Text Classification

Attention

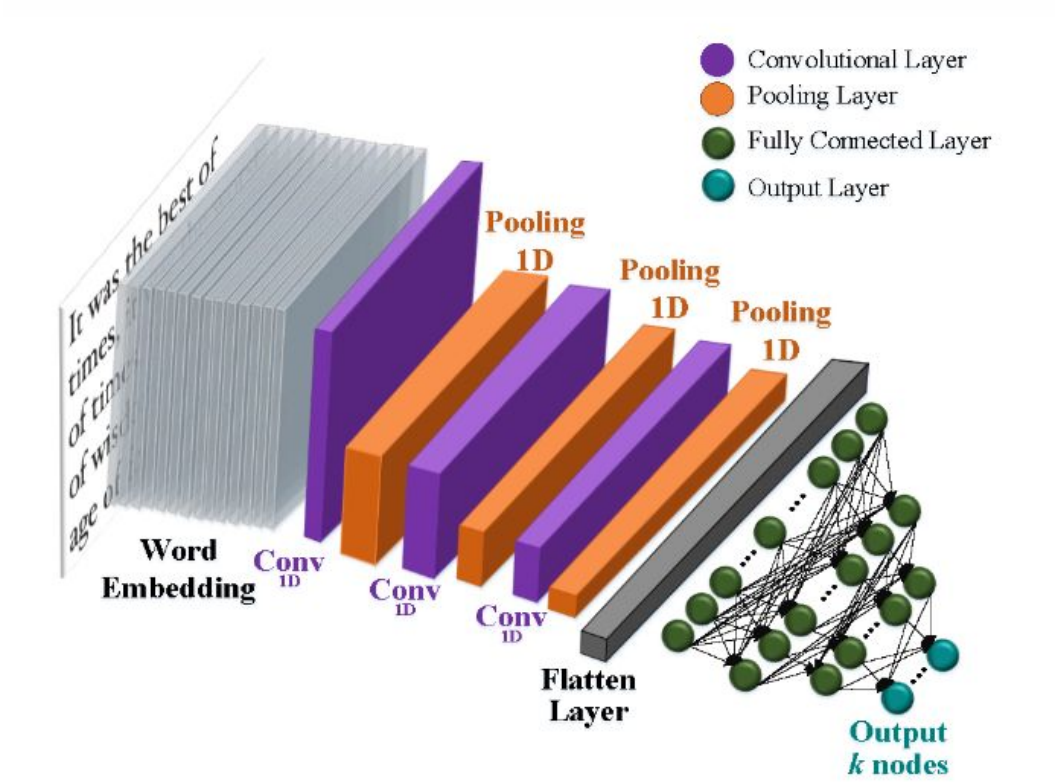
Text Classification

- Subjetivo/Objetivo
- Entailment
- Sentiment Analysis
- Topic Classification
- etc

MLP + Embeddings:

- Se suman todas los embeddings para cada una de las oraciones a clasificar y se divide por la cantidad de palabras.
- Con el resultado se alimenta una MLP, que a la salida tendrá una softmax o sigmoidea, dependiendo de las clases del problema.
- Ejemplo en : https://ai.intelligentonlinetools.com/ml/fasttext-word-embeddings-text-classification-python-mlp/

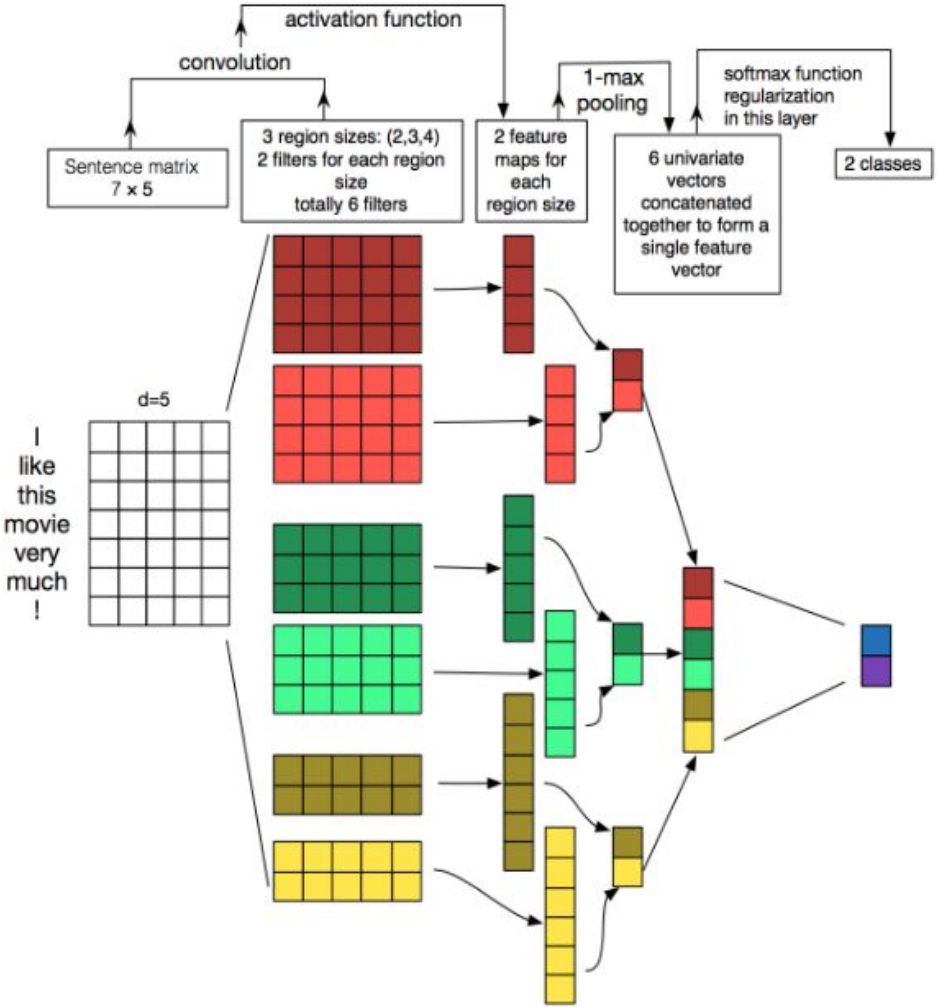
CNN


Convolutional Layer
Pooling Layer
Fully Connected Layer
Output Layer
Word Embedding
Conv 1D
Pooling 1D
Conv 1D
Pooling 1D
Conv 1D
Pooling 1D
Flatten Layer
Output k nodes

TextCNN

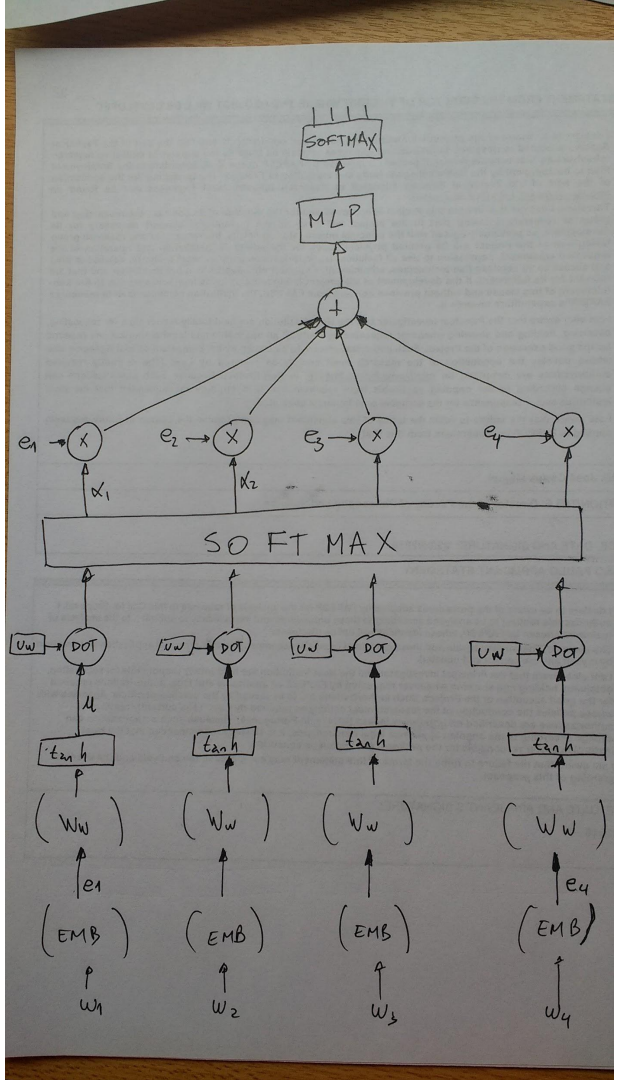
- Presentada en el paper: Convolutional Neural Networks for Sentence Classification by Yoon Kim.

  https://www.aclweb.org/anthology/D14-1181.pdf



MLP + Embeddings + Attention

- En vez de sumar todos los embeddings y promediarlos según la cantidad de palabras, podemos realizar la suma ponderada.
- Para elegir las ponderaciones de cada una de las palabras, se utiliza otra red neuronal que está “dentro” de la red neuronal original, y en base a mirar cada uno de los embeddings aprende cuales son los de mayor importancia.
- u_w se llama “vector de contexto” y puede verse como una “pregunta” que uno le hace a u_i acerca de su importancia.
- La capa densa de entrada aprende a representar a la palabra de entrada de manera que palabras importantes quedan alineadas con el vector de query u_w y las palabras menos importantes no quedan alineadas.
- A los vectores u_i se los llama “Keys” y codifican la respuesta a preguntas que podrían realizar uno o mas vectores de query.
- Una vez que se elijen los alfa, se usan para pesar cada una de las palabras (values).



MLP + Embeddings + Attention + ContextCNN

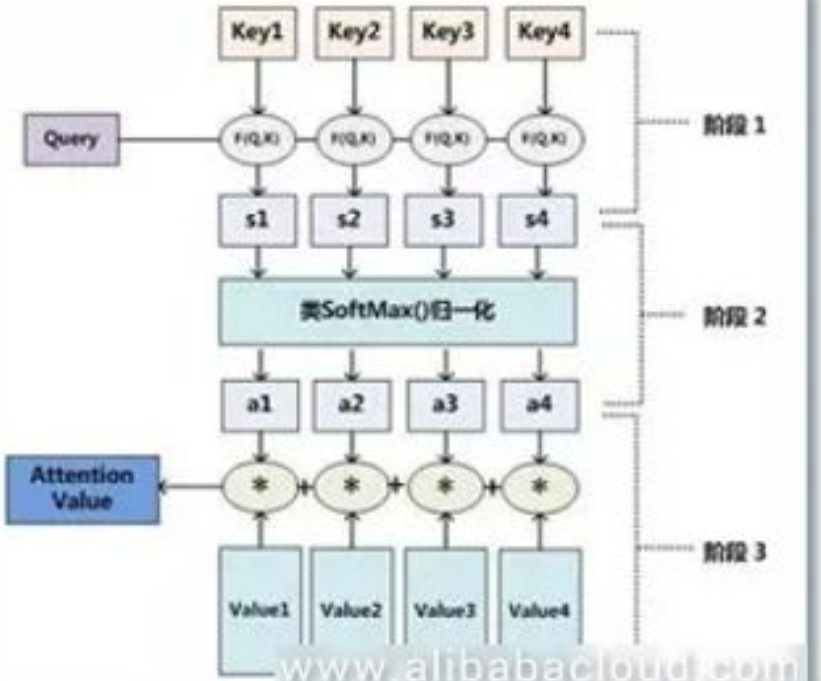
- En el caso anterior, si utilizamos embeddings no contextuales, el “Key” de cada palabra queda definido independientemente de su contexto.
- Sería bueno poder modificar la representación de cada palabra, de manera tal que dicha representación pueda “capturar” algo de su contexto.
- A tal fin se puede utilizar una CNN que vea la oración entera y nos de a la salida una representación contextualizada de cada una de las palabras.

Attention - generalización

$$f(Q,K_i)=\begin{cases} Q^T K_i & dot \\ Q^T W_a K_i & general \\ W_a[Q;K_i] & concat \\ v_a^T \tanh(W_a Q + U_a K_i) & perceptron \end{cases}$$

$$a_i = soft\max(f(Q,K_i)) = \frac{\exp(f(Q,K_i))}{\sum_j \exp(f(Q,K_j))}$$

$$\text{Attention}(Q,K,V) = \sum_i a_i V_i$$



En la mayoría de las aplicaciones de NLP, Key=Value

Hierarchical Attention

- Podemos ver la salida del último clasificador visto, antes de ingresar al MLP como una forma de hacer embeddings de oraciones.
- Podemos hacer un embedding para cada una de las oraciones que componen un texto, y a partir de ahí, volver a repetir el esquema de attention y luego ir a la capa MLP para realizar la clasificación.
- Habrá entonces un vector de query para las palabras y un vector de query para las oraciones.

Paper: https://www.cs.cmu.edu/~./hovy/papers/16HLT-hierarchical-attention-networks.pdf