A Cross Augmentation Technique in Sentiment Analysis

Victor Barbaros
School of Computer Science
McGill University
Montreal, QC, CA

Erik-Olivier Riendeau
School of Engineering
McGill University
Montreal, QC, CA

Abstract

We propose a novel, cross augmentation technique that applies to Sentiment Analysis problems. Using wordnet synsets as well as word2vec embeddings we define an augmentation model that is using synonyms to augment currently queried set, and antonyms to concomitantly augment the set of opposite polarity. The augmentation model's best performance is similar to the one obtained with no augmentation, on the same datasets. Nevertheless, the variation in accuracy, depending on the hyperparameters, revealed some interesting aspects to be considered when augmenting a data set: making the meaningful features even sparser or choosing wisely which Parts of Speech(POS) to choose when augmenting the text. Being aware of and planning to deal with these issues, represents the main result of this work.

1 Introduction

In the times of powerful machine learning models, one of the most important tasks in natural language processing is obtaining a considerable amount of high quality data. In other areas where machine learning is popular, such as image processing and face recognition, one can duplicate a dataset multiple times by rotating images, scaling images differently, adding gaussian noise and flipping images. The new augmented dataset allows a machine learning model to train on much more examples. However, no such easy solutions exists in language processing. We aim, in this project, to develop a simple but efficient way of augmenting textual data sets, i.e. enlarging the sample body with unobserved(latent) features, to maximize the extraction of useful information during the pre-processing phase, thus improving the classifier's performance.

2 Related Work

We have an abundance of data from online platforms, such as twitter posts, or comments and reviews on different platforms, to name a few. It does not represent a problem to built a corpus of text data and try to get some insights about the content through sentiment analysis, multi-class classification, or summarization.

Even with the abundance of data nowadays, we still have to deal with problems such as diversity in style, grammar and syntactic correctness, and even with comments written in different languages. A particular variation of each instance of text is the length as well. In the case of online reviews and comments, we can find reviews with size from short sentences of 5-10 words to long end elaborate ones. With such a diversity it's hard to perform an analysis that applies the same methods for all comments. In such cases, an augmentation technique is applied - the existing data set is being increased by adding new, unseen data, to compensate for the sparsity and to maximize the correct interpretation of the information.

An example of applying the augmentation techniques (Wang and Yang, 2015) on a corpus of tweeter text shows how embedded augmentation from a list of dictionaries can be used in selecting the k most similar words and obtaining a relevant improvement, compared to the baseline. In this pa-

per, (Wang and Yang, 2015) the authors use word embedding to augment their twitter corpus, showing improvements of 6.1%. We will not consider only the `Word2Vec` in our experiments, but will make use of `WordNet` as well.

(Xie et al., 2017) emulated the noise insertion for image augmentation in the context of natural language processing. However, their approach targets neural network and does not augment the context, but rather generates some noise used in the neural network calculations. We opted for a different approach targeting the input dataset instead.

Later, (Kobayashi, 2018) have pushed the text augmentation even further, defining a contextual augmentation technique where the word replacement is conditioned by it's probability of its position in the given context, in addition to the simple augmentation with synonyms only. The results reported in this case are very encouraging, although comparing the classification accuracy on the RT data set (the movies review corpus), it hits a maximum of 77.4%.

Given our recent results from the first assignment, running a Naive Bayes classifier with some parameters tuning on the RT data set allowed us to obtain an accuracy of 78.87% on the testing data.

Analyzing the misclassified reviews, we have noticed some recurring patterns, such as reviews written in a language other then English, sentences with new words(not present in training set), and probably the hardest ones - sarcastic comments.

We decide to deal with the pattern related to unknown words, through an augmentation technique designed to be easy for use and capable to augment efficiently the training data set, thus improving classifier's performance.

3 Method

3.1 Augmentation using wordnet

3.1.1 General description

As hinted previously, we aim to augment textual data through the usage of related words, synonyms and antonyms. To do so, we use Word2Vec and textual embedding, as well as WordNet and related synsets. We augment our training data such that the classifier model generalizes better on testing data, as it will have seen more words during training.

We focused our attention on the Sentiment Analysis data sets, since when having only datasets of `"pos"` and `"neg"` polarity works specifically for the augmentation technique we came up with.

The main idea is the following: if we replace some words in the original sentence with their antonyms, we can simply reverse the sense of the original sentence. On the other hand, if we use synonyms, the sense stays the same. In other words, the augmentation technique we are willing to test consists in the following: make use of the features from the set you currently query to augment not only this same set, but also to augment the dataset with opposite polarity.

We consider only a subset of categories of words to augment. The reason is, we want to focus on parts of speech which influence the most the sense of a sentence. As such, the different types of adjectives and adverbs emphasize quite well the polarity of a sentence. Thus, given the available POS(part-of-speech) tags provided by the `nltk` built-in tagger, we define the **word-of-interest** to be words tagged with the following POS tags: `'JJ' - adjective or numeral, ordinal, 'JJR' - adjective, comparative, 'JJS' - adjective, superlative, 'RB' - adverb, 'RBR' - adverb, comparative, 'RBS' - adverb, superlative`.

3.1.2 Augment dataset with similar polarity

To augment the training set with similar polarity, we generate additional sentences by using synonyms. For example, the sentence `("The complex movie was incredible.":pos)` can become `( "The elaborate movie was incredible":pos)` when augmenting for the found synonym of the word `"complex"`.

3.1.3 Augment dataset with opposite polarity

To use the sentence for the augmentation of the set with opposite polarity we need to generate a negated form of the sentence. There are a few ways in which we can negate a sentence, i.e. negating the Noun Phrase(NP), the Verb Phrase(VP), the Adjective Phrase(AdjP), or using negative forms of indefinite pronouns (”nobody”, etc).

We experimented by negating the training sentences with the antonyms of the found adjectives and adverbs. Given the sentence with positive label (`"The movie was brilliant and powerful.":pos`), by replacing all the adjectives with their antonyms we can get (`"The movie was inferior and powerless.":neg`) - a good candidate to augment the training sample with opposite(negative) polarity.

3.1.4 Augmentation Algorithm

Thus, the entire augmentation method runs as follows:

- run the POS-tagger on each sentence;
- for any word in the sentence that has a POS tag which is part of the chosen POS set (words-of-interest):
    1. replace each **word-of-interest** with its **synonym**;
    2. build a new sentence with all the **words-of-interest** replaced with their **synonyms**;
    3. use the new sentence to augment the corpus with similar polarity;
    4. replace each **word-of-interest** with its **antonym**;
    5. build a new sentence with all the **words-of-interest** replaced with their **antonym**;
    6. use the new sentence to augment the corpus with opposite polarity;
    7. keep the original sentence inside the currently augmenting corpus, as well;

3.1.5 Optimize the Augmentation Model

The way we define the augmentation of the training set leaves a lot of freedom on how exactly we can build our **augmented sentences**. Thus, some additional aspects need to be discussed before performing a complete augmentation on the training dataset.

Handling semantically similar words

When querying for synonyms and antonyms we get a list of potential choices. The word that is semantically most appropriate to the queried one is at the top of the list. Since our goal is to diminish the number of unknown words in the testing set, we make an assumption that choosing less similar words (down the list) would make the classifier hit a lesser number of unknown words in the testing set. We defined our model so that it augments the training set by indexing the list returned by `wordnet` at each of the indexes: `[0, 1, 2, 3, -1]`. Some lists from `wordnet` might have only one entry so when indexing by `0`, or `-1`, it doesn't cause any problems, but if `IndexOutOfBounds` exception is thrown, we default to `0-th` entry.

Handle sparcity due to augmentation

Once the augmented corpus is built, we preprocess the dataset through a vectorizer, so we end up with a set of most relevant tokens(since from initial setup, we use unigrams in the vectorizer). But if we have lengthy sentences and only a small number of `words-of-interest`, then we effectively double the entry without actually augmenting meaningfully the training corpus. We might even introduce more meaningless data when augmenting in this way, and making the dataset even more sparse.

To deal with this introduced sparsity we define trials for the augmentation model that accept only sentences up to a given threshold and call it SLT(sentence-length-threshold), i.e. sentences of length up to `10, 20, 30` words are used. The value of `0` threshold defaults to accepting all the sentences, irrespective of the length.

Another way to deal with the sparcity of data due to augmentation, is to use **n-grams around the augmented word**. Thus, for each augmented word, we add this word symmetrically wrapped around by n-words. So, from the options defined as `[0, 1, 2, 3]` it means that if selected is `1` we augment the testing corpus directly with the fragment of the sentence containing `"pre-augmented-word-1 + augmented-word + post-augmented-word-1"`. If the option is `0(zero)`, it defaults to not using this n-gram part of sentence. This second technique is called use n-grams for sentence (UNG).

3.2 Augmentation using Word2Vec

Other methods than `WordNet` also exists to determine word's relatedness. One such method is using word embedding, the mapping of words to a vector space. The underlying idea for such a method is to

map words in position in the vector space such that these words are close to their related words. This mapping obeys the distributional hypothesis which asserts that words that occur in the same context tend to have similar meaning.

Using these word embeddings to determine related words allows us to generate new sentences with these related words for textual augmentation. Having a second method to do so gives us the opportunity to compare the methods and to potentially use both.

To produce these word embeddings, we decided to use `Word2Vec`, which are models that use shallow 2-layers neural network. We went further and picked a pre-trained `Word2Vec` instance. This instance consists of a vocabulary of `3 million words` which was trained on around `100 billion words from Google News`.

3.3 General description of the Testing process and Datasets

Our intention was to have a starting baseline when evaluating the augmentation technique. We decided that the results we obtained from classifying the RT dataset, in the context of the assignment 1, suits very well the conditions of a reliable baseline. In assignment 1, we ran three types of classifiers on the RT dataset, selected a best-performing one, then fine-tuned the hyperparameters for this classifier.

We use the RT and the IMDB Movie reviews (Maas et al., 2011). Even though we test the augmentation model on both datasets, we performed all the hyperparameters tuning related to the augmentation model only on the validation part of the RT dataset.

3.4 Baseline setup and Fine-tune the Hyperparameters

We start our experiments on the validation set of the RT dataset.

Given the list of POS tags we defined in the Methods section, we generate all the possible combinations, thus for the `6 tags` we have $2^6 == 64$ possible POS tags. For each of the POS tag combinations we run separately, different options for the following:

- 1. max length of the augmented sentence,
- 2. n-gram part of sentence around the augmented word,
- 3. different entries from the list of synonyms and antonyms.

As a final tuning, with these 3 selected hyperparameters, we select the POS tags with the best results on the validation set.

4 Results

After tuning the necessary parameters for the two methods(augmentation with Word2Vec and WordNet), we evaluate the model on testing set. We found that the length of sentence threshold up to 30 words, and the n-gram of words equal 2, and second word in the list of synonyms and antonyms perform best. The chosen POS tags consist of: RBS, JJS, JJR, JJ (see `augmentation_model_train_and_test.py` for entire setup).

Finally, for each of the datasets, IMDB and RT, we augment the training corpus with either a) modified sentences up to 30 words length or b) n-gram words around the word-of-interest equal to 2(thus, a 5-gram sequence). In both of these cases we pick the second word from the list of synonyms and antonyms. We query for synonyms and antonyms by using Word2Vec or WordNet.

Method	WordNet Acc.	Word2Vec Acc.
Baseline	0.87	
SLT	0.8701	0.87
UNG	0.867	0.865

Table 1: Accuracy on the IMDB Dataset

Method	WordNet Acc.	Word2Vec Acc.
Baseline	0.789	
SLT	0.786	0.711
UNG	0.782	0.784

Table 2: Accuracy on the RT Dataset

For completeness of results, for each of the experimental setups, we included the baseline accuracy, which represents classifier's performance with no augmentation and with the best performing hyperparameters, as found in assignment 1 (see `augmentation_classifier.py` script).

As we can see, neither of the augmentation models outperform the baseline setup, except for the case of IMDB dataset being augmented with sentences of max 30 words in length, and querying the synonyms and antonyms from the `WordNet`, which is not statistically significant.

Otherwise, there are numerous settings of augmentation model hyperparameters that give us lower accuracy scores.

In particular, augmenting the training corpus with at least 4 POS tags, sentences with no length threshold, or threshold up to 20 words, brings the accuracy as low as 73% and 69% (see `RT_TRAIN_SELECT_SENT_LENGTH_THRESHOLD.txt`). The same pattern can be seen when selecting the best performing parameters in terms of n-grams equal to 1, 2, or 3 words (see `RT_TRAIN_SELECT_NGARM_WORD.txt`).

Observing the deteriorating results when adding too small(1-gram, 10-words sentence length) or too much(3-gram, 30-words sentence, or no threshold) context around the augmented words tells us that, in the first case we are loosing a good part of the context to which the augmented word pertains, while in the second case even if we add the augmented word, we still duplicate the original content, then at the pre-processing step by the vectorizer, we end up eventually with overemphasized values for some features.

The intuition about using for augmentation other than the top word from the list of synonyms and antonyms, in order to account for the unknown words in the testing set, seems to work. We bring our model's performance to the baseline by picking the second word from those lists, but it's not enough to surpass the baseline.

Overall, we can mention two factors that are separately introduced by the augmentation model and have an impact on the insignificant performance. One represents the increase in sparsity of meaningful features due to duplication of entries, thus attenuating the end effect of the augmentation technique. A solution would be to account more rigorously for the contextual augmentation of a given corpus or even generate synthetic parts of sentences different from the original ones, that will wrap around the augmented words-of-interest, thus preventing the duplication of actual tokens.

The other factor is not covering well enough the set of unknown words found in the testing set, even when picking semantically less similar words from the list of synonyms and antonyms. This, in part, is determined by the tools from which we query the synonyms and antonyms, and by the built-in tagger with which we select our words-of-interest about to be augmented. We can deal with these limitations by either choosing carefully between the available taggers or implementing our own, as well as using dictionary tools eventually pre-trained on bigger and more diverse corpora.

5 Statement of contributions

Victor Barbaros and Erik-Olivier Riendeau:

- evenly split the coding of the augmentation model and fine-tuning;
- equally contributed to writing and editing/reviewing each one's work on the paper;

6 Discussion and Conclusion

The cross augmentation technique that we proposed turned out to perform on par with no augmentation at all. Nevertheless, we would not call these results discouraging. While observing classifier's performance on validation data, we gained some important insights in what may be right and wrong when augmenting a corpus. For example, we found that short n-grams around the augmented word or short sentences contribute more to classifier's performance, thus revealing the importance to preserve a certain contextual information when augmenting. It turns out that not all types of adjectives and adverbs reveal equally the polarity of the sentences, thus the superlative forms of these POS tags contribute more to final accuracy. We have, finally, learned, that just a simple augmentation technique, querying for synonyms or antonyms is not enough to mitigate for the unknown words in the testing set.

This work represents a valuable beginning in developing efficient augmentation techniques, and we truly hope that either the current team, or future students in COMP550 will use it as a starting point to improve or take a new direction in applying augmentation techniques when trying to gain meaningful insights from textual data.

Acknowledgements

Thank you for an amazing semester in COMP-550!

References

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. *CoRR*, abs/1805.06201.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June. Association for Computational Linguistics.

William Yang Wang and Diyi Yang. 2015. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using #petpeeve tweets. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2557–2563. Association for Computational Linguistics.

Ziang Xie, Sida I. Wang, Jiwei Li, Daniel Lévy, Aiming Nie, Dan Jurafsky, and Andrew Y. Ng. 2017. Data Noising as Smoothing in Neural Network Language Models. *ArXiv e-prints*, page arXiv:1703.02573, March.