SSN Invente 5.0: Make-a-thon

Participant name: Andrea Celestine Solomon

Problem statement:

One of the interesting problems in NLP for healthcare is in handling medical coded term associations and spelling errors. So we are going to mimic the problem with hangman / dumb charades like NLP word games. Your ML model should predict the correct word from an intentionally obscured word and its description (Hint). Our evaluation set will have incomplete words and their descriptions.

Example:

Input masked word = "DEM_G_A_HY"

Description = "is the statistical study of populations, especially human beings."

Model prediction/output = "DEMOGRAPHY"

My Approach:

This is a problem that would benefit best from an NLP model. The inputs are sentences and obscured words. However to make the best use of NLP models, we need to feed it the right data, that is pre-processing is vital.

Pre-processing the sentence:

From the sentence describing the word given, I have chosen the words that are more likely to be present next to the actual word in the latent space. For this I have used named entity recognition using the NLTK library. (I initially set out to use spaCy but it didn't tag words properly, probably due to lack of context). The entities and their meanings are provided.

- CC coordinating conjunction
- CD cardinal digit
- DT determiner
- EX existential there (like: "there is" … think of it like "there exists")
- FW foreign word
- IN preposition/subordinating conjunction
- JJ adjective 'big'
- JJR adjective, comparative 'bigger'
- JJS adjective, superlative 'biggest'
- LS list marker 1)

- MD modal could, will
- NN noun, singular ‘desk’
- NNS noun plural ‘desks’
- NNP proper noun, singular ‘Harrison’
- NNPS proper noun, plural ‘Americans’
- PDT predeterminer ‘all the kids’
- POS possessive ending parent’s
- PRP personal pronoun I, he, she
- PRP possessive pronoun my, his, hers
- RB adverb very, silently,
- RBR adverb, comparative better
- RBS adverb, superlative best
- RP particle give up
- TO, to go ‘to’ the store.
- UH interjection, errrrrrrrm
- VB verb, base form take
- VBD verb, past tense took
- VBG verb, gerund/present participle taking
- VBN verb, past participle taken
- VBP verb, sing. present, non-3d take
- VBZ verb, 3rd person sing. present takes
- WDT wh-determiner which
- WP wh-pronoun who, what
- WP possessive wh-pronoun whose
- WRB wh-abverb where, when

I have discarded these categories for our task: CC, CD, DT, EX, IN, LS, MD, PDT, POS, PRP, TO, UH, WDT, WP, WP, WRB, as they will not provide relevant results.

A list of the appropriate words are chosen, and each word is fed into a function that calculates the cosine similarity between that word and the rest of the words in the corpus and stores the values in a dictionary. The top words from this dictionary are then compared with the obscured word to figure out if it is the obscured word.

Embeddings:

GloVe: Global Vectors for word representation is an algorithm that computes embedding vectors for a given word. Instead of opting to train a model from scratch, I have downloaded pretrained word vectors from their site https://nlp.stanford.edu/projects/glove/. It has embeddings of 400k words, and was trained on data from Wikipedia, with embeddings of dimension 300,100,200. I have gone with the 100 dimensional embeddings. Do check out https://projector.tensorflow.org/ to visualise word embeddings!

Why I didn't train a model from scratch? Computational constraints xD

Working

Although this approach does yield good results, it is not computationally efficient. And from the evaluation dataset, some of the words are not a part of the words that GloVe was trained on, leaving some blank answers. In some rare cases, with smaller word sizes different words were predicted. (for example 'six' instead of 'fix'). Overall the model performs well.

On a side note, I said during the presentation that my model wasn't predicting the output on the evaluation set properly. The problem was that the words in the evaluation set had spaces in between the letters, causing the model to get confused xD. I pre-processed the column, removed spaces and then evaluated the model using a GPU enabled notebook on Colab. It roughly took around 20 seconds per entry and computed the answers for most of the words. I have attached the csv file, 'results.csv' with the outcomes for the dataset under the column 'Answers'.

Improvement:

As you might have figured, 20 seconds per word is A MIGHTY LOT. And is computationally inefficient. To better this, I have used a clustering algorithm, namely KMeans clustering to group words into clusters, such that words from the same cluster will have similar meanings. KMeans clustering is an unsupervised learning method, popularly used in Machine Learning. I have chosen the number of clusters as 5. The algorithm is as follows:

- The words selected from the meaning will be compared to the 5 cluster centres first to determine which category it is likely to belong to. Min cosine distance corresponds to this category
- Then it is compared to the words from that category alone (using comparison) and the answer is determined

This reduces the number taken for each word by a factor of 1000! However it was able to predict answers only for a little more than half of the evaluation set. Still an improvement don't you think? I have stored these answers under the column 'Answers with KMeans' in the csv file 'results_with_kmeans.csv'.

Further improvement:

So my suggestions to improve this would be

- Using a different clustering algorithm, like DBSCAN and comparing performance
- Even with KMeans, evaluating performance for different values of k
- Using the other embedding datasets from the official site for GloVe that have more words
- Using embedding datasets from another algorithm like Word2Vec

Using this for the problem stated in the problem statement:

If this was to be used for medical purposes, one would need to train a model such as GloVe or Word2Vec on a large corpus of medical data: medical textbooks, dictionaries, etc. Then using these embeddings thus derived to compute the obscured word. Depending on the size of the corpus, such a model would also benefit from unsupervised clustering algorithms.

Conclusion:

Overall I had loads of fun playing around with different data, different functions and finding a proper solution to this problem statement. It definitely taught me a lot and made me understand different aspects of text processing and NLP models. Looking forward to more such challenges in the future!