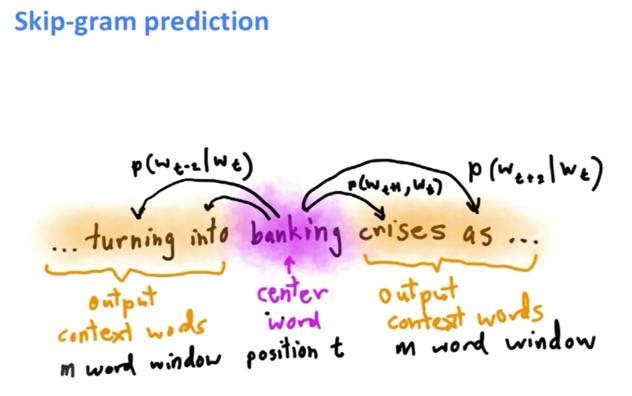
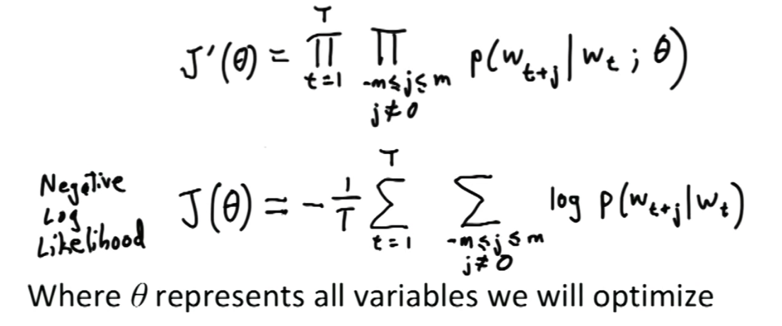
Lecture 2 | Word Vector Representation: word2vec

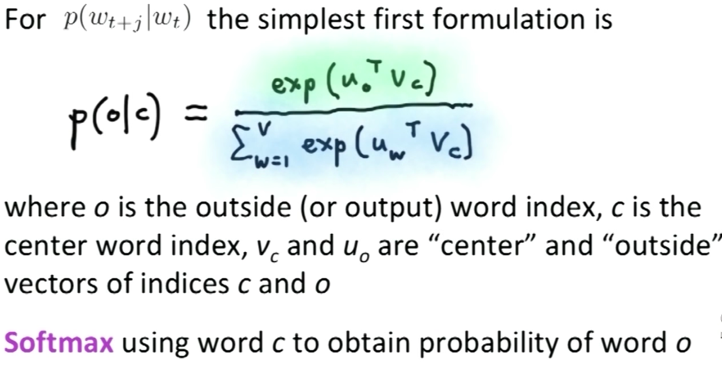
* WordNet (**from nltk.corpus import wordnet as wn**)
  + Great as a resource but missing nuances
  + Impossible to keep up-to-date, missing new words
  + Subjective in terms of what’s in wordnet
  + Hard to compute accurate word similarity
* Vast majority of rule-based and statistical NLP work regards words as atomic symbol. In vector space terms, this is a vector with one 1 and a lot of zeros (one-hot vector) – an example of a localist representation
  + **It’s a problem because it doesn’t give any inherent notion of relationships between words**
* **Distributional similarity** – You can get a lot of value for representing the meaning of a word by looking at the context in which it appears and doing something with those contexts
  + Therefore we will build a **dense vector** for each word type, chosen so that it is good at predicting other words appearing in its context
* NN word embeddings:
  + Define a model that aims to predict between a center word w(t) and context words in terms of word vectors:
    - p(context|w(t)) = ….
  + Which has a loss function
    - J = 1 – p(w(-t)|w(t))
  + We look at many positions t in a big language corpus and keep adjustinmg the vector representations of words to minimise J loss function
* Word2Vec
  + Use the theory of meaning, predict between every word and its context words
  + 2 algorithms:
    - **Skip-grams (SG)**
      * Predict context words given target (position independent)
    - Continuous Bag of Words (CBOW)
      * Predict target word from bag-of-words context
  + 2 moderately efficient training methods:
    - Hierarchical softmax
    - Negative sampling
* **Skip-grams (SG)**

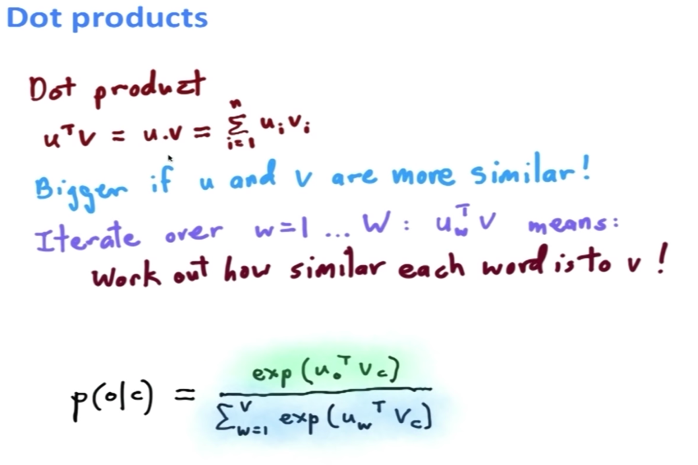
****

* + The idea for the skip-gram model is that for each estimation step, we are taking one word as the center word (banking) and try to predict words in its context out to some window size. Therefore the model is going to define a probability distribution, that is the probability of a word appearing in the context given the center word
  + For each word t = 1 …. T, predict surrounding words in a window of radius **m** of every word. Our objective function is to maximise the probability of any context word given the current center word:



* + **Theta** is the vector representation of the words and it’s the only parameters in this model
  + The window size **m** is one of the several hyperparameters
  + How do we use these word vectors to minimise the negative log likelihood?





* + To train the model, we need to compute all vector gradients
    - Define the set of all parameters in a model in terms of one long vector and do optimisation to change those parameters so as to maximise the objective function of our model
    - **Every word has two vectors**, one vector when the word is a center word and one vector when the word is a context word
    - Use gradient descent to optimise the parameters (back-propagation)
      * The derivative was derived in the lecture (if you want to see the derivation)
* When measure the similarity, you need to make sure you scale to the length of the vector instead of solely using the inner dot product – **Cosine difference**. This is to prevent inconsistency in scale measures which will result in inner dot product being inaccurate in measuring similarity
* Use stochastic gradient descent when face with large corpus so that you can update your parameters more frequently instead of waiting for it to go through the whole batches