Can hand-engineered linguistic features improve the performance of a neural model for span detection and sentiment analysis?

Florian Micliuc — Martin Kirkegaard — Magnus Sverdrup — Markus Kristoffer Sibbesen
flmi@itu.dk | marki@itu.dk | magsv@itu.dk | mksi@itu.dk

Abstract

This project examines the potential improvement of a neural model when doing span labelling for sentiment analysis, by using multiple feature extraction techniques. We test if the performance of a neural model for span detection and sentiment polarity classification is improved by the inclusion of hand-engineered linguistic features, such as affixes, part-of-speech tags and lemmatization groups, in addition to a pre-trained 100 dimensional word embedding. Compared to a baseline that only includes the word embedding, both the inclusion of learned affix representations and that of part-of-speech tags provide at best marginal improvements, if any. Including lemmatized forms of words likewise do not significantly alter the performance. We can thereby not demonstrate any significant benefit of these additional linguistic features, for the task of span labeling for sentiment analysis.

1 Introduction

The task of sentiment analysis has traditionally focused on whole-document classification, without consideration of multiple different sentiments being expressed in one text (Prabowo1 und Thelwall, 2009; Agarwal u. a., 2011). Later, however, more granular approaches emerged, such as *targeted* sentiment analysis, where a set of targets in a text are identified and the sentiments towards them classified according to polarity (Mitchell u. a., 2013; Zhang u. a., 2015). In this manner, since the sentence *"I found the hotel itself pleasant, but the traffic outside was unbearable"* cannot be labeled all-together positive or negative, it needs to be split into two sentiments with corresponding targets: a positive polarity regarding the hotel, but a negative regarding the traffic. This problem contains two distinct subtasks: *target extraction*, where the targets of opinions are identified, in this case *"hotel"* and *"traffic"*, and *polarity classification* where the sentiments regarding said targets are identified. These task are often combined in a joint model for solving both task and they frequently employ complex neural architectures together with pre-trained word embeddings, especially more recently (Li u. a., 2018; Zhou u. a., 2019). However, in researching how to improve performance, it may still be fruitful to look at each subtask individually, potentially even splitting them up further, since particular improvements may only affect one part of the problem and not the others. In this paper, we focus on an aspect of *polarity classification*, namely identifying spans of positive or negative polarity in texts that may express multiple sentiments. We seek to find out if manually constructed linguistic features can supplement pre-trained word embeddings when used to identify spans of different sentiment polarity.

2 Related Work

Traditionally, sequence labeling models have employed a mix of different feature engineering techniques, taking advantage of knowledge-based information, such as lexicons, gazetters and orthographic features (Yadav und Bethard, 2019). More recently, however, it use of large amounts of unlabeled data have made a larger array of unsupervised techniques for word representations accessible to researchers. The current state of the art approaches to sequence labeling make use of neural architectures with large pre-trained word embeddings as inputs. Recent advances in named entity recognition (NER) and sentiment analysis have shown that these pre-trained embeddings can be improved by bringing back older techniques:

Yadav u. a. (2018) demonstrated that appending affix representations to pre-trained word embeddings improved performance on NER tasks and achieved "state-of-the-art or near state-of-the-art"

performance on the CoNLL 2002/2003 datasets for multiple languages, and DrugNER datasets, respectively. Their base model also employed a BiLSTM character-based encoding of the words, demonstrating that common affix embeddings carry additional useful information, in addition to pure character-based embeddings.

Part-Of-Speech tags (POS-tags) have been utilised to enhance the performance of neural architectures in a number of instances, especially among less supported languages, like in Thai where Pasupa und Ayutthaya (2019) improved performance of a sentiment analysis task by appending, among other things, POS-tags to a pre-trained word embedding.

We amend these two features with a third, based on lemmatization. Here, we include the lemma, the unconjugated base of a word, where applicable.

3 Data

The data used in this project comes from the SemEval-2022 shared task 10 competition. It consists of a blend of hotel reviews, as mentioned in Barnes u. a. (2022), which were taken from multiple booking websites in the span of one year, from November 2012 to November 2013. (Agerri u. a., 2013)

Training data is comprised of 1744 reviews and the validation data, 249 reviews. The data is converted into a BIO-scheme, where each word is labelled depending on the polarity and where in the span it is:

B-Negative	beginning, negative meaning
B-Positive	beginning, positive meaning
I-Negative	inside, negative meaning
I-Positive	inside, positive meaning
B-Positive — I-Positive	overlapping span
B-Negative — I-Negative	overlapping span
I-Positive — B-Positive	overlapping span
I-Positive — I-Positive	overlapping span
O	out of the chunk

Table 1: The different BIO-labels in our dataset. A span begins with a B-label and all subsequent words in the span have an I-label. A span must be entirely either positive or negative

4 Methodology

The basic methodology of our experiments consists of comparing a baseline model with models that have additional features appended to the input.

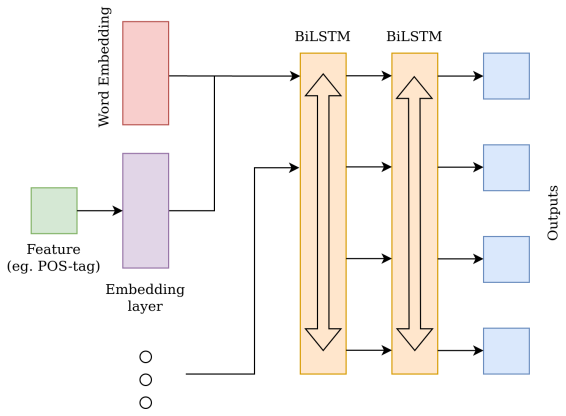

Figure 1: A visualisation of the model. The pre-trained word embeddings are concatenated with embedded linguistic features and passed through two BiLSTM layers

The baseline model of this project is a BiLSTM model which uses GloVe 100d (Global Vectors for Word Representation) as the word embedding. GloVe is a pre-trained model which provides semantic vector representations of words (Pennington u. a., 2014). The one we use was trained on Wikipedia 2014 and Gigaword 5. This is the base input to two BiLSTM layers, each with 50 neurons, using activation function *tanH*. The output layer is 9 neurons with activation function *softmax*.

When including categorical linguistic features, these are passed through an embedding layer initialized with random weights and output dimensions 10 and concatenated with the word embedding (Figure 1).

Model	BiLSTM
Training Data	OpeNER
Word Embedding	GloVe 100d
Hidden Layers	2
Neurons pr Hidden Layer	50
Learning Rate	0.01
Epochs	10
Activation Function Hidden Layer	TanH
Activation Function Output Layer	Softmax
Merge mode	sum

Table 2: Information about base model design, including hyperparameters

The hyperparameters for the baseline were chosen, by the highest macro average f1-score-score, after performing 126 experiments with multiple combinations (Table 2). These optimised values are used for each experiment, but some of the extra features have their own particular hyperparameters

which are tuned during their development. Since we are using a BiLSTM classifier, we get two outputs which needs to get merged in some way. This merging is what we call *merge mode*. Here we just sum the two values for each timestep, before passing them to the next layer.

4.1 Feature Extraction Techniques

Feature extraction can have a wide range of definitions. In the case of this paper, when talking about feature extraction, we are talking about *what additional information can one extract from a given token*

When trying to create an NLP-model, one would hope that the classifier was to pick up on the patterns in the unstructured data, and thus be able to quantify our language in a way so when new data is presented it has a generalized understanding of how to interpret it. Since we are using a neural network, then *Universal Approximation Theorem* states: "An arbitrarily wide Neural Network can approximate any continuous function on a specified range of inputs", meaning that in theory you can make your model learn anything, given enough neurons. However, we don't have infinite compute recourses, therefore we are to help the model pick up the relationship in the text quicker, by given it the building blocks, which is the features we are extracting. The goal is then to hope the model can piece together the building blocks to create a working simulation of how language is to be interpreted.

4.1.1 Affixes

The motivation for using affixes is that they capture sub-word semantic information, which is not as obviously available if only looking at words as distinct and disconnected tokens, like many pre-trained word embeddings do. A word embedding can often gather that the relationship between *"jump"* and *"jumped"* is the same as *"kick"* and *"kicked"*, but that information comes from the contexts in which those words are used in the training data. It cannot extract the meaning of the ending *"-ed"*, since that is invisible to it. Affixes could help with that, by including distinct tokens for prefixes like *"un-"* and suffixes like *"-ed"* and *"-ble"*, giving the model the ability to parse unknown words by their affixes.

We use a similar method to Yadav u. a. (2018), with learned representations of affixes. The affixes themselves are generated automatically from the training data as letter n-grams at the beginning and end of words, for prefixes and suffixes, respectively. The affixes to include are chosen based on the number of occurrences in the training data. The length of the n-grams and the minimum number of occurrences are hyperparameters tuned in the development process.

4.1.2 Part of Speech Tagging

Part-of-Speech tagging (or POS tagging) is a labelling technique used to assign words their respective part-of-speech, depending on the word definition and context. This provides a valuable insight into the sentence structure due to the context of the words. Precisely, by having a label for each word, such as whether a word is a noun or a verb, gives knowledge about the neighbouring words as well. As an example, nouns in english are usually preceded by adjectives or pronouns, while verbs are most of the time preceded by a subject, which can be either a noun or a pronoun. Therefore, by having knowledge of a word's part of speech, one can predict what precedes or succeeds it, which assists in gaining knowledge about the sentence structure.

Pointed out in (Jurafsky und Martin, 2021), part-of-speech tagging helps with disambiguation. Precisely, multiple words can have different tags, one example provided in the previously cited book is the word book, which can be a noun (a book) or it can be a verb (book a seat). By providing the correct tag, the machine is helped into identifying the right meaning of the sentence. In our implementation we employ the NLTK library's pos_tag function.

4.1.3 Lemmatization

Since the data used in this project consists of hotel reviews, the reviewers tend to use different forms of the words, due to grammatical reasons or semantical purpose. This is very problematic in a NLP setting due to the need to quantify and map words into numerical data. A word can be written differently depending on the context, while maintaining the same meaning. When transforming the words into numerical features, they tend to be very sparse and highly dimensional. Therefore, the goal is to make the model understand that these words actually provide the same meaning in an attempt to reduce sparseness.

Lemmatization is an approach of reducing sparseness by reducing the word to the base, where one uses the vocabulary and the context in order to check which would be the best approach to re-

duce the from. In this project's implementation, the POS tag mentioned beforehand is used in order to deduct the best approach. The word *"saw"* can be either a noun, or the verb *"see"* in past sense, so by using the POS tagger, one knows precisely how to reduce the words. (Manning u. a., 2008)

saw, if verb - see
saw, if noun - saw

Since this paper does sentiment analysis, the lemmatization helps with reducing words that have similar meaning, such as:

better - good

and by grouping those words together into subsets of the corpus. The idea is that the sentiment shouldn't change if the reviewer is using the word good or better.

Since every word can be lemmatized and therefor the embedding which we are concatenating onto the GloVe embedding will be equally sparse. Therefore we made a threshold to filter out sets which has low occurencies. This means words mentioned only a few times will not get a unique set. We see that words like **restaurant** and **hotel** are mentioned a lot, which makes sense when we are talking about hotel reviews. The goal of the filtering is to group the popular words together, so that the meaning of hotel and hotels is the same. We use the WordNetLemmatizer from the NLTK library for our lemmatization.

5 Results

When working with unbalanced data, odd prediction patterns can arrive if not carefully monitored. If one label is over represented, the model can pick up on this pattern and only predict the heavily favored label. This can lead to very high accuracy, despite the model essentially being useless.

This can be spotted using metrics like **macro average f1-score** and looking at the individual confusion matrices. These can be found in the appendix.

$$F1\_score = 2 \cdot \frac{precision \cdot reacall}{precision + recall}$$

$$macro\ \ average\ \ F1 = \sum_{i=1}^{n} (F1\_score_i) \cdot \frac{1}{n}$$

To get more fair and consistent results, we are training and computing the macro average f1-score 5 times and taking the average of those runs. These is the results shown in the table below.

As mentioned above, the two feature extractions *lemmatization* and *Affixes* has special hyperparameters. These are the parameters used to compute the results

Feature Extraction	Hyper param	Value
Lemmatization	Filter_amount	50
Afix	Length of affixes	3
Afix	# of occurencies	40

Table 3: Special hyper parameters for individual feature extraction methods

5.1 Baseline Model

The baseline model using word-to-vector embeddings without any additional features included has a macro average f1-score of 0.5521 for the test set averaged over the five runs.

5.2 Lemmatisation

When developing the classifier we found that the optimal filter amount is 50. Using the lemmatized groups as an additional feature to the baseline model gave an experimental result with an macro average f1-score of 0.5495 on the testing data. This is slightly worse than the baseline, but not significantly different. It can hereby be seen that our implementation of lemmatization does not have a significant impact on the results of the model.

5.3 Part-Of-Speech Tagging

Appending the PoS Tags as a feature to the baseline model, resulted in a very small improvement over the baseline model, from 0.5521 to 0.5650 for the macro average f1-score on the testing data. By these results, one can assume that the addition of an extra feature is not improving the model's performance. This can be due to various reasons, such as adding ambiguity that does not help the model improve. The PoS tags provide a good approach to provide the grammatical tagging, but these can not preserve the meaning of the word.

5.4 Affixes

In development, we arrived at an optimal n-gram length of 3, meaning that we include prefixes and suffixes of 3 letters. We included those n-grams that occurred more than 40 times in the training data. When testing our trained model on the testing

data, the results for affix features show a similarly marginal improvement over the baseline as with the previous features. Having an average macro-average f1-score only 1 percentage point above baseline, the affix features can hardly be said to add that much information not already covered by the word embedding.

Model: BiLSTM	Test set
Baseline	0.5521
Baseline — Lemmatization	0.5495
Baseline — POS-Tags	0.5650
Baseline — Affixes	0.5637

Table 4: Final results for the baseline and each feature extraction method.

6 Discussion and Error Analysis

Our results do not show the same magnitude of improvements in our task as those of Yadav u. a. (2018) and Pasupa und Ayutthaya (2019) did for their tasks. This might be due to the nature of the problems we are trying to solve. In the former of the referenced papers, the authors attempt to improve on named-entity-recognition tasks, where correctly labeled entities will often be very linguistically distinct from neighbouring words. In that case, additional linguistic features, such as affixes, might provide large amounts of new information not necessarily captured in the pre-trained word embedding, especially when the entities consist of uncommon words, like in the DrugNER datasets. The latter paper deal with another special circumstance, namely a less supported language. Here, POS-tags may provide additional semantic information, that would otherwise have been reflected in the word embedding, if that had been trained on more and better data.

7 Concluding Remarks

We cannot with any empirical evidence conclude that the chosen feature extraction methods provide any meaningful improvement over the baseline mode. Even though we get a slight improvement when adding the POS-tags and affixes, this could be down to pure chance of our predictor.

References

[Agarwal u. a. 2011] AGARWAL, Apoorv ; XIE, Boyi ; VOVSHA, Ilia ; RAMBOW, Owen ; PASSONNEAU, Rebecca: 'Sentiment Analysis of Twitter Data'. (2011)

[Agerri u. a. 2013] AGERRI ; RODRIGO ; CUADROS ; MONTSE ; GAINES ; SEAN ; RIGAU ; GERMAN: OpeNER: Open polarity enhanced named entity recognition. In: *Sociedad Española para el Procesamiento del Lenguaje Natural* Bd. 51, 2013, S. 215–218

[Barnes u. a. 2022] BARNES, Jeremy ; OBERLÄNDER, Laura ; TROIANO, Enrica ; KUTUZOV, Andrey ; BUCHMANNN, Jan ; AGERRI1, Rodrigo ; ØVRELID, Lilja ; VELLDAL3, Erik: 'SemEval 2022 Task 10: Structured Sentiment Analysis'. (2022)

[Jurafsky und Martin 2021] JURAFSKY, Dan ; MARTIN, James H.: 'Speech and Language Processing (3rd ed. draft)'. (2021)

[Li u. a. 2018] LI, Xin ; BING, Lidong ; LAM, Wai ; SHI, Bei: 'Transformation Networks for Target-Oriented Sentiment Classification'. (2018)

[Manning u. a. 2008] MANNING, Christopher D. ; RAGHAVAN, Prabhakar ; SCHÜTZE, Hinrich: *'Introduction to Information Retrieval'*. 2008

[Mitchell u. a. 2013] MITCHELL, Margaret ; AGUILAR, Jacqueline ; WILSON, Theresa ; DURME, Benjamin V.: 'Open Domain Targeted Sentiment'. (2013)

[Pasupa und Ayutthaya 2019] PASUPA, Kitsuchart ; AYUTTHAYA, Thititorn Seneewong N.: 'Thai sentiment analysis with deep learning techniques: A comparative study based on word embedding, POS-tag, and sentic features'. (2019)

[Pennington u. a. 2014] PENNINGTON, Jeffrey ; SOCHER, Richard ; MANNING, Christopher D.: 'GloVe: Global Vectors for Word Representation'. (2014)

[Prabowo1 und Thelwall 2009] PRABOWO1, Rudy ; THELWALL, Mike: 'Sentiment Analysis: A Combined Approach'. (2009)

[Yadav und Bethard 2019] YADAV, V. ; BETHARD, S.: 'A survey on recent advances in named entity recognition from deep learning models'. (2019)

[Yadav u. a. 2018] YADAV, Vikas ; SHARP, Rebecca ; BETHARD, Steven: 'Deep Affix Features Improve Neural Named Entity Recognizers'. (2018)

[Zhang u. a. 2015] ZHANG, Meishan ; ZHANG, Yue ; VO, Duy-Tin: 'Neural Networks for Open Domain Targeted Sentiment'. (2015)

[Zhou u. a. 2019] ZHOU, Yan ; HUANG, Longtao ; GUO, Tao ; HAN, Jizhong ; HU, Songlin: 'A Span-based Joint Model for Opinion Target Extraction and Target Sentiment Classification'. (2019)

Appendix

Who did what?

We all contributed to every part, but Magnus and Martin worked primarily on the lemmatization, Florian worked primarily on POS-tagging and Markus worked on the affixes. We all also did things that didn't end up in the final project. We all wrote many different parts of the report, and naming each would be impractical.

7.1 Confusion matrices

7.1.1 Baseline

	precision	recall	f1-score	support
0	0.85	0.95	0.90	5766
1	0.56	0.39	0.46	255
2	0.71	0.56	0.63	542
3	0.47	0.29	0.36	466
4	0.65	0.32	0.43	679
accuracy			0.81	7708
macro avg	0.65	0.50	0.56	7708
weighted avg	0.79	0.81	0.79	7708

7.1.2 Affixes

	precision	recall	f1-score	support
0	0.85	0.94	0.89	5766
1	0.50	0.48	0.49	255
2	0.76	0.43	0.55	542
3	0.48	0.38	0.42	466
4	0.63	0.33	0.44	679
accuracy			0.80	7708
macro avg	0.64	0.51	0.56	7708
weighted avg	0.79	0.80	0.79	7708

7.1.3 POS-tagging

	precision	recall	f1-score	support
0	0.85	0.95	0.90	5766
1	0.61	0.37	0.46	255
2	0.72	0.51	0.59	542
3	0.50	0.31	0.38	466
4	0.58	0.37	0.45	679
accuracy			0.81	7708
macro avg	0.65	0.50	0.56	7708
weighted avg	0.79	0.81	0.79	7708

7.1.4 Lemmatization

	precision	recall	f1-score	support
0	0.85	0.93	0.89	5766
1	0.53	0.51	0.52	255
2	0.73	0.49	0.59	542
3	0.43	0.34	0.38	466
4	0.57	0.36	0.44	679
accuracy			0.80	7708
macro avg	0.62	0.53	0.56	7708
weighted avg	0.78	0.80	0.79	7708

GITHUB

https://github.itu.dk/flmi/NLP2022p01g03