Capstone 1 Data Wrangling Report

Kapil Chiravarambath

January 25, 2019

0.1 Data Wrangling on Stanford Sentiment Treebank

Report on investigation of the Stanford Sentiment Treebank Dataset.

0.1.1 Data Description

The Stanford Sentiment Treebank Corpus [2] is a standardised dataset that is used in many benchmarks such as GLUE. As such we do not expect to find any data inconsistencies or incomplete or missing data in the datasets.

The Treebank consists of fully labeled parse trees that allows for a complete analysis of the compositional effects of sentiment in language. The corpus is based on the dataset introduced by [1] and consists of 11,855 single sentences extracted from movie reviews.

The sentences in the treebank were split into a train (8544), dev (1101) and test splits (2210) and these splits are made available with the data release here.

0.1.2 Parsing Sentiment Tree

A typical training sample looks like this: > (3 (2 But) (3 (2 he) (3 (2 somehow) (3 (3 (2 (2 pulls) (2 it)) (1 off)) (2 .)))))

One of the main checks on the first examination of data was to make sure that all trees could be parsed into properly formed trees. The tree nodes had to satisfy the following properties: * Each node was either a leaf or an intermediate node with exactly two children. * A Leaf Node must have a sentiment label and a word associated with it. * Leaf Nodes have no children. * An Intermediate Node must have exactly two children and a sentiment label associated with it. * Intermediate Nodes do not have any word association.

Tests were written to verify that the entire training dataset satisfied the above properties test_tree.py

0.1.3 Environment Setup

```
In [4]: from __future__ import print_function

        import os
        import sys

        import numpy as np
        import pandas as pd
```

```
In [5]: PROJ_ROOT = os.pardir

In [6]: # Add local python functions
        sys.path.append(os.path.join(PROJ_ROOT, "src"))
```

0.1.4 Parsing Example

The following code parses the tree and rewrites it back as a text.

```
In [7]: from features.tree import Tree

In [8]: x = '(3 (2 But) (3 (2 he) (3 (2 somehow) (3 (3 (2 (2 pulls) (2 it)) (1 off)) (2 .))))))'
        t = Tree(x)
        print(t.text())

But he somehow pulls it off .
```

In addition for aiding visualization in flask, a JSON conversion had to be defined.

```
In [12]: print(json.dumps(t.to_json(), indent=4, sort_keys=True))

{
    "label": 3,
    "left": {
        "label": 2,
        "left": {},
        "probabilities": null,
        "right": {},
        "word": "But"
    },
    "probabilities": null,
    "right": {
        "label": 3,
        "left": {
            "label": 2,
            "left": {},
            "probabilities": null,
            "right": {},
            "word": "he"
        },
        "probabilities": null,
        "right": {
            "label": 3,
            "left": {
                "label": 2,
                "left": {},
                "probabilities": null,
                "right": {},
```

```
            "word": "somehow"
        },
        "probabilities": null,
        "right": {
            "label": 3,
            "left": {
                "label": 3,
                "left": {
                    "label": 2,
                    "left": {
                        "label": 2,
                        "left": {},
                        "probabilities": null,
                        "right": {},
                        "word": "pulls"
                    },
                    "probabilities": null,
                    "right": {
                        "label": 2,
                        "left": {},
                        "probabilities": null,
                        "right": {},
                        "word": "it"
                    },
                    "word": null
                },
                "probabilities": null,
                "right": {
                    "label": 1,
                    "left": {},
                    "probabilities": null,
                    "right": {},
                    "word": "off"
                },
                "word": null
            },
            "probabilities": null,
            "right": {
                "label": 2,
                "left": {},
                "probabilities": null,
                "right": {},
                "word": "."
            },
            "word": null
        },
        "word": null
    },
```

```
        "word": null
    },
    "word": null
}
```

0.1.5 Caching the parsed trees

To save memory and cpu time on parsing trees a singleton object was defined DataManager

The parsed trees for all the three datasets (train, dev, test) were obtained for using asserts in the code.

References

[1] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 115–124. Association for Computational Linguistics, 2005.

[2] Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *In Proceedings of EMNLP*, pages 1631–1642, 2013.