Final NLU project report

Alessandro Zinni (mat. 229709)

University of Trento

alessandro.zinni@studenti.unitn.it

1. Introduction

This work aims at implementing a method to perform polarity classification and subjectivity detection. The proposed methods should perform better than a given baseline. The methods used to face the problem are Naive Bayes (used as baseline), BiLSTM with and without attention and also a lexicon-based supervised attention model [1] that makes use of BiLSTM with attention, but also incorporates sentiment lexicon in the process. As shown in [2] polarity classification performance can be improved by removing objective sentences from the document of interest. For polarity classification, each method is tested with and without objective sentence removal.

2. Task Formalisation

Sentiment analysis or opinion mining is a text categorization task that determines the sentiment, which is the positive or negative orientation that a writer expresses toward some object [3]. It is possible to identify two subtasks (among others) in sentiment analysis:

- **Polarity classification**: tell if a piece of text has a positive or negative sentiment.
- **Subjectivity detection**: aims to remove factual or neutral comments that lack sentiment [4].

Sentiment analysis can be performed at various levels: document level, sentence level and aspect level.
Document-level sentiment analysis is performed on a whole document and a single polarity is given to the whole document [5].
In sentence-level sentiment analysis, a polarity is given to each sentence and it is usually associated with two tasks. First, determine if a sentence is subjective or objective and then classify a subjective sentence to determine if it is positive or negative [6].
Aspect level sentiment analysis is performed on each aspect of a sentence: the goal is not only to determine text subjectivity and polarity but also what in particular the author liked or disliked about the subject of the sentence [6]. Certain words carry particular strong sentiment (either positive or negative). These lists of words can be called *sentiment lexicons*. These lexicons can be used to enrich the text representation for sentiment analysis applications by adding sentiment information to words in the text. It is worth noticing that different senses of the same term can have different opinion-related properties [7]. As an example, Esuli and Sebastiani [7] show that the word "estimable" can be considered objective (neutral) when its meaning is "may be computed or estimated", but it acquires a more positive connotation when the meaning is "deserving of respect or high regard".

3. Data Description & Analysis

For the two tasks, two datasets are given, one for subjectivity detection and one for polarity classification and both of them are from Pang and Lee. Both datasets are about movie reviews. The *subjectivity dataset* is composed of 10'000 sentences, 5'000 are subjective and 5'000 are objective, therefore the dataset is balanced, so accuracy can be safely used as a metric. The sentences are 24 words long on average, and the vocab contains 23906 words.
On the other end, the *polarity dataset* has 2'000 documents in total, 1'000 for negative reviews and 1'000 for positive reviews. Again since the dataset is balanced, accuracy can be safely used as a metric. Each element of the dataset is a document which is subdivided in sentences. Documents have on average a length of 33 sentences and the average sentence length is 24 over the whole dataset. The vocabulary is composed of 39768 words.
It is easy to see that both datasets come already tokenized, so there is no need to use a tokenizer (unless there are specific needs for tokenization).
For the LSTM-based models, I used GloVe [8] and FastText [9] word embeddings. I checked the intersection between the vocabulary of the two datasets and the vocabulary of the word embeddings to see how many words had a corresponding embedding.
For the *subjectivity dataset*, GloVe has an 85.93% vocabulary coverage[1] and a 97.65% text coverage [2], whereas FastText has an 81.73% vocabulary coverage and a 95.94% text coverage.
For the *polarity dataset*, GloVe has a 91.93% vocabulary coverage and 99.58% text coverage. For the same dataset, FastText has a coverage of 95.11% of the vocab and a coverage of 98.94% of the text.
By printing the out-of-vocabulary words in the *subjectivity dataset* it was possible to notice a pattern: all contractions (e.g. "aren't", "could've" ...) have been recognized as out of vocabulary, so after expanding these contractions using a predefined contraction map (e.g. "aren't" $\rightarrow$ "are", "not") I was able to get a little increase in the coverage for both embeddings. Similar reasoning can be applied to the *polarity dataset* where some of the words were surrounded by underscores (e.g. "_the", "_really_" ...) so again after cleaning these words I was able to get a little increase in the coverage for the embeddings.

4. Model

As briefly stated in the introduction, I am going to implement some models. Multinomial Naive Bayes will be the baseline for both: subjectivity detection and polarity classification. Then I implemented a BiLSTM with and without attention mechanism and finally, a Lexicon-Based Supervised Attention Model [1] which method that uses lexicons to guide supervision when us-

[1] **vocabulary coverage** is computed by taking the size of *set* of words in the dataset being covered by the embedding divided by the size of the vocabulary

[2] **text coverage** is computed by dividing the number of words having the embedding in the dataset by the total number of words in the dataset, counting all word occurrences (in the vocabulary coverage each word is considered once)

ing BiLSTM. From now on I will refer to this last method as LBSA. LBSA is used only for the polarity classification since it expects that training samples are documents, not sentences. In contrast, all the other methods are employed both for subjectivity and polarity datasets.

4.1. LBSA

The architecture is shown in Figure 1 and it is easy to see that is composed of one word-level BiLSTM (the one in green) and one sentence-level BiLSTM, both of which make use of an attention mechanism. The first BiLSTM produces an embedding for each sentence in one document and each sentence of a single document is processed separately by this first BiLSTM. After that, each sentence embedding is passed to the second BiLSTM to obtain a document embedding which is then passed to a classifier. As a last layer of both BiLSTM, I have added a residual module, since in the worst-case scenario it will learn the identity function without affecting the performance, but it allows to learn more sophisticated embeddings. Then in addition to the cross entropy loss on the classifier head, there are two additional losses: the word attention loss relative to the first BiLSTM and the sentence attention loss relative to the second one. Both losses are computed using the attention vector resulting from the two BiLSTMs.

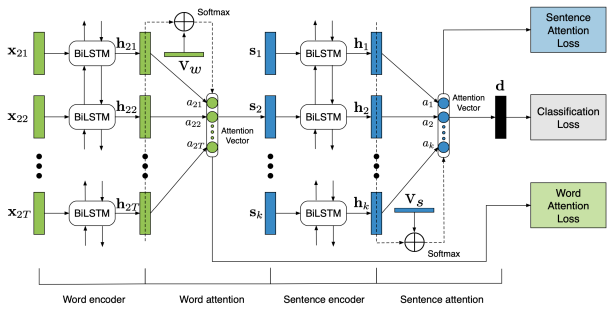

Figure 1: *Image depicting a sketch of the LBSA model*

More formally, let D be the set of documents. A document $d_m \in D$ contains k sentences $S_1, S_2, ..., S_k$. Each sentence S_i is a sequence of words $w_{i1}, w_{i2}, ...w_{it}, t \in [1, ...T_i]$ where T_i is the length of sentence S_i. After the first BiLSTM (word-level), we obtain pairs of hidden vectors $\overrightarrow{h_{it}}$ and $\overleftarrow{h_{it}}$. Then attention can be computed as follows:

$$h_{it} = (\overrightarrow{h_{it}}, \overleftarrow{h_{it}})$$

$$e_{it} = tanh(W_w h_{it} + b_w) \cdot v_w$$

$$a_{it} = \frac{exp(e_{it})}{\sum_{t=0}^{T_i} exp(e_{it})}$$

Then the sentence vector is computed:

$$s_i = \sum_{t=0}^{T_i} a_{it} h_{it}$$

After obtaining all sentence vectors s_i for a given document, $1 \leq i \leq k$, we can feed them in the sentence-level BiLSTM (Of course considering them as a part of the same sequence). After that, the document vector d is obtained [3].
The final document vector d is fed to a classifier layer, which is different from the one reported on the paper [1]: I have used a residual block because it doesn't require a lot more computation, but it can lead to better classification performance. For the sake of simplicity I am going to write the formula as a linear layer, but keep in mind that in the implementation, the residual block is used instead. The final probability estimation is:

$$p = softmax(W_c d + b_C)$$

where $p \in \mathbb{R}^C$ where C is the number of classes (in our case there are only two classes).

Then we have to compute the word-level and sentence-level *gold attention vectors* to add the supervision of the lexicon. Let L be a sentiment lexicon, having sentiment scores mapped in $score^L(w_{it}) \in [-1; 1]$ for each word, where -1 means *very bad* and *+1* means *very good*. In order to quantify the intensity of the sentiment, the *Sentiment Degree* $SD^L(w_{it})$ is defined as follows:

$$SD^L(w_{it}) = |score^L(w_{it})|, SD^L(w_{it}) \in [0, 1]$$

Then the **word-level gold attention vector** is computed:

$$a^*_{it} = \frac{exp(\lambda_w SD^L(w_{it}))}{\sum_{t=0}^{T_i} exp(\lambda_w SD^L(w_{it}))}$$

For each sentence we are going to compute the sentiment degree as follows:

$$SD^L(S_i) = \frac{\sum_{t=0}^{T_i} SD(w_{it})}{T}$$

which is simply the mean over all words in the sentence. And finally the **sentence-level gold attention vector**:

$$a^*_i = \frac{exp(\lambda_s SD^L(S_i))}{\sum_{j=0}^{k} exp(\lambda_s SD^L(S_j))}$$

These two **gold attention vectors** are used to compute the loss. Here I write the loss function **for the single sample**:

$$\mathcal{L} = -\sum_{c1}^{C} g_c log(p_c) + \mu_w \sum_{t=1}^{T} \Delta(a^*_t, a_t) + \mu_s \Delta(a^*, a)$$

where $g_c \in \mathbb{R}^C$ is the ground truth $p_c \in \mathbb{R}^C$ as stated before is the output probability of the model. a^*s are the gold attention vectors and as are word-level and sentence-level attention vectors. μ_w and μ_s are hyperparameters used to balance the training objective between loss and attention disagreement. I chose them to be 0.0005 and 0.025 respectively (slightly different with respect to the paper). Finally:

$$\Delta(a^*, a) = -\sum_{i=0} a^*_i log(a_i)$$

All of that said, the last thing to explain is how the score for each word (**sentiment degree**) is computed. As briefly mentioned before sentiment lexicons are used, more specifically I have used 5 different sentiment lexicons: SentiWordNet 3.0

[3] Formulas are skipped since they are exactly the same as for the word-level BiLSTM, obtaining one document vector for each document in the dataset

[10], mpqa effect lexicon [11], Bing Liu, Twitter lexicon [12] and SocialSent [13]. Then lexicons are combined by averaging the sentiment degrees of each word[4]. If a word appears on Mpqa, Bing Liu or Twitter is weighted as 1 for each of the lexicons it appears in. Whereas for SentiWordNet and SocialSent, contributions are computed differently. For SentiWordNet I kept the same method presented in the paper, that is summing contributions of negative and positive scores, considering the maximum and then mapping it to range $[0, 1]$ to let the scores be comparable with the others. With SocialSent, I took the scores and mapped them to range $[0, 1]$. Finally, all contributions from different lexicons are averaged for each word.

5. Evaluation

Evaluation is performed by comparing the proposed methods (i.e. BiLSTM, BiLSTM with attention and LBSA) with the baseline method obtained using the Multinomial Naive Bayes method. It is important to notice that LBSA is used only in Polarity Classification because it is required that each training instance is a document composed of sentences. Moreover, each result is an average of 10-fold cross-validation applied to each method. Folds indexes [5] are computed randomly and differently for each model. The metric used to assess the performance of each model is accuracy:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

I have chosen this metric since both datasets (subjectivity and polarity) are balanced. All deep methods are trained for 10 epochs each when trained on polarity and for 5 epochs each when trained on subjectivity detection, with the possibility to stop early the training if 100% accuracy is obtained on the training set on two consecutive epochs [6]. For each model, I have used Adam [14] as optimizer with the learning rate set to 10^{-3} and betas 0.9 and 0.999 respectively. In addition, to that, I have adopted an exponential learning rate scheduler with a gamma (i.e. a multiplicative factor of learning rate decay) of 0.8. The learning rate scheduler showed less noisy results (i.e. lower standard deviation).

5.1. Subjectivity Detection results

Model	Accuracy
Baseline	
Naive Bayes	90.75 ± 1.17
FastText Embedding	
BiLSTM	91.74 ± 0.74
BiLSTM wa	92.25 ± 0.67
GloVe Embedding	
BiLSTM	91.76 ± 0.57
BiLSTM wa	$\mathbf{92.4 \pm 0.79}$

Table 1: *Table with subjectivity detection results. "wa" means "with attention"*

[4] It is worth mentioning that if one word is in one lexicon but not in another one, the average is done by taking into account only lexicons where the word appears

[5] You can see each fold as a set of indexes for the original dataset

[6] I have noticed that when models reach 100% accuracy in the training set they do not improve or get worse anymore.

The baseline is obtained through Naive Bayes with negation marking. It is possible to appreciate that deep models have better accuracy with respect to the baseline and also these models showed to have a smaller standard deviation, meaning that they are more precise through folds. Also is worth mentioning that accuracy is slightly better when using attention, but the improvement is really small. This could be due to the fact that sentences are not really long, therefore limiting the usefulness of attention.
Another thing that can be seen is that the two models using GloVe embedding, are performing slightly better with respect to their FastText counterparts. A possible explanation of this behaviour can be given by the fact that GloVe has slightly better coverage of the vocabulary and text (Please for the explanation of text and vocabulary coverage refer to the one given in section 3-Data Description & Analysis), but given that the improvement is really small (It is also within the standard deviation), it is more likely that the difference between the two embeddings is only noise.
Even if Naive Bayes performs worse than deep models in subjectivity detection, I think that it is relevant to say that the time required to train a Naive Bayes model is really lower than deep models: Naive Bayes require just a few seconds on this dataset, whereas deep models took some minutes to converge. I think that this is important because, in a real-life application, training time might be relevant, for example, if the model needs to run on low-powered devices. In these situations, the Naive Bayes model will be a better alternative with respect to deep models because of the faster training and also the comparable results (It has been just slightly surpassed by BiLSTM-based models).-

5.2. Polarity Classification results

Firstly I am going to report Polarity classification results obtained by the previously mentioned methods, but without removing objective sentences from the corpus.

Model	Accuracy
Baseline	
Naive Bayes	81.65 ± 3.01
FastText Embedding	
BiLSTM	76.05 ± 2.82
BiLSTM wa	85.55 ± 2.66
LBSA	86.95 ± 2.11
GloVe Embedding	
BiLSTM	78.2 ± 2.83
BiLSTM wa	87.25 ± 2.27
LBSA	$\mathbf{87.45 \pm 1.96}$

Table 2: *Table with Polarity Classification*

First of all, it is easy to see that standard BiLSTM (without attention) is not able to surpass the baseline given by the Naive Bayes model. This can be due to the fact that documents are really long sequences (792 words per document on average), hence the model fails to focus on important parts of the text. To support this hypothesis, it can be seen that just by adding attention, the model is able to improve by almost 10 percentage points in accuracy in both embeddings. Talking about embeddings, it can be easily seen that GloVe performs better than FastText. This result can confirm my previous hypothesis that the coverage of the embedding impacts the performance or maybe the increase in performance may be due to the fact that GloVe

is able to encode more meaningful information for sentiment analysis.
Finally, it is possible to appreciate that both LBSA and BiLSTM with attention are able to outperform the baseline obtained using Naive Bayes. LBSA performs slightly better than BiLSTM with attention and it shows more stable results overall (lower standard deviation).
Now I am going to show the results obtained by removing objective sentences from the polarity dataset:

Model	Accuracy
Baseline	
Naive Bayes	85.35 ± 2.86
FastText Embedding	
BiLSTM	76.20 ± 3.03
BiLSTM wa	88.25 ± 2.17
LBSA	88.75 ± 1.55
GloVe Embedding	
BiLSTM	80.70 ± 2.50
BiLSTM wa	88.20 ± 1.79
LBSA	$\mathbf{89.45 \pm 1.76}$

Table 3: *Table with Polarity Classification*

As a result of removing objective sentences, all scores have been improved and the relative ranking between models remains the same. Again LBSA is the best-performing model and the one with less amount of variation in the performance through folds. This improvement can be due to the fact that by removing objective sentences the sequences are smaller and with less useless data. Even if objective sentences are removed, standard BiLSTM is still not able to surpass the baseline, this is probably to the fact that documents are still sequences that are too long.

5.3. further analysis

What I want to do in this subsection is to see why LBSA is performing better than BiLSTM with attention. To do that I have decided to inspect the attention vector of a sentence, produced by a BiLSTM with attention trained on the movie reviews dataset.

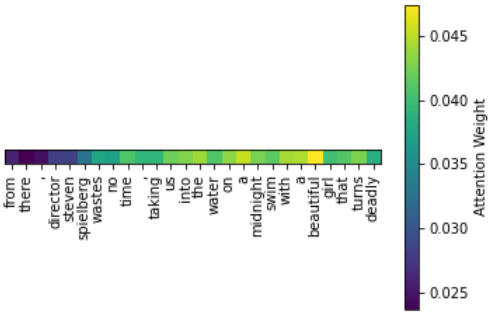

Figure 2: *Attention map of BiLSTM with attention*

Then I decided to compare this with the gold attention vector of the LBSA method for the same sentence:

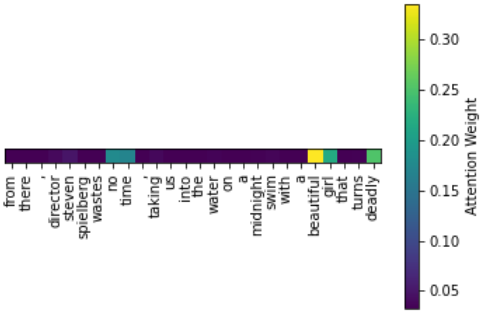

Figure 3: *gold attention vector of the LBSA method*

It is possible to see that the BiLSTM with attention is able to focus more on the word with a higher sentiment degree, which is the word "beautiful" so it is able to give attention to the most important information in the text, hence it performs similarly to LBSA, but it also gives a lot of weight to words which do not carry sentiment information, such as "the" or "a", which are just common words in the dataset. As can be seen, by the gold attention vector heatmap, LBSA adds supervision on sentiment lexemes, guiding the attention to focus more on parts of the text containing sentiment information. This allows LBSA to obtain slightly better performance with respect to the BiLSTM with attention.

6. Conclusion

In this work, I have shown how methods based on neural networks are able to perform better with respect to shallow counterparts both for subjectivity detection and polarity classification. Regarding subjectivity detection, BiLSTM with attention is able to reach the best results, compared to standard BiLSTM and Naive Bayes, even if by a small margin. In polarity, classification results are also in favour of deep models using attention, but this time with a larger margin. This is most probably due to the fact that attention helps focus on important parts of the document and it has been shown to be particularly important when dealing with long sequences (such as the ones in the polarity dataset). Moreover, the inclusion of sentiment lexicons supervision helps in guiding the attention to focus more on relevant parts of the sentences. There are some open issues that are not reported in this work, but I think are worth mentioning for a further extension of the project. For example, even if LBSA performed better in all the experiments (10-fold cross-validated) with respect to BiLSTM with attention, it would be nice to see if there is a statistical significance between the two by performing the paired t-test [15]. Another insight I want to give is that if there is a statistical significance between the two previously mentioned models, can we improve the BiLSTM with attention? For example by adding the word-level gold attention vector as an additional source of supervision, or by adding a loss component such as entropy minimization on the attention vector, encouraging attention to be more similar to a delta function (i.e. more sure about the part to which it gives attention).

7. References

[1] Y. Zou, T. Gui, Q. Zhang, and X.-J. Huang, "A lexicon-based supervised attention model for neural sentiment analysis," in *Proceedings of the 27th international conference on computational linguistics*, 2018, pp. 868–877.

[2] B. Pang and L. Lee, "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts," *arXiv preprint cs/0409058*, 2004.

[3] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 1st ed. USA: Prentice Hall PTR, 2000.

[4] I. Chaturvedi, E. Cambria, R. E. Welsch, and F. Herrera, "Distinguishing between facts and opinions for sentiment analysis: Survey and challenges," *Information Fusion*, vol. 44, pp. 65–77, 2018.

[5] M. Wankhade, A. C. S. Rao, and C. Kulkarni, "A survey on sentiment analysis methods, applications, and challenges," *Artificial Intelligence Review*, pp. 1–50, 2022.

[6] A. Westerski, "Sentiment analysis: Introduction and the state of the art overview," *Universidad Politecnica de Madrid, Spain*, pp. 211–218, 2007.

[7] A. Esuli and F. Sebastiani, "Sentiwordnet: A publicly available lexical resource for opinion mining," in *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, 2006.

[8] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[9] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the association for computational linguistics*, vol. 5, pp. 135–146, 2017.

[10] S. Baccianella, A. Esuli, and F. Sebastiani, "Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining," in *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, 2010.

[11] Y. Choi and J. Wiebe, "+/-effectwordnet: Sense-level lexicon acquisition for opinion inference," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1181–1191.

[12] D. Tang, F. Wei, B. Qin, M. Zhou, and T. Liu, "Building large-scale Twitter-specific sentiment lexicon : A representation learning approach," in *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin, Ireland: Dublin City University and Association for Computational Linguistics, Aug. 2014, pp. 172–182. [Online]. Available: https://aclanthology.org/C14-1018

[13] W. L. Hamilton, K. Clark, J. Leskovec, and D. Jurafsky, "Inducing domain-specific sentiment lexicons from unlabeled corpora," in *Proceedings of the conference on empirical methods in natural language processing. conference on empirical methods in natural language processing*, vol. 2016. NIH Public Access, 2016, p. 595.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[15] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural computation*, vol. 10, no. 7, pp. 1895–1923, 1998.