Assignment 1.1

Practical Deep Learning for Language Processing

Dr. Aseem Behl
School of Business and Economics
University of Tübingen
Wednesday, 16. November 2022

Introduction

In the lectures we are currently learning about the Word2Vec model which can be trained to learn word vector representations or word embeddings from text data. The word representations can then be applied to find semantically similar words. In practice, word vectors that are pre-trained on large datasets (for example large parts of wikipedia text) can learn better word representations with larger vocabularies.

Pretrained Word Embeddings

GloVe

CLASS torchtext.vocab.GloVe(name='840B', dim=300, **kwargs) [SOURCE]

FastText

CLASS torchtext.vocab.FastText(language='en', **kwargs) [SOURCE]

```
[docs]class GloVe(Vectors):
    url = {
        '42B': 'http://nlp.stanford.edu/data/glove.42B.300d.zip',
        '840B': 'http://nlp.stanford.edu/data/glove.840B.300d.zip',
        'twitter.27B': 'http://nlp.stanford.edu/data/glove.twitter.27B.zip',
        '6B': 'http://nlp.stanford.edu/data/glove.6B.zip',
    }

    def __init__(self, name='840B', dim=300, **kwargs):
        url = self.url[name]
        name = 'glove.{}.{}d.txt'.format(name, str(dim))
        super(GloVe, self).__init__(name, url=url, **kwargs)


[docs]class FastText(Vectors):

    url_base = 'https://dl.fbaipublicfiles.com/fasttext/vectors-wiki/wiki.{}.vec'

    def __init__(self, language="en", **kwargs):
        url = self.url_base.format(language)
        name = os.path.basename(url)
        super(FastText, self).__init__(name, url=url, **kwargs)
```

For example, Pre-trained GloVe embeddings trained on different datasets of dimension 50, 100, and 300 and fastText embeddings for English can be downloaded through `torchtext.vocab`. These pre-trained representations can then be applied to downstream language processing tasks. In this assignment, we will learn to apply pre-trained representations on two tasks.

Task A - Fine-grained Supermarket Product Segmentation

Supermarket basket data of customer orders can be analysed over time to design better targeted marketing campaigns. The data can also be used forecast sales of products in order to optimise product inventory.

In order to study customer shopping behaviour, it is important to understand what type of products end up in the basket. However, supermarkets might only maintain a very coarse-grained record of product type. For example, under the "hair care" category, one could find products of very distinct sub-categories (shampoos, conditioners, colors, oils, creams, styling products etc.). The sub-category segmentation of products could provide useful information for detecting patterns in shopping behaviour.

Therefore, in task A, we will exploit the word representations extracted from pre-trained `FastText` model to segment products into fine-grained sub-categories. Specifically, you will need to implement the following:

1. Import the provided `products.csv` and `aisles.csv` as a data-frame in your Python notebook using `Pandas`.
2. Select the products from the product data-frame of a single aisle category of your choice (for example hair care is aisle 22).
3. For each product, clean up the product name string using the provided `cleanupText()` function provided in the starter notebook.
4. For each product, split the product name into list of words and remove any *stop words*. Stop words are words which do not add much meaning to a sentence, for example, 'the', 'this', 'and', 'but', 'or' and many others. You

can use one of the many different stop word lists for the English language, for example the NLTK Stopword List.

5. After the last two filtering steps, next, for each product, you will compute the word vector representations for every word in the product name. You will use the `torchtext.vocab.FastText` package to compute the word representations. Sample code available in the starter notebook.
6. Take the element-wise mean of the word embeddings to get the embeddings of each product name.
7. Using K-means clustering approach, cluster the products using the embeddings of their names.
8. Experiment a bit with the number of clusters, analyse the final cluster members.
9. Provide a sub-category name for each cluster based on the products found in the cluster.
10. Optionally, repeat the previous steps with a different aisle category.

Task B - Genre Classification with Movie Titles

Assume in a market research survey, you want to ascertain the genre of movies your customers prefer to watch. However, asking the question directly could be noisy as customers may be subjective about genre assignments for their favourite movies. In this scenario, it might be more favourable to ask them about their favourite movies and infer the genre preference from these responses.

Towards this goal, in this exercise, you will apply the pre-trained word vectors to predict genre of movies given the title.
Using the cosine similarity of the word vectors, you will assign each movie title to a genre it is the closest to in the word vector space. Specifically,

perform the following steps:

1. Following list of genres for classification is provided in the starter notebook: *'action', 'adventure' ,'comedy', 'drama' ,'fantasy', 'horror' ,'romance', 'thriller'*. You will use the `torchtext.vocab.FastText` package to compute the word vector representations of each genre name. Sample code available in the starter notebook.
2. Following list of movie titles for classification is provided in the starter notebook: *'The Hangover', 'Shutter Island', 'Fight Club', 'Jumanji', 'Narcos', 'The Matrix', 'Rush Hour', 'The Mummy', 'Iron Man', 'Silence of the Lambs', 'Batman Begins', 'Spider Man', 'The Hobbit', 'Troy', 'Jurassic Park', 'Scary Movie', 'Mission Impossible', 'Ted', 'Eat Pray Love', 'The Notebook', 'Love Actually', 'The Terminal', 'Crazy Stupid Love', 'Twilight', 'The Martian', 'Pursuit of Happyness'*. You will use the `torchtext.vocab.FastText` package to compute the word vector representations of each movie title.
3. For titles with multiple words, you should first split the name into list of words. Next you should first remove any stop words. Stop words are words which do not add much meaning to a sentence, for example, 'the', 'this', 'and', 'but', 'or' and many others. You can use any of the many available stop word lists for English. You can use one of the many different stop word lists for the English language, for example the NLTK Stopword List.
4. Next, compute the cosine similarity of each word in the title with genre words and assign the genre with the highest average similarity. Implement your own cosine similarity function.

The task A and task B are worth **2 and 2 points respectively** and are due on **Wednesday, November 30, at 08:00 PM CET**. For submission, you only need to upload your notebook to ILIAS. Run all cells, and do not clear out the

outputs, before submitting. In order to receive credit for the assignment, please be prepared to present your solutions during the lecture on **Wednesday, November 30**. You may be asked other questions from the lecture material related to the assignment.