A REPORT ON THE REPRODUCTION OF THE PAPER "EMBARRASSINGLY SIMPLE UNSUPERVISED ASPECT EXTRACTION" BY TOLKENS AND CRANENBURGH

LUISA RIBEIRO DA SILVA

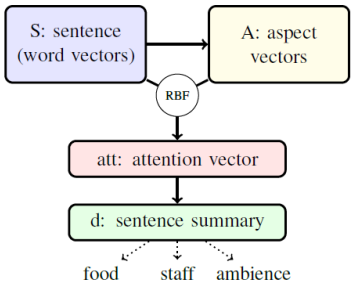
University of Tübingen
MA International Studies in Computational Linguistics
Reproducibility in Natural Language Processing
Summer Semester 2021

1. The Main Idea

Stéphan Tolkens and Andreas van Cranenburgh's goal with this project was to create a straight-forward, unsupervised method for extracting aspects of instances, since, as they claim, most existing systems are supervised and therefore require a lot of training data, which is often scarce.

During experimentation, the authors explain that "the most frequent nouns were already good aspects; any further constraint led to a far worse performance on the development dataset." (Page 2). They refer to their method as "embarrassingly simple" because it only needs a POS tagger to identify the nouns and in-domain word embeddings.

Their method basically consists of two steps: extraction of candidate aspect terms and assigning aspect labels to instances.



The way they seem to have been able to achieve good results with it was through using a single-head attention mechanism based on Radial Basis Function (RBF) kernels, referred to as CAt, Contrastive Attention:

```python
def rbf_attention(vec, memory, gamma, **kwargs):
    """
    Single-head attention using RBF kernel.

    Parameters
    ----------
    vec : np.array
        an (N, D)-shaped array, representing the tokens of an instance.
    memory : np.array
        an (M, D)-shaped array, representing the memory items
    gamma : float
        the gamma of the RBF kernel.

    Returns
    -------
    attention : np.array
        A (1, N)-shaped array, representing a single-headed attention mechanism

    """
    z = rbf_kernel(vec, memory, gamma)
    s = z.sum()
    if s == 0:
        # If s happens to be 0, back off to uniform
        return np.ones((1, len(vec))) / len(vec)
    return (z.sum(1) / s)[None, :]
```

2. The Reproduction

In order to reproduce their method, I looked into the repository cited in their paper, https://github.com/clips/cat, where not only is the code, but also some instructions on using, adapting and even reproducing their model.

Although it is said in the repository that it is possible to download the SemEval 2014, 2015 and citysearch datasets, extract the text from the XML files and reproduce the experiments by "putting it in the data/ folder and running the experiments from experiments/", when contacting one of the authors I was instructed to first extract the text from the XML files, then convert it to the CoNLLu format, then run the code presented in the /example_pipeline folder.

Basically the steps taken were the following:

1. First, I downloaded the restaurant subsets only of the mentioned datasets (SemEval 2014, SemEval 2015 and Citysearch).
2. Then, as instructed by one of the authors, I wrote script for extracting text from XML files and create new files with only the text instances which were then added to a folder called *cleanData*:

```python
from xml.etree import cElementTree as ET
import os
def extractXML(folder, out_dir):
    files = os.listdir(folder)
    for filename in files:
        if 'xml' in filename:
            dir = os.path.join(folder, filename)
            #file_name = os.path.basename(dir)
            with open(dir, 'r') as f:
                tree = ET.parse(f)
                root = tree.getroot()
                new_filename = filename.replace("xml", "txt")
            with open(out_dir + new_filename, "w") as Outfile:
                for page in root.iter('text'):
                    line = str(page.text)
                    Outfile.write(line + " \n")
                Outfile.close()
            f.close()

if __name__ == "__main__":
    extractXML("../data", '../data/cleanData/')
```

3. I also wrote a script to extract the labels from the datasets, which I did have to change running with each dataset, as the structure differed slightly:

```python
from xml.etree import cElementTree as ET
import os

def extractLabels(folder, out_dir):
    files = os.listdir(folder)
    for filename in files:
        if 'xml' in filename:
            dir = os.path.join(folder, filename)
            with open(dir, 'r') as f:
                tree = ET.parse(f)
                root = tree.getroot()
                new_filename = "labels_" + filename.replace("xml", "txt")
            with open(out_dir + new_filename, "w") as Outfile:
                for aspectTerms in root.iter('aspectTerms'):
                    for aspectTerm in root.iter('aspectTerm'):
                        line = aspectTerm.attrib['term']
                        Outfile.write(line + " \n")
                Outfile.close()
            f.close()

if __name__ == "__main__":
    extractLabels("../data", "../data/")
```

4. Next, I converted the text to CoNLLu format by cloning the *spacyconllu* repository (https://github.com/andreasvc/spacyconllu), created by one of the authors, and ran it for the POS tagging. Then, added the CoNLLu files to a folder called *my_data*.

5. Then, I ran *preprocessing.py* for training word embeddings and *run.py* in */example_pipeline*.

6. Last, I ran the files *preprocessing_embeddings.py*, *experiment_dev.py*, *grid_search.py*, *experiment_test_baseline.py* and *experiment_test.py* in the */experiments* folder.

3. Challenges, Bias and Obscurity

A possible small occurrence of obscurity is that it is not mentioned anywhere in the paper that the data had been converted into the CoNLLu format, which means one might have a hard time or have small differences replicating the results based only on the paper itself.

As for a possible occurrence of bias, in one of the tables presented on page 3 of the paper, used to illustrate and compare precision, recall and F-scores scores for each aspect in the Citysearch dataset with different methods, there seems to be some bias as the result in bold in the first column, 82.4, is actually not the best one in the table, but the one from AE-CSA, 92.6:

Aspect:	STAFF		
SERBM (2015)	81.9	58.2	68.0
ABAE (2017)	80.2	72.8	75.7
W2VLDA (2018)	61.0	**86.0**	71.0
AE-CSA (2019)	92.6	75.6	77.3
Mean	55.8	85.7	67.5
Attention	74.4	69.3	71.8
CAt	**82.4**	75.6	**78.8**

The reproduction challenges were mostly related to versions, as CAt was developed in the beginning to middle of 2020. In order to be able to run most files it was necessary to look up what the version for the libraries used and for Python were at least back in July, 2020, which is when the paper was published.

4. Results and Conclusions

These were the results reported in the paper on page 3:

Method	P	R	F
SERBM (2015)	86.0	74.6	79.5
ABAE (2017)	**89.4**	73.0	79.6
W2VLDA (2018)	80.8	70.0	75.8
AE-CSA (2019)	85.6	86.0	85.8
Mean	78.9	76.9	77.2
Attention	80.5	80.7	80.6
CAt	86.5	**86.4**	**86.4**

Table 2: Weighted macro averages across all aspects on the test set of the Citysearch dataset.

Method	P	R	F
Aspect: FOOD			
SERBM (2015)	89.1	85.4	87.2
ABAE (2017)	95.3	74.1	82.8
W2VLDA (2018)	**96.0**	69.0	81.0
AE-CSA (2019)	90.3	**92.6**	91.4
Mean	92.4	73.5	85.6
Attention	86.7	89.5	88.1
CAt	91.8	92.4	**92.1**
Aspect: STAFF			
SERBM (2015)	81.9	58.2	68.0
ABAE (2017)	80.2	72.8	75.7
W2VLDA (2018)	61.0	**86.0**	71.0
AE-CSA (2019)	92.6	75.6	77.3
Mean	55.8	85.7	67.5
Attention	74.4	69.3	71.8
CAt	**82.4**	75.6	**78.8**
Aspect: AMBIENCE			
SERBM (2015	80.5	59.2	68.2
ABAE (2017)	81.5	69.8	74.0
W2VLDA (2018)	55.0	75.0	64.0
AE-CSA (2019)	**91.4**	77.9	**77.0**
Mean	58.7	56.1	57.4
Attention	67.1	65.7	66.4
CAt	76.6	**80.1**	76.6

After tinkering and experimenting with the code, the best results I got were similar enough to the ones reported in the paper. Doing exactly as the authors mention in section 2 of the paper, I used the restaurant subsets of the SemEval 2014 and 2015 datasets as development data and did not optimize any models on the test data.

The best results I was able to reach are as follows:

- Weighted macro averages across all aspects on the test set of the Citysearch dataset:

Mean	80.0	77.0	78.1
CAt	85.2	86.0	85.9

- Precision, recall, and F-scores on the test set of the Citysearch dataset:

Food:

	P	**R**	**F-score**
Mean	90.3	71.5	86.1
CAt	91.0	91.7	91.2

Staff:

	P	**R**	**F-Score**
Mean	52.6	81.3	64.7
CAt	80.3	73.0	75.1

Ambience:

	P	**R**	**F-Score**
Mean	57.2	54.7	55.0
CAt	74.2	77.0	75.2

Interestingly, the closest results were in the food aspect, but no results were identical. As I mentioned previously, there were problems with version numbers and there might be differences in the operating systems which may or may not justify such discrepancies. In general, the method seemed very effective and indeed simple when compared to other ones.

Some of the files were too big to be pushed into the GitHub repository. They could however be shared via Google Drive or other alternatives.