02462 Project 2020

April 2020

Project description

Text classification is a fundamental task in Natural Language Processing (NLP) and is utilized in many applications, such as sentiment analysis, topic labeling, information retrieval etc. The purpose of text classification is to assign pre-defined labels or categories to text segments or documents. The goal of this project is to analyze and evaluate a specific method for text classification called *fastText* [Joulin et al., 2016]. In the fastText method, a given text is represented using word embeddings for individual words as well as bigrams and trigrams. These embeddings are learned from the data when the classifier is trained. As a baseline, you will contrast the fastText model with a simpler classifier based on pre-trained Glove embeddings. Your task is to analyze and compare the two methods using the two datasets described below and state and discuss the results in a small report in the format of a scientific paper. .

The fastText model

This project concerns a particular model for fast text classification, named *fastText*. The model takes a text as input and outputs a probability distribution over a pre-defined set of text categories. The model works as follows. First, we define a vocabulary consisting of individual words (1-grams) as well as bigrams (2-grams) and trigrams (3-grams). To each of these ngrams, we associate a D-dimensional embedding vector $\boldsymbol{b}_i \in \mathbb{R}^D$. Using this pre-defined set of ngrams, we can now represent a given text as a collection of a ngram vectors.

For example, consider the text: "Text is fun". If all the ngrams are represented in the predefined vocabulary, we can now represent this text as the collection of the following 6 vectors

$$\mathcal{D} = \left\{\boldsymbol{b}_{\text{Text}},\, \boldsymbol{b}_{\text{is}},\, \boldsymbol{b}_{\text{fun}},\, \boldsymbol{b}_{\text{Text is}},\, \boldsymbol{b}_{\text{is fun}},\, \boldsymbol{b}_{\text{Text is fun}}\right\}.$$

That is, we get a vector for each individual word, each bigram, and each trigram in the given text or document.

After having obtained the collection of ngrams vectors, $\mathcal{D}$, for the document, the idea of *fastText* is reduce this collection to a single vector by computing the mean of the vectors in $\mathcal{D}$

$$\boldsymbol{z} = \frac{1}{6}\left[\boldsymbol{b}_{\text{Text}} + \boldsymbol{b}_{\text{is}} + \boldsymbol{b}_{\text{fun}} + \boldsymbol{b}_{\text{Text is}} + \boldsymbol{b}_{\text{is fun}} + \boldsymbol{b}_{\text{Text is fun}}\right] = \frac{1}{6}\sum_{i\in\mathcal{D}} \boldsymbol{b}_i.$$

Hence, we can represent the full text using a single D-dimensional vector $\boldsymbol{z} \in \mathbb{R}^D$. In general, if we have a dataset of N documents, we can represent the n'th document as follows

$$\boldsymbol{z}_n = \frac{1}{N_n} \sum_{i \in \mathcal{D}_n} \boldsymbol{b}_i,$$

where N_n is the number of vectors in the collection for the n'th document $\mathcal{D}_n$. This provides a mechanism for generating a vector representations for texts of variable lengths. We use $\boldsymbol{B}$ to denote a matrix containing all the embedding vectors for the entire vocabulary.

The latent representation for each text $\boldsymbol{z}_n$ can now be fed into a "standard classifier". The *fastText* model combines this mechanism with a linear softmax classification layer

$$p(y_n|\boldsymbol{A}, \boldsymbol{z}_n) = \text{softmax}\left(\boldsymbol{A}\boldsymbol{z}_n\right),$$

where $\boldsymbol{A} \in \mathbb{R}^{K \times D}$ are the weight vectors for the linear classifier, K is the number classes in the classification problem and D is the dimension of the hidden representation. Note that both weight vectors $\boldsymbol{A}$ of the linear classifier as well as the word and ngram embeddings $\boldsymbol{B}$ are learned from the training data using *maximum likelihood.*

The dimension D of the embedding vectors can be considered a hyperparameter and this can be chosen using cross-validation.

The baseline method: classification using pretrained Glove embeddings

In scientific work it is important to include a relevant baseline or reference method for proper comparison. In this project, you are going to use a simple method based on pre-trained Glove word embeddings [Pennington et al., 2014]. For this baseline method, we will simply use the mean of the individual Glove embeddings vector for a given text yielding the representation

$$\boldsymbol{z}_n^{\text{Glove}} = \frac{1}{N_n} \sum_{i=1}^{N_n} \boldsymbol{w}_i,$$

where $\boldsymbol{w}_i$ is the pre-trained Glove embedding vector for the i'th word in the text. That is, to get the representation for the n'th text in a dataset, we look-up the embedding vectors for each word in the text and compute the mean of the embedding vectors. After having obtained this representation for the data set, you can combine this representation with a classification method of your own choosing (e.g. Naive Bayes, Bayesian classifiers, or something else). Note the importance difference that in the fastText method, we learn all the embedding from our training data, while for this simple baseline method, we use pre-trained embeddings.

Data

In this project, two different datasets are considered. In the first dataset, each document is a text-message (SMS) labelled as either *spam* (unwanted and unsolicited message) or non-spam. In the second data-set, each document is a short news report, labelled as being *sports*, *world*, *business* or *science & technology* news. Both datasets have been divided into a training and a test set. Note that all documents are in lower case.

Approach

You are encouraged to use any material covered in the course, including methods and code used in exercises and assignments.

- Implement the baseline method as the first step. This is a good first step in order to get familiar with the data.
- You are expected to conduct a principal component analysis (PCA). Use PCA to visualize the latent representations for both methods and data-sets. Identify the PC directions that are most useful for classifying the data-sets. Comment on how the directions can be interpreted.
- You are expected to test, evaluate and compare classification accuracy for both classifiers on both data-sets. Describe and motivate your approach and evaluation setup. Analyze and conclude on your findings.
- Evaluate and discuss the model capabilities. You can base your discussion on the following investigations:

  The news data

  - Find a few examples of news reports on your own, that complements the news data-set. Of interest can be examples that are not defined by any of the four categories or span multiple of the categories. Comment on the choice of texts, how you would label them and the classification probabilitites (softmax scores) from the model.
  - Visualize the examples in the same PCA plot together with the news data-set. Comment and discuss the result and choice of texts.
  - Locate the nearest neighbors in each category to each of your own examples. Report and discuss the classification probability for each class.

  The spam data

  - Investigate which words and n-grams that are strong predictors for spam and non-spam.
  - Consider some of the texts that are classified as being spam. Can you add another sentence to it such that it becomes not-spam?
  - Is this spam classifier useful for filtering e-mails?
  - Comment on adversarial strategies to 'fool' the spam classifier.

Formalities

- The project must be conducted in groups of preferably 3 students.
- The project deliverables consist of a written report, your source-code and a pre-recorded video presentation (max 5 minutes, individual for each student!).
- The project deliverables must be handed in as a group assignment on *Inside*. **Deadline for handing in is May 12th 2020**.
- The report must be at most 6 pages, including all figures.

- The report must be self-contained, preferably organized according to the IMRaD (Introduction, Methods, Results, and Discussion) structure.
- The source-code must be well-structured, and comments should be added to aid readability.
- Based on your code and report it should be possible to reproduce your findings.
- The project is part of the final assessment in the course and the oral exam will include questions regarding the project and your report.

Code

You can use the following code to load the news dataset:

```
news_data = np.load('./news_data.npz', allow_pickle=True)
train_texts = news_data['train_texts']
test_texts = news_data['test_texts']
train_labels = news_data['train_labels']
test_labels = news_data['test_labels']
ag_news_labels = news_data['ag_news_label']
```

You can use the following code to load the SPAM dataset:

```
spam_data = np.load('./spam_data.npz', allow_pickle=True)
train_texts = spam_data['train_texts']
test_texts = spam_data['test_texts']
train_labels = spam_data['train_labels']
test_labels = spam_data['test_labels']
spamlabels = spam_data['labels']
```

The fastText method is implemented in the module `text_classifier.py`. You can use it as follows:

```
from text_classifier import TextClassifier

# training the classifier
classifier = TextClassifier(train_texts, train_labels, embed_dim, num_epochs=5)

# making predictions
output_label = classifier.predict(text)
output_prob = classifier.predict(text, return_prob=True)

# looking up embeddings
emb = classifier.get_text_embedding(text)

# does word exists in vocab?
print(classifier.word_in_vocab('exam'))
print(classifier.word_in_vocab('fictiouswordthatdoesntexist'))
```

You may have to install the following Python packages: `torch`, `torchtext`, `tqdm` to run the fastText code.

References

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification, 2016. URL `http://arxiv.org/abs/1607.01759`. cite arxiv:1607.01759.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *In EMNLP*, 2014.