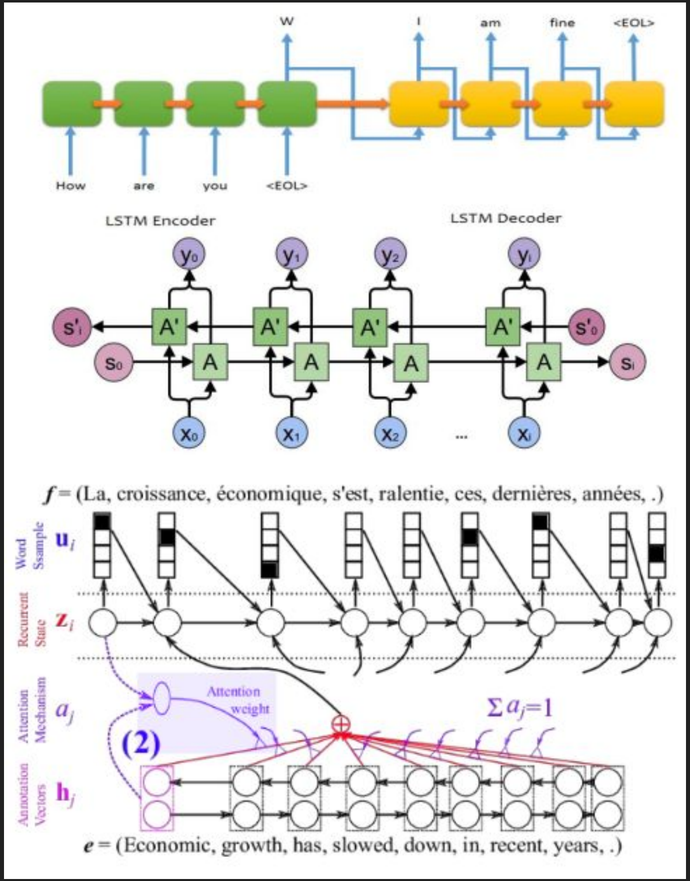
Text Analysis with Convolutional Neural Nets

What is currently used?

The "go-to" for Text analysis right now is Recurrent Neural Networks:

- Slow (er than CNNs)
- Cannot parallelize RNNs
- Reads the words sequentially
- Complicated (if you want the whole sentence to be considered in computation)
- Long term and backward dependencies imply almost double the computation
- Needs streamed sentences



Why use Convolutional Neural Nets for Text?

- Faster than RNNs
- CNNs have a larger receptive field (area of “vision” each neuron can see)
- Have a lot of knowledge and tools from the field
- Local invariance is useful for applications like Sentiment Analysis
- Compositionality of structures is inherent in grammar.
- And because why not?

How do we convert some text into a 2D format?

We use some word embedding systems such as

- One-Hot Representations
- BoW : Bag of Words
- Word2Vec
- GloVe: Global Vectors

Word2Vec Quickly Explained

Properties of Word2Vec:

- Keep context of surrounding words
- Find common phrases in a vocabulary
- Creating a more compact representation of words and phrases.
- Embedding is created using **a simple feed forward neural net**

Achievements of word2vec:

- Encoding semantic properties of words into the representations
- Allowing for a mapping of relations in the corpus via a vector space of high dimensions.

Word2Vec Advantages

Allows for algebraic operations through the vector representations

w2v(King) = w2v(Queen) - w2v(woman) + w2v(man)

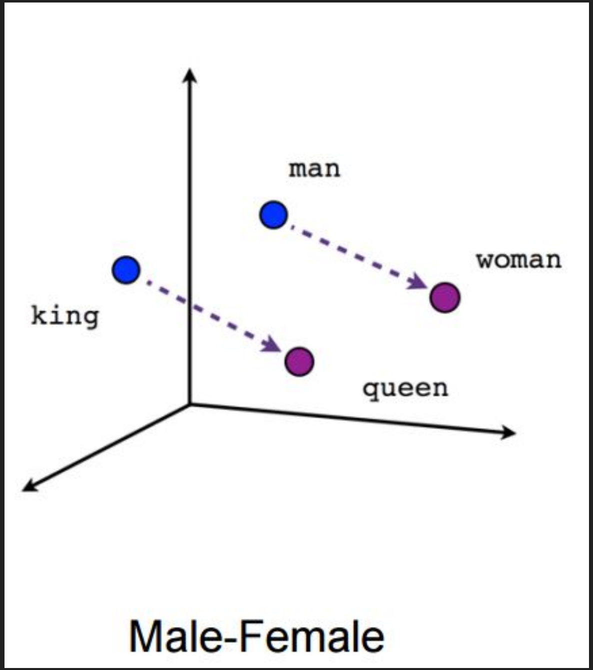
Is valid.

Applications of Word2Vec:

- Semantic Relations
- Word Suggestions
- Language Translations

Aim/Applications of a Skip-Gram word2vec model:

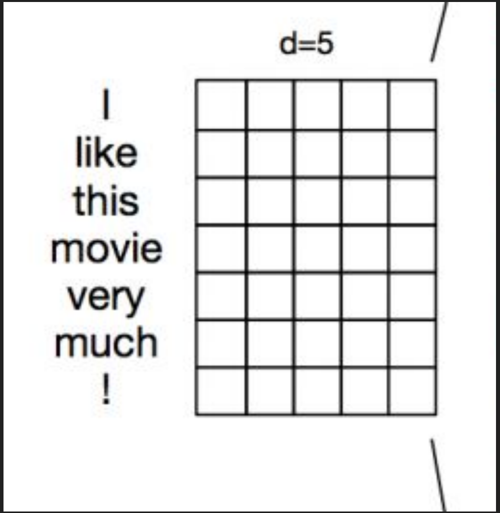
Given Context, return the missing word!



Embedding to 2D (word2vec)

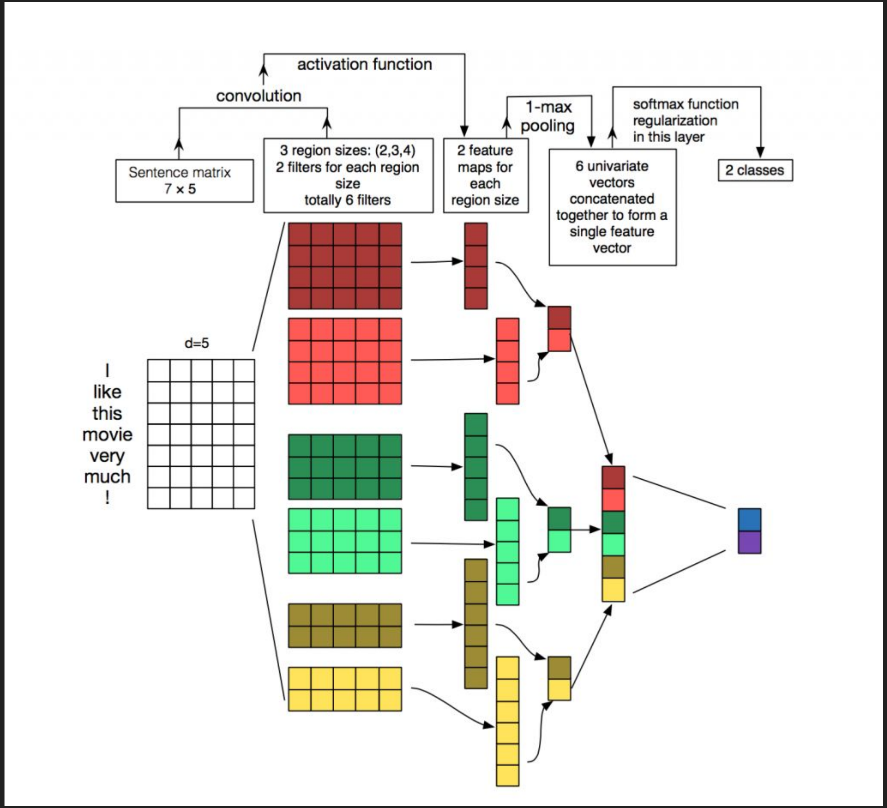
How to turn a sentence into 2D

1. Take the vector representation of each word in the sentence (d)
2. Take all the words in the sentence (n)
3. Create a matrix of size (n x d) which represents the sentence



Convolution on a text based matrix

- Convolutions are done over the whole width of the matrix
- Filters are of different sizes (n-gram distances)
- Max Pooling is often done over the whole feature map
- Softmax like functions bring down number of classes


activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
d=5
I like this movie very much !

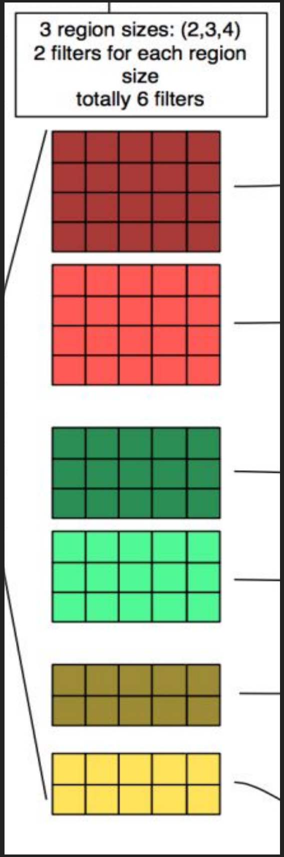
Text CNNs Filters

- Different sizes of filters
- Width = width of embedding
- Height = number of words wanted in the receptive field
- Capture different aspects of the sentence

For example:

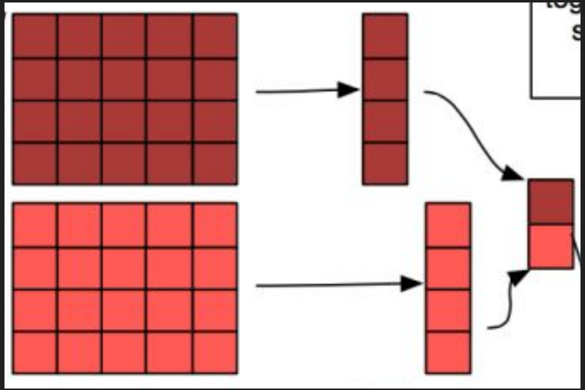
Size 2 filter: captures a negation “not amazing”

Size 4 filter: captures a target “potatoes are not amazing”



Text CNNs Pooling: The “not amazing” example

- Max Pooling is done after Convolutions
- Max Pooling Reduces dimensionality
- Obtains the core features of a sentence
- Max pooling is **done over the whole filter**



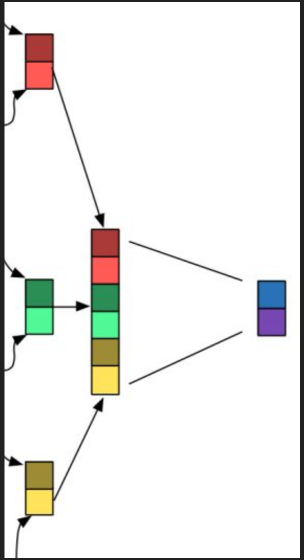
Negation Example:

In a sentence that contains “not amazing” as a phrase, we want to obtain the core idea that the sentence has a negation in it. We max pool to extract that information, rather than ‘noise’ in the rest of the sentence. (which is picked up by other filters)

Text CNNs Large Receptive Fields

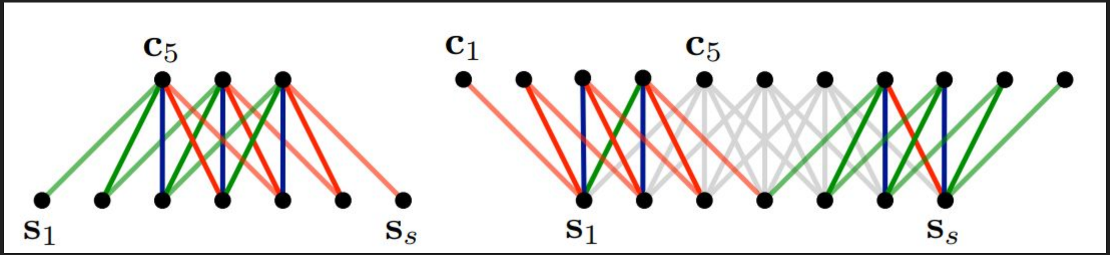
A great advantage of using CNNs is large receptive fields

- Various sizes of filters grab a large range of details
- Max pooling removes noise that exists in text (“is,the”)
- Allows the final layers to use information from each part of the sentence, and keep the context at the end influence the start of the sentence’s meaning.



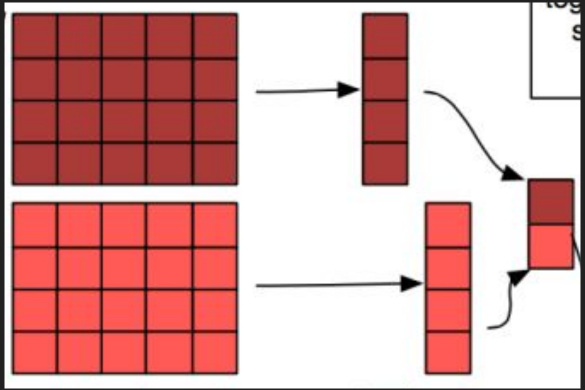
Hyper Parameters of CNNs

- Various stride sizes allow us to mimic properties of a RNN, where we have large overlaps between contexts creating internal tree structures
- Wide / Narrow Convolutions: Allow us to alter how much of the embedding we want to use. Weighing certain relations and properties more than the others



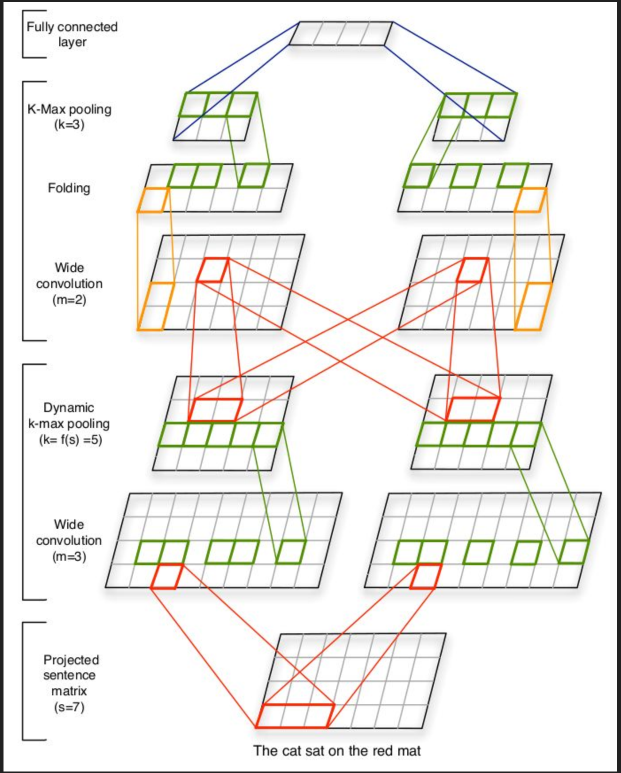
Text CNNs Pooling: k-Max Pooling

- Similar to Pooling, but returns k-Max numbers rather than 1
- Pooling and Convolution lead to a Feature Map
- ‘K’ is a hyper parameter for the network, or can be made as a function of embedding/convolution width
- Higher ‘k’ values implies we pick more than one important feature per filter.
- k=1 Max pooling might respond saying we have a negation “not” = Not Good!
- k=2 Max pooling shows we have a negation of “not bad” = Good!



Dynamic K-Max Pooling CNNs

- Type of CNN that uses K-Max Pooling
- Layers alternate between wide convolution and k - max pooling to create non linearity
- Mostly uses 1D convolutions
- Dimensionality of embeddings and width of convolutions are hyper parameters
- Focuses on creating and using Feature maps and feature graphs

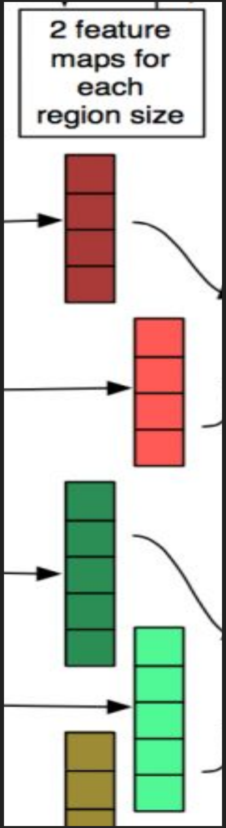

Fully connected layer
K-Max pooling (k=3)
Folding
Wide convolution (m=2)
Dynamic k-max pooling (k= f(s) =5)
Wide convolution (m=3)
Projected sentence matrix (s=7)
The cat sat on the red mat

Feature Maps and Feature Graphs

- Feature Maps act as feature detectors per filter
- Can be computed in parallel
- Independent until a top fully connected layer

Dynamic CNN’s induce an internal feature graph

- Nodes in higher layers are connected to lower layers if lower layer convolution feeds the higher layer
- Nodes that are not selected in pooling are dropped from the graph
- K-pooling allows for drawing graphs between different positions in the sentence far apart


2 feature maps for each region size