Explainable Sentence-Level Sentiment Analysis for Amazon Product Reviews

Salvatore Stefano Furnari
Politecnico di Torino
Torino, Italy
s290057@studenti.polito.it

Giuseppe Gallipoli
Politecnico di Torino
Torino, Italy
s291086@studenti.polito.it

Marco Tasca
Politecnico di Torino
Torino, Italy
s285174@studenti.polito.it

***Abstract*—In the last few years, the e-commerce market has experienced an exponential growth, with the direct consequence of an unprecedented generation of a huge amount of data. One of the most important insights that can be extracted from a product review is the writer's feeling: this is the objective of sentiment analysis task. Among its variants, we focus on sentence-level sentiment analysis with an additional study of the model's interpretability. In particular, our model is based on a BiLSTM architecture enriched by a sentiment lexicon and an attention mechanism. The sentiment lexicon is used to take into consideration the sentiment polarity of each word while the attention layer allows to assign weights to words both within a single sentence and across an entire corpus, with a particular focus on the most relevant aspect terms and sentimental words. We conduct experiments on three benchmark datasets: the first contains product reviews, the second movie reviews and the third product reviews written in different languages for which we implement a multilingual extension of our method. In all three cases, the proposed approach achieves promising results that demonstrate the effectiveness of the solution and its capability of handling reviews belonging to different text domains (i.e., products, movies) and in a multilingual setting. Code can be found at https://github.com/gallipoligiuseppe/SentiModel**

***Index Terms*—sentiment analysis, product reviews, BiLSTM, attention, interpretability**

I. Problem statement

In this project we perform sentence-level sentiment analysis on a dataset of Amazon product reviews: starting from the work of Li et al. [1], we extend it by considering both another benchmark dataset and a multilingual variation.
The task consists of extracting the writer's emotions, feeling and opinion from a piece of text: this objective is particularly useful in the context of e-commerce, where customers are eager for information and opinions regarding the products they could buy. For this reason, customers' reviews are welcome and encouraged by the platform owners. However, if the number of reviews keeps growing, users could get lost in the massive amount of opinions, having difficulties in grasping the general idea of the community. Thanks to sentiment analysis, sellers can automate the process of extracting the latent rating (e.g., from 1 to 5) or feeling (e.g., positive, negative or neutral) about the products, which represents the final message conveyed by the review. In this way, users can avoid reading huge amount of text and sellers can easily show statistics about the experience of previous customers.

Various techniques can be adopted, from traditional rule-based approaches to most recent ML and DL solutions: in this project we rely on DL architectures, both for automatic feature extraction and for the sentiment analysis model. Sentiment analysis can be classified both on the basis of the type of text from which we want to extract the feeling and on the desired level of granularity: in this work we focus on a sentence-level approach, by considering the whole review as a single sentence, and we assign a unique feeling for the entire text.
We rely on a supervised learning approach, which requires the availability of annotated samples to train the model. The dataset in use presents a score from 1 (very negative rating) to 5 (very positive rating) for each review: since we aim to perform binary classification to detect a positive or negative writer's feeling, we substitute scores 4 and 5 with label 1 (i.e., positive) and the lower ones with label 0 (i.e., negative).

In this work we also explore ways to make the predictions of the neural network we developed explainable: this is a critical point for DL solutions, where architectures are often seen as "black-boxes" for which it is difficult to understand the reasons leading to the produced output. For this reason, we attempt to define the importance of words in spotting a positive or negative sentiment, exploiting the attention weights learned by the corresponding layer in the network. Indeed, the implemented self-attention mechanism allows words in a sentence to look each other and to determine how much each word is crucial for writer's opinion.
We investigate explainability at two levels of granularity:

- document-level: by considering attention weights for each word in a review;
- corpus-level: by analyzing attention weights of aspect terms (i.e., entities involved in the description of a product) and sentimental words (i.e., adjectives and adverbs describing the qualities of a product) in the reviews corpus.

II. Methodology

A. Pre-processing

As a first operation, reviews text is cleaned by deleting analphabetic sign, lowercasing all words, applying lemmatization and stopwords removal. Considering the particular task of sentiment analysis, we preserve some terms that are generally

considered stopwords, such as negations (e.g., *no*, *not*) and negative forms of auxiliary and modal verbs (e.g., *isn't*, *can't*), which are crucial to preserve the actual meaning of sentences.

In order to identify the set of aspect and sentimental terms needed for the explainability study, we first apply POS tagging to each review, then we leverage TF-IDF to select the most relevant words across the whole corpus: we sort the words in decreasing order of score and then we filter the top-160 nouns for the aspect set and the top-160 adjectives and adverbs for the sentimental set, avoiding duplicates. According to TF-IDF, this allows us to identify words which are frequent in some specific reviews but less frequent in the whole corpus, in order to avoid considering meaningless words.

With the recent advancements of NN in NLP, it has been exploited their use for feature extraction: one of the most important techniques is word embedding, whose objective is to encode each word of the corpus vocabulary into a vector of latent features with a predefined size. In particular, we use a pre-trained BERT [2], which is an architecture for contextualized word embeddings, where each word is encoded into a different vector on the basis of the sentence it belongs to.

Moreover, we also apply a more traditional NLP technique to enrich the data representation given by BERT: we define a sentiment lexicon, which is a model based on human-generated features, tailored on the specific task of sentiment analysis. This can provide a high accuracy, but it can reduce the versatility of the model with respect to text domain. The lexicon we rely on is SentiWordNet [3], which outputs sentiment weights for each word of its vocabulary: specifically, each word can have multiple triplets of weights on the basis of its meaning into text, which are scores from 0 to 1 (summing to 1) with respect to a positive, negative or neutral feeling. In view of scaling the embedding vectors, we combine the weights of each triplet by defining a unique score in the range $[0.5, 1.5]$ with the following formula:

$$0.5 \cdot (pos_{score} - neg_{score} + 2) \tag{1}$$

This is done to avoid having 0 in the interval, otherwise some embeddings would be scaled by 0, making them indistinguishable. Furthermore, we assign a weight of 1 to words not present into the lexicon vocabulary and we average the unique scores defined above across all the triplets of the same word. Finally, word embeddings provided by BERT are multiplied by the corresponding sentiment weight.

B. Model architecture

Our model, displayed in Figure 1, mainly consists of a BiLSTM layer enriched by an attention layer and a softmax layer for final predictions.

Recurrent Neural Networks (i.e., RNN) are one of the most used architectures for NLP tasks, since they allow to perform feature extraction on sequential data such as text. Word vectors are transformed one by one by RNN layers, taking as input also the transformation of the previous word (i.e., the hidden state) to keep track of the sentence sequence across

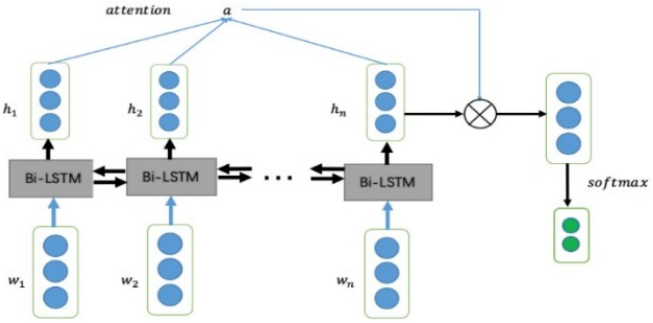

Figure 1. Model architecture

the different time steps. However, due to the well-known problem of vanishing gradient, backpropagation through a RNN encounters higher difficulties and satisfactory results can be achieved only with a limited input sequence.

To solve this issue, RNN layers are modified to turn the network into a Long Short-Term Memory (i.e., LSTM) network, which better keeps memory of the sentence sequence and mitigates the vanishing gradient, even with longer input sequences. We adopt a bidirectional implementation, where the input sequence is processed both from left to right and in the backward way. This allows the model to have a more complete view of the sentence, generating hidden states which take into account both the preceding and the subsequent word.

After the input sequence has gone through the BiLSTM layer, self-attention mechanism is applied on the final hidden states: please note that the attention layer implemented in this work is different from the one of Transformer architecture. First, attention scores are computed for each hidden state h_i as follows:

$$u_i = tanh(W \cdot h_i + b) \tag{2}$$

where W is a weight matrix and b is a bias vector. Then, we normalize the dot products between attention scores u_i and the attention score of the final hidden state of the [CLS] token embedding u_w through a softmax function to get the attention distribution. Indeed, the [CLS] embedding provided by BERT acts as a global context vector which takes into account the characteristics of the whole sequence. Mathematically, attention weights for each word are defined as follows:

$$a_i = \frac{e^{u_i^T \cdot u_w}}{\sum_i e^{u_i^T \cdot u_w}} \tag{3}$$

Having done so, we scale the output of the BiLSTM layer for the corresponding a_i and we send the scaled hidden state of the last time step to a dense layer with ReLU activation function which halves the number of features. Finally, we perform binary classification through the final softmax layer.

During attention weights computation, if the review which is being processed at that time contains one of selected words by TF-IDF, the computed a_i is stored in view of the interpretability study which will be detailed in Section III-E2.

C. Loss function

The model is trained by optimizing a linear combination of a Cross Entropy (i.e., CSE) function and a Mean Square Error (i.e., MSE) function. In particular, for each batch, we have:

$$CSE = -\frac{1}{N}\sum_i y_i \cdot log(f(x_i)) + (1-y_i)\cdot log(1-f(x_i)) \quad (4)$$

which is the usual cross-entropy loss, where N is the batch size, y_i is the ground truth (0 for negative, 1 for positive), x_i is the review sample and $f(x_i)$ is the probability that x_i has a positive sentiment, as assigned by the softmax layer.

$$MSE = \sqrt{\frac{1}{N}\sum_i (y_i - f(x_i))^2} \quad (5)$$

with the terminology above. Indeed, if the ground truth is 1, we want $f(x_i)$ to be the highest possible, whereas if the ground truth is 0 we want the opposite.

So the loss function to be minimized is the following:

$$\mathcal{L} = \frac{1}{2}CSE + \frac{1}{2}MSE \quad (6)$$

III. Experiments

A. Dataset

Our network is trained on the Amazon Musical Instruments Reviews Dataset [4], which consists of 10.261 samples of text. The class distribution is highly unbalanced, with a predominance of positive reviews which amount to the $\approx 88\%$ of the total (considering as positive the reviews labelled with ratings 4 and 5). This will probably lead the model to mostly predict a positive sentiment, so achieving very high performance with respect to this class but having difficulties to infer the reviews associated with a negative sentiment.

B. Performance metrics

We consider the usual metrics for a classification problem. For each class, let TP, FP, TN, FN the number of true/false positives and true/false negative, respectively, we define accuracy a, precision p, recall r and F_1 score as follows:

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

$$p = \frac{TP}{TP + FP} \qquad r = \frac{TP}{TP + FN} \qquad F_1 = \frac{2 \cdot r \cdot p}{r + p}$$

The above metrics (apart from accuracy) are defined for each class but they can also be aggregated to obtain a unique score to evaluate the global performance (i.e., on both classes) of the model. In our experiments, we report results both for the positive class, according to [1], and we aggregate metrics using `macro` average.

Table I
NUMBER OF EPOCHS

n_epochs	batch_size	dropout_rate	`macro` F_1	$F_{1_{pos}}$
8	32	0.2	0.593	0.929
10	32	0.2	0.620	0.923
12	32	0.2	0.535	0.933
17	32	0.2	0.614	0.904

Table II
BATCH SIZE

n_epochs	batch_size	dropout_rate	`macro` F_1	$F_{1_{pos}}$
10	32	0.2	0.620	0.923
10	34	0.2	0.663	0.931
10	36	0.2	0.597	0.928
10	38	0.2	0.616	0.929

C. Hyperparameters tuning

We divide the dataset into three parts: 70% is used to train the model while the remaining 30% is equally divided into a validation set, which is used to tune the hyperparameters, and a test set, which is used to evaluate the model's performance. All splits are performed in a stratified fashion, thus preserving the original class distribution.

We consider three hyperparameters: number of epochs, batch size and dropout rate. Since performing a complete grid search on the configuration space would be onerous, we explore the different configurations following a sequential approach: by keeping fixed the batch size and the dropout rate, we first try different values for the number of epochs and we fix it to the value achieving the best results, then we repeat the procedure with the remaining two hyperparameters.
As a starting configuration, we consider `n_epochs=8`, `batch_size=32` and `dropout_rate=0.2`.

Since it is important to assess the performance on both classes, configurations are primarily evaluated on the `macro` F_1 score average and, in case of a tie, on the F_1 score for the positive class.

1) Epochs: The number of epochs defines how many iterations over the training set are performed, in order to search the configuration of hyperparameters achieving the optimum of the loss function. We need to find a significant trade-off between a low number, which does not allow to properly learn a good model for the task, and a high number, which leads the network to overfitting over the training dataset. This mostly depends on the size and variety of data and on the complexity of the model (i.e., the number of parameters that it has to learn). From Table I, we fix `n_epochs=10`.

2) Batch size: This represents the number of samples on which we compute the loss function and backpropagate its gradient to update model's parameters, so optimizing the loss through Stochastic Gradient Descent. This trade-off involves two aspects: a higher number allows to better optimize the loss function, since the estimation of the gradient improves, but it can negatively impact on the computational performance if not enough memory resources are available. From Table II, we set `batch_size=34`.

Table III
DROPOUT RATE

n_epochs	batch_size	dropout_rate	macro F_1	$F_{1_{pos}}$
10	34	0.2	0.663	0.931
10	34	0.4	0.614	0.924
10	34	0.6	0.540	0.930
10	34	0.8	0.497	0.935

Table IV
BILSTM + ATTENTION – RESULTS ON TEST SET

a	p_{pos}	r_{pos}	$F_{1_{pos}}$	macro F_1
0.943	0.916	0.988	0.951	0.687

3) Dropout rate: Dropout is a powerful technique to improve the generalization ability of the model, by removing at random at each training iteration a certain percentage (i.e., the dropout rate) of cells in the hidden layers of the network. This strategy induces the network to make good predictions even in the absence of features extracted by certain neurons, making the model more robust. However, a too high rate could lead the network to update rarely cells weights and so falling into underfitting. Finally, from Table III, we fix `dropout_rate=0.2`.

The best configuration found is `n_epochs=10`, `batch_size=34` and `dropout_rate=0.2`. The model is then evaluated on the test set obtaining the results displayed in Table IV. As it is possible to see from `macro` F_1 score, the model does not achieve particularly high results: this is due to the severe imbalance in the dataset which causes predictions to be biased towards the positive class, showing instead poor performance in identifying negative reviews. This is also confirmed by the high scores of the different metrics for the positive class and it underlines the importance of considering, especially in case of unbalanced datasets, appropriate metrics such as `macro` F_1 to correctly assess the quality of a model.

D. Comparison with baselines

To have a better awareness of the performance reached by our model, we compare it with three baselines: SVM, Multinomial Naive Bayes and LSTM. In particular, the first two implement a traditional machine learning approach, using only TF-IDF scores computed on the reviews corpus after pre-processing, while the third one uses the same architecture of our network with the only difference that the LSTM layer is not bidirectional. As for the hyperparameters configurations of SVM and Naive Bayes, they are left to their default values in the adopted implementation, while LSTM uses the same best configuration found for our model.

Baselines performance on the test set are reported in Table V: results are again affected by the problem of class imbalance, nonetheless we can see that the proposed model outperforms the considered baselines with a $\approx 20\%$ improvement on the `macro` F_1 score. This highlights the efficacy of the method even though its potentiality can not be fully appreciated due to the unbalanced dataset in use. This is one of the main motivations for applying the model on another, possibly balanced, dataset, as discussed in Section IV.

Table V
BASELINES – RESULTS ON TEST SET

baseline	a	p_{pos}	r_{pos}	$F_{1_{pos}}$	macro F_1
SVM	0.913	0.881	0.979	0.927	0.489
NB	0.910	0.874	0.981	0.924	0.468
LSTM	0.908	0.910	0.922	0.916	0.615

Figure 2. Sentence attention weights – sample review

Further details on the execution environment (i.e., hardware and libraries used), execution times and the full set of model's parameters can be found in the Appendix.

E. Model explainability

As previously anticipated, we investigate model's interpretability at two levels of granularity.

1) Sentence-level attention weights analysis: After network training, we show the attention weights distribution for test reviews extracted at inference time: this allows to identify which words in a review have greater importance according to the network. Figure 2 shows the attention weights distribution for a sample review and it can be noticed that the words associated with higher weights are either sentimental terms (e.g., *nice*, *good*) or the related aspects (e.g., *cable*, *material*) while verbs and less relevant nouns (e.g., *make*, *foot*) have less importance in the sentence.

2) Aspect terms and Sentimental words attention weights analysis: During network training, we store the attention weights computed by the attention layer of the words belonging to the aspect and sentimental terms sets: in particular, only the weights extracted during the last epoch are stored, so as to consider the knowledge of the network at the end of the training process. Each word can appear more than once in the whole reviews corpus and it will probably have different attention weights depending on the specific review: for this reason, we need to aggregate these weights and this can be done considering their maximum, sum or average. The third approach is the one we follow in our experiments. Figures 3 and 4 display the top-10 attention weights for both sets of words: specifically, the most relevant terms of the first set all belong to the music field, with some technical terms (e.g., *impedance*, *reverb*), while the sentimental words that emerge from the second plot are both common words to express a positive sentiment (e.g., *best*, *cool*) and adjectives to describe the qualities of an article (e.g., *affordable*, *durable*). It can also be noticed that all the selected words are positive and this is due to the class imbalance already mentioned.

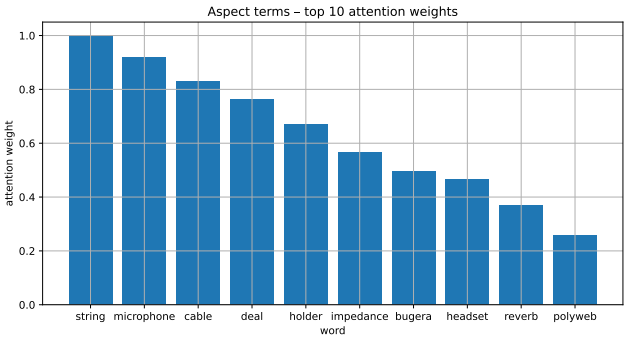

Figure 3. Aspect terms attention weights

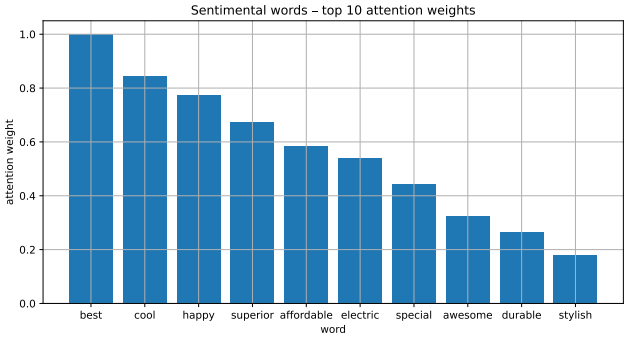

Figure 4. Sentimental words attention weights

IV. Extension I: Data Enrichment

Motivated by the moderate performance achieved by the model caused by the highly unbalanced dataset used, we decide to test the implemented approach on another dataset. By doing this, we can also verify its applicability on reviews of a different domain.

A. Dataset

In this section of our work, the network is trained on the IMDb Movie Reviews Dataset [5], which consists of 50.000 movie reviews. The dataset is perfectly balanced, so there is an equal number of samples associated with a positive and a negative sentiment. Due to computational constraints, the size of the dataset is reduced from its original size to its half, so we consider 25.000 samples with the original class distribution.

B. Pipeline variations

Since the dataset was collected from IMDb website, reviews texts are equipped with HTML tags that need to be filtered out to avoid the presence of meaningless tokens that could alter the following steps of the pipeline. After parsing the whole reviews corpus, the same pre-processing operations described in Section II-A are applied before the training of the network.

C. Experiments

The dataset is split following the same methodology of Section III-C and the same set of hyperparameters is considered. Results on the validation set are reported in Table VI

Table VI
Hyperparameters tuning – IMDb Movie Reviews Dataset

n_epochs	batch_size	dropout_rate	macro F_1	$F_{1_{pos}}$
8	32	0.2	0.863	0.860
10	32	0.2	0.853	0.845
12	32	0.2	0.842	0.839
8	34	0.2	**0.867**	0.860
8	34	0.4	0.864	0.862
10	34	0.2	0.853	0.846
12	34	0.2	0.843	0.851
17	34	0.2	0.854	0.845
8	36	0.2	0.861	0.865
10	36	0.2	0.851	0.847
12	36	0.2	0.854	0.848
8	38	0.2	0.861	0.867
10	38	0.2	0.864	0.860

Table VII
Results on test set – IMDb Movie Reviews Dataset

	a	p_{pos}	r_{pos}	$F_{1_{pos}}$	macro F_1
BiLSTM + Attention	0.886	0.866	0.894	0.880	**0.885**
SVM	0.810	0.785	0.832	0.808	0.806
NB	0.791	0.796	0.782	0.789	0.791
LSTM	0.831	0.812	0.861	0.836	0.829

from which we find as the best configuration `n_epochs=8`, `batch_size=34` and `dropout_rate=0.2`.

D. Results

The model is evaluated on the test set together with the same baselines defined in Section III-D. As it is possible to see from the results in Table VII, the proposed solution outperforms the other approaches considered. Furthermore, it shows a $\approx 20\%$ improvement with respect to the results obtained on the Amazon Musical Instruments Reviews Dataset: this is thanks to the availability of a bigger dataset (from 10.261 to 25.000 samples) which is also balanced, thus allowing the network to learn correctly how to classify positive and negative reviews.

Interpretability analysis can be found in the Appendix.

V. Extension II: Multilingual Extension

An important feature that can enlarge the range of applicability of a model is the possibility to apply it in a multilingual context. We implement one of the possible techniques which allow to deal with multilingual data, which is a fundamental requirement in today's world where global e-commerce platforms are extremely common.

A. Dataset

The dataset we consider is the Multilingual Amazon Reviews Corpus [6] which contains reviews in six different languages and each record includes several fields: the review text, the review title, the star rating (from 1 to 5), two anonymized reviewer and product IDs, the coarse-grained product category (e.g., *books*, *electronics*) and the review language. The corpus is balanced across stars, so each star rating constitutes 20% of the reviews in each language. For each language there are 200.000 samples but, due to computational limitations, we

Table VIII
HYPERPARAMETERS TUNING – MULTILINGUAL AMAZON CORPUS

n_epochs	batch_size	dropout_rate	macro F_1	$F_{1_{pos}}$
8	32	0.2	0.816	0.815
10	32	0.2	0.804	0.801
12	32	0.2	0.797	0.806
8	34	0.2	0.830	0.831
8	34	0.4	0.830	0.822
10	34	0.2	0.821	0.821
12	34	0.2	0.807	0.818
8	36	0.2	**0.835**	0.835
8	36	0.4	0.824	0.817
10	36	0.2	0.823	0.818
12	36	0.2	0.811	0.821
8	38	0.2	0.824	0.821
10	38	0.2	0.811	0.816
12	38	0.2	0.801	0.809

restrict it to 30.000 reviews: in particular, we consider reviews written in four languages (i.e., English, French, Spanish and German) belonging to the *home* product category.

To allow a better comparison with the results obtained in Section IV-D, we need to consider a balanced portion of the dataset: since the corpus is balanced across stars and the same is almost verified within the chosen product category, we filter out reviews with star rating 3 and we map those with 4 and 5 stars to the positive class and the ones with rating 1 and 2 to the negative class. By doing this, the final subset we consider is balanced both across languages and, for each language, the class distribution is also balanced.

B. Pipeline variations

After removing fields which are not useful for our task, we handle the contemporaneous presence of reviews written in different languages by defining a target language into which we translate all the samples. Specifically, since word embeddings are generated by a BERT model pre-trained on the English language, we choose English as target language. Translation is performed by relying on pre-trained Opus-MT [7] models, which are based on the Neural Machine Translation framework Marian [8]. Since the translation process is computationally demanding, the translated version of the dataset is serialized into a file to allow for faster successive executions. After the reviews translation, the same preprocessing steps described in Section II-A are applied before the network training.

C. Experiments

The dataset is split following the same methodology of Section III-C and the same set of hyperparameters is considered. Results on the validation set are reported in Table VIII from which we find as the best configuration `n_epochs=8`, `batch_size=36` and `dropout_rate=0.2`.

D. Results

The best configuration found is then evaluated on the test set together with the baselines defined in Section III-D. Results are reported in Table IX: the developed model achieves better performance than the baselines considered, thus confirming

Table IX
RESULTS ON TEST SET – MULTILINGUAL AMAZON CORPUS

	a	p_{pos}	r_{pos}	$F_{1_{pos}}$	macro F_1
BiLSTM + Attention	0.852	0.850	0.852	0.851	**0.854**
SVM	0.782	0.787	0.775	0.781	0.782
NB	0.777	0.770	0.786	0.778	0.776
LSTM	0.813	0.801	0.832	0.816	0.812

the superiority of the proposed solution even in a multilingual context. This is also thanks to the high-performing translation model which was employed to obtain reviews in a unique language while the slight decrease with respect to the results in Table VII is probably caused by the fact that, in some cases, the sentiment of a review is not perfectly preserved during the translation process.

Interpretability analysis can be found in the Appendix.

VI. CONCLUSIONS AND TAKEAWAYS

Online retail data is a valuable resource both for retailers and customers that can exploit the available data to guide their decisions: here sentiment analysis comes into play, allowing to understand reviewers' opinions on a certain product.

In this work, we perform sentiment analysis by means of a neutral network based on a BiLSTM architecture, enriched by a sentiment lexicon that allows to take into account the polarity of words. We test our method on three benchmark datasets obtaining overall satisfactory results: in particular, the first dataset is the most challenging due to a severe imbalance in the class distribution. This highlights the need for handling unbalanced data via ad-hoc techniques or by enriching the available number of samples, for example through a generative model that could represent a future research direction. The efficacy of the solution is then evaluated on a second dataset, demonstrating its applicability also on the movie reviews domain. Finally, the model is extended to allow the processing of multilingual data: this is done by leveraging a Machine Translation approach which allows to achieve good results.

To conclude, we also focus our study on the model's explainability by the definition of attention weights: interpretability analysis is performed both at a document and at a corpus level. This is a critical point for DL solutions and, in our task, it allows to understand which are the most relevant aspects in terms of products and features on which customers focus more. This can be of particular interest for producers and retailers which can use this knowledge to drive their business decision-making.

References

[1] Xuechun Li et al. "Explainable Sentence-Level Sentiment Analysis for Amazon Product Reviews". In: *CoRR* abs/2111.06070 (2021). arXiv: 2111.06070. URL: https://arxiv.org/abs/2111.06070.

[2] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805 (2018). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

[3] Stefano Baccianella, Andrea Esuli, and Fabrizio 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)*. Valletta, Malta: European Language Resources Association (ELRA), May 2010. URL: http://www.lrec-conf.org/proceedings/lrec2010/pdf/769_Paper.pdf.

[4] Julian J. McAuley et al. "Image-based Recommendations on Styles and Substitutes". In: *CoRR* abs/1506.04757 (2015). arXiv: 1506.04757. URL: http://arxiv.org/abs/1506.04757.

[5] Andrew L. Maas et al. "Learning Word Vectors for Sentiment Analysis". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2011, pp. 142–150. URL: http://www.aclweb.org/anthology/P11-1015.

[6] Phillip Keung et al. "The Multilingual Amazon Reviews Corpus". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. 2020.

[7] Jörg Tiedemann and Santhosh Thottingal. "OPUS-MT — Building open translation services for the World". In: *Proceedings of the 22nd Annual Conferenec of the European Association for Machine Translation (EAMT)*. Lisbon, Portugal, 2020.

[8] Marcin Junczys-Dowmunt et al. "Marian: Fast Neural Machine Translation in C++". In: *Proceedings of ACL 2018, System Demonstrations*. Melbourne, Australia: Association for Computational Linguistics, 2018, pp. 116–121. URL: http://www.aclweb.org/anthology/P18-4020.

APPENDIX

MODEL EXPLAINABILITY

We perform the same interpretability analysis of the model also on the two additional datasets we considered in Sections IV and V. In particular, for each dataset we report both the attention weights of one sample review and the attention weights distribution of the words belonging to the sets of aspect terms and sentimental words.

A. IMDb Movie Reviews Dataset

ix	1	2	3	4	5	6	7	8	9	10
word	davis	electrify	performance	hard	remember	female	player	perfect	part	wonderful
attention weight	0.047242	0.132870	0.202740	0.018956	0.038704	0.055603	0.108019	0.127337	0.150193	0.118335

Figure 5. Sentence att. weights – IMDb dataset sample review

As it is possible to see from the sample review shown in Figure 5, the words showing higher weights are either nouns (e.g., *performance*, *part*) or adjectives that convey a certain sentiment (e.g., *electrifying*, *perfect*) while less relevant terms are associated with lower weights. For visualization purposes, the sampled review has been truncated to the first 10 tokens.

Top-10 attention weights for aspect terms and sentimental words are displayed in Figures 6 and 7: from the first it is possible to notice that they are mainly cinematic terms (e.g., *rating*, *spoiler*), with the addition of some words probably related to movie characters or titles (e.g., *superhero*, *Matrix*). As for the second plot, almost all the selected words are adjectives commonly used to express a feeling but there are also more specific terms involving movie genres (e.g., *romantic*, *western*) or the nationality of a film/actor (e.g., *French*). From both plots we can see that, differently from the results obtained on the positively-biased Amazon Musical Instruments Reviews Dataset, negative aspects and sentimental words also appear (e.g., *waste*, *boring*).

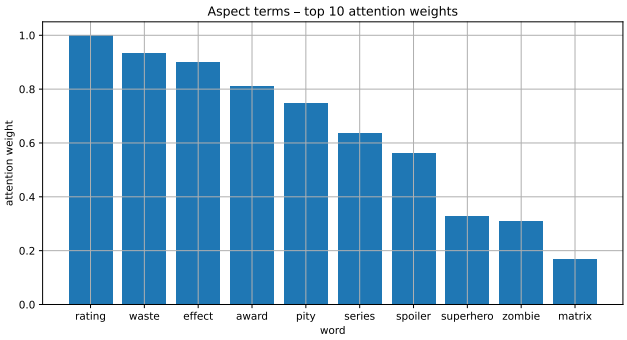

Figure 6. Aspect terms att. weights – IMDb Movie Reviews Dataset

B. Multilingual Amazon Reviews Corpus

Figure 8 shows the translated version of a sample review and its corresponding attention weights: also in this case the network seems to focus more both on words describing the product bought by the customer (e.g., *coffee*) and on

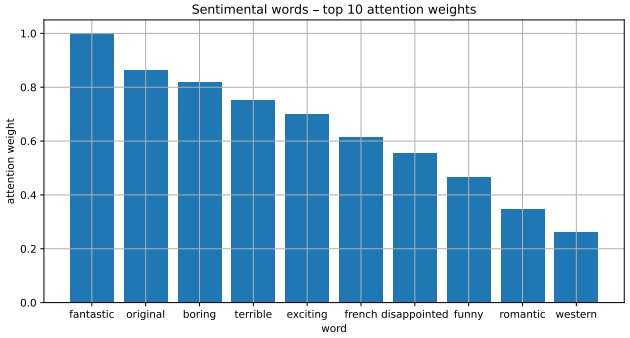

Figure 7. Sentimental words att. weights – IMDb Movie Reviews Dataset

ix	1	2	3	4	5	6	7	8	9	10
word	disappointed	enough	coffee	mill	throw	half	away	not	recommend	product
attention weight	0.158814	0.049000	0.119626	0.019194	0.060020	0.031756	0.129640	0.181143	0.156706	0.094101

Figure 8. Sentence att. weights – Amazon Multilingual dataset sample review

adjectives or verbs expressing the sentiment of the text (e.g., *disappointed*, *not recommend*).

The attention weights distribution of aspect terms and sentimental words are shown in Figures 9 and 10: the words displayed in the first plot involve both product features that, according to the model, have a higher importance (e.g., *design*, *price*) and other terms, maybe better refining the product category (e.g., *wedding*, *gadget*). Among sentimental words, it is possible to find both terms expressing the reviewer's opinion (e.g., *great*) and adjectives defining the qualities of the item (e.g., *practical*, *reliable*). Also in this case, negative aspects and sentimental words are included into the two sets (e.g., *defect*, *unsuitable*).

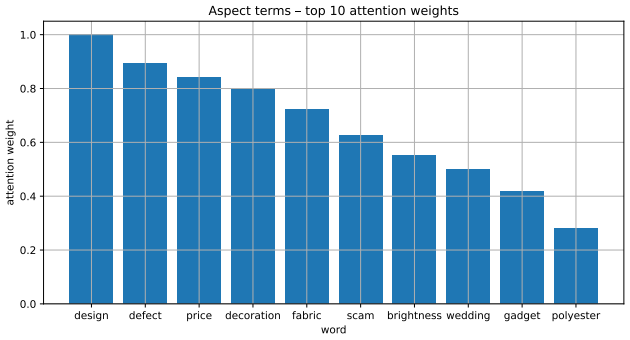

Figure 9. Aspect terms att. weights – Multilingual Amazon Corpus

DIFFERENCES WITH THE PAPER

With respect to the reference paper [1], due to partly missing information, we made the following assumptions/variations during our implementation:

- Samples from the Amazon Musical Instruments Reviews Dataset are mapped as follows: reviews with ratings $\{1, 2, 3\}$ to the negative class and ratings $\{4, 5\}$ to the positive class;

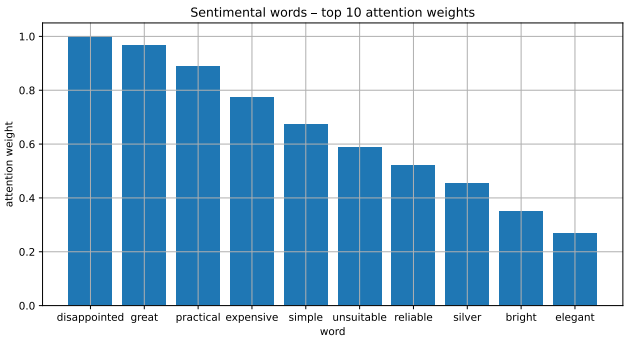

Figure 10. Sentimental words att. weights – Multilingual Amazon Corpus

- The sentiment lexicon is based on the scores retrieved from SentiWordNet since WordNet does not seem to have a publicly available version which can provide polarity scores;
- Polarity scores are aggregated according to the formula in 1 to produce a unique value in the desired range;
- As word embeddings technique, we use BERT instead of Word2Vec to exploit its ability to provide contextualized embeddings;
- As global context vector, we consider the [CLS] token embedding provided by BERT;
- In case of unbalanced datasets, reporting results only with respect to the majority class can lead to a wrong assessment of the performance of a model: for this reason, we define the classification metrics with their usual formulation for both classes. According to the reference paper, we report detailed results for the positive class but we also consider the performance of the method on the negative class, by reporting the `macro` F_1 score;
- As for the pipeline used by the baselines, we consider the approach described in Section III-D.

Execution environment

A. Hardware and execution times

The model was implemented and trained using the two following hardware configurations:

- NVIDIA Tesla K80 on Intel Xeon 2.3 GHz
- NVIDIA Quadro M4000 on Intel Xeon 2.6 GHz

Execution times of the best configurations are approximately the following, with no significant difference between the two hardware configurations:

- Amazon Musical Instruments Reviews Dataset: 7 min/epoch
- IMDb Movie Reviews Dataset: 17 min/epoch
- Multilingual Amazon Reviews Corpus: 30 min/language (translation process) + 20 min/epoch

B. Libraries

The model was developed in `Python 3.8` with the following libraries:

- `pandas 1.4.1`
- `numpy 1.22.3`
- `matplotlib 3.5.1`
- `seaborn 0.11.2`
- `Jinja2 3.1.0`
- `beatifulsoup 4.10`
- `nltk 3.7`
- `scikit-learn 1.0.2`
- `tensorflow 2.8.0`
- `tensorflow-hub 0.12`
- `tensorflow-text 2.8.1`
- `datasets 2.0.0`
- `easynmt 2.0.1`

Model's parameters

In the following table, there are listed the additional parameters used in the implementation of our solution: specifically, they concern TF-IDF technique, word embeddings, network architecture, network training and baselines.

Table X
Additional parameters

	parameter	value
TF-IDF	`min_df`	`5`
word embeddings	`max_seq_len`	`64`
	`embedding_size`	`768`
network architecture	`n_hidden_layers`	`1`
	`n_units`	`256`
	`recurrent_dropout`	`0.0`
network training	`optimizer`	`adam`
	`learning_rate`	`0.001`
	`validation_step`	`1`
SVM	`C`	`1`
	`kernel`	`rbf`
	`degree`	`3`
Multinomial Naive Bayes	`alpha`	`1`