Paladin: an annotation tool based on active and proactive learning

Minh-Quoc Nghiem[1,2], **Paul Baylis**[2], **Sophia Ananiadou**[1]
[1] National Centre for Text Mining
School of Computer Science, The University of Manchester, United Kingdom
[2] Bott and Co Solicitors
{minh-quoc.nghiem, sophia.ananiadou}@manchester.ac.uk
p.baylis@bottonline.co.uk

Abstract

In this paper, we present Paladin, an open-source web-based annotation tool for creating high-quality multi-label document-level datasets. By integrating active learning and proactive learning to the annotation task, Paladin makes the task less time-consuming and requiring less human effort. Although Paladin is designed for multi-label settings, the system is flexible and can be adapted to other tasks in single-label settings.

1 Introduction

Labelled data is essential in many NLP tasks based on Machine Learning. Manually annotating such data is time-consuming, and require a lot of human effort. *Active learning* has been used to ease this process by choosing the data points for annotation instead of annotating all instances of the unlabeled data (Settles, 2009). Some recent research has also utilized *proactive learning*, in which the system is allowed to assign specific unlabeled instances to specific annotators (Li et al., 2019). The annotators, in these scenarios, only have to annotate a small set of representative and informative data which they can provide reliable labels. It helps reduce the labelling effort and at the same time makes the best use of available annotators.

To date, there are many tools available for active learning, such as the TexNLP (Baldridge and Palmer, 2009), the Active-Learning-Scala (Santos and Carvalho, 2014), the JCLAL (Reyes et al., 2016), the LibAct (Yang et al., 2017) libraries, the Vowpal Wabbit[1]. These tools, however, focus only on the active learning algorithms and provide no user interface thus making it difficult to use for the end-users. On the other hand, several tools have been made with user-friendly interface such as BRAT (Stenetorp et al., 2012), WebAnno (Yimam et al., 2013), PubAnnotation (Kim and Wang, 2012), doccano[2]. Some of the tools offer active/proactive learning such as APLenty (Nghiem and Ananiadou, 2018), DUALIST (Settles and Zhu, 2012), AlpacaTag (Lin et al., 2019), Discrete Active Learning Coref (Li et al., 2020a). Currently, these tools support sequence labelling/coreference resolution tasks but not document classification tasks. To the best of our knowledge, there is no such tool for document classification which supports active/proactive learning. Prodigy[3] supports active learning for both sequence labelling and document classification tasks but it is a commercial product.

To compensate for the lack of available document-level annotation tool, we develop **Paladin** (**P**ro**a**ctive **l**earning **a**nnotator for **d**ocument **in**stances), an open-source web-based system for creating labelled data using active/proactive learning[4]. The main innovation of Paladin is the combination of a user-friendly annotation tool with active/proactive learning. Specifically:

1. Active/proactive learning integration: Paladin makes annotation easy, time-efficient, and require less human effort by offering active and proactive learning.

2. An easy-to-use interface for annotators: Paladin adapts the interface of doccano, making annotation intuitive and easy to use.

3. Suitable for multi-label document annotation tasks: Paladin is best used for multi-label document annotation tasks, although it can be used for other single-label classification problems.

[1] http://hunch.net/~vw/

[2] https://github.com/doccano

[3] https://prodi.gy/

[4] The source code is publicly available at https://github.com/bluenqm/Paladin

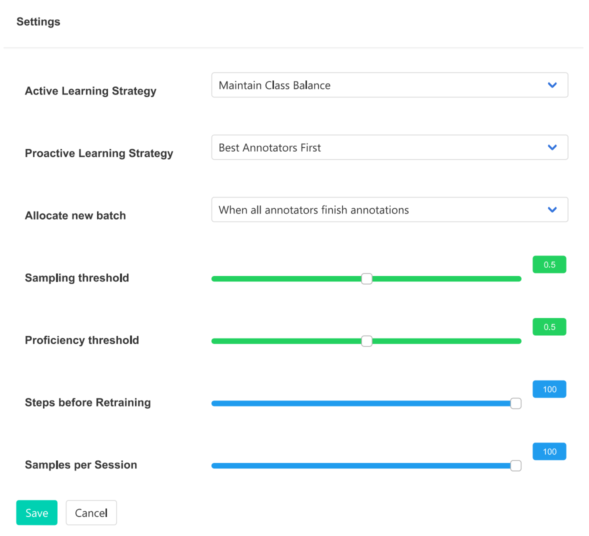
The remainder of this paper is organized as follows. Section 2 presents details of Paladin. Section 3 describes a case study of using Paladin for a multi-label document annotation task. Section 4 concludes the paper and points to avenues for future work.

2 System Descriptions

Paladin is a web-based tool implemented in Python using Django web framework and Vue.js. The main user interface consists of a project management page and an annotation page. Below, this section describes Paladin in detail.

2.1 Project management

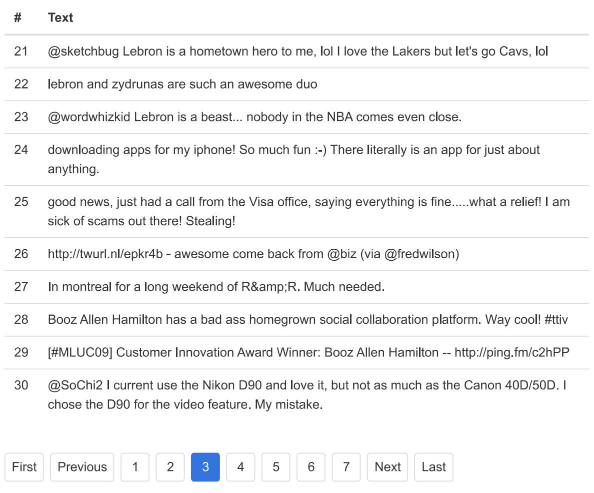
In Paladin, there are two main types of user role: the project manager role and the annotator role. A project manager can create/customise annotation projects and add annotators to the projects. The annotators can annotate text assigned to them. The interface allows the project manager to: 1. create a project 2. define the tagset 3. upload the seeding and unlabelled data to the webserver 4. assign annotators to a project 5. choose the active/proactive learning strategy. The project manager can additionally set how the batch is allocated, the sampling and proficiency thresholds, the steps before retraining and samples per session as illustrated in Figure 1.


Settings

Active Learning Strategy	Maintain Class Balance
Proactive Learning Strategy	Best Annotators First
Allocate new batch	When all annotators finish annotations
Sampling threshold	0.5
Proficiency threshold	0.5
Steps before Retraining	100
Samples per Session	100

Figure 1: Project Settings

When creating a new annotation project, the project manager needs to upload two datasets (in Tab Separated Values format) to the server. The first dataset is the seeding dataset, which will be used by the system to train the classifier and estimate the annotators' proficiency. The second dataset is the unlabelled dataset, on which the system chooses the text to assign to the annotators. If there is no seeding data, the system will select random text from the unlabelled dataset for annotation in the first batch. Figure 2 shows the text when successfully uploaded to the system.



#	Text
21	@sketchbug Lebron is a hometown hero to me, lol I love the Lakers but let's go Cavs, lol
22	lebron and zydrunas are such an awesome duo
23	@wordwhizkid Lebron is a beast... nobody in the NBA comes even close.
24	downloading apps for my iphone! So much fun :-) There literally is an app for just about anything.
25	good news, just had a call from the Visa office, saying everything is fine.....what a relief! I am sick of scams out there! Stealing!
26	http://twurl.nl/epkr4b - awesome come back from @biz (via @fredwilson)
27	In montreal for a long weekend of R&amp;R. Much needed.
28	Booz Allen Hamilton has a bad ass homegrown social collaboration platform. Way cool! #ttiv
29	[#MLUC09] Customer Innovation Award Winner: Booz Allen Hamilton -- http://ping.fm/c2hPP
30	@SoChi2 I current use the Nikon D90 and love it, but not as much as the Canon 40D/50D. I chose the D90 for the video feature. My mistake.

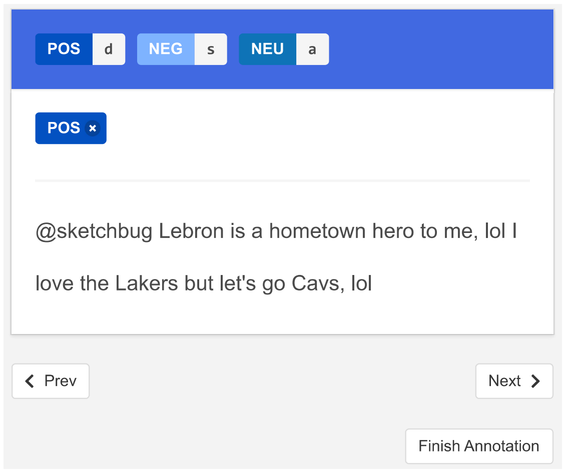
Figure 2: Dataset/Seed Dataset

2.2 Annotation interface

For annotation and visualization of annotated documents, we adapted the doccano annotation interface. The annotation interface displays a set of documents that are assigned to the annotator, one at a time as illustrated in Figure 3. The annotator can navigate to next or previous documents during annotation using the "Prev" or "Next" buttons. When working on Paladin, the annotator uses the mouse or keyboard shortcut to select label(s) for the current document. When finishing the assigned documents, the annotator can click on "Finish Annotation". The system will validate the annotated documents, retrain the classifier, and assign new documents to the annotator. Each annotator can only see the documents assigned to him/her in the current batch.

2.3 Active learning

Depending on the project manager's settings, the system chooses different document instances to send to the annotators. The project manager can choose to prioritise the most informative instances for the classifier or to maintain the balance between the number of instances in each class. With the first option, the system prioritises the most



@sketchbug Lebron is a hometown hero to me, lol I love the Lakers but let's go Cavs, lol

Figure 3: Annotation interface. The displayed sentence was taken from the Sentiment140 dataset. All labels are shown in the blue rectangle box with the shortcut keys next to them. Annotated labels are shown above the sentence.

informative documents, regardless of the class. Paladin currently employs the least confidence uncertainty-based strategy (Culotta and McCallum, 2005) based on the classification outputs from a Transformer model (Devlin et al., 2019). A linear model is added to the embedding output to predict the score for the labels. Previous research has established that active learning can increase the performance of Transformer-based text classifiers (Grießhaber et al., 2020). With the second option, the system uses the same classification outputs but unlabelled instances are taken from each class in equal amounts. The default option in Paladin is the second one. This setting aims to minimise the unbalanced data problems where we have unequal instances for different classes.

Paladin uses pool-based sampling scenario, where the data samples are chosen for labeling from the unlabeled dataset. The project manager, however, can upload additional unlabeled data to an existing annotation project at anytime.

2.4 Proactive learning

In many annotation tasks, we assume that the annotators are experts who always provide correct annotations. But in reality, different annotators have different levels of expertise in different domains. It has been demonstrated that proactive learning is helpful for task allocation in crowdsourcing setting where the level of expertise varies from annotator to annotator (Donmez and Carbonell, 2010; Li et al., 2017, 2019, 2020b). Proactive learning is useful in modelling the annotator reliability which can be used to assign the unlabelled instances to the best possible annotators.

Before any annotation, Paladin estimates the proficiency of the annotators for each class by assigning the documents in the seed dataset to all annotators. When the annotators finish labelling these seed documents, the system calculates the likelihood that a particular annotator provides a correct label for a particular label. Then, when assigning new documents to the annotators, Paladin will assign the documents to the best possible annotators by combining the predicted label(s) and the likelihood that the annotator provides a correct label for a particular label. The system will update the estimation after every annotation batch.

3 Use cases

The typical use cases of Paladin are as following:

1. A user wishes to add more data to an existing dataset to improve model performance: the user can use the existing labelled dataset as the seed to train the initial model, the labels will be automatically extracted from the labelled dataset. The model will select instances from the unlabelled dataset and then distribute them to the annotators for annotation.

2. A user wishes to create a labelled dataset from scratch: the user needs to provide the tag set and the unlabelled data. The first iteration will select unlabelled instances for annotation randomly. After the first iteration, the process is the same as the previous use case.

3. A user wishes to add more data to an existing unbalanced dataset: the user can choose "maintain class balance" option in Settings. With this option, the model will try to select more data from the potential minority classes for annotation.

4 Experiments and Results

4.1 Simulated Annotators

We used the Toxic Comment Classification Challenge dataset[5] for this experiment. The dataset contains Wikipedia comments which have been manually labelled for toxic behaviour. There are

[5] https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data

six classes: toxic, severe toxic, obscene, threat, insult, and identity hate. In the experiment, we used 60 comments as the initial training data (seed), 600 comments as test data, and 18,000 for unlabelled data. The instances forming the seed and test data are randomly taken from the original data but we make sure that each class has at least 10 instances and 100 instances in the seed and test data respectively.

We compare three settings in this case study. The first one is Random Sampling: the system randomly chooses the next documents for annotation. The second one is Active Learning: the system uses the output of the trained model to assign new documents to an expert (annotator who always provide correct labels). The third one is Proactive Learning: same as Active Learning, but we have two annotators, one expert, and one fallible annotator (annotator who makes mistakes with a probability of 0.1). Figure 4 shows the F1 scores on the test set. In all cases, active/proactive learning setting outperformed Random Sampling setting.

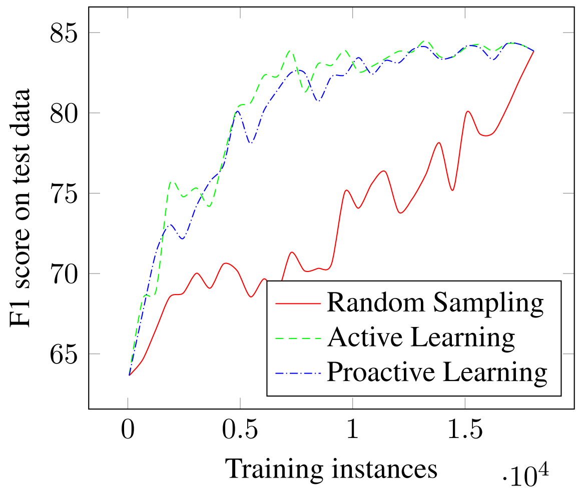


Figure 4: Learning curve

4.2 Real-World Annotators

For this experiment, we worked with a consumer law firm analysing 6,880 emails. Each email can have one or more labels from a predefined list which consist of 15 labels. Some examples are "update query", "payment query", and "fee query". Given an email, the annotator had to annotate all labels that are applicable to that email. There are a total of 2,000 emails which were already annotated.

This dataset is an unbalanced dataset where nearly two-thirds of the emails belong to the 5 most common labels while less than 7 percent of the emails come from the 5 least common labels. In the experiment, we used 1,000 emails as the initial training data, 1,000 emails as test data, and the rest (4,880) as unlabelled data. The purpose of the experiment was to investigate the performance of Paladin with an unbalanced seed dataset.

Using Paladin, we created an annotation project with four annotators and in each annotation session, an annotator must annotate 20 emails. All annotators are members of the law firm with legal background. We used "maintain class balance" and "best annotators first" for active learning strategy and proactive learning strategy respectively. We stopped when a total of 1,000 emails were annotated. Figure 5 shows the F1 scores and the stacking percentages of label instance count. The results showed that the F1 score and percentage of minority classes were gradually increased after each annotation batch.

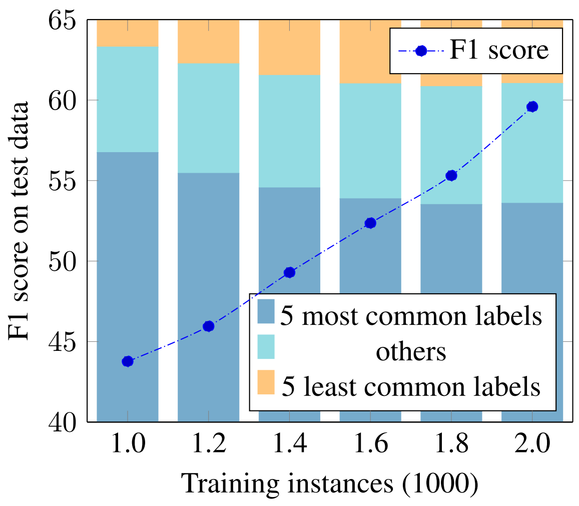


Figure 5: F1 scores and percentages of label instance count. We grouped 5 labels together for readability.

We used an Intel Core i9 9820X Linux server with 64GB RAM and a Titan RTX GPU. When allocating a new annotation batch (retraining the model, predicting the unlabelled instances, selecting new instances for annotation), Paladin runs consistently at the rate of around 0.01 to 0.02 seconds per document and it takes less than two minutes to get results. The average level of satisfaction (with ratings from 1 to 5 of three aspects: responsiveness, easy to annotate, easy to navigate) of the annotators with the annotation tool is 4.5/5.

5 Conclusion

We introduced Paladin, a web-based open environment for constructing multi-label document-level datasets using active and proactive learning. Paladin can support the quick development of high-quality labelled data needed to train and evaluate NLP tools for different applications.

Considerably more work will need to be done to further enhance Paladin to work with other active/proactive learning algorithms. Besides that, a natural progression of this work is to evaluate Paladin in a large scale annotation project.

Acknowledgments

This research has been carried out with funding from KTP11612. 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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Daniel Grießhaber, Johannes Maucher, and Ngoc Thang Vu. 2020. Fine-tuning BERT for low-resource natural language understanding via active learning. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1158–1171, Barcelona, Spain (Online). International Committee on Computational Linguistics.

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.

Belinda Z. Li, Gabriel Stanovsky, and Luke Zettlemoyer. 2020a. Active learning for coreference resolution using discrete annotation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8320–8331, Online. Association for Computational Linguistics.

Maolin Li, Arvid Fahlström Myrman, Tingting Mu, and Sophia Ananiadou. 2019. Modelling instance-level annotator reliability for natural language labelling tasks. 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)*, pages 2873–2883, Minneapolis, Minnesota. Association for Computational Linguistics.

Maolin Li, Nhung Nguyen, and Sophia Ananiadou. 2017. Proactive learning for named entity recognition. In *BioNLP 2017*, pages 117–125, Vancouver, Canada,. Association for Computational Linguistics.

Maolin Li, Hiroya Takamura, and Sophia Ananiadou. 2020b. A neural model for aggregating coreference annotation in crowdsourcing. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5760–5773, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Bill Yuchen Lin, Dong-Ho Lee, Frank F. Xu, Ouyu Lan, and Xiang Ren. 2019. AlpacaTag: An active learning-based crowd annotation framework for sequence tagging. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 58–63, Florence, Italy. Association for Computational Linguistics.

Minh-Quoc Nghiem and Sophia Ananiadou. 2018. APLenty: annotation tool for creating high-quality datasets using active and proactive learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 108–113, Brussels, Belgium. Association for Computational Linguistics.

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.

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

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.

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.