Weak Supervision Text Classification using Cosine Similarity and SVM for Hardware Constrained Systems

Anonymous ACL submission

Abstract

Weakly supervised text classification is the ability to classify large, diverse types of unstructured text data while requiring only a small amount of manual guidance. With open-source pre-trained language models becoming widely available in the last couple of years, the weak supervision text classification domain has received renewed interest due to the potential for transfer learning. Recent weak supervision methods proposed using pre-trained language models have performed well against the popular WRENCH benchmark datasets (Zhang et al., 2021), demonstrating the capability of transfer learning. However, these methods use pre-trained language models that are computationally expensive to perform inference with and are unfeasible to finetune without specialized accelerated hardware. Methods that don't require fine-tuning often require repeated inference or large storage needs to achieve their results. In this paper, an alternative solution is proposed that uses a single inference step, has minimal storage and memory requirements, doesn't require accelerated hardware, and can provide competitive results to much more hardware-intensive methods.

1 Introduction

Finding access to large amounts of clean labeled data is not common in real use cases, and requiring domain experts to manually label more than a few samples can quickly become an expensive and time-consuming drain on resources. The text classification goal of users can also shift over time to reflect new data or new demands. An example is a customer complaint system; business users may want to track specific customer complaints based on a multitude of criteria, and this criterion is likely to change over time. Weak supervision promisesthe flexibility that this text classification task would require.

Transformer architecture and hardware advances have allowed for capable pre-trained language models (PLM) to become available to the public. Large companies (i.e. Microsoft, Google) train them on expensive hardware over massive amounts of data, and then smaller organizations and individuals can directly use them for a variety of natural language processing (NLP) tasks. Autoencoder PLMs convert text by embedding it into dense, high-dimensional vectors which incorporate rich contextual meaning. This context captures the different meanings between words and allows for accurate semantic comparisons between words using these embeddings. It has been shown that averaging these word embeddings per document can retain the meaning of the overall document (Reimers and Gurevych, 2019); this is a useful tool for reducing memory and computational complexity of each document, by collapsing the words into a single pooled embedding.

Cosine similarity is a formula to calculate the similarity between two vectors and has been successfully used for text document comparisons for many years (Mikawa et al., 2011). It has been used more recently with PLMs as a way to match class labels with document embeddings, demonstrating its effectiveness when regular clean samples are limited (Schopf et al., 2023).To use cosine similarity between the document vectors generated by a PLM, the PLM must be finetuned on a cosine similarity goal for pairs of sentences in order to generate a meaningful vector space (Reimers and Gurevych, 2019). SBERT (Reimers and Gurevych, 2019) and SimCSE (Gao et al., 2021) are two popular options for performing this fine-tuning step. Contextually similar words in a meaningful vector space have high cosine similarity, and dissimilar words have a low cosine similarity.

2 Previous Work

There are two recent categories of weak supervision methods that have had strong performance, but with contrasting design and hardware requirements. The first category requires fine-tuning on a PLM to obtain results. They often don't generate their own weakly supervised data, and instead are methods to improve the accuracy of pre-generated noisy data. The second category of weak supervision methods do not require PLM fine-tuning or pre-generated noisy data, and instead only require PLM inference. They are designed to use a low amount of manually provided words for each class to perform labeling. These methods work using a word-based analysis, and they need the full corpus of token embeddings available to function.

The first category is a best fit for many different text classification tasks, including topic classification, sentiment analysis, name entity recognition, relation classification, etc. However, as shown by a recent survey of these types of weak supervision methods (Zhu et al., 2023), these methods still require clean samples to perform model selection and validation. This category was chosen due to its benchmark setting performance on many weak supervision datasets. COSINE (Yu et al., 2021) was chosen for comparison in this paper, as it was shown to be the best overall performing weak supervision method in the survey (Zhu et al., 2023). COSINE uses "roberta-base" as its PLM for inferencing and fine-tuning.

X-Class (Wang et al., 2021) belongs to the second category of text classification methods and doesn't require fine-tuning a PLM or traditional clean samples. It only requires a one-word class label and performs very competitively on certain datasets. It is part of the one-word and low-word classification methods that work without using any traditional clean samples. It only requires a PLM to transform the text into embeddings, and can perform classification without fine-tuning or additional inference, as long as the entire corpus of token embeddings can be stored and retrieved. This category was selected due to both its strong performance and its minimal labeling requirements. X-Class was chosen for comparison in this paper because it is a top-performing method from this category. X-Class uses "bert-base-uncased" as its PLM for inferencing.

3 Methodology

This paper's algorithm combines a few different methods to obtain the final classification result: MPNet (Song et al., 2020), SBERT, cosine similarity, and SVM. MPNet fine-tuned with SBERT is used as the Autoencoder PLM for creating the initial document embeddings. SBERT fine-tuning is necessary for the embeddings generated by MPNet to be used directly with cosine similarity. Cosine similarity has been commonly used in the document scoring domain for many years. SVM is a machine learning classifier that learns boundary points and optimizes a best-fit margin between the different classes. It has shown to be one of the most powerful classifiers for a variety of domains (Cervantes et al., 2020), and it has the advantage of being memory and compute efficient versus PLMs when trained with limited, selective input data.

A benefit to using the embeddings directly is that once they are generated by the PLM, they can be stored and reused indefinitely. As shown in Table 1, storing the document embeddings instead of the token embeddings greatly minimizes the storage and memory requirements. The PLM is only necessary for further inference when new data is provided, class labels change, or the clean samples change. This removes the need for repeated inference and limits the usage of the compute-intensive PLM. The consequence is that once the initial corpus is embedded for the first time, the system can be modified near real time by users as their data monitoring needs progressively change.

There is only one hyper parameter to modify; the number of standard deviations above the median class scores of the top cosine similarity scores per class ($\boldsymbol{\alpha}$). The hyper parameter $\boldsymbol{\alpha}$ will be explained in detail below. For $\boldsymbol{\alpha}$, using anywhere from 2-3 is sufficient for most applications.

3.1 PLM And Embedding Details

The first step is to use MPNet to generate the embedding vectors for the text corpus. MPNet is an alternative to the encoder transformer network BERT, which instead uses a different pre-training method. This alternative pre-training method maximizes the amount of information the network receives for each input versus the traditional masked language modeling approach with BERT. This MPNet model was then fine-tuned using SBERT and a cosine similarity goal to create a meaningful vector space that can be compared using cosine similarity. The specific model used for this paper was ”all-mpnet-base-v2”, since it used some of the largest amount of training data and therefore produced the most accurate and detailed embeddings.

The standard tokenizer for MPNet is used to turn the input text into tokens to feed into MPNet. Any document with text over the 512 token limit of MPNet has the extra text truncated, and the extra text is not used. Once the 768-dimensions text embeddings are generated by MPNet for each token from the text, they are immediately pooled together as an average so that there is a single 768-dimension vector per document.

The second step is to then do the same MPNet embedding and pooling technique to the labeled text samples that will be provided by the domain expert. The class labels also need to be provided by a domain expert and then embedded into a vector as well. These class labels can be one word or multiple words. Now that all the relevant text has been turned into pooled MPNet embeddings, they can now be compared to each other using cosine similarity.

3.2 Document Scoring

There are two separate cosine similarity scoring processes that are combined and averaged together for each document’s score per class (DS): the highest sample cosine similarity score per class for each document (S1), and the class label cosine similarity score per document (S2). To get the first score, the highest cosine similarity amongst the samples for each class is obtained against each document. To get the second score, it’s simply the class label embedding cosine similarity for each class against each document.

$$L_k,\ k \in \left\{1,\ \ldots,\ p\right\},\ \text{where p = number of classes}$$
$$A_{ki},\ i \in \{1,\ \ldots,\ n\},\ \text{where n = samples per class}$$
$$D_j,\ j \in \left\{1,\ \ldots,\ m\right\},\ \text{where m = number of documents}$$
$$S1_{kj} = max\left(\left\{cos\ sim\left(A_i, D\right) : i = (1, \cdots, n)\right\}\right)$$
$$S2_{kj} = cos\ sim(L, D)$$
$$DS_{kj} = mean(s1, s2)$$

Figure 1: Document score (DS) for a given class ‘k’ and document ‘j’.

3.3 Selecting Top Scores

Once each document has a score for each class, the highest score for a document amongst all the classes is chosen as the class winner. Then, the top portion of scorers for each class are selected to be used. Specifically, the scores that are higher than Equation 3 are chosen. You can lower $\boldsymbol{\alpha}$ to lower the accuracy and widen the selection of samples returned or increase $\boldsymbol{\alpha}$ to improve the accuracy and reduce the number of samples returned. As the next step is to feed this data through SVM, it may be worth lowering the overall accuracy to increase the data available for SVM to use.

$$DS_{kj} > median(DS_k) + \alpha\sigma$$

Figure 2: Minimum document score DS required per class ‘k’

These high-confidence samples for each class are used to train the SVM algorithm implemented in the sklearn library using the radial basis kernel. Different kernels were examined, and the radial basis kernel provides the best overall results without having to modify the default hyperparameters. This SVM classifier is then used to do the final predictions on the full dataset.

4 Results

A variety of datasets were used to test the method proposed in this paper, and ensure it has flexibility amongst a variety of challenges. While cosine similarity + SVM is rarely a top-performing solution, it’s highly adaptable to many topic classification and sentiment analysis datasets without complex hyperparameter tuning, and produces competitive results compared against two well-performing weak supervision text classification methods, X-Class and COSINE. X-Class uses a one-word class label to classify the data and doesn’t require clean samples to classify the data. COSINE does require clean samples, but only as validation samples for selecting a final model. For X-Class, the numbers from the original paper are used for the dataset if available, else the publicly available code is used to generate the results.

4.1 Hardware Requirements

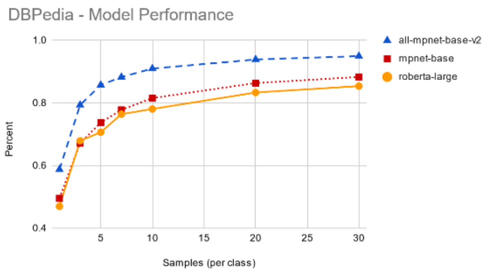
Cosine similarity + SVM hardware cost is greatly minimized compared to many popular weak supervision methods. Common class label-based methods require using the entire corpus word embeddings, which can quickly grow to many GBs for even a small corpus of documents. By only requiring the averaged document

Dataset	Type of Task	Classes	Split	Samples	Embedding Generation Time CPU (hours)
AG News	Topic Classification	4	Train	120000	2.4
DBPedia	Topic Classification	14	Test	70000	1
20Newsgroup	Topic Classification	5	Both	17870	0.9
20Newsgroup	Topic Classification	20	Both	18845	1
Yelp	Sentiment Analysis	2	Both	38000	1
IMDB	Sentiment Analysis	2	Both	50000	2.2

Table 1: Details for the datasets used in this paper. The CPU used was an Intel Core i9-9900k @ 3.6 GHz processor with 32 GB of RAM. The GPU used was a Nvidia RTX 2080 Ti. Batch size for embedding is 128 for GPU, 1 for CPU.

Dataset	Letters per Document (Mean)	Corpus Size	Samples per Class	Total Samples	Sample Embedding Time (s)	Training Time (s)	Testing Time (s)	Total Time (s)
20Newsgroup	1838	18845	1	20	5	6	~0	11
	1838	18845	5	100	21	8	~0	29
	1838	18845	10	200	40	10	~0	50
DBPedia	1293	70000	1	14	1	4	2	7
	1293	70000	5	70	3	4	5	12
	1293	70000	10	140	3	5	4	12

Table 2: Training and testing time requirement for cosine similarity + SVM with two different datasets using only a CPU. The CPU used was an Intel Core i9-9900k @ 3.6 GHz processor with 32 GB of RAM. Batch size for embedding is 1.



Model Name	Model Size
all-mpnet-base-v2	438 MB
mpnet-base	532 MB
roberta-large	1400 MB

Figure 3: The performance of different models using SVM with document embeddings, along with estimates of their model size for inference.

embedding, storage requirements grow linearly with document count 'm', and it is not dependent on document size.

Large models are not required to achieve competitive results when the PLM is fine-tuned using SBERT and a cosine similarity goal. The SBERT fine-tuned "all-mpnet-base-v2" model used in this paper performs significantly better with SVM than either equal sized or larger models that were not fine-tuned using SBERT. Fine-tuning SBERT with a cosine similarity goal for the embedding space may allow the embeddings to be more linearly separated in high dimensions, which could account for the significant baseline improvement with SVM. This advantage further minimizes both the memory and storage needs required.

By minimizing the model size, the inference time and computational requirements are consequently reduced as well. The post-corpus embedding training and testing runtime is measurable in seconds, and the primary runtime bottleneck is the initial corpus embedding process. SVM can potentially be a source of slowdown if too many values are fed to it, but this can be controlled by the $\boldsymbol{\alpha}$ value. Cosine similarity + SVM's runtime is broken down in Table 2.

4.2 Testing Procedure

Due to the variance from selecting clean samples to use, each sample count was tested 20 times with a different sample selection each time. The average macro f1 for those 20 epochs are shown. $\boldsymbol{\alpha}$=3 is used for the topic classification datasets, except for AGNews as it was the largest corpus so $\boldsymbol{\alpha}$=3.5 was used to accelerate testing. $\boldsymbol{\alpha}$=2 was used for sentiment analysis datasets, because there wasn't enough data selected with $\boldsymbol{\alpha}$=3. All SVM implementations use the default hyperparameters for C and gamma. Cosine similarity + SVM can start at 0 samples by only using the class label score (S2). Cosine similarity + SVM and SVM are always the average of 20 runs. COSINE data is taken from the original paper and is the average of 5 runs. X-Class data is taken from the original paper except for the two untested datasets IMDB and 20Newsgroup (20 class), where it is the result of a single run.

4.3 Datasets

A variety of common text classification datasets were used to evaluate this method.

- AGNews (Zhang et al., 2015) is a topic classification dataset from a set of short news story summaries.
- DBPedia (Zhang et al., 2015) is a topic classification dataset made of a list of short article descriptions from DBPedia.
- 20Newsgroup (Lang, 1995) is a topic classification dataset made of a collection of news organized into 6 main groups or 20 sub-groups.
- IMDB (Maas et al., 2011) is a sentiment analysis dataset from a list of movie reviews from IMDB.
- Yelp (Zhang et al., 2015) is a sentiment analysis dataset from a list of business reviews.

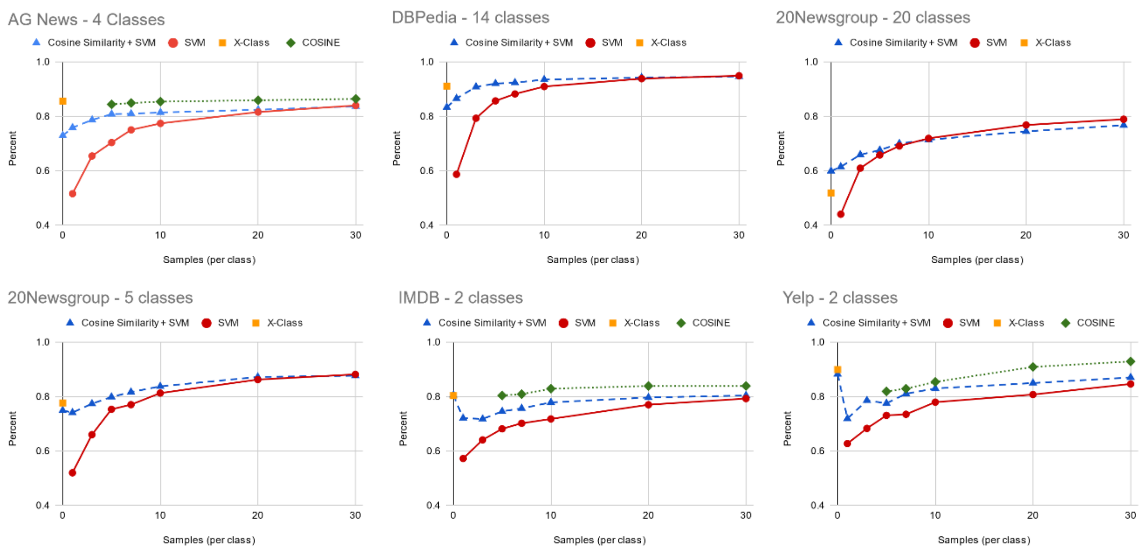


Figure 4: The results of SVM, cosine similarity + SVM, and two recent weak supervision methods.

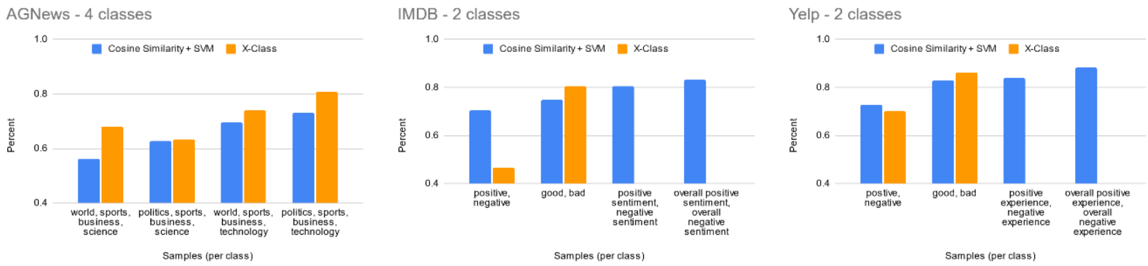


Figure 5: The effect of changing the class label on the macro f1 accuracy when 0 clean samples are available for cosine similarity + SVM and X-class.

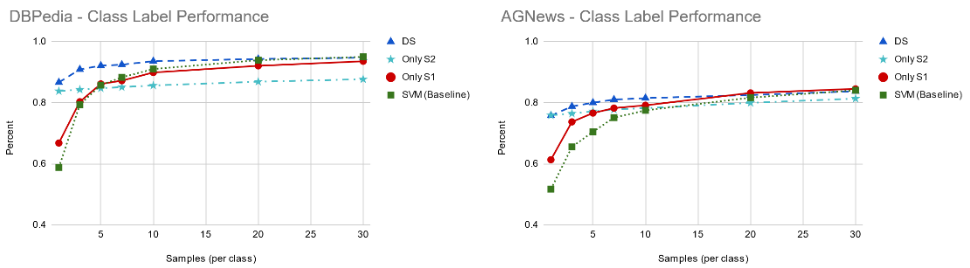


Figure 6: Ablation study of different scoring components. For topic classification tasks, using DS (S1 & S2) provides the best performance over using S1 or S2 separately.

5 Analysis

5.1 Ablation Study

The performance of cosine similarity + SVM is better than standalone SVM when the number of samples per class is less than ten for all topic classification datasets, at which point it tends to converge. S1 and S2 provide two weak supervision signals that work together better than as separate methods for all topic classification datasets, as shown in Fig 3.

The S2 score performed best for the sentiment analysis datasets, and including the S1 score reduced the accuracy. Due to sentiment analysis being a higher-level concept than simple topic matching, it may be harder for document comparisons to identify sentiment. It may be more effective to capture the sentiment with a summary label of the objective versus using examples of the objective. This is reinforced by the drop in accuracy from the class label score when samples are provided for Yelp and IMDB datasets.

5.2 Comparison to Previous Work

Cosine similarity + SVM stays competitive with the two other high-performing weak supervision methods reviewed in this paper. X-Class is dependent on the quality of the class label, and it can have a large effect on the final accuracy. Even semantically similar labels have this effect, as shown in Fig 2. Cosine similarity + SVM is also affected by the quality of the label, but it can be offset by clean samples for topic classification tasks. The performance of cosine similarity + SVM is less dependent on the specific words in a particular corpus, and is instead determined by both accuracy and descriptiveness, which makes for a more simple, general application across varied datasets. An example of this behavior is found in Fig 2; "positive" and "negative" perform similarly for both IMDB and Yelp for cosine similarity + SVM but have extreme variance with X-Class.

For sentiment analysis tasks, overall goal summarization with cosine similarity + SVM performs much better with only the class label (S2) score. Cosine similarity + SVM with only the class label can match and outperform the accuracy of COSINE with 5 clean samples. For topic classification with AGNews, cosine similarity + SVM closely approaches COSINE performance.

6 Conclusion

Fine tuning PLMs is a resource and time intensive process. Even organizations that can afford to support expensive finetuning methods may want to instead repurpose older and more limited existing infrastructure. The ideal solution would be transfer learning without modification, where any general PLM trained by a large group with resources can be used directly for any common NLP task.

As shown by the results, averaged document embeddings from a meaningful vector space provide competitive performance for topic and sentiment classification tasks while minimizing computational, storage, and memory requirements. It suggests that averaged dense embedding vectors have all the information needed to reach a similar level of performance versus more complex, hardware-expensive methods.

7 Future Work

Comparing documents and class labels is a simple process; further processing methods could yield improvements to the overall accuracy. The clean samples provided can be broken down into smaller pieces to provide more signals to refine the weak supervision labels. Alternative pooling methods could be explored for the word pooling step, such as maximum, minimum, median, etc.

8 Potential Risks

Pre-trained language models can have biases and unintended associations, and these can be especially present in models finetuned to produce semantically similar words. This risk is most present in low resource areas such as weak supervision, where there is little room for human correction. However, by allowing for multi-word class labels, using document averages, and including two different scoring methods, individual biases amongst particular words is greatly reduced. Classification can also be directly adjusted by including samples of a particular incorrect classification as part of the sample set for the correct class. This can mitigate the risk of unintended classifications due to inherent model biases.

9 Limitations

There were only two types of text classification examined in this paper. Other types of text classification, such as relation classification, may not perform well with this paper's semantic similarity-based method without additional processing.

The ability to retain a near real-time ability to modify the classifier is severely diminished with larger datasets over one million samples, without careful consideration of the amount of data being fed into the SVM classifier. The α parameter becomes more sensitive when datasets start to increase in size, and it could cause over-sampling if it's not carefully monitored. The need to perform cosine similarity against all documents in a larger dataset may also limit the near real-time scalability of this method.

References

Jair Cervantes, Farid Garcia-Lamont, Lisbeth Rodrıguez-Mazahua, and Asdrubal Lopez. 2020. A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing*, pages 408:189– 215.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, *Punta Cana, Dominican Republic* pages 6894–6910, Association for Computational Linguistics.

Kenta Mikawa, Takashi Ishida and Masayuki Goto, A proposal of extended cosine measure for distance metric learning in text classification. *2011 IEEE International Conference on Systems, Man, and Cybernetics*, *Anchorage, AK, USA, 2011*, pages. 1741-1746

Ken Lang. 1995. Newsweeder: Learning to filter netnews. *In Machine Learning, Proceedings of the Twelfth International Conference on Machine Learning*, *Tahoe City, California, USA, July 9-12, 1995*, pages 331–339. Morgan Kaufmann.

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

Nils Reimers, and Iryna Gurevych. 2019. SentenceBERT: Sentence embeddings using Siamese BERT-networks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Hong Kong, China.* pages 3982–3992, Association for Computational Linguistics.

Tim Schopf, Daniel Braun, and Florian Matthes. 2023. Evaluating unsupervised text classification: Zero-shot and similarity-based approaches. *In Proceedings of the 2022 6th International Conference on Natural Language Processing and Information Retrieval, NLPIR '22, page 6–15, New York, NY, USA.* Association for Computing Machinery.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: masked and permuted pre-training for language understanding. *In Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS'20, Red Hook, NY, USA.* Curran Associates Inc. 1

Zihan Wang, Dheeraj Mekala, and Jingbo Shang, 2021. X-class: Text classification with extremely weak supervision. *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3043–3053, Association for Computational Linguistics.

Yue Yu, Simiao Zuo, Haoming Jiang, Wendi Ren, Tuo Zhao, and Chao Zhang. 2021. Fine-tuning pretrained language model with weak supervision: A contrastive-regularized self-training approach. *In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 1063–1077. Association for Computational Linguistics

Jieyu Zhang, Cheng-Yu Hsieh, Yue Yu, Chao Zhang, and Alexander Ratner. 2021b. WRENCH: A comprehensive benchmark for weak supervision. *In Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual.*

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *In Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 649–657

Dawei Zhu, Xiaoyu Shen, Marius Mosbach, Andreas Stephan, and Dietrich Klakow. 2023. Weaker than you think: A critical look at weakly supervised learning. In Rogers, A., Boyd-Graber, J., and Okazaki, N., editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Toronto, Canada.* pages 14229–14253, Association for Computational Linguistics.

A Appendix A: Score Distribution

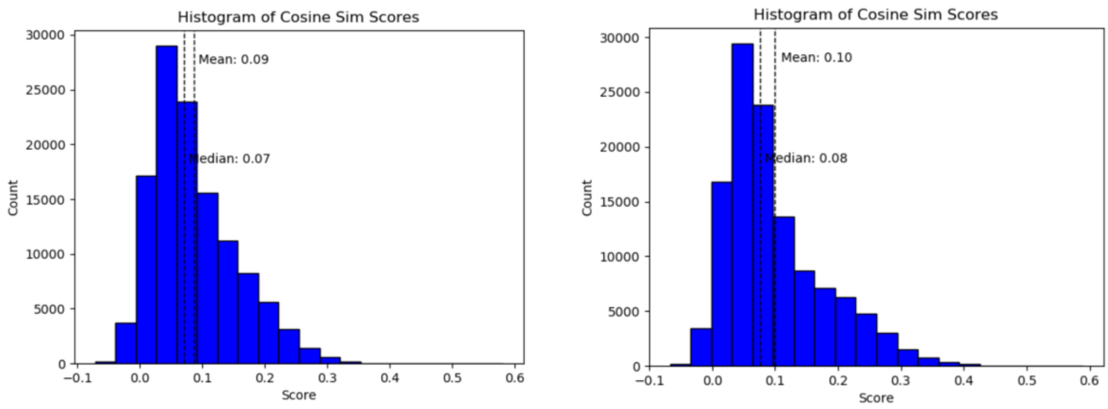


Figure 6: The score distribution for a single class of document scores (DS_j) with 5 clean samples on the left, and with 10 clean samples on the right for the AGNews dataset.

B Appendix B: Python Libraries

- The hugging face transformers library was used to run the mpnet model.
- The sklearn library was used to implement SVM.
- The pandas library was used to load csv files.
- The sentence_transformers library was used to perform cosine similarity comparison.