APLenty: annotation tool for creating high-quality datasets using active and proactive learning

Minh-Quoc Nghiem, Sophia Ananiadou
National Centre for Text Mining
School of Computer Science, The University of Manchester, United Kingdom
{minh-quoc.nghiem, sophia.ananiadou}@manchester.ac.uk

Abstract

In this paper, we present APLenty, an annotation tool for creating high-quality sequence labeling datasets using active and proactive learning. A major innovation of our tool is the integration of automatic annotation with active learning and proactive learning. This makes the task of creating labeled datasets easier, less time-consuming and requiring less human effort. APLenty is highly flexible and can be adapted to various other tasks.

1 Introduction

Labeled data play a critical role in many machine learning problems. Obtaining such data is difficult, time-consuming, and require a lot of human effort. Many researchers have utilized *active learning* or *proactive learning*, in which a learning algorithm is allowed to choose the data from which it learns (Settles, 2009). The annotators, in this scenario, only have to annotate a reasonable-size set of representative and informative data. It helps reduce the human labeling effort and at the same time reduces the cost of training the machine learning models.

In recent years, there has been an increasing amount of libraries and systems that focus on active learning, such as the JCLAL (Reyes et al., 2016), the Active-Learning-Scala (Santos and Carvalho, 2014), or the LibAct (Yang et al., 2017) libraries. They implement several state-of-the-art active learning strategies in single-label and multi-label learning paradigms. These libraries, however, have not treated sequence labeling tasks (part-of-speech tagging, information extraction, ...) in much detail. Due to the nature of sequence labeling tasks, the learning algorithm usually gets not a single label but a sequence of labels from the annotators. Besides, to the best of our knowledge, no system offers support for proactive learning.

Up to now, far too little attention has been paid to the interaction between the annotators and the active learning algorithm. The main point of active learning is that a learning algorithm must be able to interactively query the annotators to obtain the desired outputs at new data points. The available systems fail to deliver this by providing over-simplified user-interfaces for the annotators (i.e., showing the feature vector of an instance and asking for the label). Such user-interfaces are not suitable for the task since the annotators need to know the context of every instance to provide accurate annotations. Some tools provide excellent visualization front end, such as brat (Stenetorp et al., 2012), PubAnnotation (Kim and Wang, 2012) or WebAnno (Yimam et al., 2013), but unfortunately these tools provide no support for active learning.

To compensate for the lack of learning environment in the well-known annotation tool, we develop **APLenty** (**A**ctive **P**roactive **Le**ar**n**ing Sys**t**em), a web-based system for creating annotated data using active/proactive learning. The main innovation of APLenty is the combination of a well-known annotation tool (brat) with active/proactive learning. Specifically:

1. Proactive learning integration: APLenty makes annotation easy, time-efficient, and require less human effort by offering automatic and proactive learning.

2. An easy-to-use interface for annotators: APLenty adapts the interface of the brat rapid annotation tool, making annotation intuitive and easy to use.

3. Suitable for sequence labeling: APLenty is best used for sequence labeling tasks, although it can be used for other classification problems.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (System Demonstrations), pages 108–113
Brussels, Belgium, October 31–November 4, 2018. ©2018 Association for Computational Linguistics

The remainder of this paper is organized as follows. Section 2 provides a brief overview of the related work. Section 3 presents details of APLenty. Section 4 describes a case study of using APLenty for named-entity recognition task. Section 5 concludes the paper and points to avenues for future work.

2 Related work

There are many tools available for active learning, such as the JCLAL (Reyes et al., 2016), the Active-Learning-Scala (Santos and Carvalho, 2014), or the LibAct (Yang et al., 2017) libraries. Among those, JCLAL includes the most state-of-the-art strategies for active learning. Other tools such as Vowpal Wabbit[1] or TexNLP (Baldridge and Palmer, 2009) also include some active learning methods. These tools, however, do not focus on the interaction with the annotators (the user-interface).

BRAT (Stenetorp et al., 2012) is one of the most well-known annotation tools that offer easy-to-use user-interfaces. BRAT has been developed for rich structured annotation and uses a vector graphics-based visualization component for rendering. BRAT can, at the same time, display many annotation layers. WebAnno (Yimam et al., 2013) improves the annotation interface of BRAT by letting the annotators choose the annotation layer(s) for rendering. WebAnno offers a purely web-based generic annotation tool and supports distributed annotation. PubAnnotation (Kim and Wang, 2012) also offers a web-based annotation interface but its main focus is to improve the reusability of corpora and annotations. These tools do not support active/proactive learning.

DUALIST (Settles, 2011; Settles and Zhu, 2012) and Prodigy[2] are most closely related to APLenty. DUALIST is an active learning annotation paradigm that offers annotation interface for semi-supervised active learning. Prodigy is a commercial product which provides an annotation tool powered by active learning. Unfortunately, both DUALIST and Prodigy do not support proactive learning.

3 APLenty

APLenty is a web-based tool implemented in Java using Apache Wicket web framework and JavaScript. The overall architecture of APLenty is depicted in Figure 1. The user interface consists of a project management page and an annotation page. Below, this section describes APLenty in detail.

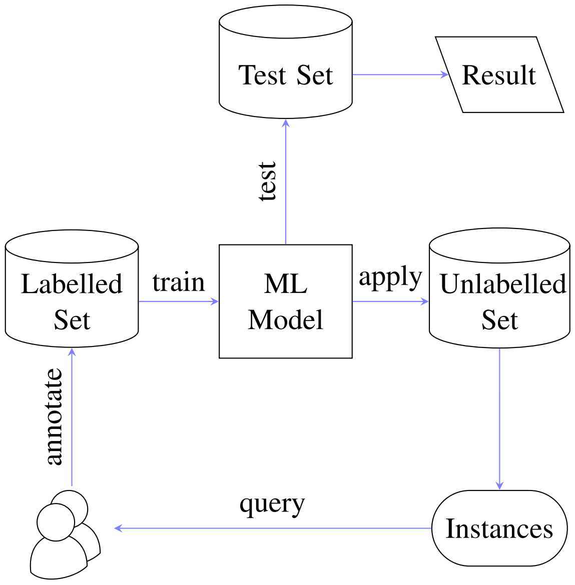


Figure 1: System architecture

3.1 Project management

In APLenty system, we have two main types of user: annotator and project manager. The annotators can only annotate text assigned to them while a project manager can create and customize annotation projects. The interface lets the project manager to:

1. create a project
2. upload the training, test, and unlabelled data to the web server
3. define the tagset
4. assign annotators to a project
5. choose the active/proactive learning strategy.

The system predefines some common tags, but the manager can override these by uploading a tagset file.

There are three types of data that the project manager can upload. The first one is the training data, which will be used to train the machine learning model. The second one is the testing data, which will be used to test the performance of the system after each annotation step. The last one is

[1] http://hunch.net/~vw/
[2] https://prodi.gy/

the unlabelled data, on which the annotators will work. Training and testing data is not required, but unlabelled data is mandatory. When there is no training data, the active learning algorithm will choose the longest sentences for annotation.

APLenty currently supports data using CoNLL-2003 format and XML Metadata Interchange (XMI) format. Since APLenty is based on the Apache Unstructured Information Management Architecture (UIMA) framework[3], new formats can be supported easily. UIMA enables the application to be decomposed into components. Out of which, a collection reader component is responsible for reading documents in a specific format. By swapping the collection reader component, one can allow APLenty to support data in different format.

The test set is optionally used to evaluate the annotation process. By providing the test set, the project manager can see the learning curve of the active learning method. This evaluation step is skipped if there is no test set.

3.2 Annotation interface

For annotation and visualization of annotated documents, we adapted the WebAnno annotation interface, which in turn, adapted the brat rapid annotation tool. Since the initial purpose of APLenty is sequence labeling, the smallest unit we consider is a sentence. The annotation interface only displays one sentence to the annotator at a time. It helps the annotator to focus on the current sentence. Figure 2 shows the annotation interface.

When working on APLenty, the annotator selects a span of text on the displayed sentence and chooses a tag for that span. The annotator does not need to save the annotation since every annotation is automatically sent to the web server (via Ajax using JSON format). The annotator has the possibility to skip annotating a sentence. By choosing skip, the algorithm marks the sentence as “skipped” and does not consider that sentence in the next annotation round for this specific annotator. When the annotator completed an active learning iteration step, APLenty will trigger the training process with newly annotated data and update the sentences for the next annotation batch.

The annotation ends when stopping criteria are met. A project manager can have several ways to define the stopping criteria: the algorithm reaches a predefined number of iteration, or the dataset reaches a predefined number of instances, or the result on the test set reaches a predefined amount. The learning process has been completed when the stopping criteria are met. The annotators, however, can stop anytime they want and resume the process later.

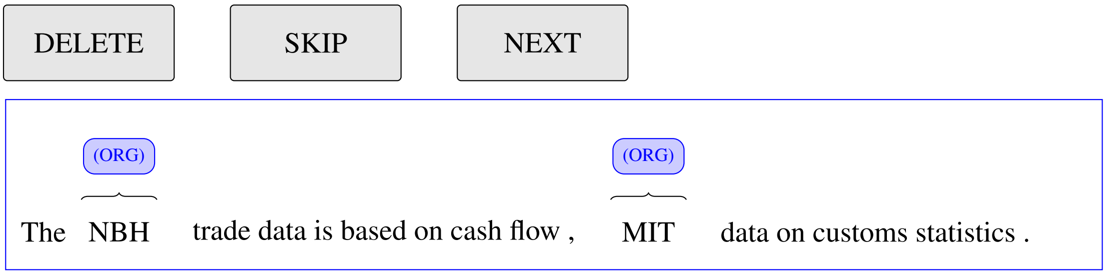
The project manager can increase the annotation speed by turning on automation. APLenty, in this case, automatically annotates certain spans of text (based on the model available from the previous active learning iteration round). The annotators are then required to label only uncertain sequences. This approach was proved to reduce the number of tokens to be manually annotated about 60% compared to its fully supervised counterpart, and over 80% compared to a totally passive learning (Tomanek and Hahn, 2009). The project manager can set a threshold θ for automatic annotation. If the probability of an output sequence from the machine learning model is larger than θ, the output sequence is accepted as valid annotation and is used as a training instance for later active learning iterations.

3.3 Active learning

Depending on the project manager’s settings, the system will choose a query strategy for active learning. Generally, a machine learning algorithm uses the instances in the labeled set (training set) to train a model. The system then uses the model to label the instances in the unlabeled set. Every instance now has a score indicating how informative or representative it is. Finally, the system aggregates these scores to get the score of the whole sentence. The system sends the most informative sentences to the annotators, based on the sentences’ scores. When the system receives the annotations, a new iteration round starts.

Active learning for sequence labeling can use different query strategies. Most common query strategies are Least Confidence (Culotta and McCallum, 2005), Margin Sampling (Scheffer et al., 2001), Entropy Sampling (Mann and McCallum, 2007), Vote Entropy (Dagan and Engelson, 1995), Kullback Leibler Divergence (Settles and Craven, 2008), Expected Gradient Length (Settles et al., 2008), Information Density (Settles and Craven, 2008) strategies. Among which, no query strategy is completely outperformed other strategies (Settles and Craven, 2008). APLenty cur-

[3] https://uima.apache.org/



The NBH trade data is based on cash flow , MIT data on customs statistics .

Figure 2: Annotation interface

rently employs the least confidence uncertainty-based strategy for sequence models based on the sequence outputs from a Conditional Random Fields model (Okazaki, 2007).

3.4 Proactive learning

Active learning assumes that annotators have similar level of expertise and nobody is fallible. But in reality, different annotators have different levels of expertize in a specific domain. Proactive learning has been proposed to model many types of annotators (Donmez and Carbonell, 2010; Li et al., 2017). Proactive learning assumes that there is at least one expert and one inexperienced annotator (fallible). The expert always provides correct annotations while the fallible annotator might provide incorrect annotations in certain cases.

At each iteration step, the proactive learning algorithm selects the sentences for the annotators based on the probabilities that the annotator will provide correct labels for a sequence in a sentence. A recent study by Li et al. (2017) estimated the performance of each annotator based on a benchmark dataset. The system calculates the probability that an annotator provides a correct label when annotating a sentence by combining the class probability and the likelihood that the annotator provides a correct label for the tokens in the sentence.

In the real-time annotation scenario where speed is one of the most important factors, APLenty uses a simple threshold ϱ to distribute sentences to annotators. If the probability of an output sequence from the machine learning model is smaller than ϱ, APLenty considers the sentence a hard case and sends it to the expert annotator. Otherwise, the sentence is sent to the fallible annotator. This reduces the cost of annotation since the time of the expert is more expensive than the time of the fallible annotator.

4 Case study

One use case that describes the best use of APLenty is the named entities annotation. This is a multiple span annotation task.

We used the English CoNLL-2003 named entity dataset (Tjong Kim Sang and De Meulder, 2003) for the case study. The dataset contains newswire articles annotated with four entities: LOC (locations), MISC (miscellaneous names), ORG (organizations), and PER (persons). In the experiment, we used 1,200 sentences as the initial training data, 3,622 sentences as test data, and the rest for unlabelled data. θ is set to 0.8, ϱ is set to 0.2, batch size is set to 100.

We compare four settings in this case study. The first one is Random Sampling: the system randomly chooses the next sentences for annotation. The second one is Active Learning: the system uses the output of CRF model to assign sentences to an expert. The third one is Proactive Learning: same as Active Learning, but we have two annotators, one expert, and one fallible annotator. The last one is Active Learning with Automation: the system automatically assigns labels for sequences based on the threshold θ.

Figure 3 shows the learning curves of the four settings. In all cases, active/proactive learning setting outperformed Random Sampling setting. It can be seen that the last three settings achieved peak performance when we reached 50th iteration. Combining active learning and automation lead to best performance. This result may be explained by the fact that the system got more reliable data for training after every iteration.

5 Conclusion and future work

We introduced APLenty, a web-based open environment for constructing annotated datasets using active and proactive learning while leveraging the functionalities of the popular annotation edi-

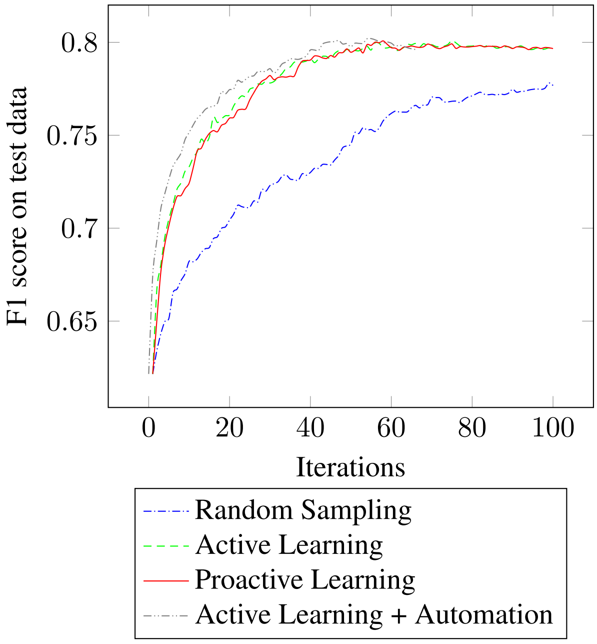


Figure 3: Learning curve

tor brat. APLenty can support the quick development of high-quality labeled data needed to train and evaluate text mining tools for different applications. Existing annotation editors do not provide such an integrated environment which can quickly produce labeled data while at the same time taking into account different levels of expertise of annotators.

A key feature of APLenty is how it supports the automation for sequences with high confidence to be included (certain sequences), thus allowing the annotators to focus only on the uncertain ones. We have demonstrated that this feature enables annotators to create high-quality labeled datasets in less time than other settings.

Considerably more work will need to be done to: 1. extend our work for link annotation; 2. further enhance APLenty to work with other active/proactive learning criteria; 3. evaluate APLenty in a complete data creation; 4. enhance centralize repository of annotation such as PubAnnotation.

Acknowledgments

This research has been carried out with funding from BBSRC BB/P025684/1 and BB/M006891/1. We would like to thank the anonymous reviewers for their helpful comments.

References

Jason Baldridge and Alexis Palmer. 2009. How well does active learning actually work?: Time-based evaluation of cost-reduction strategies for language documentation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 296–305. Association for Computational Linguistics.

Aron Culotta and Andrew McCallum. 2005. Reducing labeling effort for structured prediction tasks. In *AAAI*, volume 5, pages 746–751.

Ido Dagan and Sean P Engelson. 1995. Committee-based sampling for training probabilistic classifiers. In *Machine Learning Proceedings 1995*, pages 150–157. Elsevier.

Pinar Donmez and Jaime G Carbonell. 2010. From active to proactive learning methods. In *Advances in Machine Learning I*, pages 97–120. Springer.

Jin-Dong Kim and Yue Wang. 2012. Pubannotation: a persistent and sharable corpus and annotation repository. In *Proceedings of the 2012 Workshop on Biomedical Natural Language Processing*, pages 202–205. Association for Computational Linguistics.

Maolin Li, Nhung Nguyen, and Sophia Ananiadou. 2017. Proactive learning for named entity recognition. *BioNLP 2017*, pages 117–125.

Gideon S Mann and Andrew McCallum. 2007. Efficient computation of entropy gradient for semi-supervised conditional random fields. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 109–112. Association for Computational Linguistics.

Naoaki Okazaki. 2007. CRFsuite: a fast implementation of Conditional Random Fields (CRFs).

Oscar Reyes, Eduardo Pérez, María Del Carmen Rodríguez-Hernández, Habib M Fardoun, and Sebastián Ventura. 2016. JCLAL: a Java framework for active learning. *The Journal of Machine Learning Research*, 17(1):3271–3275.

Davi P Santos and André CPLF Carvalho. 2014. Comparison of active learning strategies and proposal of a multiclass hypothesis space search. In *Hybrid Artificial Intelligence Systems*, pages 618–629. Springer.

Tobias Scheffer, Christian Decomain, and Stefan Wrobel. 2001. Active hidden Markov models for information extraction. In *International Symposium on Intelligent Data Analysis*, pages 309–318. Springer.

Burr Settles. 2009. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison.

Burr Settles. 2011. Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1467–1478. Association for Computational Linguistics.

Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1070–1079. Association for Computational Linguistics.

Burr Settles, Mark Craven, and Soumya Ray. 2008. Multiple-instance active learning. In *Advances in neural information processing systems*, pages 1289–1296.

Burr Settles and Xiaojin Zhu. 2012. Behavioral factors in interactive training of text classifiers. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 563–567. Association for Computational Linguistics.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. BRAT: a web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107. Association for Computational Linguistics.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

Katrin Tomanek and Udo Hahn. 2009. Semi-supervised active learning for sequence labeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1039–1047. Association for Computational Linguistics.

Yao-Yuan Yang, Shao-Chuan Lee, Yu-An Chung, Tung-En Wu, Si-An Chen, and Hsuan-Tien Lin. 2017. libact: Pool-based active learning in Python. Technical report, National Taiwan University. Available as arXiv preprint `https://arxiv.org/abs/1710.00379`.

Seid Muhie Yimam, Iryna Gurevych, Richard Eckart de Castilho, and Chris Biemann. 2013. WebAnno: A flexible, web-based and visually supported system for distributed annotations. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 1–6.