An Active Learning-Driven Model for Subtle Sentiment Analysis

Anonymous NAACL submission

Abstract

Sentiment analysis is one of the most popular applications of Natural Language Processing and there are off-the-shelf models for this task. However, most of these models are trained on medium-to-large datasets containing examples generally expressing obvious explicit feelings or sentiments, such as tweets, reviews and social media posts. Our model deals with comments in which the sentiment is harder to predict due to the subtlety of the comments and forum idiosyncrasy. We make use of syntactic, semantic, and lexical information of the text for features that would capture the relevant information. In order to obtain labeled data efficiently, we use active learning which allows us to get it faster and at a low cost, while employing minimum resources. Our customized model achieves better results that a general out-of-the-box sentiment model on subtle employee comments from a corporate forum.

1 Introduction

The Internet has increasingly become the platform for expressing opinions. Social media acts as public forum in which people share information and opinions with others. From social media to discussion forums, and on-line stores, we can see individuals writing opinionated comments about many different topics ranging from feedback to news articles to product reviews. Most of the time, the opinions are not restricted or limited to any structure, allowing writers to express their feelings in an accurate and complex way. Such venues generate large labeled datasets that are mostly freely available, and can be used to model the writer's sentiment based on the comment. This has driven the development of great algorithms and trained out-of-the-box open source models for sentiment analysis, among them we have: The Stanford University sentiment analysis model (Socher et al., 2013) that is part of CoreNLP (Manning et al., 2014). Textblob (Loria, 2017). Microsoft and Google also provide trained analyzers through MicrosoftML and Cloud Natural Language, respectively.

However, there are public web forums where there are explicit and/or implicit restrictions, rules, or etiquette that the forum's user are expected to follow when expressing an opinion. These forums can be found as part of a more formal environment or community, such as employee corporate websites, formal political debates, and academic research forums. On these websites, ethics and code of conduct make the writer choose a different, more formal way of expressing their emotion, especially when they have a negative opinion.

In this work, we focus on the comments made by employees on articles published on the internal corporate website of a Midwestern insurance company with headquarters in Madison, WI.

A company's internal website is an important vehicle to facilitate internal communication and collaboration. Often, these internal websites have options for employees to leave comments on published articles. Measuring employee response to the articles is important, since it tells the communication team what kind of articles are well received by the internal community. These insights can be used to improve employee engagement.

Since the articles and comments are generated in a corporate environment, the comments are formal and with a professional undertone, mostly polite even when they are conveying a negative or oppositional opinion. Furthermore, Midwesterners are known for using passive aggression when communicating. Overly descriptive terms are often avoided, and the Midwestern culture is quite familiar with the art of disagreeing as discreetly as possible. Another characteristic of the comments is that they are mostly related to the insurance domain, which has a unique set of terms or jargon. Because of these reasons, it is difficult (as we show in our empirical experiments) for existing standard sentiment analysis models to make the correct sentiment prediction.

On the other hand, lack of labeled data due to confidentiality of the topics and comments makes it difficult to follow the conventional pipeline to train customized sentiment analysis models. To solve this problem, we implemented a system that improves labeling efficiency while minimizing labelers' time. This allows us to collect data to train a customized model to predict sentiment for the subtle comments generated as a response to the articles published in the internal corporate website.

We tested this algorithm against a pre-trained sentiment analyzer, and found that our algorithm performs better at predicting the sentiment even when training with a relatively small number of training samples.

2 Subtle Sentiment Analysis

In a more formal environment, people tend to add subtle hints in their comments to get their sentiment across without risking offense to anyone in the vicinity. Conventional sentiment analyzers tend to mislabel such comments, unless they are are provided with a large labeled set of such examples. In our case, we had a large set of comments from previously published articles, but we did not have the corresponding labels indicating the sentiment.

For example,
"*I agree that the healthy food options in the cafeteria are the most expensive options. Also, water and ice being more accessible would be nice. With a busy work day having water and ice in more of the work areas rather than just the cafeteria would be beneficial.*"

Pre-trained sentiment analyzers, trained on comments made in informal settings, would predict the sentiment of this comment to be positive or neutral, while it actually is a negative comment.

In order to deal with comments carrying less obvious sentiments, we decided to look into the lexical, syntactic, and semantic aspects of the text, and use these as features for training our model. We also had to make sure that our model can handle the words that did not occur in the training data, as there is higher chance of missing out on vocabulary due to a small training set size.

3 Text Representation

Due to our smaller training set, we needed a text representation that would encompass the essence of the comments in the training data, as well as work on newer texts with minimum loss of information. To get such a representation, we went for a combination of bag of words (to capture words with high emotional expression), positional vectors (to capture placement of words), and text embedding (to make the representation semantically sound).

3.1 Bag of Words

We used a trained nltk part-of-speech tagger (Bird et al., 2009) to extract nouns, verbs, adjectives, and adverbs from the comments. Words belonging to these categories hold most of the information that is present in the text. We, then, computed a term frequency-inverse document frequency (tf-idf) matrix that would capture the word occurrence based on word count and its frequency in the training data (Salton and McGill, 1986) (Pedregosa et al., 2011).

3.2 Text Embedding

In order to capture the meaning behind the comment, we implemented an algorithm that would compute text embedding of the given comment by averaging across the vectors of the words occurring in the comment. The result is the center of the collection of words that we treat as the unweighted text embedding or text2vec.(Wieting et al., 2015)

For word embedding, we used the word2vec representation (Mikolov et al., 2013). We chose this representation, as it has performed really well in generating text embedding and word-related operations such as odd-word-out and contextually-similar words. They capture the semantic qualities of the word, which are reflected in the vector.

3.3 Positional Features

Bag-of-words captures the most informative words, while text2vec captures the semantic sense of the comment. However, what these two features lack is the positional information that is a vital area to detect subtlety of the comment. For example,
"*Good article. I was disappointed by response*"
and
'*Disappointing article. I liked the response.*"

Without positional information, the two examples will have very similar text representation, which would lead to similar classification output.

In order to solve this issue, we included a position-based text representation similar to the one proposed in (Rosales et al., 2010). We use a list of important (enriched) words, and compute distance of other words with respect to these words in the text. In a nutshell, a positional feature considers relative distances from any document word w_i to all words $\tilde{w_j}$ in a dictionary D of predefined relevant words, if they appear together within a window size. The distance is measured by how many words apart w_i and $\tilde{w_j}$ are in the sentence. If the distance is larger than the given window size, the value is ignored.

4 Efficient Data Labeling

The most difficult part of machine learning with unstructured data (and most machine learning problems in general) is acquiring a sufficiently large set of representative and good quality labels. To make the process easier and more efficient, we leveraged the crowd-sourcing/active-learning platform NEXT (Jamieson et al., 2015) to implement active learning algorithms and deploy them at the scale of many users.

4.1 Active learning

Active learning is a subset of machine learning that addresses the issue of how efficiently algorithms can learn by choosing which samples in the dataset to train with (Settles, 2010). The typical setup for pool-based active learning for classification is: an unlabeled pool of examples $\mathcal{U}$, a labeled pool $\mathcal{L}$ of example-label pairs (x, y_x), an oracle that can supply the label of any $x \in \mathcal{U}$, and a querying strategy that selects which example in $\mathcal{U}$ the oracle should label based on the current state of $\mathcal{L}$. *Passive learning* would just sample uniformly from $\mathcal{U}$ as its querying strategy. In our case, human annotators provide the role of the oracle. Active learning is particularly useful when human annotation is involved, because one can substantially reduce the cost and time needed to label a sufficient subset of $\mathcal{U}$.

The goal of the active learning querying strategy is to select $x^* \in \mathcal{U}$ such that $\mathcal{L}^* = \mathcal{L} \cup \{(x^*, y_{x^*})\}$ yields the maximum information gain versus $\mathcal{L} \cup \{(x, y_x)\}$ for any other $x \in \mathcal{U}$(MacKay, 1992). Maximal information gain is usually defined as maximally changing the predicted distribution towards the true distribution, which could

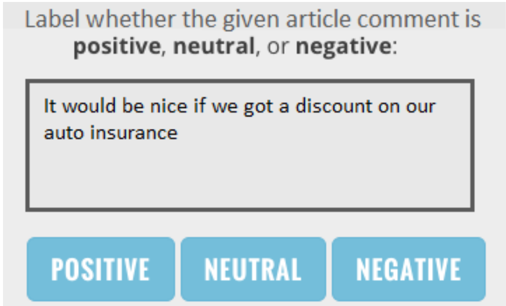

Figure 1: NEXT UI. We label the comment as *positive*, *neutral*, or *negative* sentiment.

be quantified by greatest decrease in risk:

$$x^* = \text{argmin}_{x \in \mathcal{U}} \mathbb{E}\left[\ell(f_{\hat{\mathcal{L}}})\right]$$

Where $\mathbb{E}\left[\ell(f_{\hat{\mathcal{L}}})\right]$ is the expected loss of the classifier trained on the labeled set $\hat{\mathcal{L}} = \mathcal{L} \cup \{(x, y_x)\}$ over the true distribution of your samples.

For this paper, we implemented *uncertainty sampling*, which is a querying strategy where $\mu(x)$ captures the classifier's uncertainty about the class of x (Lewis and Gale, 1994).

The intuition is that not much information is gained if a classifier gets a new label for which it already agreed with the truth with high certainty.

For a linear classifier, this is equivalent to finding unlabeled samples that lie closest to the decision hyper-plane and can be implemented in $\mathcal{O}\left(|\mathcal{U}| \log |\mathcal{U}|\right)$ time.

4.2 Active learning implementation: NEXT

NEXT (Jamieson et al., 2015) provides the architecture to implement active learning algorithms and scale the labeling process to as many users as necessary. The application itself is stateless, jobs are run asynchronously, and all variables and computations are stored in a database or cache accessible by all instances, meaning it is easy to distribute the processes over a cluster to scale with the number of labelers.

In order to streamline the labeling experience, we do not retrain the algorithm in real-time for every label. Instead, we maintain a queue of unlabeled examples chosen by the active learning algorithm. When a user is ready to label, they are simply served the first query in the queue and an asynchronous job is started to refill the queue.

An important part of efficiently gathering labels is knowing when to stop, so you may conserve labor costs or move on to new concepts. Performance diagnostics and stopping conditions are

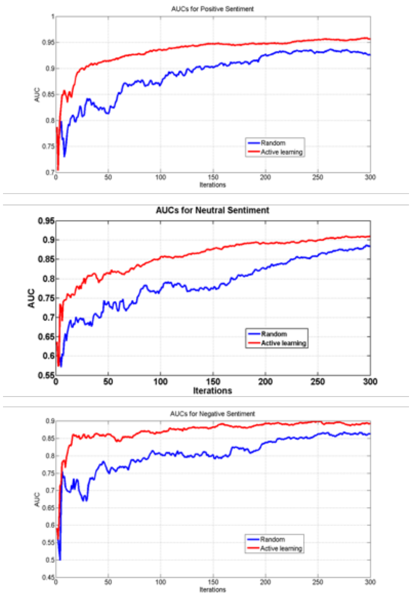

Figure 2: Comparison of AUC performance progression of active learning vs random selection for (a) positive, (b) neutral, and (c) negative comments vs the rest classifier.

necessary to evaluate the progress of the label acquisition and resulting model. NEXT provides the ability to build a customized dashboard for this purpose.

5 Experimental Results

When initially exploring options for our solution, we tested an open source pre-trained sentiment analysis model from the CoreNLP library (Manning et al., 2014) but the results were not satisfactory.

We deployed NEXT to obtain labels for comments from 10 members of the communications team, who are in charge of publishing articles on the internal company website. We collected labels for approximately 3500 comments to be used for training and testing our models. Additionally, for a disjoint set of 250 comments we obtained labels from each of 4 labelers to be used as a special testing set for this project. By obtaining labels from 4 different labelers, we were able to perform a more in-depth analysis of the model performance by examining inter-labeler agreement. The comments were labeled in as negative, positive and neutral, and we trained a one-vs-all classifier. The average test set AUC was 0.90. Figure 2 show the effectiveness of the active learning algorithm which achieves high performance faster than randomized sampling.

	L1	L2	L3	L4	SST	**SSA**
L1	1	0.86	0.81	0.86	0.35	0.69
L2	0.86	1	0.82	0.82	0.36	0.654
L3	0.81	0.82	1	0.84	0.38	0.70
L4	0.86	0.82	0.84	1	0.38	0.71
SST	0.35	0.35	0.38	0.38	1	0.44
SSA	0.69	0.65	0.70	0.71	0.44	1

Table 1: Spearman's rank correlation between each labeler, the Stanford model and Subtle Sentiment Analysis. $L_i, (i = 1, \ldots, 4)$ denote the labelers.

In order to further assess performance of the model, we computed Spearman's rank correlations among the labelers, the Stanford Sentiment Model (SSM) (Socher et al., 2013), and the model produced by subtle sentiment analysis (SSA) over the 250 testing comments. Results are summarized in Table 1. Our model performs close to the human labelers and significantly outperforms the generic Stanford Sentiment model.

On further evaluation, we discovered that our model performed really well on identifying positive and neutral comments. When it came to "negative" comments, we discovered that most of the wrong predictions declared the comments as "neutral". This is in line with our hypothesis of subtlety of expression of negative sentiment.

6 Conclusions and Future Work

We have highlighted some of the challenges that arise when training a sentiment analysis model using comments made in a formal environment where writers express their opinion in a subtle way. We also showed how we can speed up the labeling process using active learning by obtaining labels for fewer, but more influential, samples. This is important in cases where we have short amount of time and a small number of labelers.

Currently, our work treats comments independent from the source upon which they were made. We intend to explore the effect of including the context information into the model for the given comment. The context is usually the source, such as article or original comment, on which the current comment is being made. We also intend to do emotional analysis of these comments for detailed understanding of the writer's feelings in the given comment. Once we obtain a sufficient number of labeled samples, we also want to explore shallow and deep neural networks for predicting subtle sentiment analysis.

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

Kevin G Jamieson, Lalit Jain, Chris Fernandez, Nicholas J. Glattard, and Rob Nowak. 2015. Next: A system for real-world development, evaluation, and application of active learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, Curran Associates, Inc., pages 2656–2664.

David D. Lewis and William A. Gale. 1994. A sequential algorithm for training text classifiers. In *SIGIR '94*. Springer-Verlag, pages 3–12.

Steven Loria. 2017. Textblob. http://textblob.readthedocs.io/en/dev/index.html.

David J. C. MacKay. 1992. Information-based objective functions for active data selection. *Neural Comput.* 4(4):590–604. https://doi.org/10.1162/neco.1992.4.4.590.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, System Demonstrations*. pages 55–60. http://aclweb.org/anthology/P/P14/P14-5010.pdf.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of ICLR Workshop 2013*.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Rómer Rosales, Faisal Farooq, Balaji Krishnapuram, Shipeng Yu, and Glenn Fung. 2010. Automated identification of medical concepts and assertions in medical text. *Proceedings for AMIA Annual Symposium* 2010:682–6.

Gerard Salton and Michael J. McGill. 1986. Introduction to modern information retrieval .

Burr Settles. 2010. Active learning literature survey. Technical report.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, pages 1631–1642. http://www.aclweb.org/anthology/D13-1170.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Towards universal paraphrastic sentence embeddings. *CoRR* abs/1511.08198.