CSC 412/2506:
Probabilistic Learning and Reasoning

Week 10 - 1/2 Embeddings I

Michal Malyska

University of Toronto

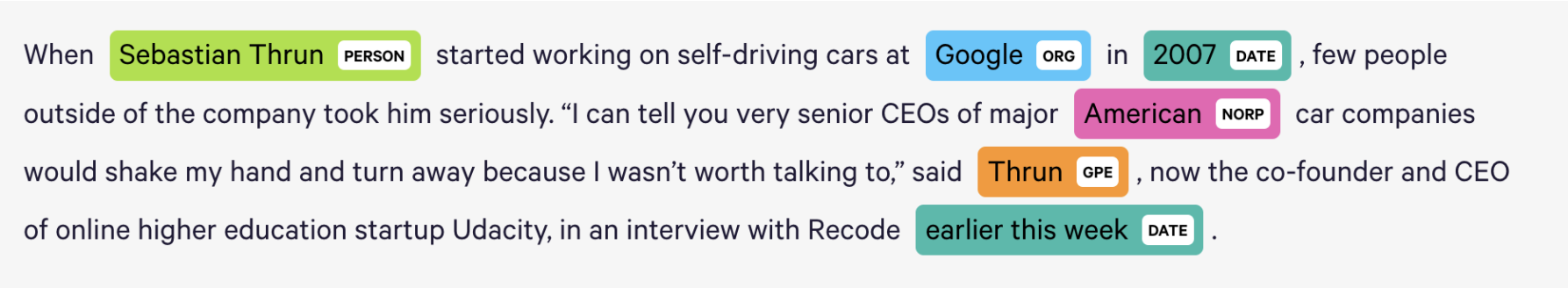
Overview

- Embeddings
  - Word2Vec
    - Skip Gram
    - CBoW
- Negative Sampling

Word Embeddings

Previously, we covered document embeddings that had different properties (indicators, counts, relative frequencies).

- What if we need to model text at a word / token level?



- What can we use from the previous lecture?
- What properties would it have?
- What properties would it be missing?

[1] from SpaCy: https://explosion.ai/demos/displacy-ent

Word2Vec

We start with a fairly strong assumption: "Words that have similar meanings will occur in simmilar contexts" Based on that we define a **context** of size **k** of token $w_{i,j}$ [1]as a set of tokens:

$$\text{context}(w_{i,j}) = \{w_{i,j-k}, w_{i,j-(k-1)}, \ldots, w_{i,j-1}, w_{i,j+1}, \ldots w_{i,j+k}\}$$

Then given a set of datapoints $x_{i=1:N}$ consisting of R_i words each, and a vocabulary $V_{r=1:M}$ we define an unsupervised learning task of predicting what words occur in the context of each word in the vocabulary. More formally, given a sequence of training words $w_1, \ldots, w_T$ we want to maximize the average log probability:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{j\in\text{context}(t)} logp(w_j|w_t)$$

[1]This is also called a **skip-gram**

Skip Gram Continued

The basic formulation of $p(x|w)$ uses the softmax function:

$$p(x|w) = \frac{\exp\left((u(w)^T(v(x)\right)}{\sum_{m=1}^{M} \exp\left((u(w))^T(v(V_m))\right)}$$

where $u(w)$ is the "word" and $v(w)$ is the "context" representation of word w. What are those representations?

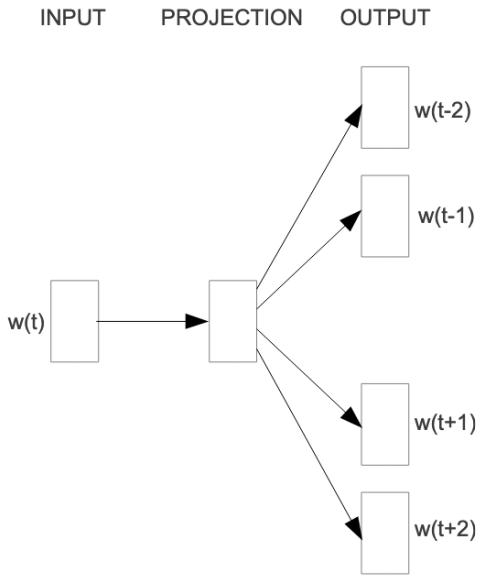
- In our particular case we will take u and v to be simple linear projections of the **one-hot** (binary BoW) encoding of the word, and context respectively.

$$u(w) = bBoW(w)U^T$$

The matrix U will be of size $R \times e$ where **e** is the embedding dimension, which is a hyperparameter you chose.

Skip Gram Continued

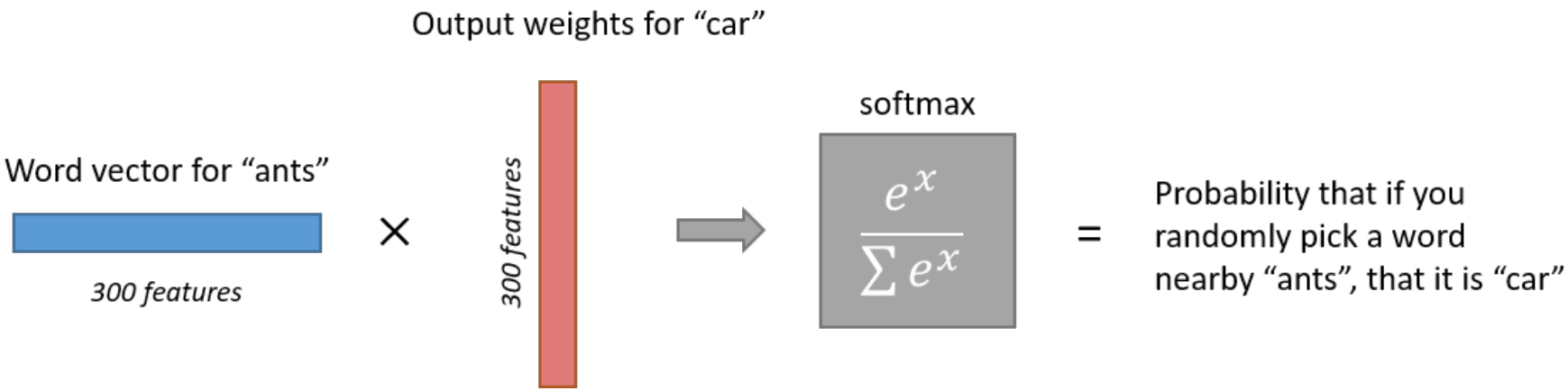
Similarily we define $v(w)$ to be a linear projection that back from $u(w)$ to predict each word in the context.



[1]"Efficient Estimation of Word Representations in Vector Space", Mikolov et al

Skip Gram Vis

How to estimate $p(\text{"car"}|\text{"ants"})$?



[1] Image from: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

Word2Vec

The training process is then as follows:

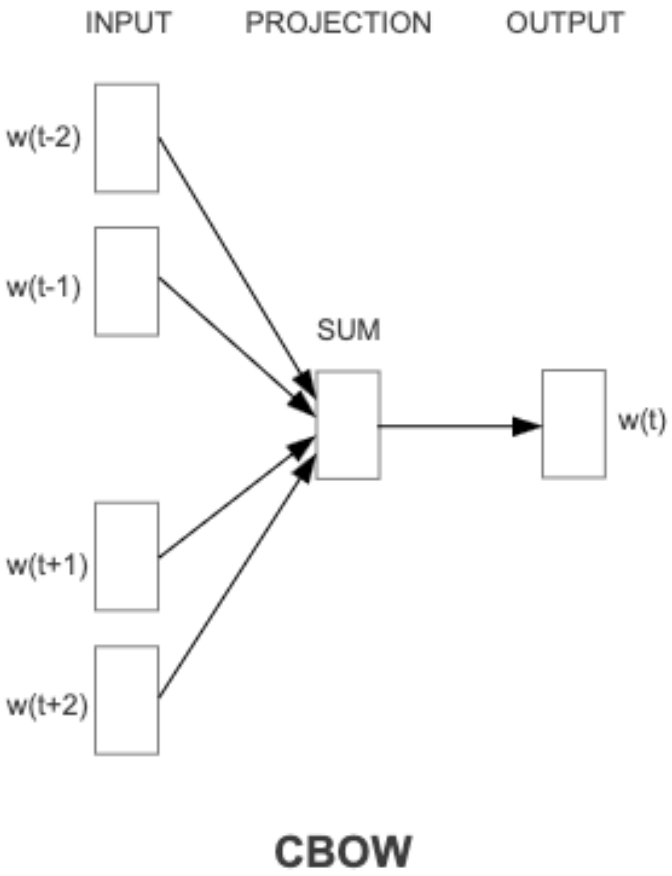
- Initialize the two matrices
- Sample pairs of words (w_i, w_j) and compute the objective
- Gradient Descent based on the objective.
- After convergence keep only the matrix U
- Embedding of word at vocabulary index i is just the i-th row of U

$$[0 \quad 0 \quad 0 \quad \boxed{1} \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ \boxed{10} & \boxed{12} & \boxed{19} \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

[1] Image from:
http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

CBoW

Instead of predicting all context words from the word, we can predict the word from it's context. This objective is called the **Continuous Bag of Words**



Some Cool properties of W2V

Let $h(x)$ be the Word2Vec embedding of the word x. We can perform some vector algebra to find that:

- $$h(\text{"Athens"}) - h(\text{"Greece"}) + h(\text{"Germany"}) \approx h(\text{"Berlin"})$$
- $$h(\text{"Mice"}) - h(\text{"Mouse"}) + h(\text{"Dollar"}) \approx h(\text{"Dollars"})$$
- In the original paper they propose a set of 14 categories and evaluate accuracy on these kinds of "algebraic" operations to find an accuracy of $\sim 55 - 60\%$

You can just download the original Word2Vec embeddings and play around!

Word2Vec notes

- In practice this training procedure is not feasible - we would have to compute softmax over the entire vocabulary at every step.
- There are a lot of tricks and improvements over the years - really worth reading the original paper.
- What are the properties of these embeddings?
- What is missing?

Negative Sampling

Instead of evaluating the softmax with a target of 1 for one dimension and 0 for all others, only take a small sample of negative examples in the vocabulary.

- For large datasets 2-5 negative examples works well
- For smaller datasets 5-20
- How to sample?

The original authors went through a number of options and experimentally settled on:

$$p(w_i) = \frac{C_{w_i}^{3/4}}{Z}$$

Token Classification and Embeddings continued

- To classify tokens, we can just take the Word2Vec embedding of each token as an input to e.g. Linear Regression / Multinomial Naive Bayes, and estimating probabilities that the token belongs to a certain category.
- We can combine Word2Vec representations into document level representations.
- We can combine Different embedding methods! Nothing is stopping you from taking TF-IDF vector of a document and "stacking" the average of all Word2Vec vectors in the document.

Generative Tasks

- What if our output should also be in the form of text?
- Re-frame the "context" to only feature words before the input
- Train in the unsupervised setting of CBoW, with the modified context.
- Idea: Sample the next word, conditional on the previous k based on the CBoW softmax.
- What kind of model is that?

Similarity

- What does it mean for 2 words to be similar?
- What does it mean for 2 datapoints to be similar?
- The most common way to measure similarity in NLP is via the cosine similarity

We define **cosine similarity** between two vectors to be:

$$\text{cosine sim}(x, y) = \frac{x \cdot y}{||x|| \cdot ||y||} = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2}\sqrt{\sum_i y_i^2}}$$

This is especially convenient for binary BoW.