A Replication Study on Baseline Results of ConWea

Yacun Wang
yaw006@ucsd.edu

Abstract

In the context of weakly supervised text classification, researchers have put extensive efforts into investigating models that only require a few human-annotated seed words per class. In particular, ConWea (Mekala and Shang, 2020) has proposed an impactful method to contextualize the data, expand seed word sets. It has also compared the framework to various baseline results. This paper aims to replicate two baseline results of ConWea: IR-TF-IDF and Word2Vec. We also explored the best classification setting possible for the two methods including corpus preprocessing, hyperparameter settings, etc., since the original paper did not explicitly state them. We have found that among our replications, we are able to catch up with most baseline performances and provide a few settings with better performances in other tasks than in the original ConWea paper.

1 Introduction

In recent years, researchers have been extensively investigating the setting of weakly supervised text classification, which is the possibility to classify texts based on fewer human annotation, as annotating a huge corpus require plenty of human expertise in the domain of the corpus, high-quality labels, and most importantly substantial efforts. In particular, a common setting of weak supervision require users to give a few words highly representative of the class label (i.e. seed words).

Examples of such seed-driven methods (Tao et al., 2018; Meng et al., 2018) utilizes an iterative framework with generating pseudo-labels, learning relevance mapping, and expanding seed word sets. More advanced methods like ConWea (Mekala and Shang, 2020) focuses on resolving the issue of multi-meaning words and layers contextualization methods with the iterative framework. This approach is able to achieve much better classification accuracy and has justified that contextualization is key to learn and disambiguate different meanings for words that appear under different contexts. The paper also compares the classification performance between the ConWea framework and various baseline results.

Table 1: Dataset Statistics

Dataset	# Docs	# Coarse	# Fine
NYT	11,527	5	26
20News	18,259	6	20

In this paper, we aim to replicate two of the baseline results using the same datasets as ConWea and directly compare the baseline results to those experimented in the ConWea paper as well as the best results achieved by the ConWea framework. In particular, we experiment with: (1) IR-TF-IDF, which finds the relevance between document and label by aggregated TF-IDF values of seed words; (2) Word2Vec (Mikolov et al., 2013), which learns context-free vector representations of each word that appeared in the corpus, finds label and document representation by aggregating the vector representations, and predict the label with the best cosine similarity. Note that in this paper we have constrained ourselves to not use any neural models that will surely improve the performance when applying to the baseline models such as Word2Vec as a downstream task.

We find out after extensive experiments that we are able to reach slightly lower or similar results in most tasks presented in the original ConWea paper, and exceeds their performance in other tasks such as fine-grained datasets in the Word2Vec setting. We also identified plenty of drawbacks about the baseline models themselves and the experiment settings that we chose during the process.

2 Data

Following the ConWea paper (Mekala and Shang, 2020) that we are targeting to replicate, this paper uses the same two news datasets in the experiments. The basic statistics are shown in 1, with slight differences detected from the original paper. Appendix A shows detailed distributions of coarse labels:

- **The New York Times (NYT):** The NYT dataset consists of news articles published by the New York Times. There are 26 fine-grained categories in 5 coarse genres: arts, sports, business, politics, science. The coarse categories are imbalanced with mostly sports news.

- **The 20 Newsgroups (20News):** The 20News dataset [1] is also a collection of news articles categorized into topics. There are 6 imbalanced coarse-grained categories with labels harder to count as words: alt, comp, misc, rec, sci, soc, talk. There are 20 fine-grained categories.

3 Evaluation Metrics

For all results sections, both micro-F1 and macro-F1 scores are reported. In particular, when computing classification accuracy, micro-F1 scores treat each observation equally and therefore have larger impact on larger classes. It is also equivalent to accuracy. On the other hand, macro-F1 scores treat each class equally regardless of the prevalence, and therefore will be more objective for imbalanced classes as we have.

4 IR-TF-IDF

4.1 Method

The first experiment with the baseline model involves TF-IDF values, a popular metric describing the importance of a particular word to each piece of text, computed by the Term Frequency (TF) and Inverse Document Frequency (IDF). In particular, given a word w and a document d:

$$\mathrm{tf}(w,d) = \text{Term Frequency of } w \text{ in } d$$

$$\mathrm{idf}(w) = \log\left(\frac{\text{\# Documents}}{\text{\# Documents containing } w}\right)$$

where if any word w appears across documents, the IDF score will be low, leading to less importance of the word. For example, articles such as *the* is used very frequently in most texts, and will be considered not indicative of the sentiment with a low IDF score close to $\log 1 = 0$. Then the TFIDF score is defined as:

$$\mathrm{tfidf}(t,d) = \mathrm{tf}(t,d) \cdot \mathrm{idf}(t)$$

Adapting to this weak supervision setting, we form the vocubulary set using all seed words from all labels, and used the `sklearn` text feature extractor and transformer `tfidfVectorizer` to transform the seed words into vectors of TFIDF scores.

[1] `http://qwone.com/~jason/20Newsgroups/`

To classify the document d, we first find the relevance score $\delta_{d,l}$ by aggregating TF-IDF scores in 1 of the 3 different ways for each seed words set S_l of that label l:

$$\text{Mean:} \quad \delta_{d,l} = \frac{1}{|S_l|}\sum_{s\in S_l} \mathrm{tf}(s,d)\cdot \mathrm{idf}(s)$$

$$\text{Mean Sq:} \quad \delta_{d,l} = \frac{1}{|S_l|}\sum_{s\in S_l} (\mathrm{tf}(s,d)\cdot \mathrm{idf}(s))^2$$

$$\text{Max:} \quad \delta_{d,l} = \max_{s\in S_l}(\mathrm{tf}(s,d)\cdot \mathrm{idf}(s))$$

Then when we obtain a relevance between d and all labels $l \in L$, we classify the document based on the most relevant label:

$$\hat{l}_d = \operatorname*{argmax}_{l\in L} \delta_{d,l}$$

4.2 Experiment Settings

Different from usual settings where we use the TF-IDF values for all possible words and used them for downstream tasks, it is only possible to use TF-IDF values for all seed words in the seed-driven weak supervision setting, as they are the only information we know that relates documents with labels.

To address the issue of same-root words such as sports and sport being counted separately, we used the Snowball Stemmer [2] in `nltk.stem` module to get the stem for each word in the corpus and the seed words.

In particular, in order to keep the relevance, we tested IR-TF-IDF method on the following settings:

- Stemming: True, False;
- TF-IDF Aggregation: Mean, MeanSq, Max.

Note: The fine-grained 20News dataset is only evaluated without stemming, as the duplicate seed word "window" after stemming causes confusion on multiple meanings in different contexts.

4.3 Results and Discussions

The results are shown in Table 2. From the results, we can see that in most cases, our replications results are getting close to the best-tuned models presented in the papers, and some results are able to keep up with the pace. This shows that our effort in tuning the choices have been successful in some, but not all cases. Fortunately, this method

[2] `https://snowballstem.org/`

Table 2: IR-TF-IDF experiment results using different classification settings

IR-TF-IDF Settings		**NYT**				**20 Newsgroup**			
		5-Class (Coarse)		**26-Class** (Fine)		**6-Class** (Coarse)		**20-Class** (Fine)	
Stem	Agg	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
True	Mean	**0.66**	0.51	**0.53**	0.56	**0.48**	**0.51**	-	-
True	MeanSq	0.66	**0.51**	0.52	0.55	0.48	0.51	-	-
True	Max	0.65	0.51	0.47	0.51	0.47	0.51	-	-
False	Mean	0.65	0.50	0.53	**0.58**	0.44	0.50	**0.49**	**0.53**
False	MeanSq	0.65	0.50	0.52	0.57	0.44	0.50	0.49	0.53
False	Max	0.65	0.50	0.48	0.53	0.44	0.50	0.48	0.52
ConWea: IR-TF-IDF		0.65	0.58	0.56	0.54	0.49	0.48	0.53	0.52

is generally effective as it's performance is much better than random guessing.

From the classification setting choices, we can see that the regular mean approach in aggregating TF-IDF values obtain the best performance with stemming, showing that stemming does help identify words from the same root. However, in most situations, the choices do not affect the results by a lot.

The results from different datasets also indicates that one method might not work well for all datasets, especially when the datasets contain seed words of different quality. For example, from the seed word information presented above for 20News, stemming such words might not lead to a big effect since they are not proper English words by themselves.

4.4 Limitations

There are plenty of limitations of the IR-TF-IDF method and the design choices, apart from the fact that none of the baseline models take contextualized words into account.

TF-IDF is essentially a counting method, and most counting methods share the limitation of following exact matches. From words of different forms but stemmed from the same word root will be treated separately. Even with stemming applied and improved the result by a bit, these stemming algorithms are mostly deterministic, so it may successfully find the root for some words but also undermines other words by taking away essential parts of the word. More importantly, when the words in vocabulary are not regular English words, such as acronyms and short expressions, the stemming algorithm will fail.

TF-IDF scores only determines importance of words, which are mostly words that: (1) occur a lot in certain documents; (2) occur rarely in other documents. These criteria might be confusing when classifying texts, as some words might occur repeatedly but only in certain classes, then they should be highly label-indicative but will have low TF-IDF scores. Similarly, words that distinctly occur in only a few documents might be too specific to the few documents themselves instead of representing the entire class, and specific words might be shared by classes because they share details. For example, the word "sport" could occur in most sports articles but have low IDF values; while the word "staples" might be too specific as it could represent the stadium where both basketball games and concerts take place.

5 Word2Vec

5.1 Method

The second experiment with the baseline model involves Word2Vec algorithms (Mikolov et al., 2013) that learns continuous vector representations for each word based on their neighboring words, and derives similarity between words because their neighboring word set is close to each other. Using the `Word2Vec` module from `gensim.models` package, we train Word2Vec models based on the input corpus then obtain the vector representations for each word in the corpus and the seed word sets.

To predict the label, we separately find the document and label representations by aggregating the vector representations:

$$v_l = \frac{1}{|S_l|}\sum_{s \in S_l} v_s, \quad v_d = \frac{1}{|W_d|}\sum_{w \in W_d} v_w$$

where S_l is the seed word set for label l, W_d is the words in document d.

Finally we find the relevance by taking the cosine similarity between documents and labels. In particular, we to predict the label $\hat{l}$ for document d, we find:

$$\hat{l}_d = \max_{l \in L} \text{cosine}(l, d) = \max_{l \in L} \frac{\langle v_l, v_d \rangle}{\|v_l\| \cdot \|v_d\|}$$

where L is the set of labels.

5.2 Experiment Settings

In the Word2Vec model, there are plenty of variations and hyperparameters to tune, and under different datasets the optimal setting will vary from each other. In particular, we have experimented with:

- Different preprocessing techniques including using the same Snowball Stemming algorithm;
- Different model framework including using Skipgram and Continuous Bag of Words (CBOW);
- Different training algorithms including Negative Sampling and Hierarchical Softmax, both presented in the original Word2Vec paper;
- Different hyperparameter settings including neighborhood window size, vector dimension, etc.

We have also added the START and END token for each document to represent the starting and endpoint of sentences and let the models take into account of these information.

5.3 Results and Discussion

Since there are extensive number of experiments, we first provide an overview of the settings we obtained as the final choice(s):

- Model Framework: CBOW. Due to the nature of CBOW that models only know co-occurrence, CBOW trains much faster and is able to give better and quicker results;
- Training Algorithm: Hierarchical Softmax. We observe that HS gives better results and has fewer hyperparameters to tune, given the setting that we're preferring higher-dimensional latent factors;
- Epochs: Since Word2Vec is a neural approach, we need more epochs to converge the algorithm, so we choose all training settings to run 75 epochs;
- Minimum Count: To better capture the meaning of words based on neighbors, we choose to ignore all words that have less than 10 occurrences in the entire corpus;
- Window Size: After experimenting with the hyperparameter, we have recognized that larger window sizes will lead to better results, so we set size to be 10. Intuitively, since we wish to understand the topic of news articles instead of semantics of particular phrases or sentences, we need to include more information that can be accessed by each word;
- Vector Dimension: We present the results with different vector dimensions in each dataset trained, but in general we prefer more dimensions so the latent factors are able to capture more information in each of their entries;
- Stemming: We choose to stem each word and seed word since we wish to give more context information for each word root, and it turns out this leads to better model performances.

We have presented the results with respect to different datasets and different vector dimensions in Table 3. From the table, we can see that other than the NYT Coarse dataset, we have outperformed the baseline results in all other datasets presented in the original paper.

Interestingly, we see that the largest dimension 256 only shows an advantage in the NYT dataset, which mostly indicates that the dataset contains enough general information for the Word2Vec model to learn from. In contrast, the smallest vectors become clearly better in the 20News dataset. It might be due to the fact that the labels of this dataset are not English words and therefore it's hard to find their neighbors even after stemming.

We can also observe that both fine-grained datasets and the 20News are harder to predict, which indicates that the model is struggling more with more fine-grained or ambiguous seed words. This again stresses the importance of high-quality seed words. They are essential to such weak supervision settings as they serve as the only human annotation on the ground truth values.

5.4 Limitations

One of the most pivotal drawback of the Word2Vec model is that the neighborhood window is chosen

Table 3: Word2Vec experiment results using CBOW, Hierarchical Softmax, but different latent vector dimensions.

Word2Vec	**NYT**				**20 Newsgroup**			
	5-Class (Coarse)		**26-Class** (Fine)		**6-Class** (Coarse)		**20-Class** (Fine)	
Dimension	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
64	0.92	0.76	**0.75**	0.63	**0.72**	**0.65**	**0.56**	**0.54**
128	0.91	0.75	0.73	0.64	0.70	0.63	0.55	0.54
256	**0.92**	**0.77**	0.72	**0.66**	0.65	0.60	0.51	0.51
ConWea: W2V	0.92	0.83	0.69	0.47	0.51	0.45	0.33	0.33

without any context about different word meanings, and therefore learning the meaning for those words will be extremely hard and ambiguous to the model. The ConWea framework directly addresses this problem.

The Word2Vec model also fails to address the importance of order in natural languages as the semantics brought by word orderings lead to drastically different meanings. In the Word2Vec framework, the model only cares about the words that appear in the neighbor window instead of the order they appear. More advanced neural architectures such as recurring structures used in LSTM, GRU will perform much better by addressing this issue.

An important limitation to our design choices is that the `Gensim` package does not support GPU, so training large neural models like Word2Vec becomes extremely time-consuming and inefficient in CPUs. This becomes the main reason that we skipped the time-consuming algorithms and refrained from working on extensive hyperparameter search to try most combinations.

6 Conclusion

In conclusion, we have achieve close or better classification accuracy results in both baseline models we experimented and identified limitations both in the models themselves and the design settings we choose. From these results, we're able to recognize the advantages of the ConWea framework itself as it addresses plenty of drawbacks presented in these baseline models.

Acknowledgements

This paper draws inspiration from Mekala and Shang, 2020 and aims to replicate two of the baseline models provided by the paper, namely IR-TF-IDF and Word2Vec. The results presented in this paper could be directly compared to the original paper.

References

Dheeraj Mekala and Jingbo Shang. 2020. Contextualized weak supervision for text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 323–333, Online. Association for Computational Linguistics.

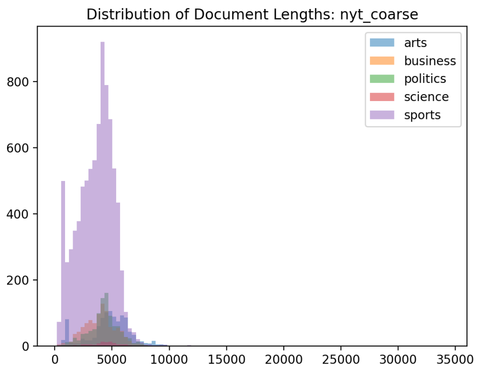
Yu Meng, Jiaming Shen, Chao Zhang, and Jiawei Han. 2018. Weakly-supervised neural text classification. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 983–992, New York, NY, USA. Association for Computing Machinery.

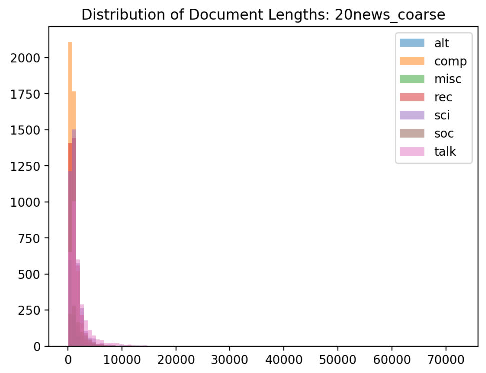
Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Fangbo Tao, Chao Zhang, Xiusi Chen, Meng Jiang, Tim Hanratty, Lance Kaplan, and Jiawei Han. 2018. Doc2cube: Allocating documents to text cube without labeled data. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 1260–1265.

A Dataset Distributions

We show the distribution of document categories and lengths with respect to coarse-grained categories.




Distribution of Document Lengths: 20news_coarse
alt
comp
misc
rec
sci
soc
talk
0
250
500
750
1000
1250
1500
1750
2000
0
10000
20000
30000
40000
50000
60000
70000