Neural Networks for Natural Language Processing

Lecture 3 – Word Embeddings

Jun.-Prof. Sophie Fellenz

Plan today

- Word meaning
- Word2vec intro
- Word2vec, more details
- Glove vectors
- Evaluation of word vectors
- Sentence embeddings

How can we encode the features as input into a neural network?

- One-hot encodings

the	dog	sits	on	the	sofa
0	1	2	3	0	4
[1,0,0,0,0]	[0,1,0,0,0]	[0,0,1,0,0]	[0,0,0,1,0]	[1,0,0,0,0]	[0,0,0,0,1]

- Dense encodings

→ Word embeddings

3

Word meaning

4

How do we represent the meaning of a word?

- Meaning of word “***Meaning***” (according to Webster dictionary):
  - the idea that is represented by a **word**, phrase, etc.
  - the idea that a person wants to express by using **words**, signs, etc.
  - the idea that is expressed in a work of writing, art, etc.
  - Most common linguistic way of thinking of meaning:
    - signifier (symbol) $\Leftrightarrow$ signified (idea or thing)

How do we have usable meaning in a computer?

- Common solution: Use e.g. Wordnet
- Wordnet is a database of words and their synonym sets and hypernyms ("is a" relationships).

e.g. *synonym sets containing "good":*

noun: good (benefit)
noun: good, goodness (moral excellence or admirableness)
noun: commodity, trade good, good (articles of commerce)
adj: good
adj: full, good
adj: estimable, good, honorable, respectable
adj: beneficial, good (promoting or enhancing well-being) …
adj: good, just, upright (of moral excellence)
adverb: well, good
adverb: thoroughly, soundly, good

e.g. *hypernyms of "panda":*

```
[Synset('procyonid.n.01'),
Synset('carnivore.n.01'),
Synset('placental.n.01'),
Synset('mammal.n.01'),
Synset('vertebrate.n.01'),
Synset('chordate.n.01'),
Synset('animal.n.01'),
Synset('organism.n.01'),
Synset('living_thing.n.01'),
Synset('physical_entity.n.01')]
```

Problems with Wordnet

- Great as a resource but missing nuance
  - e.g. "proficient" is listed as a synonym for "good".
  - This is only correct in some contexts.

- Missing new meanings of words
  - e.g., wicked, badass, nifty, wizard, genius, ninja, bombast
  - Impossible to keep up-to-date!

- Subjective
  - Can't be used for tasks that require Natural Language Understanding for e.g. Question answering, text generation etc.
  - Solution : To learn directly from large amount of available text using word embeddings.

Discrete Word Representations

- Localist representation (traditional rule-based and statistical NLP) : Words are represented as discrete atomic symbols: hotel, conference, walk.
- Example of discrete word representation - one-hot vectors.
  - Consider a document of three words : dog, theatre, cat
  - Vocabulary size = 3 (all the distinct words in a document)
  - One-hot representations of these words are :

dog = [1, 0, 0]
theatre = [0, 1, 0]
cat = [0, 0, 1]

Dimension of the one-hot vector = Vocabulary size

- Problem: Dimensionality can be huge, hundreds of thousands of words

Problem with words as discrete symbols

- Example: in web search, if user searches for "Seattle motel", we
- would like to match documents containing "Seattle hotel".
- But:

  motel = [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
  hotel = [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]

- These two vectors are orthogonal.
- There is no natural notion of similarity for one-hot vectors!
- Solution:
  - Could rely on WordNet's list of synonyms to get similarity?
  - Instead: learn to encode similarity in the vectors themselves

Representing words by their context

- <u>Distributional semantics</u>: A word's meaning is given by the words that frequently appear close-by
  - *"You shall know a word by the company it keeps"* (J. R. Firth 1957: 11)
  - One of the most successful ideas of modern statistical NLP!
- 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*

…government debt problems turning into	banking	crises as happened in 2009…
…saying that Europe needs unified	banking	regulation to replace the hodgepodge…
…India has just given its	banking	system a shot in the arm…

These context words will represent ***banking***

Word Vectors

We will build a dense vector for each word, chosen so that it is similar to vectors of words that appear in similar contexts.
Also: the word vector should be good at predicting other words appearing in its context

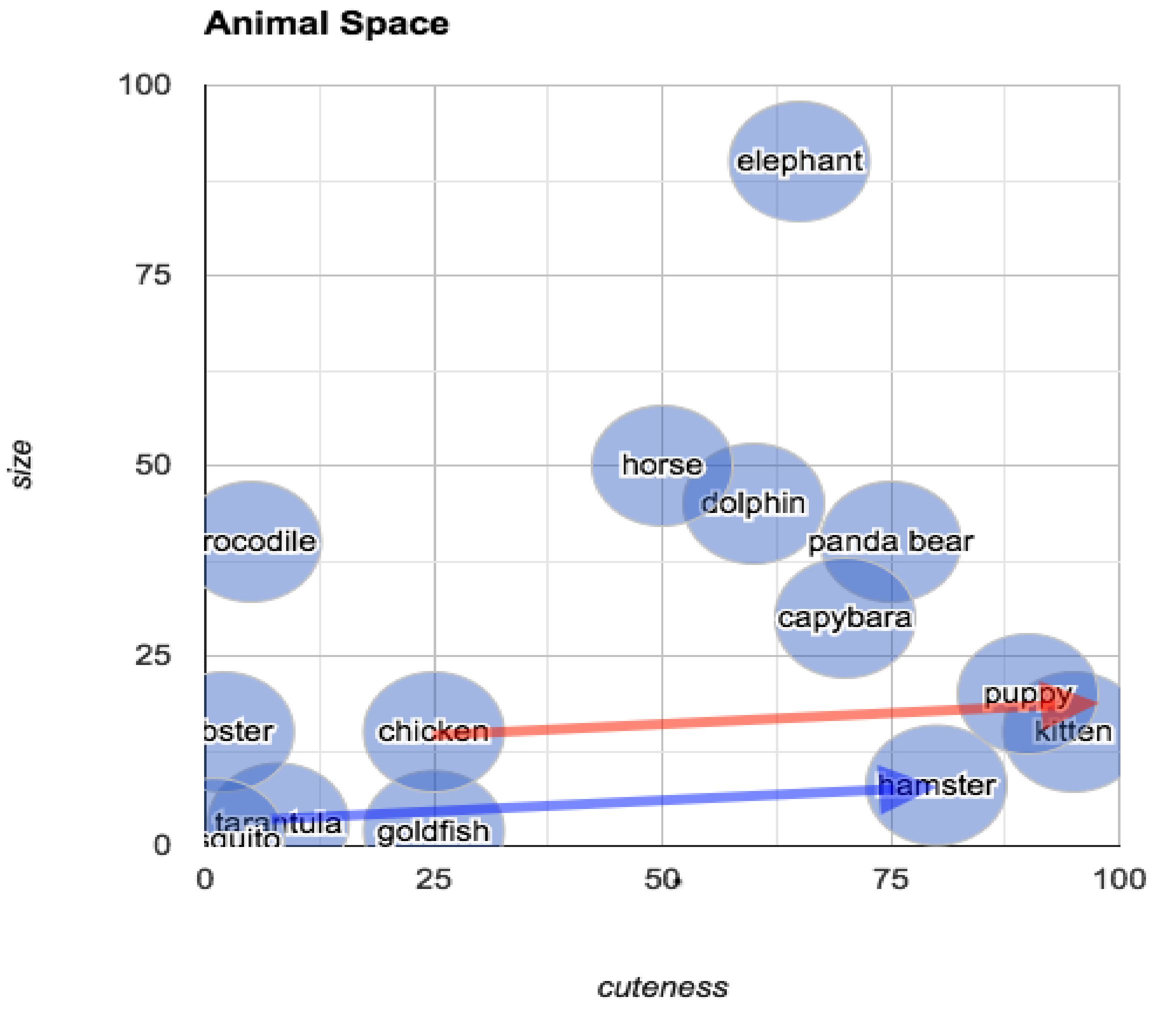
$$\text{linguistics} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

Note: word vectors are sometimes called word embeddings or word representations.

Word Embeddings

- Distributional semantics (as opposed to other theories of meaning of words)
- Distributed representation (as opposed to one-hot representation)
- **Definition**: word embeddings represent the mapping between words from a dictionary to vectors. (A dictionary is a collection of all the words in a document corpus)

Visualizing Word Vectors



[source: https://medium.com/analytics-vidhya/deep-dive-into-word2vec-7fcefa765c17]

Types of word vectors

1. Frequency based word vectors
    - Count vectors
    - TF – IDF
    - Co-occurrence matrix
2. Prediction based word vectors (Word2Vec)
    - CBOW (Continuous Bag of Words)
    - Skip – Gram Model

Count Vectors

- Idea: Represent a word as a vector of frequencies of it's occurrences in every document of a corpus.
- Overview:
  - Let C be a corpus of D documents $\{d_1, d_2, \dots, d_\mathrm{D}\}$ and N be the size of the dictionary.
  - Count vector matrix M is a matrix of size $D \times N$ where each column N represents a word vector.
  - E.g. **D1** : "Dog sat on the couch. Dog slept." **D2** : "Dog slept on the floor."
  - Count Vector Matrix :

	Dog	sat	on	the	couch	slept	floor
D1	2	1	1	1	1	1	0
D2	1	0	1	1	0	1	1

Count vectors

TF-IDF

- Example: Consider the term frequency count of the following two documents

	This	is	about	Dog	Cat
D1	1	1	2	4	0
D2	1	2	1	0	1

TF (This, D1) = 1/8

TF (This, D2) = 1/5

IDF (This) = log(2/2) = 0

IDF (Dog) = log(2/1) = 0.301

TF-IDF(This, D1) = (1/8) * (0) = 0

TF-IDF(This, D2) = (1/5) * (0) = 0

TF-IDF(Dog, D1) = (4/8) * (0.301) = 0.15

- TF-IDF heavily penalizes a common word “this” but assigns higher weight to “Dog”. This can be understood as “Dog” is an important word for D1.

Co-Occurrence Matrix

- Measures co-occurrence of words.

- Idea: Words having similar context appear together.

- E.g. Dogs and cats are pet animals. Here “Dogs” and “cats” tend to have similar context i.e. “pet animals”.

Co-occurrence Matrix

- **<u>Overview:</u> Given a corpus of documents, co-occurrence matrix calculates the frequency of co-occurrence of two terms say *w1* and *w2* in a given context window (number of terms that are preceding or succeeding a term *t*).**

- Word vectors are obtained by decomposing the co-occurrence matrix using dimensionality reduction techniques such as PCA, SVD etc.

- E.g. Quick brown fox jump over the lazy dog

- For word "Fox" and context window = 2, co-occurrence of the words in green box will be counted.

Co-occurrence Matrix

- Co-occurrence Matrix: (She is happy. She is well. She is not dull.)

	She	Is	Happy	Well	Not	dull
She	0	5	2	2	1	0
Is	5	0	2	2	1	1
Happy	2	2	0	0	0	0
Well	2	2	0	0	0	0
Not	1	1	0	0	0	1
dull	0	1	0	0	1	0

Problem: such a matrix is sparse and therefore inefficient for computation.

Co-occurrence vectors

- Simple count co-occurrence vectors
  - Vectors increase in size with vocabulary
  - Very high dimensional: require a lot of storage (though sparse)
  - Subsequent classification models have sparsity issues -> Models are less robust
- Low-dimensional vectors
  - Idea: store “most” of the important information in a fixed, small number of dimensions: a dense vector
  - Usually 25–1000 dimensions, similar to word2vec
  - How to reduce the dimensionality?
    - -> SVD
    - Running an SVD on raw counts doesn’t work well
    - Problem: function words (the, he, has) are too frequent -> syntax has too much impact.

Prediction based Embeddings

- Prediction based embeddings are obtained by predicting the probability of the occurrence of a word.
- Word2Vec (Mikolov et al. 2013) is a framework for ‘learning’ word vectors.
- Generally, it is a dense vector such that it is closer to the embeddings of other words that appear in similar context.
- E.g., the word embedding of dog will be closer to that of cat than of theatre.

Word2Vec

- Word2Vec (Mikolov et al. 2013) is a framework for learning word vectors
- <u>Idea:</u>
  - large corpus of text
  - every word in a fixed vocabulary is represented by a vector.
  - Let c be a center word and o be the context words and t is the position of the center word in the text.
  - Use the similarity of the word vectors for c and o to calculate the probability of o given c (or vice versa)
  - We adjust the word vectors (learn) to maximize this probability.

Word2Vec : Two variants

- Word2Vec (Mikolov et al. 2013) is a framework for learning word vectors

  1. Skip-grams

  Predict context ("outside") words (position independent) given center word

  2. Continuous bags of words (CBOW)

  Predict center word from (bag of) context words

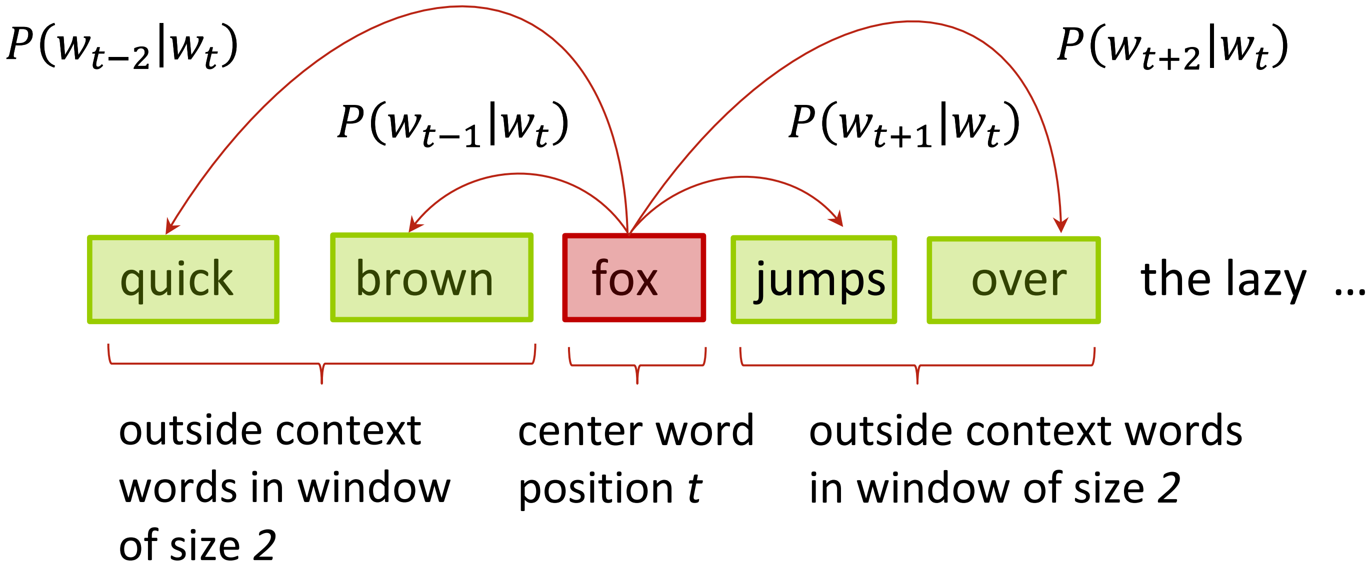
- Two training methods

  1. Hierarchical softmax

  2. Negative sampling

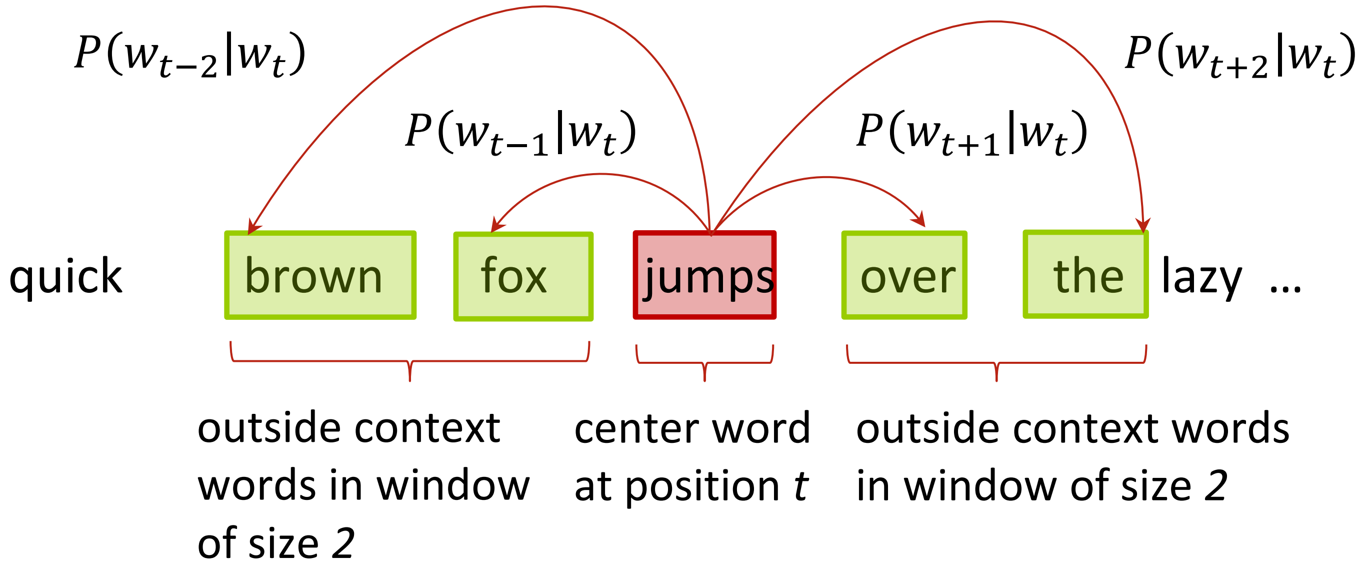
Word2Vec

Example windows and process for computing P(w_{t+j} | w_t)



Word2Vec

Example windows and process for computing P(w_{t+j} | w_t)



Word2Vec : Objective function

- For each position $t = 1, \dots, T$, predict context words within a window of fixed size m, given center word w_{j} .

$$L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \neq 0}} P\left(w_{t+j} \,\middle|\, w_t \,;\, \theta\right)$$

- Data log-likelihood:

$$J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log P\left(w_{t+j} \,\middle|\, w_t \,;\, \theta\right)$$

- The objective function $J(\theta)$ is the (average) negative log likelihood:

Minimizing objective function ⇔ Maximizing predictive accuracy

Word2Vec : Objective function

We want to minimize the objective function:

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T} \sum_{\substack{-m\leq j\leq m \\ j\neq 0}} \log P\left(w_{t+j} \mid w_t \,;\, \theta\right)$$

<u>Question:</u> How do we calculate $P(w_{t+1}|w_t;\theta)$?

<u>Answer:</u> We will use two vectors per word w

v_w when w is a center word

u_w when w is a context word

Then for a center word c and context word o: $P(o|c) = \frac{\exp(u_o^T v_c)}{\Sigma_{w\in V}\exp(u_w^T v_c)}$

Word2Vec : prediction function

Exponentiation makes anything positive

Dot product compares similarity of o and c.
$u^T v = u.v = \Sigma_{i=1}^{n} u_i v_i$
Larger dot product = larger probabilities

$$P(o|c) = \frac{\exp(\boxed{u_o^T v_c})}{\boxed{\Sigma_{w \in V} \exp(u_w^T v_c)}}$$

Normalize over entire vocabulary to give probability distribution.

$$P(deep|learning) = \frac{\exp(u_{deep}^T v_{learning})}{\sum_{w \in V} \exp(u_w^T v_{learning})}$$

Word2Vec : prediction function

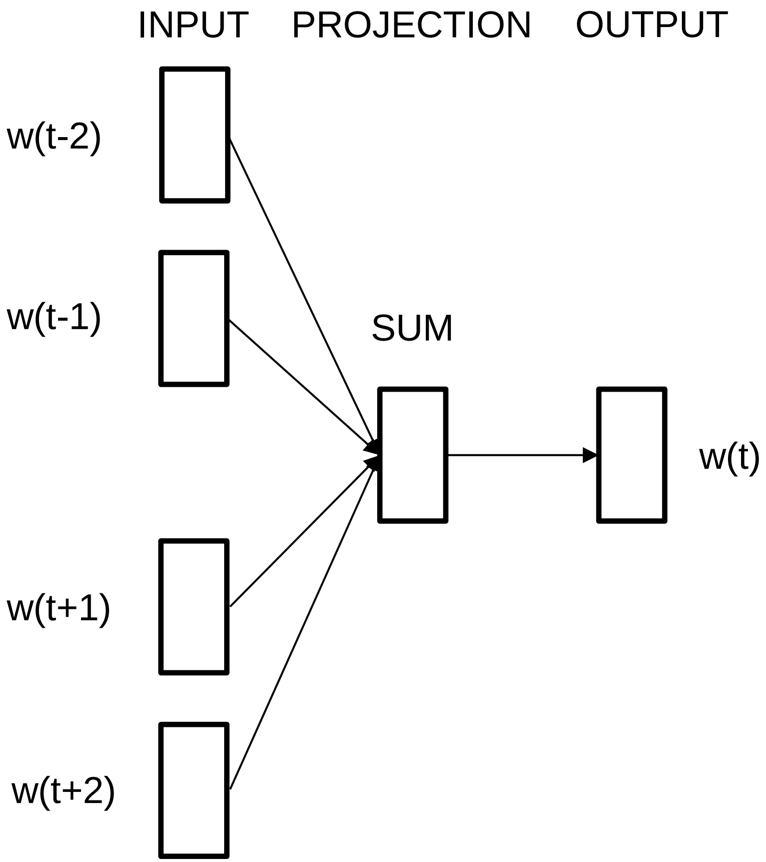
$$P(o|c) = \frac{\exp(u_o^T v_c)}{\Sigma_{w \in V} \exp(u_w^T v_c)}$$

This is an example of a softmax function $\mathbb{R}^n \rightarrow (0,1)^n$

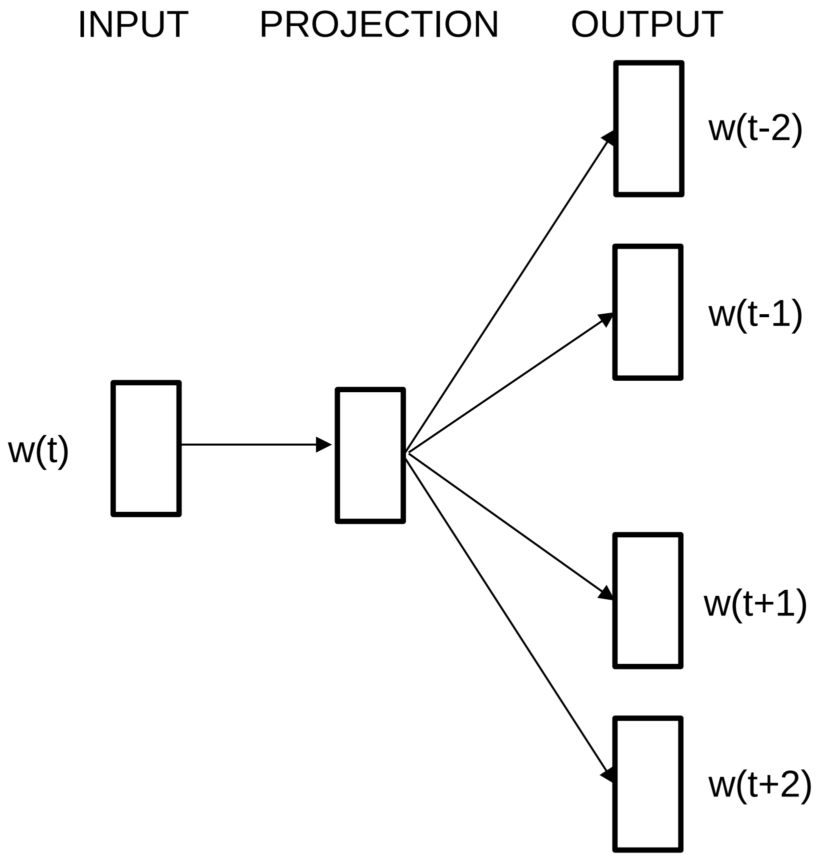
$$\boxed{\boldsymbol{softmax(x_i) = \frac{exp(x_i)}{\Sigma_{j=1}^{n} exp(x_j)} = p_i}}$$

- The softmax function maps arbitrary values x_i to a probability distribution p_i.
  - "max" because amplifies probability of largest x_i
  - "soft" because still assigns some probability to smaller x_i
  - Frequently used in Deep Learning

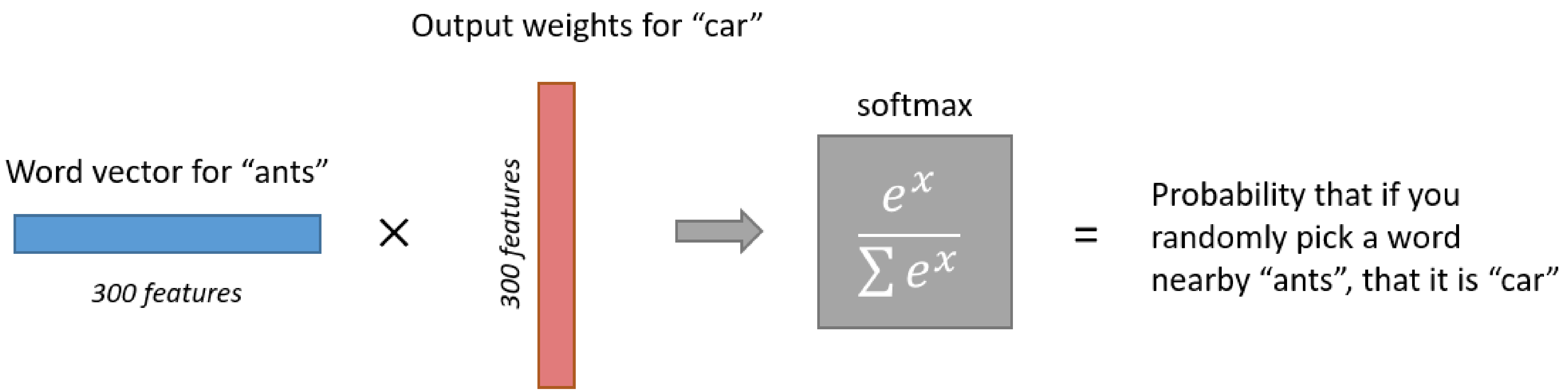
Word2Vec : CBOW



Word2Vec : Skip-Grams



Skipgram



Source: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

Skip gram gradient

Minimize: $J(\theta) = -\frac{1}{T}\sum_{t=1}^{T} \sum_{\substack{-m\leq j\leq m \\ j\neq 0}} \log P\left(w_{t+j} \,\middle|\, w_t \,;\, \theta\right)$

where $P(o|c) = \frac{\exp(u_o^T v_c)}{\Sigma_{w\in V} \exp(u_w^T v_c)}$

$$\frac{\partial}{\partial v_c} \log \frac{\exp(u_o^T v_c)}{\Sigma_{w\in V} \exp(u_w^T v_c)}$$

$$= \underbrace{\frac{\partial}{\partial v_c} \log \exp(u_o^T v_c)}_{\boxed{1}} - \underbrace{\frac{\partial}{\partial v_c} \log \sum_{w\in V} \exp(u_w^T v_c)}_{\boxed{2}}$$

Skip gram gradient

$\boxed{1}$ $$\frac{\partial}{\partial v_c} \log \exp(u_o^T v_c) = \frac{\partial}{\partial v_c} u_o^T v_c = u_o$$

$\boxed{2}$ $$\frac{\partial}{\partial v_c} \log \sum_{w \in V} \exp(u_w^T v_c) = \frac{1}{\sum_{w \in V} \exp(u_w^T v_c)} \frac{\partial}{\partial v_c} \sum_{x \in V} \exp(u_x^T v_c)$$ chain rule

$$= \frac{1}{\sum_{w \in V} \exp(u_w^T v_c)} \sum_{x \in V} \frac{\partial}{\partial v_c} \exp(u_x^T v_c)$$ move derivative inside sum

$$= \frac{1}{\sum_{w \in V} \exp(u_w^T v_c)} \sum_{x \in V} \exp(u_x^T v_c) \frac{\partial}{\partial v_c} u_x^T v_c$$ chain rule

$$= \frac{1}{\sum_{w \in V} \exp(u_w^T v_c)} \sum_{x \in V} \exp(u_x^T v_c)\, u_x$$

Skip gram gradient

$$\frac{\partial}{\partial v_c} \log \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)} = u_o - \frac{1}{\sum_{w \in V} \exp(u_w^T v_c)} \sum_{x \in V} \exp(u_x^T v_c)\, u_x$$

$$= u_o - \sum_{x \in V} \frac{\exp(u_x^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)} u_x$$

$$= u_o - \sum_{x \in V} p(x|c) u_x$$

=observed – expected

This is the gradient for the center vector parameters, similarly you can compute gradients for the output vector parameters

The skip-gram model with negative sampling

- The normalization term is computationally expensive

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

- Hence, in standard word2vec and HW2 you implement the skip-gram model with negative sampling

- Main idea: train binary logistic regressions for a true pair (center word and a word in its context window) versus several "noise" pairs (the center word paired with a random word)

The skip-gram model with negative sampling

From paper: "Distributed Representations of Words and Phrases and their Compositionality" (Mikolov et al. 2013)

- Overall objective function (to maximize): $J(\theta) = \frac{1}{T}\sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \ne 0}} J_{t,j}(\theta)$
- $J_{t,j}(\theta) = \log \sigma(u_o^T v_c) + \sum_{k \in \{sampled\ indices\}} \log \sigma(-u_k^T v_c)$

- The logistic/sigmoid function: $\sigma(x) = \frac{1}{1+e^{-x}}$
- We maximize the probability of two words co-occurring in first log and minimize probability of noise words in second part
- We take k negative samples (using word probabilities)
- Maximize probability that real outside word appears; minimize probability that random words appear around center word

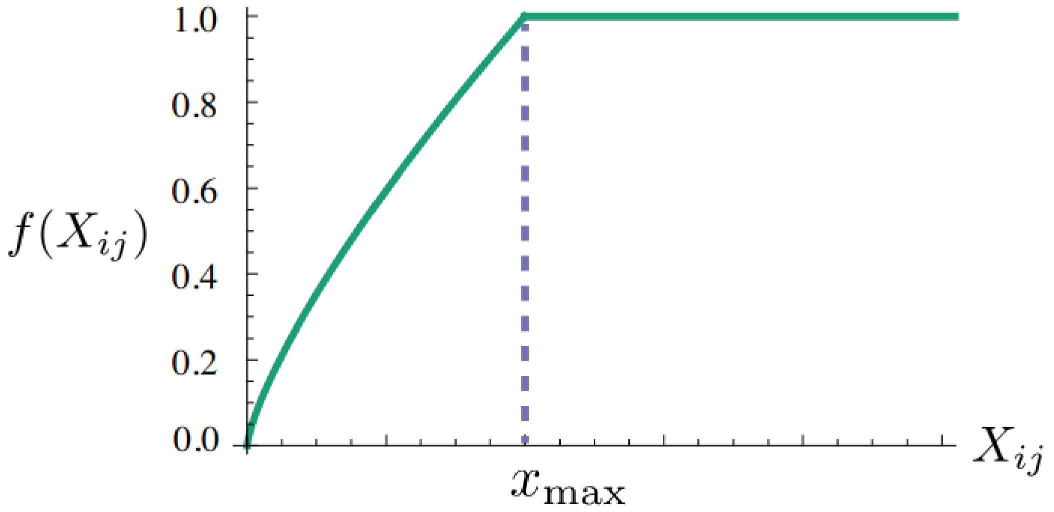
Count-based vs prediction-based WE

Count based	Prediction based
• Fast training • Efficient usage of statistics • Primarily used to capture word similarity • Disproportionate importance given to large counts	• Scales with corpus size • Inefficient usage of statistics • Generate improved performance on other tasks • Can capture complex patterns beyond word similarity

Combining the best of both worlds – GloVe [Pennington, Socher, and Manning, EMNLP 2014]

$$J(\theta) = \frac{1}{2}\sum_{\mathrm{i},j=1}^{W} f(X_{ij})\left(u_i^T v_j - \log X_{ij}\right)^2$$

- Fast training
- Scalable to large corpora
- Good at modeling rare words
- Good performance



Commmon choice: $\alpha = \frac{3}{4},\ x_{max} = 100$

$$f(x) = \begin{cases} \left(\frac{x}{x_{max}}\right)^{\alpha} & if\ x < x_{max} \\ 1 & otherwise \end{cases}$$

GloVe results

Nearest words to frog:

1. frogs
2. toad
3. litoria
4. leptodactylidae
5. rana
6. lizard
7. eleutherodactylus

litoria

leptodactylidae

rana

eleutherodactylus

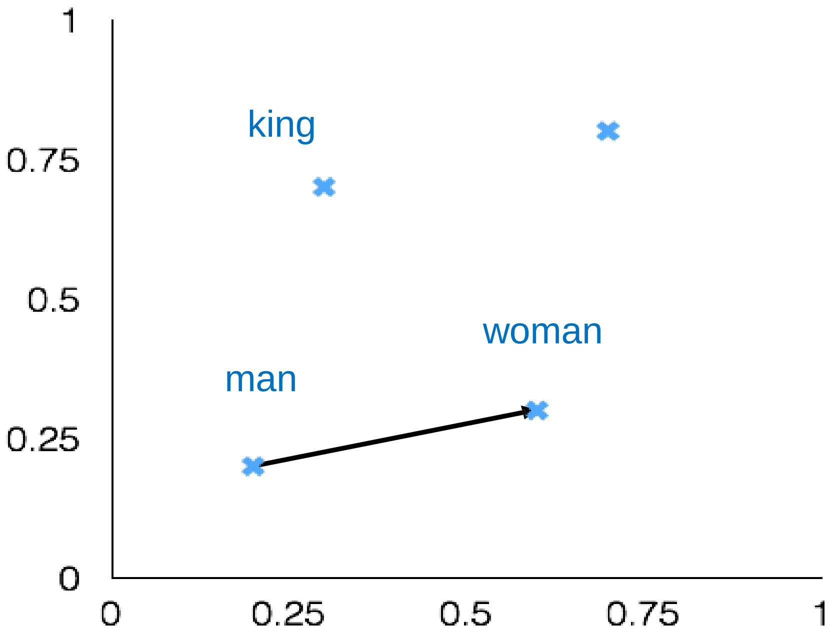
[source: https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1214/slides/cs224n-2021-lecture02-wordvecs2.pdf]

How to evaluate word vectors

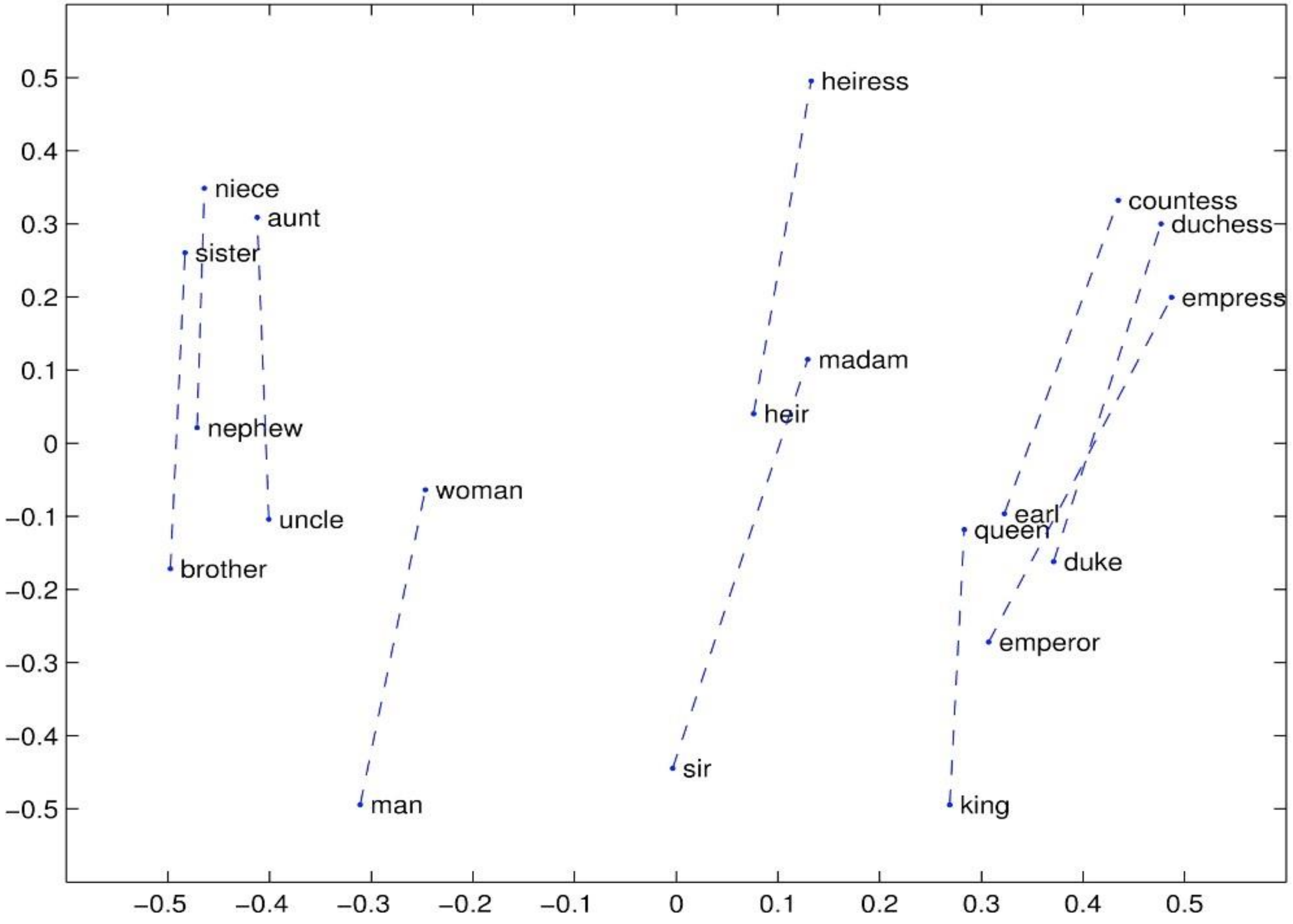
- Related to general evaluation in NLP: Intrinsic vs extrinsic
- Intrinsic:
  - Evaluation on a specific/intermediate subtask
  - Fast to compute
    - Helps to understand that system
    - Not clear if really helpful unless correlation to real task is established
- Extrinsic:
  - Evaluation on a real task
    - Can take a long time to compute accuracy
    - Unclear if the subsystem is the problem or its interaction or other subsystems
    - If replacing one subsystem with another improves accuracy it is winning

Intrinsic word vector evaluation

- Word vector analogies
- a : b = c : ? $\longrightarrow$ $d = \arg\max_i \frac{(x_b - x_a + x_c)^T x_i}{||x_b - x_a + x_c||}$
- man : woman = king : ?
- Evaluate word vectors by how well their cosine distance is after addition captures intuitive semantic and syntactic analogy questions
- Discarding the input words from the search!
- Problem: What if the information is there but not linear?

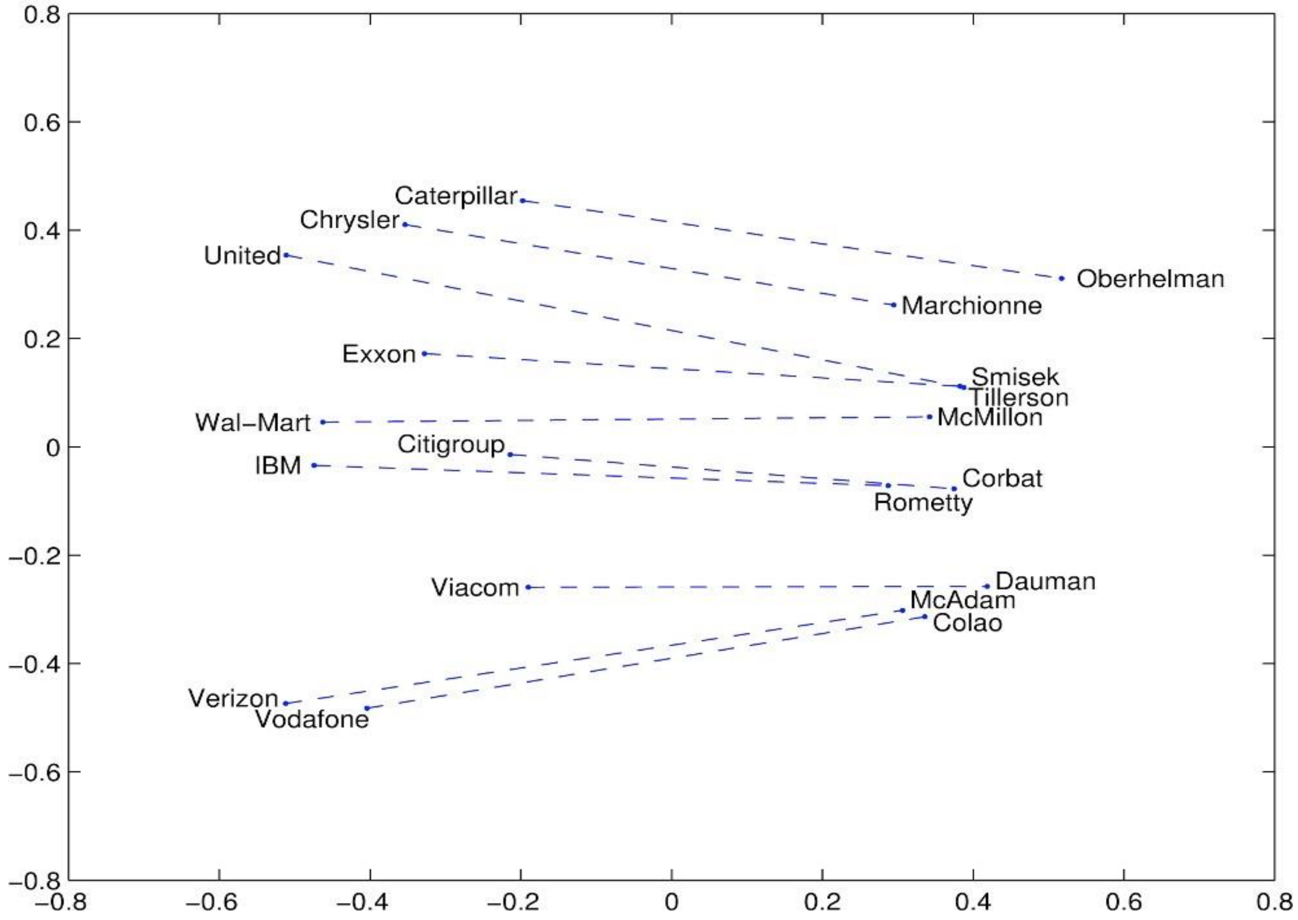


GloVe Visualizations



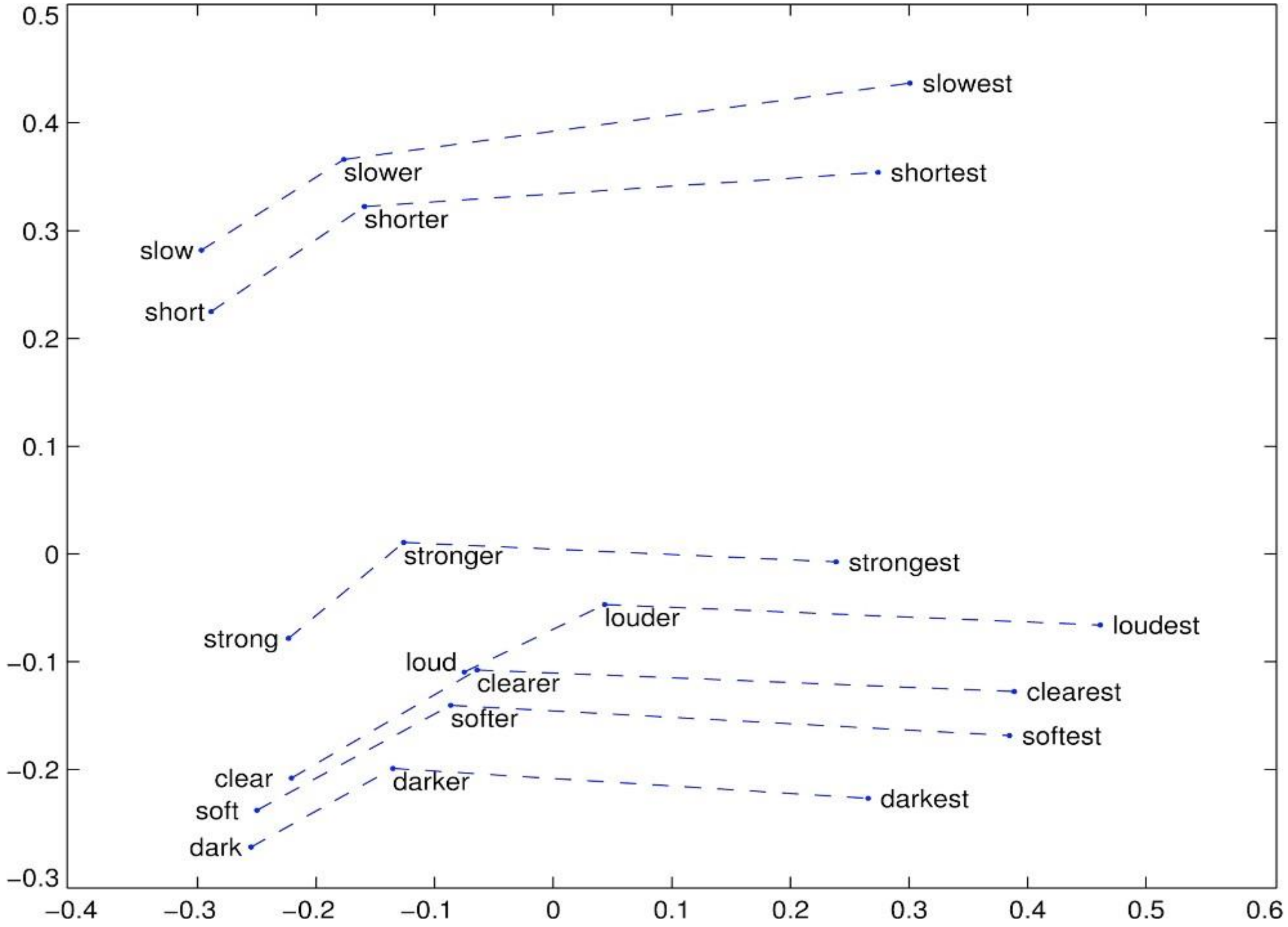
[source: https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1214/slides/cs224n-2021-lecture02-wordvecs2.pdf]

GloVe Visualizations: Company - CEOs



[source: https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1214/slides/cs224n-2021-lecture02-wordvecs2.pdf]

GloVe Visualizations: Comparatives - Superlatives



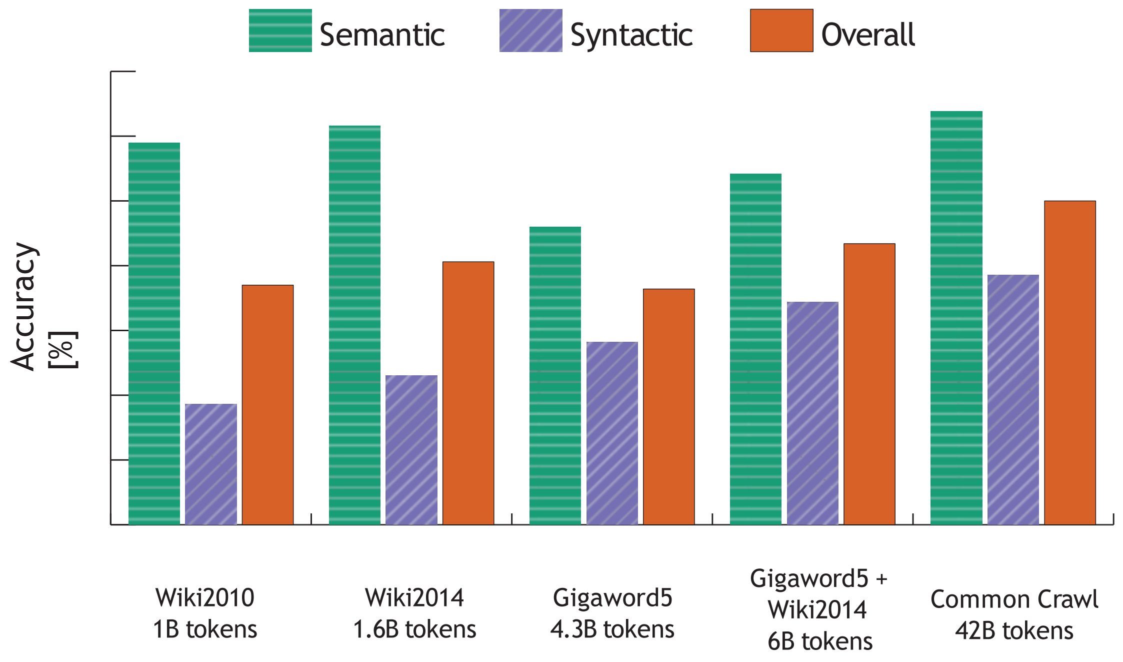
[source: https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1214/slides/cs224n-2021-lecture02-wordvecs2.pdf]

Some fun word2vec analogies

Expression	Nearest token
Paris – France + Italy	Rome
Bigger – big + cold	colder
Sushi – Japan + Germany	bratwurst
Cu – copper + gold	Au
Windows – Microsoft + Google	Android

Analogy evaluation and hyperparameters

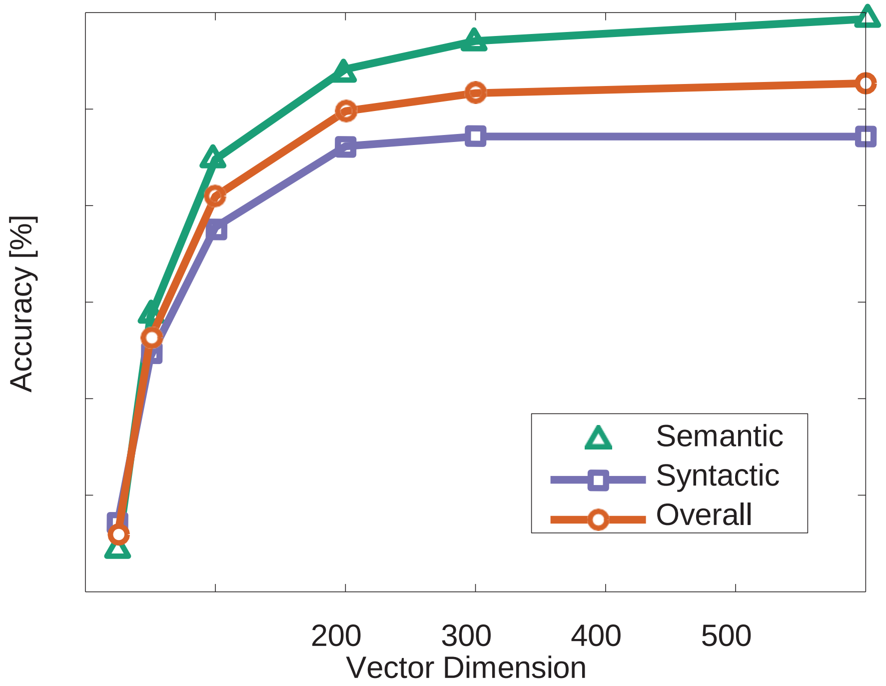
- More data helps
- Wikipedia is better than news text!



[source: https://aclanthology.org/D14-1162.pdf]

Analogy evaluation and hyperparameters

- Dimensionality
- Good dimension is ~300



[source: https://aclanthology.org/D14-1162.pdf]

Another intrinsic word vector evaluation

RPTU

- Word vector distances and their correlation with human judgments
- Example dataset: WordSim353
  http://alfonseca.org/eng/research/wordsim353.html

Word 1	Word 2	Human (mean)
tiger	cat	7.35
tiger	tiger	10
book	paper	7.46
computer	internet	7.58
plane	car	5.77
professor	doctor	6.62
stock	phone	1.62
stock	CD	1.31
stock	jaguar	0.92

Extrinsic word vector evaluation

- Extrinsic evaluation of word vectors: All subsequent NLP tasks in this class. More examples soon.

- One example where good word vectors should help directly: **named entity recognition**: identifying references to a person, organization or location

Word senses and word sense ambiguity

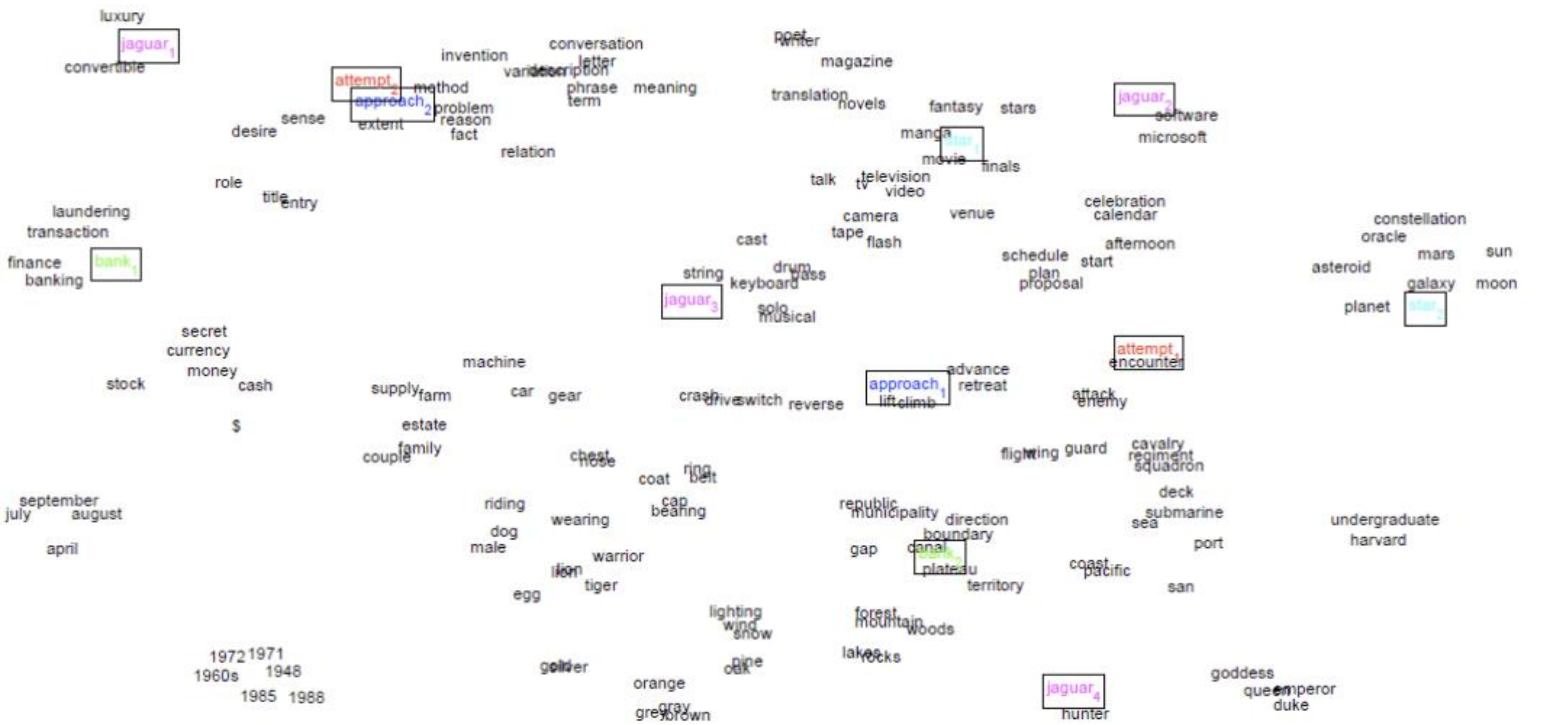
- Most words have lots of meanings!
- Especially common words
- Especially words that have existed for a long time
- Example: pike
- Does one vector capture all these meanings or do we have a mess?

pike

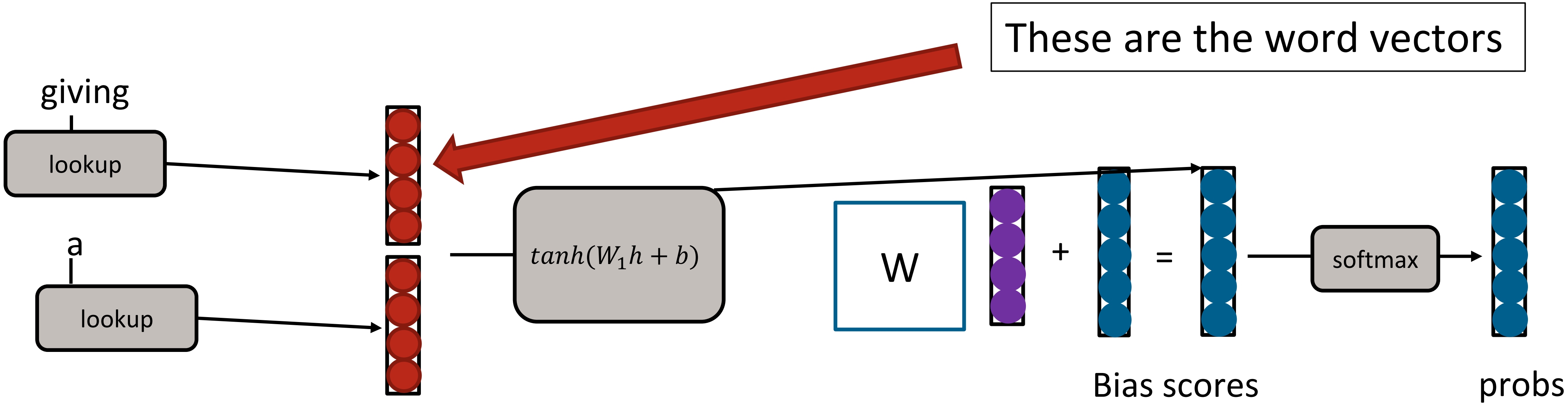
- A sharp point or staff
- A type of elongated fish
- A railroad line or system
- A type of road
- The future (coming down the pike)
- A type of body position (as in diving)
- To kill or pierce with a pike
- To make one’s way (pike along)
- In Australian English, pike means to pull out from doing something: I reckon he could
- have climbed that cliff, but he piked!

Improving Word Representations Via Global Context And Multiple Word Prototypes (Huang et al. 2012)

Idea: Cluster word windows around words, retrain with each word assigned to multiple different clusters bank1 , bank2 , etc.



How are word vectors used in neural networks?



- Pretraining with e.g. word2vec
- Fine tuning on the classification task (or not)

Sentence embeddings

<u>Sentence embeddings:</u> Sentences are mapped to numerical vectors (similar to word embeddings).

<u>Idea:</u> Semantically similar sentences must have similar embeddings.

In simpler words, similar sentences with different words must have similar sentence embeddings.

Example :

1. Water is neither acidic nor basic.
2. Water is neutral.

Sentence embeddings - techniques

1. Smooth Inverse Frequency (Arora S. et. al.):
   - Weighted average of word vectors as sentence representation.
   - Performs well on classification and entailment tasks.

2. Bag of random embedding projections (Wieting & Kiela, 2019):
   - Projects word embeddings to higher dimensional space in order to make them linearly separable and pooling the projections.
   - Performs well on classification tasks.
   - Ignores the order of words.

3. Contextual sentence embeddings:
   - Obtaining intermediate representation of a sentence using a BERT model.

Applications of sentence embeddings

- Information retrieval – To compare the meaning of the text snippets
- Machine Translation – Uses sentence embeddings as “intermediate language” between source and target language.
- Classification
- Tagging

Summary

- Word vectors :
    - count-based vectors
    - word2vec
    - glove
- Word vectors allow to reduce dimensionality of the feature space
- Word vectors encode the meaning of the words
- Word vectors are used as input features in neural networks

Next lecture

Intro to Neural Networks

References

- Efficient Estimation of Word Representations in Vector Space (Word2Vec paper)
- Distributed Representations of Words and Phrases and their Compositionality (negative sampling)
- GloVe: Global Vectors for Word Representation (GloVe paper)

Acknowledgements

- Stanford Course “Natural Language Processing with Deep Learning”
- Asmita Bhat