Iterative machine learning applied to annotation of text datasets

Thiago Abdo[0000−0003−3086−1461] and Fabiano Silva[0000−0001−5453−6175]

Informatics Department, Federal University of Paraná, Curitiba, Brazil
{tjabdo,fabiano}@inf.ufpr.br

Abstract. The purpose of this paper is to analyze the use of different machine learning approaches and algorithms to be integrated as an automated assistance on a tool to aid the creation of new annotated datasets. We evaluate how they scale in an environment without dedicated machine learning hardware. In particular, we study the impact over a dataset with few examples and one that is being constructed. We experiment using deep learning algorithms (Bert) and classical learning algorithms with a lower computational cost (W2V and Glove combined with RF and SVM). Our experiments show that deep learning algorithms have a performance advantage over classical techniques. However, deep learning algorithms have a high computational cost, making them inadequate to an environment with reduced hardware resources. Simulations using Active and Iterative machine learning techniques to assist the creation of new datasets are conducted. For these simulations, we use the classical learning algorithms because of their computational cost. The knowledge gathered with our experimental evaluation aims to support the creation of a tool for building new text datasets.

Keywords: Machine learning · Natural Language Processing · Sentiment Analysis.

1 Introduction

The creation of new annotated datasets for machine learning is a slow and costly process. Usually, this process is done in a manual and error-prone manner. Tools that make the process of building a new annotated dataset less error-prone, faster, and straightforward [1, 5, 16] are of great interest. In this paper, we present an experimental evaluation of machine learning techniques to support the development of such tools. The first one can suggest annotations based on past annotations (iterative learning) [4, 8, 11]. And, the second, choose which is the next example that should be annotated to improve the performance of a machine learning algorithm (active learning) [9, 13, 15, 17].

In this paper, we provide the rationale behind choosing a machine-learning algorithm to support the development of a web application to be used as an annotation tool. The experimental evaluation is conducted to evaluate both iterative and active learning using a Brazilian Portuguese dataset with 4333 annotated

examples. For evaluating the machine-learning algorithm, we have to test not only the performance but also have to account for the environment where this algorithm is going to be running. We choose BERT [3] to represent deep learning algorithms, as it is easy to find different versions trained in several languages. We also executed a full training using our dataset.

To represent classical learning algorithms, we choose to combine the feature extractors, Word2Vec (W2V) [10] and GloVe [12], with the classifiers Random Forest (RF) [7] and Support Vector Machine (SVM) [2]. We tested each of the feature extraction algorithms with the dataset and also with extra content.

Although deep learning strategies have a better performance than the classical approach, it also has an increased computational cost that is not viable to the environment where the tool is executed. Because of that, we choose to use classical learning algorithms even with a performance hit that can be as big as 15% in our tested datasets.

To evaluate the iterative and active learning, we created a simulation using our annotated dataset [16]. At first, we used few examples of each class, after each round, we added more examples, randomly for iterative learning and by uncertainty for active learning.

2 Materials and methods

In all experiments, we use a database about social and racial quotas under construction by the Communication Department of Federal University of Paraná (UFPR). This database has 4333 comments from social networks. They are heterogeneous in size and format. From this database were manually created 22 datasets (*https://gitlab.c3sl.ufpr.br/mestr/datasets*). We are using these three datasets in our tests: positioning, rationality, and theme. To ensure that each dataset was balanced these actions were necessary to: remove the classification "does not apply" from the dataset "positioning", as this had only 289 examples and reduce the "absent" classification of the "rationality" dataset since it had 2536 examples (about 41 % more than the other most present class).

We also created a new dataset by reducing the "theme" dataset called "reduced theme". The table 1 shows the possible classifications for each dataset with the number of examples.

Table 1. Datasets with the number of examples in each class.

Dataset	**Classes**
Positioning	neutral (1501), favorable (1298), and contrary (1245)
Rationality	present (1797) and absent (1797)
Theme	off topic (1181), relational (1213), structural (937), and unknown (1002)
Reduced Theme	off topic (off topic and unknown) (2183) and on topic (relational and structural) (2150)

2.1 The machine learning algorithm

To choose the machine learning technique that is the best for the tool, we developed two experiments. In the first one, we tested five different BERT training sets. The goal is to evaluate the performance of each Bert's model concerning the F1 metric (using different amounts of data in the pre-training) and the training time of the algorithm. In the second one, we combined feature extractors (W2V and GloVe) with non-linear classifiers (SVM and RF) and used three training sets. The second experiment aims to evaluate the performance concerning the F1 metric, the total training time for algorithms that do not have deep learning, such as W2V and GloVe, and if adding more data about the subject of the dataset improves the performance.

In the first experience, we use BERT due to its training process be split into two parts (pre-training and fine-tuning). So, it's easy to find versions with the pre-training completed on the internet for different languages, and it's competitive with state-of-the-art for several natural language processing problems. We tested five BERT's pre-training: pre-training only with data from the database, pre-training with texts extracted from the internet, pre-training with texts extracted from the internet together with the data from the base data, pre-trained with texts in different languages by Google, and pre-trained with texts in Portuguese by Neuralmind [14]. All models went through fine-tuning using only the quota database.

- Bert 1: pre-training with the dataset examples only.
- Bert 2: pre-training with texts about quotas extracted from the internet.
- Bert 3: pre-training with texts about quotas extracted and dataset examples.
- Google: multi-language provided by Google.
- Neuralmind: single language provided by Neuralmind.

In the models that we did the complete training (Bert 1, 2, and 3), the pre-training was tested with variations of the following parameters: the size of the internal layers, number of layers, number of attention mechanisms, and input's max size. The tested values for each parameter are exhibited in the table 2. In the fine-tuning, we tested every model generated with the combinations of these parameters. The size of each inner layer, the number of layers, and the input's max size determine respectively: the number of neurons in each inner layer, the number of inner layers, and the maximum size of the input that will be processed. These parameters are common to most deep learning algorithms. The number of attention mechanisms is a specific parameter of BERT and determines the number of attention mechanisms per inner layer.

The attention mechanism is a BERT instrument that allows each word analyzed by this mechanism to have a connection of some intensity with other words in the sentence/text that is analyzed. Another peculiarity of BERT is that the size of the inner layer needs to be multiple of the size of the attention mechanism. The initial values were inspired for these parameters by the values that the Google model used for pre-training. We added variations and reductions due to the size of our datasets and the computational cost.

For fine-tuning, we made variations in the following parameters: entry size, learning rate, and the number of epochs. The tested values for each parameter

Table 2. Variation of each parameter tested in the pre-training of BERTs 1, 2, and 3.

Parameters	**Tested value**
Size of the internal layers	288, 516, 768
Number of layers	4, 8 e 12
Number of attention mechanisms	4 e 12
Input's max size	256 e 512

are presented in the table 3. These parameters are common to most machine learning algorithms. We choose the inputs max sizes according to the number of elements in the dataset: 54.60% has the size of up to 128, 77.75% of up to 256 and, 91.14% up to 512. We defined the number of epochs and the learning rate empirically. We also performed preliminary tests with more epochs (7) and a smaller learning rate (2^{-7}), but we only performed the complete tests with the values in the table 3.

Table 3. Variation of each parameter tested in the fine-tuning phase.

Parameters	**Tested value**
Input's max size	128, 256 e 512
Learning rate	2^{-3} e 2^{-5}
Number of epochs	3 e 5

In the second experiment, we are evaluating machine learning models that do not have deep learning. We developed an experiment combining feature extractors (W2V and GloVe) and non-linear classifiers (RF and SVM). Also, we tried three different variants for training the feature extractors, using: only the dataset for training (represents an extractor without prior knowledge); a set of texts without classification on the same subject of the dataset combined with the dataset (represents an extractor with knowledge of the subject of the dataset); a set of 17 datasets in Portuguese (represents an extractor with language knowledge) [6]. We are trying these 3 types of training data for the feature extractors because our datasets are small and we wanted to do a fair comparison with the Bert models. Due to the nature of the dataset, comments from internet pages, we tried combinations of these two pre-processors: data cleaning (removing links, numbers, excessive repetition of characters and HTML structures) and spelling correction.

In the following list, we have the acronyms for each combination of feature extractors plus training sets that were tested:

- W2V 1: W2V without prior knowledge
- W2V 2: W2V with knowledge of the subject of the dataset
- W2V 3: W2V with knowledge of the language
- GloVe 1: GloVe without prior knowledge
- GloVe 2: GloVe with knowledge of the subject of the dataset

– GloVe 3: GloVe with knowledge of the language

2.2 Techniques to improve learning

We tested two techniques that aim to improve learning speed and feedback to the annotator. Iterative machine learning (IML) [4, 11, 8] can be used both in the context of creating new datasets and in the creation or improvement of a machine learning system. In the context of creating a new dataset, IML is used to assist in labeling examples.

Active machine learning (AML) [13, 17, 9, 15] is a technique that can be useful to reduce the number of examples needed for a machine learning system to achieve the desired performance. In the context of creating a new dataset, AML works together with IML reducing the number of examples that need to be classified before IML can make good classification guesses.

To evaluate the impact of these learning techniques and how the success rate of each model behaves, we developed a simulation. This simulation consists of evaluating the performance of the models with only a part of the dataset, and at each round adding a fixed amount of new examples annotated from each class. How we choose the examples that are added depends on the technique that we are simulating. When we are simulating AML, we add examples that the current model has more uncertainty. When we are simulating IML, we add examples randomly. Allowing us to test the evolution of the performance of the models as if we were creating a new dataset. This way, we do not need to involve human annotators, thus occur in a fraction of the time. The main difference of this simulation for reality is the time it takes to include more examples.

Each simulation started with ten examples of each class. After the training and testing process, we add ten more examples from each class until there are no more examples. We are using the datasets "positioning with data cleaning" and "reduced theme with data cleaning" because they are the ones with the worst and best performance in our tests respectively. In these simulations, we evaluated feature extractors W2V 3 and GloVe 3.

3 Experimental results

In this section, we present the results for the experiments to choose the machine learning algorithm and the simulations with techniques to improve annotation speed and quality.

In our testing, Bert-based algorithms performed better in all scenarios. They had up to 15% more f1-score than the approach with classic algorithms. But their computational cost is too high for the environment where we intend to use it. The two best Bert models that we tested took more than 28 hours to complete fine-tunning. For this reason, even with worse overall performance, we choose to continue our tests of techniques to improve annotation and speed using classic algorithms.

In our simulations of techniques to improve annotation speed and quality, we simulated AML and IML. AML showed that we could use it to speed up learning while using fewer examples. Improving how fast we can start using IML and acting as a balancer to keep the dataset balanced with meaningful examples.

3.1 Choosing a machine learning algorithm

For all models, the best overall parameters setting is in table 4, but for the models that we only performed fine-tuning, we have a N/A (not applicable) in the parameters exclusive for pre-training. We measured overall performance by each model accumulating points depending on the f1 score it had in one test. The model with the highest f1 score accumulates ten points, the second nine. Until the tenth that accumulates 1 point. The models chosen to have the results presented in table 5 are the models that had the best overall performance.

Table 4. Best overall parameters for pre-trainned and fine-tunned BERTs

	Bert 1	**Bert 2**	**Bert 3**	**Google**	**Neuralmind**
Size of the internal layers	768	288	768	N/A	N/A
Number of layers	8	8	8	N/A	N/A
Number of attention mechanisms	4	4	4	N/A	N/A
Input's max size in pre-training	512	512	256	N/A	N/A
Input's max size in fine-tunning	512	512	512	512	512
Learning rate	2^{-5}	2^{-5}	2^{-5}	2^{-5}	2^{-5}
Number of epochs	5	5	5	3	3

Table 5. F1 score for each dataset

	Bert 1	**Bert 2**	**Bert 3**	**Google**	**Neuralmind**
Reduced theme	88.19%	84.95%	**89.81%**	89.35%	88.89%
Theme	59.76%	60.67%	58.81%	63.07%	**64.81%**
Positioning	56.92%	49.17%	59.43%	56.94%	**61.80%**
Rationality	74.02%	75.12%	75.39%	77.65%	**79.89%**

In table 5, Neuralmind had the best performance in three datasets, only in the reduced theme it is worse than Google's pre-training. In addition, Bert 2 and 3 performed better than Google in the positioning dataset. Even with the usage of hardware, time, and data for pre-training and training being lower.

Due to high computational cost, algorithms based on BERT are not ideal for our scenario since the tool is hosted on web servers without access to resources to speed up learning. To illustrate how high computational cost is a problem to run tests on non-optimized hardware such as a server in the cloud, we performed the fine-tuning tests of the Google and Neuralmind models on a four processors

virtual machine. These models took more than 28 hours to complete the fine-tuning, and they had a consumption of 16 Gigabytes of RAM and occupied all the processing of the virtual machine. With a high computational cost, algorithms based on deep learning are not feasible for scenarios that do not have specific hardware for reducing learning time.

The results presented in tables 6, 7, 8 and 9 are using the random forest as the non-linear classifier. In bold text, we have the best model for a pre-processor (the best in a row), and in underlined text, we have the best model pre-processor for a model (the best in a column). In addition to having the best F1 performance in 6 of 12 combinations of dataset and pre-processing, this classifier has a lower computational cost than support vector machines.

Table 6. F1 score for classic models in the dataset "reduced theme"

	W2V 1	**GloVe 1**	**W2V 2**	**GloVe 2**	**W2V 3**	**GloVe 3**
No pre-processing	85.40%	81.70%	**85.64%**	82.86%	81.44%	79.84%
Data Cleaning	**84.25%**	83.09%	83.08%	81.94%	80.78%	81.70%
Spell correction	83.32%	82.62%	**85.87%**	82.15%	82.14%	79.14%
Both	81.94%	80.29%	**84.49%**	82.40%	81.48%	80.32%

For the dataset "Reduced Theme", we can see in table 6 that there is no performance gain from using both pre-processors in any model, and using data cleaning or spell correction brought performance gains in 4 of 6 models tested. When comparing extractors that used the same training data (eg. W2V 1 with GloVe 1 and W2V 2 with GloVe 2), the models using W2V as extractors had a better performance than GloVe for this dataset in 11 of 12 comparisons.

Table 7. F1 score for classic models in the dataset "theme"

	W2V 1	**GloVe 1**	**W2V 2**	**GloVe 2**	**W2V 3**	**GloVe 3**
No pre-processing	52.12%	52.30%	**53.42%**	51.53%	51.79%	48.54%
Data Cleaning	52.95%	**56.36%**	50.80%	53.01%	47.06%	50.38%
Spell correction	50.33%	**53.75%**	51.74%	53.64%	49.05%	49.64%
Both	52.56%	**54.79%**	50.14%	51.14%	49.39%	49.52%

For the dataset "Theme", we can see in table 7 that the pre-processing "data cleaning" improved 3 of 6 models tested. When comparing extractors that used the same training data, the models using GloVe as extractors had a better performance than W2V for this dataset in 10 of 12 comparisons.

For the dataset "Positioning", we can see in table 8 that the pre-processing "data cleaning" improved 3 of 6 models tested, and "both", for 2 of 6. When comparing extractors that used the same training data, the models using GloVe as extractors had a better performance than W2V for this dataset in 7 of 12 comparisons. Also, using an extractor with knowledge of the language (W2V 3

Table 8. F1 score for classic models in the dataset "positioning"

	W2V 1	**GloVe 1**	**W2V 2**	**GloVe 2**	**W2V 3**	**GloVe 3**
No pre-processing	47.61%	48.63%	45.70%	50.92%	51.56%	**53.08%**
Data Cleaning	50.89%	48.97%	49.70%	50.99%	53.59%	**53.96%**
Spell correction	51.31%	48.48%	46.05%	49.57%	**51.62%**	51.23%
Both	49.38%	50.16%	48.61%	47.98%	**54.83%**	51.20%

Table 9. F1 score for classic models in the dataset "rationality"

	W2V 1	**GloVe 1**	**W2V 2**	**GloVe 2**	**W2V 3**	**GloVe 3**
No pre-processing	**68.58%**	67.47%	67.00%	67.67%	67.23%	63.49%
Data Cleaning	66.23%	65.44%	**66.36%**	64.05%	65.68%	65.19%
Spell correction	**69.49%**	64.62%	65.97%	65.24%	68.20%	64.36%
Both	69.11%	66.95%	66.97%	62.39%	**69.67%**	65.71%

and GloVe 3) is better than using extractors with dataset knowledge for every tested pre-processing.

For the dataset "Rationality", we can see in table 9 that the no pre-processing was better in 3 of the 6 models tested. When comparing extractors that used the same training data, the models using W2V as extractors had a better performance than GloVe for this dataset in 11 of 12 comparisons.

Overall, the usage of all the pre-processors increased the performance in 50% of the use cases. The pre-processing that has the best performance most of the time is "data cleaning" in 8 of the 24 tests and the second-best is "no preprocessing" in 7 of 24 tests for these datasets. The performance difference between the classical approach and deep learning models can be as big as 15% (for the "Rationality" dataset) in favor of deep learning models.

For classic models, when trained on the virtual machine, the tests that took the longest run were those that used the combination of feature extractor with language knowledge and SVM, lasting, on average, 2 hours, and using less than one Gigabyte of RAM. Tests with random forests took an average of 20 minutes and used less than one Gigabyte of RAM. The time to train for deep learning approaches in the same machine took more than one day to complete (28 hours).

The chosen model to be used in an annotation tool must be able to train its model daily for all the datasets. With this in mind, we chose to continue the simulation experiments using only the classical approach, once using a deep learning model would not be possible in this kind of tool. Also, we chose to use W2V 3 and GloVe 3 because this model is pre-trained, therefore saving us the time for training the feature extractor.

3.2 Simulation of techniques to improve learning

In tables 10 and 11 we present the performance results in stages of the simulation of IML, with 10, 50, 100, 200, 400, 800, and all the examples of each class of the dataset. The values shown in the tables are the average values of 10 runs. As the

Table 10. F1 score as the "positioning" database with "data cleaning" size increased. In parentheses, the difference of the rate for the same model of the previous line.

# Examples	W2V + SVM	GloVe + SVM	W2V + RF	GloVe + RF
30	33.67%	36.27%	37.31%	37.03%
150	39.72% (6.05%)	40.47% (4.20%)	42.04% (4.73%)	41.45% (4.42%)
300	42.88% (3.15%)	44.38% (3.91%)	43.49% (1.45%)	43.62% (2.17%)
600	44.47% (1.59%)	46.25% (1.87%)	46.05% (2.57%)	45.06% (1.44%)
1200	45.79% (1.32%)	48.48% (2.22%)	46.76% (0.71%)	45.47% (0.41%)
2400	51.19% (5.41%)	50.55% (2.07%)	47.58% (0.82%)	47.12% (1.65%)
3640	53.60% (2.41%)	50.99% (0.44%)	47.03% (-0.55%)	47.45% (0.33%)

Table 11. F1 score as the "reduced theme" dataset examples added with data cleaning. In parentheses, the difference of the rate for the same model of the previous line.

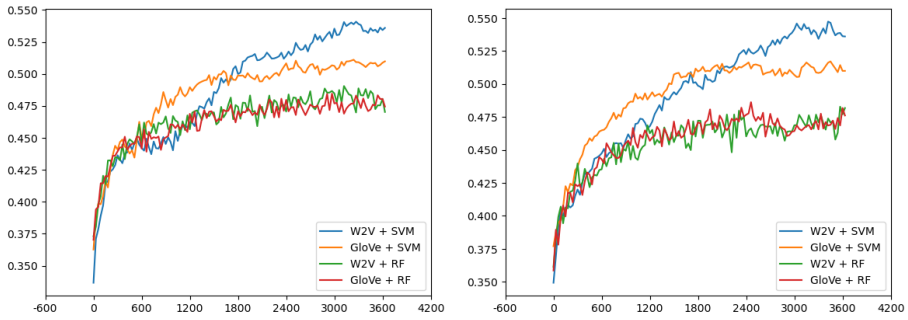
# Examples	W2V + SVM	GloVe + SVM	W2V + RF	GloVe + RF
20	61.13%	64.88%	66.01%	72.12%
100	68.16% (7.03%)	67.92% (3.04%)	75.63% (9.62%)	76.90% (4.79%)
200	69.40% (1.24%)	70.39% (2.47%)	77.43% (1.80%)	78.16% (1.26%)
400	71.15% (1.75%)	73.06% (2.67%)	78.23% (0.80%)	79.10% (0.94%)
800	72.22% (1.07%)	77.93% (4.87%)	79.68% (1.44%)	78.88% (-0.22%)
1600	76.39% (4.16%)	80.04% (2.11%)	80.18% (0.50%)	80.71% (1.83%)
3900	81.06% (4.68%)	81.94% (1.90%)	80.73% (0.55%)	81.25% (0.53%)

number of classes in each database is different, the total amount of examples in each one also differs, but the amount of examples per class is the same (except for the last step, which uses all the examples in the training dataset). In parentheses, we have the step difference in the f1 score rate from the current line to the previous one. We have, in the second line of each table, that the impact of adding 40 examples of each class at the beginning of training (totaling 50 for each class) is greater than adding 200 examples from each class to a dataset that has 200 examples (totaling 400 examples for each class). The average increase in the F1 score by adding the forty examples of each class in a dataset that has ten examples is 4.85% for the "positioning with data cleaning" dataset and 6.12% for the "reduced theme with data cleaning" dataset. The average increase of the F1 score by adding 200 examples of each class to a dataset that already has 200 examples, is 1.86% for the "positioning with data cleaning" dataset and 1.54% for the "reduced theme with data cleaning" dataset.

The impact of adding new examples decreases with dataset size. When adding 100 examples to a dataset that previously had 10. the dataset size is increased by approximately ten times. When adding 100 examples to a dataset that has 1000. the size of the base is increased by 0.1 times. Besides, the larger the dataset, the greater the chance that a new example is similar to an existing one and not contribute to increased performance.

The standard variance starts at about 5%, as is expected for small datasets. However, when adding examples, the standard variance decreases. With 300

examples of each class, it is around 1%. When adding more examples, it drops to less than 1% for all models.



(a) Added randomly (b) Added by uncertainty

Fig. 1. F1 score of classifiers with feature extractors as adding new examples in the dataset "positioning with clean".

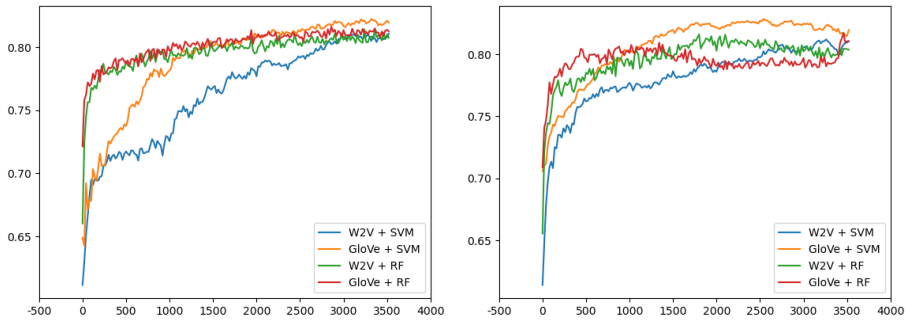
In figures 1 and 2 we have the results of the AML simulation. These graphs are the average performance evolution of each tested model. The standard variant behaves in the same way as the simulation for iterative learning. The X-axis is the total number of examples in each run. The Y-axis is the F1 score rate of each run in the range from 0 to 1.

In figure 1 we can see that active machine learning did not change how the learning curve behaves, bringing little or no performance gain at the beginning of execution when compared to not choose the next examples. Although the impact was small, there was a small change in the learning curve of the models that use the SVM classifier.

In figure 2 we can see that active machine learning changed the learning curve, increasing the learning speed of models that use the SVM classifier. The model combining GloVe with SVM has fast learned at the beginning of the dataset (it used fewer examples) and it reached the maximum performance faster.

4 Discussion

Machine learning techniques using deep learning have better performance in our evaluation, in some cases, it has over 15% of advantage over classic alternatives. But the computational cost makes it impractical for the tool that is being built. It took over 28 hours to complete fine-tunning. Our testing determined that using machine learning techniques without the usage of deep learning was a viable solution. We choose to use the combination of the Random Forest classifier and the Word2Vector feature extractor, because of its lower computational cost and better performance in the tested database.



(a) Added randomly (b) Added by uncertainty

Fig. 2. F1 score of classifiers with feature extractors as adding new examples in the dataset "reduced theme with data cleaning"

Our simulations of iterative machine learning confirmed that most of the performance of a machine learning system comes from the initial examples. As we increase the training dataset, we have small performance breakthroughs. The active learning simulations showed that the Support Vector Machine had better use of the reordering of the examples than the Random Forest in the tested datasets. But it can improve both, speeding up the initial learning and anticipating breakthroughs.

In the annotation tool, we can use active machine learning to propose an order of examples to be annotated. By doing it, we can reduce the amount of annotated examples needed for a machine learning algorithm that can be used in an iterative machine learning scenario. So, we can use iterative machine learning to suggest possible annotations, for example, improving the quality and speed of the annotation process.

References

1. Bontcheva, K., Cunningham, H., Roberts, I., Tablan, V.: Web-based collaborative corpus annotation: Requirements and a framework implementation. New Challenges for NLP Frameworks pp. 20–27 (2010)
2. Boser, B.E., Guyon, I.M., Vapnik, V.N.: A training algorithm for optimal margin classifiers. In: Proceedings of the Fifth Annual Workshop on Computational Learning Theory. pp. 144–152. COLT '92, ACM, New York, NY, USA (1992). https://doi.org/10.1145/130385.130401, http://doi.acm.org/10.1145/130385.130401
3. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019), https://www.aclweb.org/anthology/N19-1423

4. Dudley, J.J., Kristensson, P.O.: A review of user interface design for interactive machine learning. ACM Transactions on Interactive Intelligent Systems (TiiS) **8**(2), 1–37 (2018)
5. El-Assady, M., Sevastjanova, R., Gipp, B., Keim, D., Collins, C.: Nerex: Named-entity relationship exploration in multi-party conversations. Computer Graphics Forum **36**(3), 213–225 (2017). https://doi.org/10.1111/cgf.13181, https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13181
6. Hartmann, N.S., Fonseca, E.R., Shulby, C.D., Treviso, M.V., Rodrigues, J.S., Aluısio, S.M.: Portuguese word embeddings: Evaluating on word analogies and natural language tasks. In: XI Brazilian Symposium in Information and Human Language Technology and Collocated Events. pp. 122–131. SBC, Sociedade Brasileira de Computação, Uberlândia, Brazil (Oct 2017), https://www.aclweb.org/anthology/W17-6615
7. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer Series in Statistics, Springer New York Inc., New York, NY, USA (2001)
8. Kim, B., Glassman, E., Johnson, B., Shah, J.: ibcm: Interactive bayesian case model empowering humans via intuitive interaction. Tech. rep., MIT-CSAIL, Cambridge, MA 02142-1209 (2015)
9. Kranjc, J., Smailović, J., Podpečan, V., Grčar, M., Žnidaršič, M., Lavrač, N.: Active learning for sentiment analysis on data streams: Methodology and workflow implementation in the clowdflows platform. Information Processing & Management **51**(2), 187–203 (2015)
10. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)
11. Mishra, S., Diesner, J., Byrne, J., Surbeck, E.: Sentiment analysis with incremental human-in-the-loop learning and lexical resource customization. In: Proceedings of the 26th ACM Conference on Hypertext & Social Media. pp. 323–325 (2015)
12. Pennington, J., Socher, R., Manning, C.: GloVe: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1532–1543. Association for Computational Linguistics, Doha, Qatar (Oct 2014). https://doi.org/10.3115/v1/D14-1162, https://www.aclweb.org/anthology/D14-1162
13. Settles, B.: Active learning literature survey. Tech. rep., University of Wisconsin-Madison Department of Computer Sciences (2009)
14. Souza, F., Nogueira, R., Lotufo, R.: Bertimbau: Pretrained bert models for brazilian portuguese. In: Brazilian Conference on Intelligent Systems. pp. 403–417. Springer (2020)
15. Vitório, D., Souza, E., Oliveira, A.L.I.: Evaluating active learning sampling strategies for opinion mining in brazilian politics corpora. In: Moura Oliveira, P., Novais, P., Reis, L.P. (eds.) Progress in Artificial Intelligence. pp. 695–707. Springer International Publishing, Cham (2019)
16. Yimam, S.M., Biemann, C., Majnaric, L., Šabanović, Š., Holzinger, A.: Interactive and iterative annotation for biomedical entity recognition. In: Guo, Y., Friston, K., Aldo, F., Hill, S., Peng, H. (eds.) Brain Informatics and Health. pp. 347–357. Springer International Publishing, Cham (2015)
17. Zimmermann, M., Ntoutsi, E., Spiliopoulou, M.: Incremental active opinion learning over a stream of opinionated documents. WISDOM 2015 (KDD'15) (10 2015)