[Learning Note] StarSpace For Multi-label Text Classification

Ceshine Lee Jan 21, 2018 · 4 min read

StarSpace is an ambitious model that attempts to solve a wide range of entity-embedding-related problems. It has been created and open-sourced by Facebook AI

intends to be a straight-forward and efficient strong baseline, that is, the first model you'd train for a new dataset or a new problem.

In this post, I'll write down how I (tried to) get StarSpace to work with the dataset from Kaggle's Toxic Comment Classification Challenge. This dataset represents a multi-label text classification problem, i.e., more than one labels can be assigned to a single comment. As fastText does not exactly support multi-label classification, I thought StarSpace might be a good alternative.

Spoilers / tl;dr

I've found two main problems of StarSpace in its current state that make it not ready as a baseline for this dataset:

1. No `predict` command: Currently only `test` is provided, which gives descriptive model evaluation. Extra work is needed to be done by yourself to obtain clean output of label probabilities of each comment.
2. Evaluation metric is not customizable: Because of the lack of clean prediction output, we are depending on the model evaluation from `test` command. But I don't see a way to get it to show *mean column-wise log loss* that we care about for this dataset.

In addition, the documentation of StarSpace is not very comprehensive. It's reason why this post was written in the first place.

Install StarSpace

Starspace is written in C++, and you have to build it yourself. It's quite straight-forward in my Linux Mint 18.3 environment. Just clone the Git repo and run `make` :

```
git clone https://github.com/facebookresearch/Starspace.git
cd Starspace
make
```

In other environment, you might need to jump through additional hoops to meet the requirements. Please check the link for further instructions.

Python Environment

- Python 3.6
- pandas 0.22.0, joblib 0.11, tqdm 4.19.5 (very basic APIs are used, so the exact versions shouldn't matter. Check if there are any major updates after the publish time of this post if you run into any problem.)
- spacy 2.0.5
- scikit-learn 0.19.1

Prepare Dataset

It goes without saying that you need to download the dataset to your computer first. I usually put the dataset inside `data` sub-folder under the project root, and put everything that derives from the raw dataset inside `cache` sub-folder. We'll use the same setting below.

The next step is to clean and tokenize the comments. Here I use the English tokenizer from spacy, remove newline characters and some other punctuations, and trim the comments to 20,000 characters. (Cleaning schemes are often dataset-dependent. There are definitely better scheme for this dataset than the simplistic one presented in this post. You could filter out stop words, for instance.)

```
""" Tested with Python 3.6 """
import re

import pandas as pd
import spacy
import joblib
from tqdm import tqdm

nlp = spacy.load('en')

```

```
        text = re.sub(
            r"[\*\"“”\n\\…\+\-\/\=\(\)‘•:\[\]\|’\!;]", " ", str(text))
        text = re.sub(r"[ ]+", " ", text)
        text = re.sub(r"\!+", "!", text)
        text = re.sub(r"\,+", ",", text)
        text = re.sub(r"\?+", "?", text)
        if (len(text) > 20000):
            print(text)
            print(len(text))
            text = text[:20000]
        lists_of_tokens.append(nlp.tokenizer(text))
    return lists_of_tokens


def main():
    train = pd.read_csv('data/train.csv')
    train_tokenized = tokenize(train["comment_text"].tolist())
    joblib.dump(train_tokenized, "cache/train_tokenized.pkl")
    del train_tokenized, train
    test = pd.read_csv('data/test.csv')
    test_tokenized = tokenize(test["comment_text"].tolist())
    joblib.dump(test_tokenized, "cache/test_tokenized.pkl")


if __name__ == "__main__":
    main()
```

After the comments are tokenized, we combine these tokens with their associated labels, and save them in a format that StarSpace can recognize (Here we use fastText format).

```
import pandas as pd
import joblib
from sklearn.model_selection import train_test_split


LABELS = ["toxic", "severe_toxic", "obscene",
          "threat", "insult", "identity_hate"]
EMPTY_ID = len(LABELS)


```

```
        for i, l in enumerate(LABELS):
            if row[l]:
                parts.append("__label__{}".format(i))
                flag = True
        if flag is False:
            parts.append("__label__{}".format(EMPTY_ID))
        return " ".join(parts)


    def main():
        train = pd.read_csv('data/train.csv', usecols=[2, 3, 4, 5, 6, 7])
        train_tokens = joblib.load("cache/train_tokenized.pkl")
        train["comment_text_cleaned"] = [
            " ".join([str(x) for x in tokens if str(x).strip() != ""]) for tokens in train_tokens]
        train, val = train_test_split(train, test_size=0.25, random_state=24)
        train_lines = train.apply(create_labeled_string, axis=1)
        with open("cache/train.txt", "w") as f:
            f.write("\n".join(train_lines))
        val_lines = val.apply(create_labeled_string, axis=1)
        with open("cache/val.txt", "w") as f:
            f.write("\n".join(val_lines))


if __name__ == "__main__":
    main()
```

We use 25% of the train dataset as validation. The results are saved to `cache/train.txt` and `cache/val.txt` respectively. Note because I haven't found a way to extract predictions easily, the test dataset is not processed here.

The fastText format is pretty simple. Put a space between consecutive tokens, one comment/instance per line, and labels at the end. Labels are encoded numerically, and a dummy label is created for comments that are not toxic (StarSpace does not accept empty list of labels).

As an example, this is the first comment in the train dataset:

After processing, it becomes:

```
Nonsense ? kiss off , geek . what I said is true . I 'll have your
account terminated . __label__0
```

Train and Evaluate Model

To train a model, run this command in the command line:

```
starspace train -ngrams 2 -minCount 10 -thread 4 -trainFile
cache/train.txt -model cache/starspace.model
```

This model uses unigram and bigram, requires a token to appear at least 10 times to be consider, and use 4 threads. More parameters can be found via `starspace -h`.

StarSpace save the model to `cache/starspace.model`, and the word embedding and label embedding vectors to `cache/starspace.model.tsv`. You can analyze or visualize those embedding vectors for further insights.

To evaluate the model with the validation dataset, run this command:

```
starspace test -testFile cache/val.txt -model cache/starspace.model -
predictionFile cache/starspace.pred
```

Besides command line outputs, it'll also write the case-by-case evaluation to `cache/starspace.pred`. An example:

```
Example 68:
LHS:
IT 'S HER QUOTE . WHAT CAN'T YOU UNDERSTAND ? Here is the official
press release Now , please , can we stop this nonsense ?
RHS:
```

```
(--) [0.0467122]        __label__2
(++) [-0.0127831]       __label__0
(--) [-0.23463] __label__1
```

Another example:

```
Example 116:
LHS:
STOP , you had better stop putting all that stupid junk on my IP page
. read it . you 'll see what i 'll do if you do it again .
RHS:
__label__0
Predictions:
(++) [0.529969] __label__0
(--) [0.416814] __label__2
(--) [0.388696] __label__4
(--) [0.34913]  __label__3
(--) [0.240833] __label__1
```

Note that most of the comments in the dataset are not toxic (*__label__6*), so I'm really cherry-picking examples here.

The meaning of the values assigned to each label is unclear. I guess the answer can be found in the the paper.

Wrapping Up

It's unfortunate that StarSpace is not ready to serve as a baseline for the toxic comment dataset yet. This post keeps track of the steps needed to get as far as we can for future reference. Hopefully StarSpace will continue to be developed and be truly ready for production. We'll come back for more then.