Using Weakly Supervised Learning to Enhance Text Classification

Stanford CS224N Custom Project

Pooja Chopra
Department of Computer Science
Stanford University
chopra14@stanford.edu

Abstract

In today's day and age, companies are making concerted efforts every day to be digitally advanced by automating their processes to increase accuracy, save time, and cut costs. They are trying to incorporate predictive algorithms that rely on machine learning modules to increase revenue and extract valuable insights that allow them to stay relevant and competitive in today's market. As a senior data scientist working to help clients achieve these goals, a problem I see every day is how predictive models are limited in their performance by the amount of data provided. My goal is to leverage weakly supervised learning to generate data that can not only increase performance, but also prevent overfitting (common in model trained with very less data). To close this gap, I explore how data generation can amplify performance of classification models. I create a pipeline that aims to advance results shown in a researched approach by Meng, Yu, et al. I combine language generation through a combination of skip-gram and LSTM models which uses keywords and existing labeled documents to build a vocabulary and learn semantics to generate new data mimicking the data provided with a BERT classification model. The results show that my proposed hybrid approach outperform both the researched WeSHClass method and BERT alone.

1) Key Information

- TA Mentor: Anna Goldie
- External collaborators: No
- External mentor: No
- Sharing project: No

2) Introduction

To see reliable performance for models utilizing machine learning algorithms, a common thread is that a lot of training data is required. Machine learning models varying from Logistic Regression to Deep Neural Nets all rely heavily on large number of labeled datasets to converge and stabilize in their performance. However, "the reliance of these models on massive sets of hand-labeled training data" comes at a not only a high financial cost, but also is time consuming [3]. Creating a quality hand-labeled dataset also requires domain expertise. These reasons serve as an impediment to achieving high performance metrics. Further population data is often not completely balanced, and models can be prone to overfitting and leaning towards overpopulated classes.

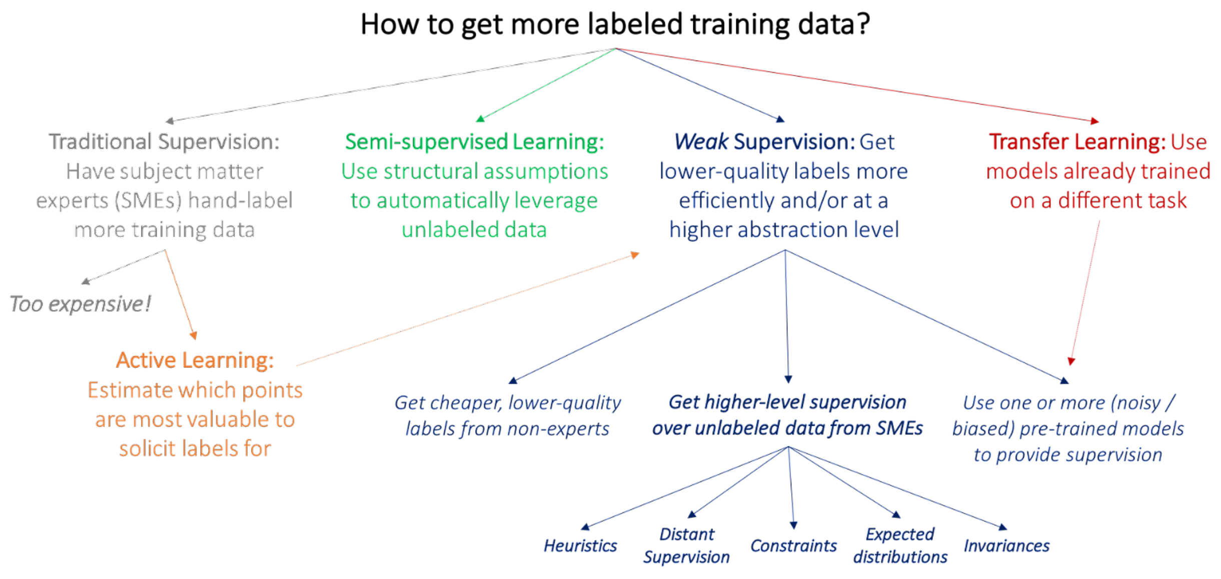

How to get more labeled training data?
Traditional Supervision: Have subject matter experts (SMEs) hand-label more training data
Semi-supervised Learning: Use structural assumptions to automatically leverage unlabeled data
Weak Supervision: Get lower-quality labels more efficiently and/or at a higher abstraction level
Transfer Learning: Use models already trained on a different task
Too expensive!
Active Learning: Estimate which points are most valuable to solicit labels for
Get cheaper, lower-quality labels from non-experts
Get higher-level supervision over unlabeled data from SMEs
Use one or more (noisy / biased) pre-trained models to provide supervision
Heuristics
Distant Supervision
Constraints
Expected distributions
Invariances

Image 1: How to get more labeled training data? [2]

The graphic above shows that there are multiple ways of getting more labeled training data. This includes Traditional Supervision, Semi-supervision Learning, Weak-Supervision, and Transfer Learning. Traditional Supervision is a common approach, however, costly, time consuming, and necessitates SMEs to hand label large datasets. Active Learning is used to label those data points that could lie close to model decision boundary lines to make the effort of hand labeling “worth it”. Semi-supervised learning relies on the idea that structural assumptions of few labeled data points can help make assumptions of large unlabeled datasets, therefore making the large dataset valuable. Transfer learning utilizes large, labeled datasets, and “pre-train” before “fine-tuning” on a more specific dataset. Weak Supervision is an alternative to the above approaches and relies on the same concept: leverage a smaller quality labeled dataset to create a larger labeled dataset that can be used in ML algorithms. [2] I am motivated by the idea of using weakly supervised learning to leverage data generation, therefore, effectively overcoming a common impetus: lack of training data.

3) Related Work

This paper explores language generation through a combination of skip-gram and LSTM models. It uses keywords and existing labeled documents to build a vocabulary and learn semantics to generate new data mimicking the data provided. It is an interesting approach to data generation that aims to replicate natural language. It is taking a step forward to enhance classification models by leveraging weakly supervised learning to generate data.

4) Approach

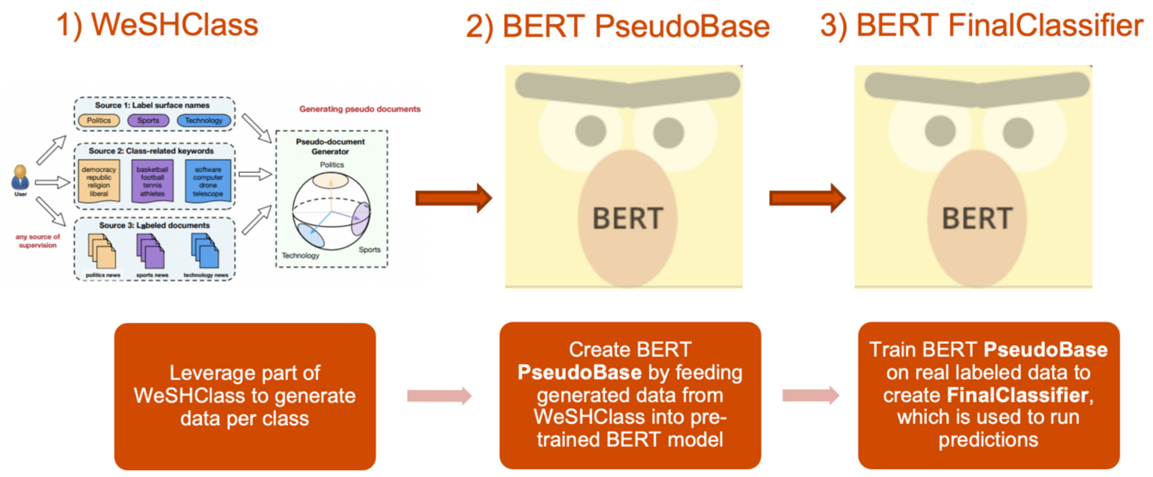

1) WeSHClass
2) BERT PseudoBase
3) BERT FinalClassifier
Source 1: Label surface names
Politics
Sports
Technology
Source 2: Class-related keywords
democracy republic religion liberal
basketball football tennis athletes
software computer drone telescope
Source 3: Labeled documents
politics news
sports news
technology news
User
any source of supervision
Generating pseudo documents
Pseudo-document Generator
Politics
Technology
Sports
BERT
BERT
Leverage part of WeSHClass to generate data per class
Create BERT **PseudoBase** by feeding generated data from WeSHClass into pre-trained BERT model
Train BERT **PseudoBase** on real labeled data to create **FinalClassifier**, which is used to run predictions

Image 2: Hybrid Model

The goal of my project is to explore the efficacy of the classifier in the WeSHClass pipeline researched by Meng, Yu, et al. Above I have created my own model pipeline that takes pieces from the WeSHClass pipeline. For reference, I have included the WeSHClass pipeline in the Appendix (Image A1). To improve the efficacy of the WeSHClass pipeline I have decided to research how replacing a sequence of RNN and CNN classifiers with a BERT classification model can improve published metrics on three different datasets.

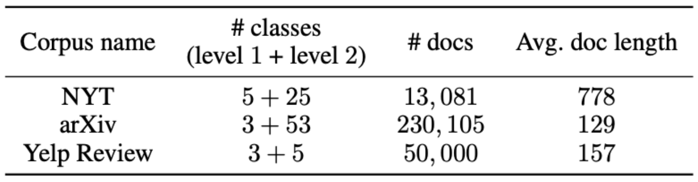
WeSHClass uses keywords to generate pseudo documents for model pre-training. The "user-given weak supervision" whether it be labeled documents or the keywords, are necessary to generate pseudo documents [1]. The paper outlines two distinct ways to retrieve keywords per subclass: keywords can be provided as inputs or extracted using tf-idf weighting from labeled documents. To generate data, the WeSHClass model utilizes a skip-gram and long short-term memory (LSTM) model to generate new text. First, a skip-gram model is trained on the entire corpus. This allows the model to learn the dimensional space of each word in the corpus within the vector space. Next an LSTM model is trained on the entire corpus and is the key component to create pseudo documents. For each subclass, an embedding vector is sampled from the distribution per class. As a reminder, this distribution is created by training the skip-gram model on the corpus to understand how each word relates to the other in higher dimensional space. The closest word to this embedding vector is picked as the start word of the sequence. The LSTM model is then able to generate the next word and attach it to the sequence recursively. [1]

The hybrid approach leverages aspects of the WeSHClass model and serves as a precursor input into the BERT model. To accomplish this task, there are three main steps in my approach. In the WeSHClass methodology, it is important to remember it has two parts: 1) generation of pseudo documents, and 2) a built in deep neural network classifier (Appendix Image A1). For the hybrid approach, I use the first part of the WeSHClass model and pull the generated pseudo documents per subclass. I replicate the approach outlined by Meng, Yu, et al. to generate this data.

These pseudo documents are then fed into a BERT model. I leverage the idea of "pre-trained models" with the following approach. The BERT model is already pre-trained, but I have created another "base model": a pre-trained BERT model finetuned on pseudo documents from WeSHClass. In the graphic above, I refer to this as **PseudoBase** in step 2. It is noteworthy to mention the increase in training data available at this point to create **PseudoBase**. In Step 3, I have further finetuned the **PseudoBase** model, which now serves as the new "base model". My **FinalClassifier** model is the **PseudoBase** model trained on the real labeled documents and is the model that runs predictions on a true test set outlined in results.

5) Experiments

Data. Because my goal is to enhance the approach highlighted in my chosen research paper, I compare results of my proposed approach to the paper's results on two of the three datasets used in the paper. I focus on Yelp Reviews and NYT dataset. Below is a summary of the datasets taken from the paper.



Corpus name	# classes (level 1 + level 2)	# docs	Avg. doc length
NYT	$5+25$	$13,081$	778
arXiv	$3+53$	$230,105$	129
Yelp Review	$3+5$	$50,000$	157

Table 1: Data Summary [1]

Using train_test_split from sklearn, I use an 85/15 split of data for train and validation respectively to train the **PseudoBase** model. For both the yelp and NYT dataset, I use 1,000 documents and feed it into the WeSHClass model. The WeSHClass model creates 500 additional pseudo documents per subclass, resulting in 2,500 and 12,500 additional documents for the Yelp Review and NYT dataset, respectively. The original 1,000 labeled documents for each dataset are then used again to "fine-tune" and create the **FinalClassifier** model. To test the **FinalClassifier** model, I take 20% of the real labeled dataset and run my final model predictions of the hybrid approach on this holdout set.

Evaluation Method. F1 Score. This metric is used in the paper as a way of measuring model performance of the binary classifiers for each class. I have used this metric not only for comparison but also because of its ability to assess model performance for classification models. Imbalanced classes and the distribution of false positives and negative can give a falsely high accuracy, and to use reliable metrics to evaluate model performance, for classification it is recommended to use Precision, Recall, and F1 Score as our evaluation metrics. Specifically, I choose to focus on Macro F1 Scores. The paper I am researching reports both Macro and Micro F1 scores for each dataset (with the Micro scores averaging higher than the Macro F1 Scores). Statistically, Micro F1 scores calculate accuracy because it is the number of correctly classified instances divided by total instances. With 5 subclasses in the Yelp dataset and 25 subclasses in the NYT dataset, the overall Micro F1 score can be skewed if certain classes perform far better than other. To get a more holistic representation of how the WeSHClass, BERT, and Hybrid model compare, I rely solely on the Macro F1 score. However, in future work it would be worth investigating more granular details of performance metrics.

Experimental details. For the WeSHClass portion of my hybrid approach, I replicated the parameters the researchers use on the Yelp and NYT dataset. For the BERT classification model, I followed these parameters: optimizer = AdamW(model.parameters(), lr=1e-5,eps=1e-8) for 5 epochs. I chose to keep the same parameters for both layers of BERT training. In terms of model performance, my goal is to overcome a data sparsity issue by leveraging weakly supervised learning to generate data that could not only increase performance but also prevent overfitting. To prevent overfitting, I made sure to include a dropout layer (probability = 0.1) in my neural network in the BERT model during training (it is excluded in test to ensure replicability of classification predictions). The main benefit of dropout is that it randomly zeroes nodes, therefore spreading out the weights and ensuring the neural network does not rely on any one feature for predictions [4]. It is a commonly used regularization technique to prevent overfitting while training. Further, I outline my training times below for both datasets.

Model	**# Documents**	**# Classes**	**Training Time with GPU**
WeSHClass	1,000	5	36 min
BERT Layer 1	2,500	5	7 min
BERT Layer 2	1,000	5	3 min

Table 2: Training times for Yelp

Model	**# Documents**	**# Classes**	**Training Time with GPU**
WeSHClass	1,000	25	2 hrs 10 minutes
BERT Layer 1	12,500	25	28 min
BERT Layer 2	1,000	25	2 min

Table 3: Training times for NYT

Unfortunately, I had to exclude the arXiv dataset. For both the Yelp and NYT dataset, I trained the WeSHClass on the provided 1,000 rows of data. The arXiv dataset showed NaN probabilities in the spherical distribution generated by the skip-gram model when I cut the given dataset of ~230,000 documents to even 10,000. Further, training on all ~230,000 rows showed an estimated training time of 19 hours per epoch for a total of 25 epochs. Given this constraint, I chose not to pursue this dataset. More importantly, my specific point of interest is how to increase classification performance metrics given a small amount of training data. The necessity of training over all ~230,000 documents for arXiv deviated from my research goal.

Something to note here is that the prediction time was quite fast (within few seconds). This is important to note because the biggest limiting factor of deploying a transformers model in the client environment is that though they are trying to increase accuracy, it is not at the cost of increased time. With such a fast prediction time, this model would be eligible for deployment in a client environment.

Results. This table below lays out the different model approaches and corresponding F1 scores for both the Yelp and NYT datasets.

Model	**Data**	**Macro F1 Score**
WeSHClass	Real + Generated Yelp Data	42.3%
BERT	Real Yelp Data	55%
WeSHClass + BERT Layer 1	*Real + Generated Yelp Data*	*98.6%*
Hybrid Model	**Real + Generated Yelp Data**	**58%**
Hybrid Model + tf-idf	**Real + Generated Yelp Data + additional keywords**	**59%**

Table 4: Model Results for Yelp Dataset

Model	Data	Macro F1 Score
WeSHClass	Real + Generated NYT Data	63.2%
BERT	Real NYT Data	59%
WeSHClass + BERT Layer 1	*Real + Generate NYT Data*	*92.3%*
Hybrid Model	**Real + Generated NYT Data**	**66.8%**

Table 5: Model Results for NYT Dataset

WeSHClass: These metrics are reported from the researched paper and serve as a base comparison. Specifically, I choose Macro F1 scores from the reported metrics for aforementioned reasons.

BERT: As a base comparison, I run a BERT model classifier only on the real labeled datasets for Yelp and NYT. This also provides a good baseline to compare the Hybrid model against.

WeSHClass + Bert Layer 1: This intermediary step correlates to Step 2 from Image 1. For both the Yelp and NYT datasets, this layer gives a much higher F1 score of 98.6% and 92.3%, respectively. The most apparent explanation for this is that the WeSHClass uses keywords for each subclass to build a vocabulary. These keywords are prominently used in the generated text, therefore, making the distinction between text in different classes far more distinct than the real data would. This step is built and tested on only pseudo documents and does not give a full picture as the model at this point has not been tested on real labeled documents mimicking population data.

Hybrid Model: These results are the most interesting to look at. For both Yelp and NYT, the hybrid approach outperforms the WeSHClass model and BERT model alone giving a Macro F1 score of 58% and 66.8%, respectively. Not only did the hybrid approach achieve higher performance metrics, it also is trained on substantively more training data then BERT alone and is a step in the right direction for overcoming overfitting.

Hybrid Model + tf-idf: Given the constraint with the arXiv dataset, I instead chose to focus on understanding how manipulation of keywords can increase model performance. For the yelp dataset, I cleaned the text and removed stopwords and then ran a tf-idf vectorizer over each subclass to extract highly correlative features of each class. As a result, I was able to add a few keywords to each class. This shows a menial increase of 1% in overall Macro F1 score and was not as fruitful of an approach as expected. An analysis of this is further highlighted below.

6) Analysis

The hybrid approach leverages data generation and increase performance metrics by replacing a combination of a RNN and CNN with a transformers approach. A key limitation to note is that the performance metrics are still relatively low. The hybrid approach does yield higher metrics for both Yelp and NYT than the researched method, however, it still does not hit a threshold of 80%. Specifically for the Yelp dataset, I qualitatively assessed the output of the tf-idf max features and the actual text of Yelp reviews per class to understand the root cause of seemingly low metrics. Below I have outlined the results of words I selected are qualitatively interesting to look at.

Subclass	Common Keywords extracted from tf-idf
great	‘amazing’, ‘delicious’, ‘favorite’, ‘fresh’, ‘good’, ‘great’, ‘special’, ‘wonderful’
good	‘delicious’, ‘favorite’, ‘fresh’, ‘good’, ‘great’, ‘pretty’, ‘well’, ‘worth’
average	‘better’, ‘friendly’, ‘pretty’, ‘price’, ‘service’, ‘special’
bad	‘best’, ‘better’, ‘decent’, ‘friendly’, ‘good’, ‘great’, ‘small’
terrible	‘cold’, ‘friend’, ‘money’, ‘poor’, ‘pretty’, ‘wait’

Table 6: tf-idf on Yelp dataset

It is apparent, that there is a strong overlap of commonly appearing words amongst the five subclasses. After looking at the actual dataset there were numerous examples of a negative word placed in front of a positive word for the classes “bad” and “terrible”. For example, both the “bad” and “good” subclases show the common words: ‘good’ and ‘pretty’, however, the “bad” reviews will have phrase such as “not so good” or “pretty bad”. It might be useful to investigate what the results will look like if we concatenate a tf-idf vectorizer with bi- and tri- grams with BERT embeddings as additional context could be provided in training if we include common n-grams. All in all, the common keywords are very telling as to why a metric of 80% was difficult to achieve, the different subclasses were simply no unique enough for BERT alone to discern while classifying.

7) Conclusion

	BERT	WeSHClass	Hybrid
Enough Training Data		✓	✓
Increased Performance Measure	✓		✓
Prevents Overfitting			✓
Fast Training Time	✓	✓	✓
Fast Prediction Time	✓	✓	✓

Table 7: Advantages of each model

Above, I choose five important factors I typically use to assess machine learning models before deployment: the model is trained on enough training data, shows substantial performance, is not prone to overfitting, and has a fast training and prediction time. Relatively speaking, based on each of these factors the hybrid model outperforms a BERT model only trained on labeled data and the researched WeSHClass pipeline.

For future work, I have a keen interest in developing the hybrid model further to reach a threshold of 80% F1 Score. My next steps would include a thorough analysis of mis-classified instances. It would be telling to look at incorrectly labeled predictions in my validation set to which if there are specific classes in which majority of my errors fall. An example of this would possibly be if a “bad” review was repeatedly mis-labeled as “good” or vice versa based on the analysis shown above. Based on this investigation, some sort of feature engineering could be used to increase performance measures. For example, length of text could be an additional feature to include as an input i.e., it is possible there could be a distribution of length of text amongst the 5 categories. Any of these avenues would be worth investigating to mold the model to the specifics of the population dataset.

It would also be useful to take a more granular look at the performance metrics. Right now, I have focused on Marco F1 Scores because a high-level look at the model performance was important in my model iterations to look at performance increase. However, to understand the pitfalls, looking at F1 Scores per class along with precision and recall would be an effective way to assess how to improve model performance.

References

[1] Meng, Yu, et al. "Weakly-supervised hierarchical text classification." *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. No. 01. 2019.

[2] http://ai.stanford.edu/blog/weak-supervision/

[3] Devlin, Jacob et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” *ArXiv* abs/1810.04805 (2019): n. pag.

[4] Srivastava, Nitish & Hinton, Geoffrey & Krizhevsky, Alex & Sutskever, Ilya & Salakhutdinov, Ruslan. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. 15. 1929-1958.

Appendix

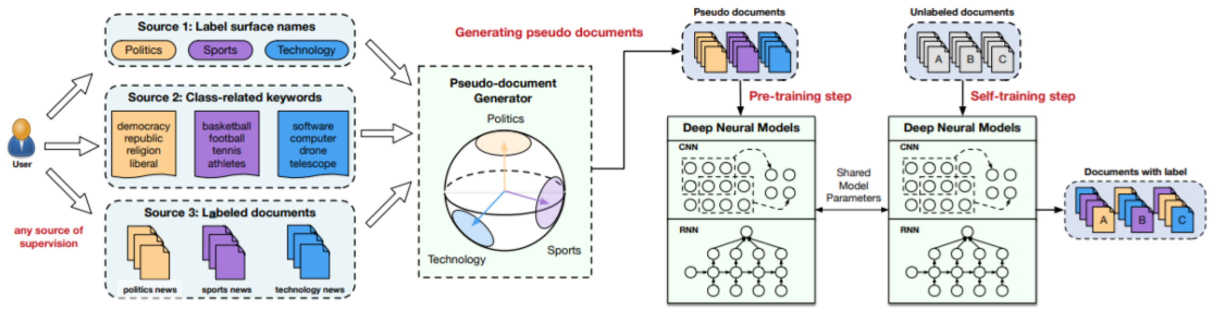

Source 1: Label surface names
Politics
Sports
Technology
Source 2: Class-related keywords
democracy republic religion liberal
basketball football tennis athletes
software computer drone telescope
Source 3: Labeled documents
politics news
sports news
technology news
User
any source of supervision
Generating pseudo documents
Pseudo-document Generator
Politics
Technology
Sports
Pseudo documents
Pre-training step
Deep Neural Models
CNN
RNN
Shared Model Parameters
Unlabeled documents
Self-training step
Deep Neural Models
CNN
RNN
Documents with label

Image A1: WeSHClass Pipeline [1]