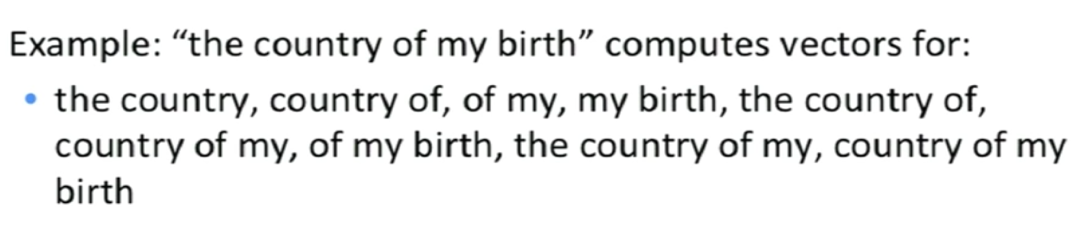
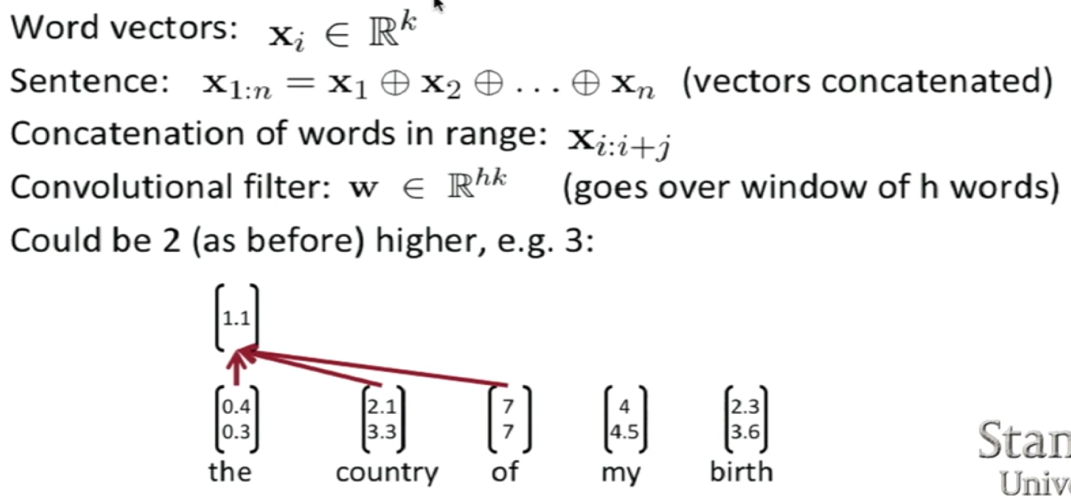
Lecture 13 | Convolutional Neural Networks (CNN)

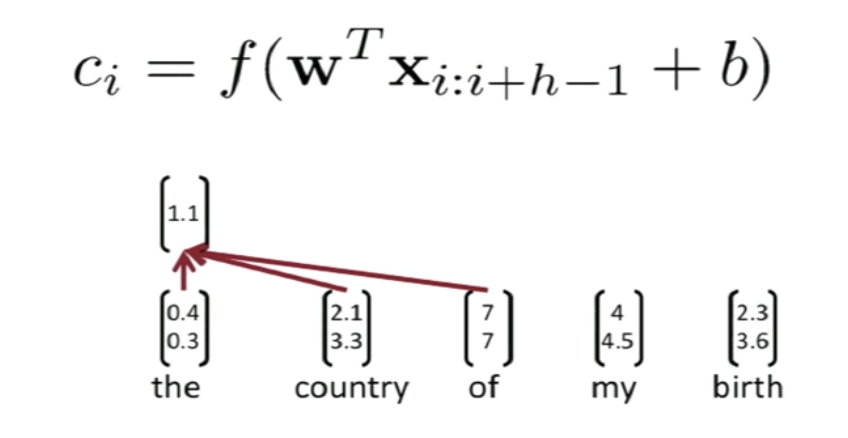
* Recurrent neural nets cannot capture phrases without prefix context (often capture too much of last words in final vector), therefore it’s difficult to capture phrases in isolation
* Sometimes you just want to identify that certain phrases exist in the overall document
* CNN compute phrase vectors for every possible phrase in a sentence, regardless of whether the phrase is grammatically correct. Then you group them afterwards

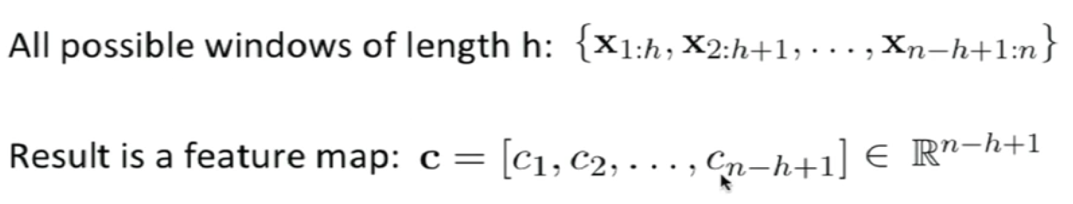


* CNN with NLP
  + Single layer CNN, a simple variant using one convolutional layer and pooling



* + The filter is a vector, size of filter is a hyperparameter
  + To compute feature for CNN layer:





* + Pooling layer (max-pooling)
    - Capture the most important activation from feature map!
    - But we want more features
    - Therefore, we can use multiple filter weights **w** and different window sizes **h** and because of max pooling, length of c is irrelevant, meaning we can have some filters looking at unigrams, bigrams, trigrams etc
  + Multi-channel idea
    - Initialise with pre-trained word vectors
    - Start with two copies
      * Backprop into only one set, keeping the other ‘static’
      * Both channels are added to c\_i before max-pooling
  + To obtain final feature vector, concatenate all the max pool filtered feature
  + Then a simple final softmax layer
  + CNNs + Highway Network over characters can extract rich semantic and structural information
* Dropout
  + Randomly mask/dropout/set to 0 some of the feature weights z
  + Firstly, you create masking vector r of Bernoulli random variables with probability p being 1. Therefore, you delete features during training to prevent co-adaptation (overfitting)
  + Therefore, at training time, gradients are backpropagated only through those elements of z vector for which r\_i = 1
  + At test time, there is no dropout, so feature vectors z is larger. Hence, we scale final vector by Bernoulli probability p
* Find hyperparameters based on dev set! During training, keep checking performance on dev set and pick highest accuracy weights for final evaluation
* Tree-LSTMs obtain better performance on sentence datasets