 0 0 0 0
2	food	0 0 1 0 0 0 0
3	was	0 0 0 1 0 0 0
4	horrible	0 0 0 0 1 0 0
5	restaurant	0 0 0 0 0 1 0
6	great	0 0 0 0 0 0 1

Words represented as “features”, Embedding Matrix

Embedding size

Word features	The	food	was	**horrible**	restaurant	great	**atrocious**
Sentiment adjective	-0.013	-0.2	0.003	0.97	0.1	0.99	0.99
Positive sentiment	0.02	0.1	-0.02	-0.99	0.2	0.98	-0.98
Something related to eating food	~0	.898	~0	~0	0.98	~0	~0
…...	…..	….					

Vocabulary size

Word embeddings

The magic of creating “word features” from large amounts of unlabeled free texts. Learning an “embedding matrix”.

“Word features” = distributed representation of words = word embedding = word vector

Bengio, Yoshua, et al. "A neural probabilistic language model." Journal of machine learning research 3.Feb (2003): 1137-1155.

A model that learns 1) distributed representations of words and 2) the probability function for word sequences, expressed in terms of these representations.

Neural Language Model

An article generated with Open AI language model.

3 ways to improve your work day, and brightest minds will realize these advantages quickly. Test Your Brain for The Reasons How Effective Do People Work? Every sentence now gains relevance when you realize your genius is at its most effective. Remember that the most effective men are at most 50% smarter and more loyal than people general abilities are. And once you learn that you and your career partner are constantly learning and learning to see that crucial things like working memory, motivation, and understanding are essential to mental success—you can tackle your bottleneck effortlessly and make massive technical gains ...

word2vec

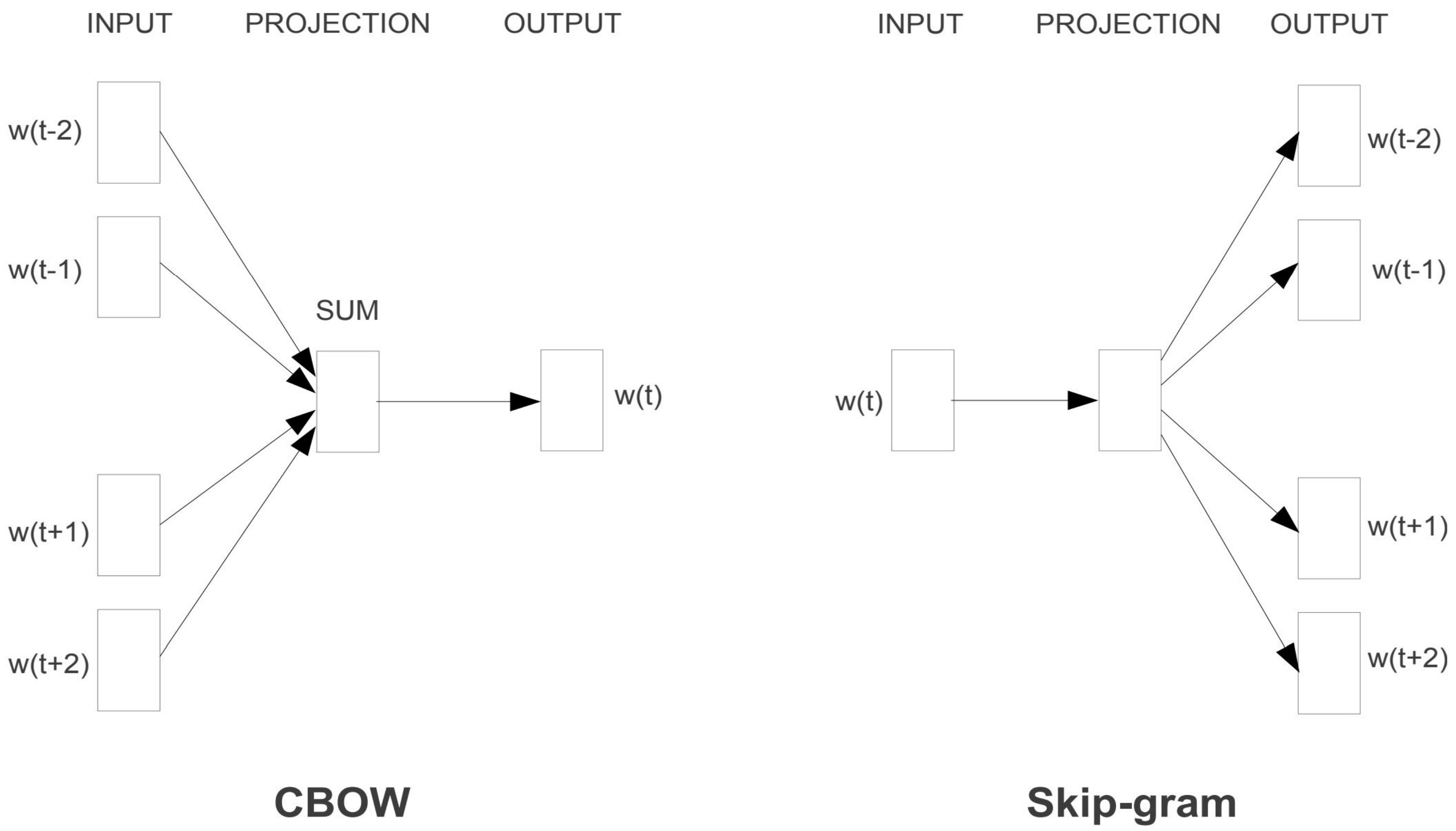
Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." arXiv preprint arXiv:1301.3781 (2013).

Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." Advances in neural information processing systems. 2013.

A significant speedup compared to previous work, open source. An optimized single-machine implementation can train on more than 100 billion words in one day. A big hit in the NLP community (than 27,000 google scholar citations).

Technically, not a “deep learning” model, there are no hidden layers, a single-layer linear transformation.

"*Everything should be made as simple as possible, but not simpler.*" Einstein



The skip-gram model

[This place is wonderful], ***authentic****, [fresh made Italian dishes] with house made pasta.*

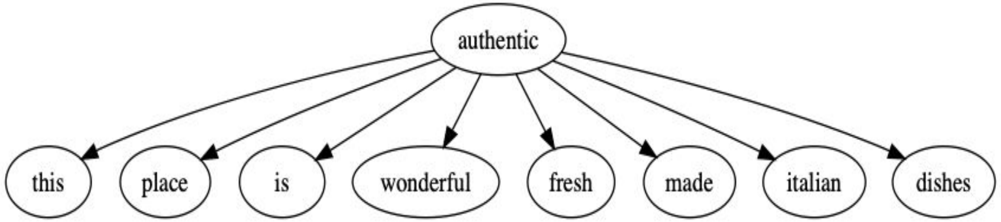
The objective is to maximize the average log probability (c=4, w_t='authentic'):

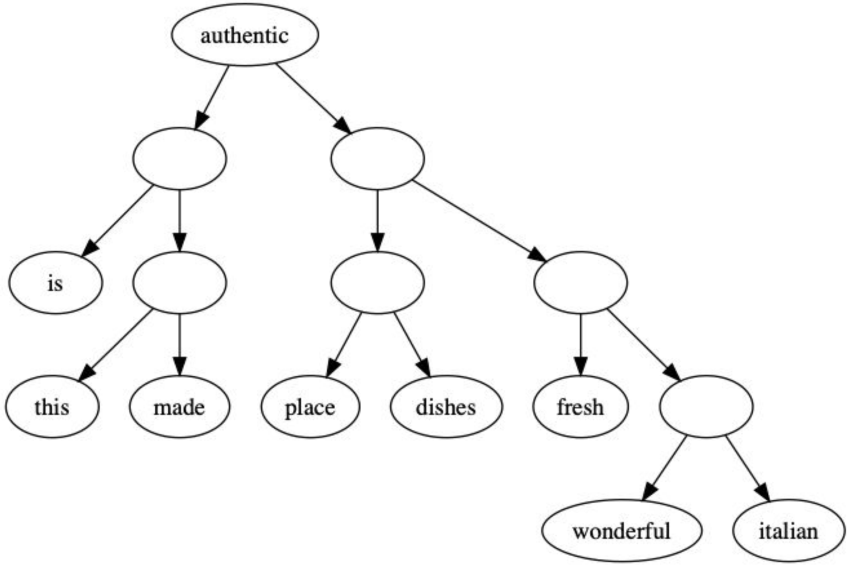
$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t)$$

Max Entropy (Maxent) model, think of it as logistic regression for more than 2 classes. Logistic regression = probabilistic model for 2 classes. Both logistic regression and MaxEnt are log linear models.

Hierarchical softmax

Computing probabilities for a large vocabulary (100,000+ words) is computationally expensive. Hierarchical softmax optimizes the computation (instead of evaluating W nodes for each vocabulary word, it evaluates $Log_2(W)$ nodes).





Further optimizations

Negative Sampling

- Distinguish the target word from randomly selected k negative samples (k is smaller for large dataset (2-5), and larger for small datasets (5-20)
- Logistic regression

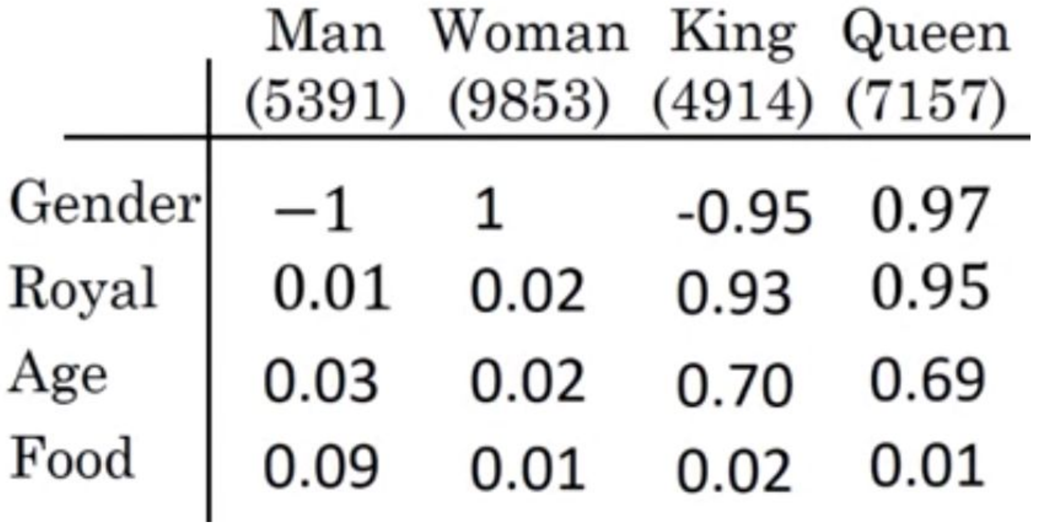
Context	**Word**	**Target?**
authentic	dishes	1
authentic	computation	0
authentic	blue	0

Subsampling

Use less examples of frequent words (e.g. *is, the, a*)

Word Vector Math

Cosine distance between word vectors: a measure of similarity. Two vectors with the same orientation have a cosine similarity of 1 (cosine of 0° is 1), two vectors diametrically opposed have a similarity of -1.



	Man (5391)	Woman (9853)	King (4914)	Queen (7157)
Gender	−1	1	-0.95	0.97
Royal	0.01	0.02	0.93	0.95
Age	0.03	0.02	0.70	0.69
Food	0.09	0.01	0.02	0.01

Embedding Evaluations

Evaluation using semantic and syntactic analogies datasets:

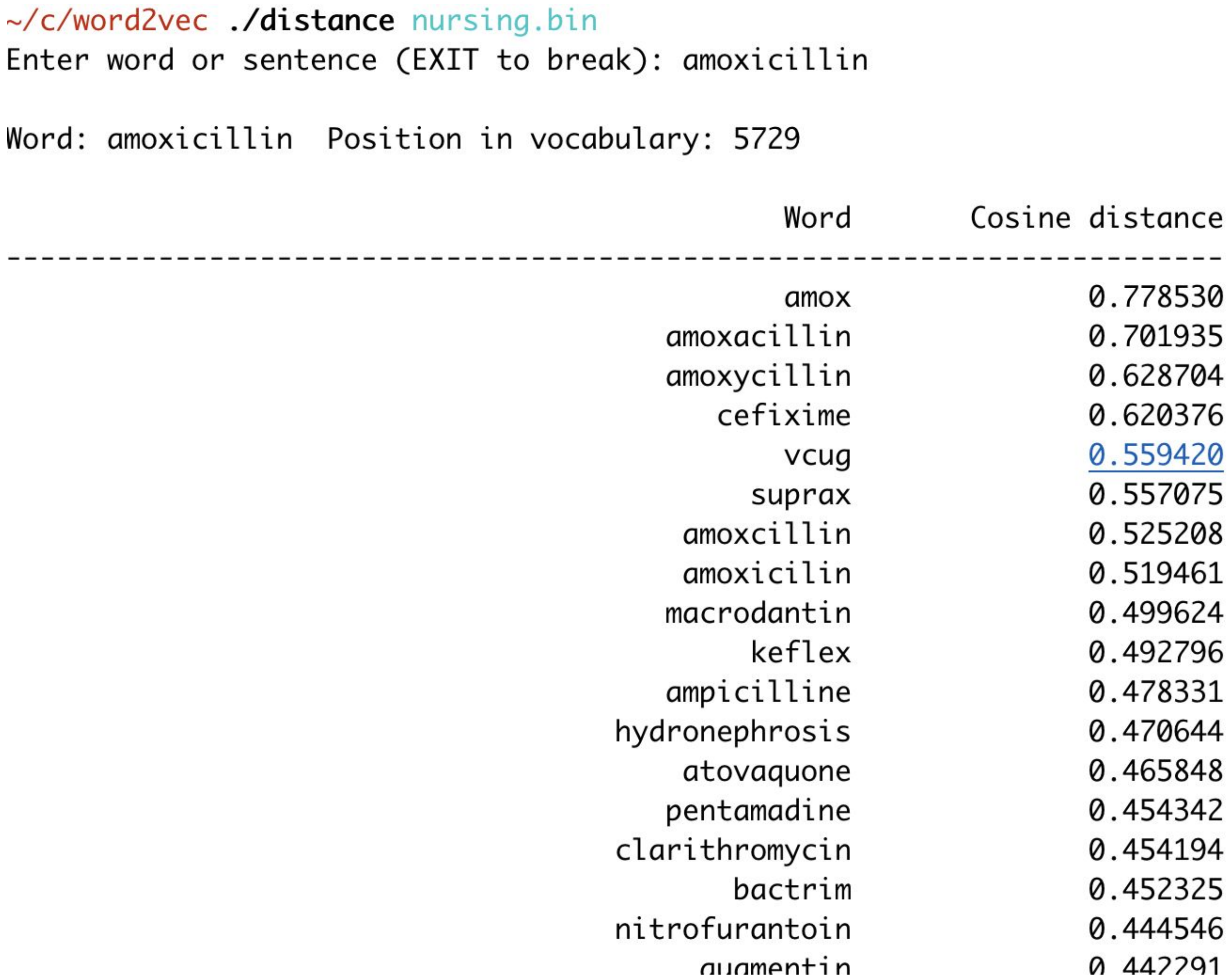
Man is to Woman what King is to ___?

Embedding(Queen) = Embedding(King) + (Embedding(Man)-Embedding(Woman))

"dancing" is to "danced" what "going" is to "went"

Embedding(went) = Embedding(going) + (Embedding(dancing)-Embedding(danced))

Evaluation based on utility in respect to an NLP application (e.g. text classification, NER, etc.)



```
~/c/word2vec ./distance nursing.bin
Enter word or sentence (EXIT to break): amoxicillin

Word: amoxicillin  Position in vocabulary: 5729

                                              Word       Cosine distance
------------------------------------------------------------------------
                                              amox              0.778530
                                       amoxacillin              0.701935
                                       amoxycillin              0.628704
                                          cefixime              0.620376
                                              vcug              0.559420
                                            suprax              0.557075
                                        amoxcillin              0.525208
                                        amoxicilin              0.519461
                                       macrodantin              0.499624
                                            keflex              0.492796
                                       ampicilline              0.478331
                                    hydronephrosis              0.470644
                                        atovaquone              0.465848
                                       pentamadine              0.454342
                                    clarithromycin              0.454194
                                           bactrim              0.452325
                                    nitrofurantoin              0.444546
                                         augmentin              0.442291
```

Closest embeddings to the word *amoxicillin* (a type of antibiotic) based on embeddings generated from 2 million clinical text corpus. Note abbreviations, misspellings, alternate antibiotic names and brand names.

Required reading:

- Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." Advances in neural information processing systems. 2013.
- Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). 2019.

Encouraged reading:

- Pennington, Jeffrey, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation." Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014. OR Andrew Ng's 11-minute talk on Glove (not sure, but you might need to create a Coursera.org account if you don't have one…)
- Peters, Matthew E., et al. "Deep contextualized word representations." Proceedings of NAACL-HLT. 2018.