MIC: A Multi-task Interactive Curation Tool

Shi Yu[1]*, Mingfeng Yang[2], Jerrod Parker[3], and Stephen Brock[2]
[1]Broadridge Financial Solutions, shi.yu@broadridge.com
[2]Vanguard Group, {mingfeng_yang, stephen_brock}@vanguard.com
[3]Thomson Reuters, jerrodparker20@gmail.com

Abstract

This paper introduces MIC, a **M**ulti-task **I**nteractive **C**uration tool, a human-machine collaborative curation tool for multiple NLP tasks. The tool aims to borrow recent advances in literature to solve pain-points in real NLP tasks. Firstly, it supports multiple projects with multiple users which enables collaborative annotations. Secondly, MIC allows easy integration of pre-trained models, rules, and dictionaries to auto label the text and speed up the labeling process. Thirdly, MIC supports annotation at different scales (span of characters and words, tokens and lines, or document) and different types (free text, sentence labels, entity labels, and relationship triplets) with easy GUI operations.

1 Introduction

With the recent advances in many frontiers, high-quality annotations are essential to the success of NLP applications. Numerous organizations have accumulated vast amounts of unlabeled text data that they want to utilize in NLP applications. However, for many of these tasks (text summarization, relation extraction, named-entity recognition), aquiring labels can be very costly and susceptible to error. Furthermore, domain adaptation (Han and Eisenstein, 2019), which is the common approach of fine-tuning gigantic domain agnostic NLP models on a small amount of domain-specific labeled data, commonly has difficulty on new emerging / specific domains that lack similar labeled datasets and still requires annotations from scratch. Meanwhile, adoption of accelerated ML solutions have shown to reduce the workload and budget required for manual labeling. Example techniques include active learning, weak supervision, data augmentation, and many others. To concur the time-consuming, labor-intensive, and expensive annotation challenges, recent trends in annotation tool development (Lin et al., 2019; Lee et al., 2020) focus on cost-effective and human-machine collaborative mechanisms, which leverage the processing power of state-of-the-art models pre-trained on large corpora and high-accuracy human intelligence on rare ambiguous incidents.

Please visit *www.textmic.com* using username *ds* and password *demouser123* to visit the demo system. A screencast video is at https://youtu.be/pHxt5k_mLvw. The github repo of MIC is at https://github.com/cyberyu/textmic.

2 Related Work

In the past decade, there were about 30 popular annotation tools published. Among them, there are well-known tools like BRAT (Stenetorp et al., 2012), which supports a wide variety of NLP tasks, including entity recognition, event extraction, and POS (part-of-speech) tagging. GATE Teamware (Bontcheva et al., 2013) is both a desktop application and a web tool that focuses on user management and supports multi-user roles. Yedda (Yang et al., 2018) is a recent tool built on Python that offers auto-labeling via machine learning and provides both command line and web-based interfaces. SANTO (Hartung et al., 2018), which is designed primarily for slot-filling tasks, enables the formation of relational structures from an ontology. It also visualizes the annotations of every user at once to help project owners monitor the quality of annotations. TALEN (Mayhew and Roth, 2018) specialises in the annotation of rare entities. EasyTree (Tratz and Phan, 2018) is specifically designed for the annotation of dependency trees and is integrated with the Amazon Mechanical Turk crowdsourcing platform. AlpacaTag (Lin et al., 2019) and LEANLIFE (Lee et al., 2020) leverage machine learning models, active learning, and weak supervision, respectively, to provide annotation recommendations to reduce annotation costs.

The annotation visualization design of our tool

*All authors contributed equally

System	Annotation Type	Adjud -ication	Intelligent Interactive Annotation	External Dependencies	Programming Language
MIC	Classify, Link	Yes	pre-trained models, crowd -sourcing weak-supervision	MongoDB, PostgreSQL	Python,Django,Vue.js
RedCoat (Stewart et al., 2019)	Classify, Link	Yes	hierarchical entities	MongoDB	Javascript, Python
SANTO (Hartung et al., 2018)	Link	-	ontology-driven	Apache,MySQL	PHP
TALEN (Mayhew and Roth, 2018)	Classify	Yes	entity propagation, internet search	-	Unknown
EasyTree (Tratz and Phan, 2018)	Classify,Link	Yes	crowd-sourcing	Amazon Turk	Java Servlet, Javascript
AlpacaTag (Lin et al., 2019)	Classify	Yes	recommendation, crowd-sourcing	-	Python, Django
LEAN-LIFE (Lee et al., 2020)	Classify, Link	Yes	crowd-sourcing, weak-supervision	-	Python
SLATE (Kummerfeld, 2019)	Classify, Link	Yes	terminal-based annotation	-	Python
BRAT (Stenetorp et al., 2012)	Classify, Link	Yes	-	Apache	Python,Javascript
GATE (Bontcheva et al., 2013)	Classify, Link	Yes	-	-	Java
YEDDA (Yang et al., 2018)	Classify	Yes	-	-	Python
WAT-SL (Kiesel et al., 2017)	Classify	Yes	-	Apache	Java
SAWT (Samih et al., 2016)	Classify	-	-	-	Python, PHP
GraphAnno (Gast et al., 2015)	Classify, Link	-	-	-	Ruby
CorA (Bollmann et al., 2014)	Classify	-	-	-	PHP,Javascript
WebAnno (Yimam et al., 2013)	Classify, Link	-	-	-	Java
Anafora (Chen and Styler, 2013)	Classify, Link	Yes	-	-	Python
ANALEC (Landragin et al., 2012)	Classify, Link	-		-	Java
LabelStudio `labelstud.io`	Classify, Link	Yes	text, images, video, audio	Commercial	React, MST, Python
Prodigy `https://prodi.gy/`	Classify, Link	Yes	active learning in annotation	Commercial	Python
Tagtog `tagtog.com`	Classify, Link	Yes	support annotations in PDF	Commercial	Java, Python
LightTag `lighttag.io`	Classify, Link	Yes	support inter-annotator, project management	Commercial	Python

Table 1: A comparison of annotation tools released recently. MIC supports classification (sentence, NER) and link prediction (relationship); Adjudication: MIC encourages human-machine collaborative annotation; thus, human annotators can correct mistaken machine-generated labels. Relying on role configuration, experienced reviewers can also correct/reject any individual human annotator's labeling results, or even reject the entire annotation results from a specific annotator and ask for re-annotation.

is inspired by RedCoat (Stewart et al., 2019), a web-based annotation tool that supports the stacking and inheritance of hierarchical entities using flexible Javascript visualization. We applied the same visual design style in MIC using Vue.js to display a large number of stacked annotations from different human curators, hand written rules, and models. Besides sharing many common features, the proposed annotation tool has some unique characteristics and strengths compared to all existing tools. We summarize and compare the main characteristics of MIC with other tools, including some commercial products, in Table 1. Recent advances in annotation tool development focus more on intelligent capabilities such as auto-annotation, recommendation, crowd-sourcing, weak-supervision, and many other STOA aspects.

3 System Description and Key Features

Rather than specifying a task such as named entity recognition or sentence labeling, MIC is designed to flexibly support any annotations that can be formulated as one of three annotation types: sentence-level labeling, word/phrase-level labeling, or entity-relation-entity triplet labeling. These can be applied to items that are single documents, lines, phrases, tokens, or token combinations. Furthermore, MIC is able to manage annotation tasks associated with interactions among various annotation types. For example, one can restrict the annotation task on named entities among sentences having positive sentiment score, or limit the findings of relationship triplets that contain the entity *Olympics* with the entity type as *event*.

One major novelty of MIC is its support of human-machine interactive annotation via a flexible user interface. For each annotation task, the human annotator can be empowered by a set of pre-trained ML models to quickly generate machine-annotated labels. These pre-trained models are either built-in models from popular data science packages or novel open-source implementations from Github. Pre-trained models can be configured by administrators via the MIC backend interface and each annotation project can associate with multiple models. Pre-trained models are grouped in three categories: sentence labeling, NER labeling and relationship labeling. All pre-trained models are hosted as RESTFUL API endpoints, and for each annotation category the input/output parameters of all endpoints are required to follow the same standards so various models can be interchanged easily. Though machine generated labels cannot be directly considered as ground-truth annotations, the advantages here are two fold. First, instead of requiring the annotator to write everything from scratch, they start from the most likely machine-generated outputs, and MIC supports quick and intuitive editing operations. This helps the human annotator to spend effort effectively and focus on correcting difficult examples. Second, if the human annotator load and apply multiple machine-generated outputs on the same text, those machine outputs can be exported as noisy labels to train a

consensus model using weak supervision.

Finally, MIC has been designed to manage annotation tasks for multiple projects and multiple users. Multi-project setting allows MIC to be configured flexibly to support diverse annotation tasks. For each project, new textual data can be loaded to seek curations at sentence level, named entity level or relationship level, or any combinations of them. From MIC's backend user interface, one can associate a number of relevant ML models/dictionaries/rules to a project to allow quick generation of machine labels. The textual data can be fully unlabeled, partially labeled, or integrated with ground truth labels. In cases where the data is partially or fully labeled with ground truths, the administrator can setup a built-in validation process to monitor performance of annotations as the task continues. Annotation performance can be evaluated by comparing ground truth labels with human/machine generated labels, or comparing ground truth labels against consensus labels learned by weak supervision. The multi-user setting allows MIC to involve multiple parties in the annotation pipeline. Each project can allow users with different roles such as Administrator, Curator, Data Scientist, Reviewer, etc., and their operational accesses are categorized and limited by roles to ensure the integrity of the annotation task.

In conclusion, the main novelties of MIC are (1) Extendable framework to integrate customized annotation models; (2) Multi-project and multi-user management; (3) Support of multi-layer annotations from sentences to entities and relations; (4) State-of-art user interface design for annotations.

4 MIC Annotation

4.1 General Architecture

MIC is a web-based annotation system that was developed using Django, Quasar and Vue.js frameworks. As its conceptual framework shows (Figure 1), the frontend of MIC relies on Quasar and Vue.js to provide a flexible and interactive user interface. The main web application is developed in Django, therefore we have a python native environment and an integrated backend administrative panel. One of MIC's most notable features is its web-based project management interface which allows users to set up an annotation project, invite annotators/reviewers, define task scope, setup machine models, and quickly manage exports/imports of annotations and text. These management features were achieved efficiently through Django's admin panel. MIC uses PostgreSQL to store the textual data, manage project/user data, record annotation progress, and store all annotations.

Besides a series of Django REST Web APIs that establish the backbone of the tool, MIC can be extended to include a wide range of interactive curation APIs for specific annotation tasks. This means MIC plays the role of a web annotation server while other endpoints can be distributed on multiple machines as machine labeling servers to optimize the computational balance and latency of annotations.

4.2 Annotation Interface

The main annotation layout is composed of three connected areas: automatic labeling zone (left panel), annotation zone (central panel), and summarization zone (right panel), as shown in Figure 2.

MIC provides three types of automatic labeling tools to speed up annotation: machine models, dictionaries and rules. At the same time, MIC also supports annotation at three different levels: sentences, entities and relationships. Users can freely choose the most appropriate auto-tool to annotate text at the most relevant level. For each annotator, MIC allows arbitrary stacking of human/machine labels on the text. To successfully save the results of an annotator's work, MIC will check whether there are any contradictory labels assigned to a unique token sequence. For example, the entity labels of *Nikolaus van der Pas* can be saved as (*Nikolaus: Person*), (*van der Pas: Person*), (*Nikolaus: Entity*) and (*Pas: Entity*). MIC allows saving all four different annotations though they overlap on each other. However, MIC will ask for resolution if the unique token sequence (sentence position dependent) *Nikolaus van der Pas* has two contradictory labels. The reason of allowing flexible annotations as such is to minimize the burden of human resolutions. As a matter of fact, lots of similar conflicts can be considered as noise, and can be resolved successfully by well-designed machine learning models.

With MIC, annotators can create three levels of labels simultaneously, which means users can switch back and forth among the three levels and complete the labels for each page. In another way, annotators can focus on sentence-level annotation of all pages first and save the results, and the revisit and finish annotations for the other two layers later. Every annotation task can be scoped as arbitrary

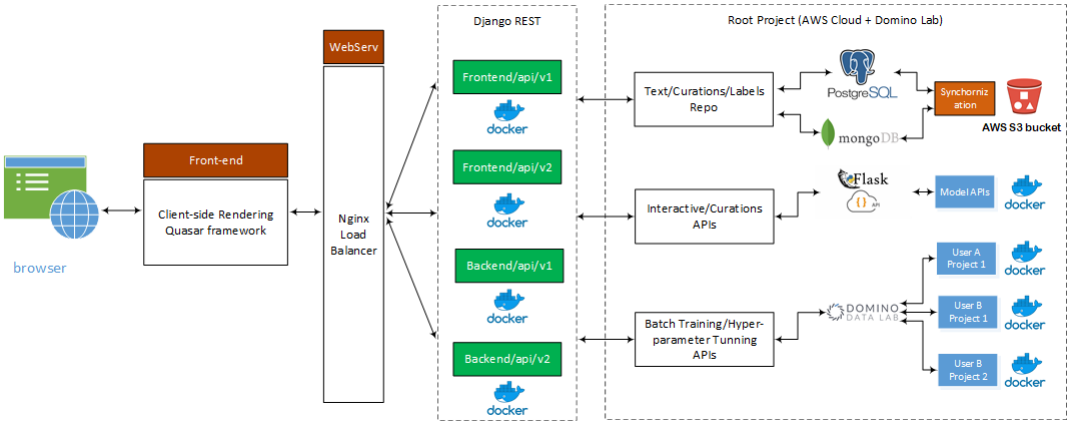

Figure 1: An overview of general architecture of MIC

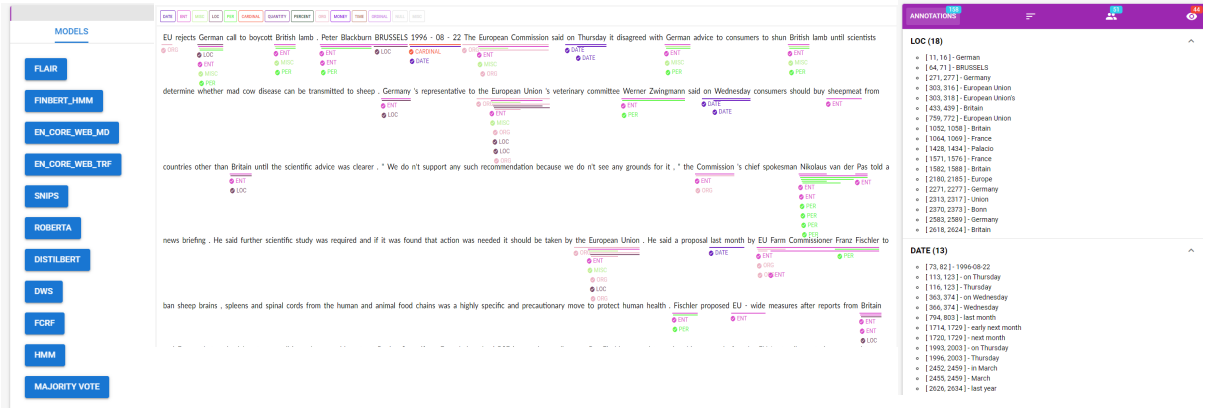

Figure 2: Annotation Interface of MIC

combinations of labels from the three levels. This feature make MIC very useful to gather important annotations from different perspectives for a same data set iteratively, which is commonly desired in industrial applications.

4.3 Annotation Summarization Panel

It is worth highlighting that MIC contains a well-designed annotation summarization panel (right panel) to efficiently and concisely provide valuable information about the annotations provided by multiple users. The panel has four controllable head icons: (1) Annotations, (2) Sorting, (3) Users, and (4) Issues. If a reviewer wants to group all annotations by categories, she can review all labels using the Annotations icon. Click-in will expand into all individual labels, and reviewer acceptance and rejection can be applied here. MIC supports all annotations being associated with confidence scores. For human annotators we can fix the score as 1 or allow them to explicitly score their confidence per each annotation. For machine learning models, the API Endpoints must return an additional output parameter representing uncertainty. Thus, a reviewer can rank all candidate annotations by their uncertainty scores using the sorting function. The third function *Users* allows the reviewer to group all annotations by annotator. Here, the reviewer can accept or reject all annotations from a specific annotator. This feature, combined with the ease of integrating weak labelers, makes MIC a great weak supervision data preparation tool. For example, the annotator can quickly try out multiple weak labelers, view some of their annotations, choose to reject the noisy labelers, and then export the remaining labels to be fed into an offline model to denoise the weak labels. The last function *Issues* is used to highlight potential conflicts that may be of concern for the reviewer such as contradictory annotations on the same unique token sequence. Another feature in this summarization panel is that clicking on any annotation listed here will redirect and highlight the corresponding tokens in the original text. This feature is very useful to quickly review and correct the annotations.

5 Case Studies

5.1 Market News Insider Trade Annotation

In the case study demo, the goal is to use MIC to extract facts about potential insider trades from a financial news feed. We assume the user is a subject matter expert (SME) curator with some basic knowledge of NLP and machine learning. The annotator has several goals. Firstly, from all the news feeds she needs to select those that are relevant to insider selling or buying. Since there is no machine learning model classifier distinguishing the insider trading concept at hand, the annotator decides to use a simple rule *inside buy/sell* to quickly generate machine labels. The rule is defined as a SpaCy rule in the admin panel such that if a sentence after lemmatization contains both words *inside* and *buy* or *sell*, then the machine auto-generated label will be set to *Inside Trade*. The annotator reviews results at the sentence-level panel, and manually corrects some mistaken predictions. Then, she saves the sentence labeling results and hides all sentences that are not related to insider trading. She switches the annotation panel from sentence to NER labeling, and uses several out-of-the-box NER models (i.e. from FLAIR (Akbik et al., 2019) or SpaCy) to quickly generate automatic NER tags for persons, organizations, locations, and others.

Next, the annotator switches the panel from *NER* to *Relation* to extract semantic relationships about insider trading. Her goal here is to extract *buy* and *sell* relationships that occur between two entities (usually the subject *person* entity is defined as the head and the object *stock* entity is defined as the tail). Instead of spending tedious effort to find desired relationships manually, the annotator applies an open relation extraction (OpenRE) model to automatically extract candidate relationships. If the annotator wants to further designate the extracted relationships as one of the three pre-defined types *buy*, *sell* and *own*, she can change the relation type to any text in the confirmation menu.

The MIC OpenRE model is based on the MaMa open information extraction (OpenIE) model described in (Wang et al., 2020) and built using the code from (theblackcat102, 2020). The OpenRE model carries out several steps such as named entity recognition, verb phrase pattern matching, pretrained language model inference, and triplet post-processing. Since each step may produce uncertainty in its output, the OpenRE approach tends to generate noisy candidate relationships.

In our demo, we published an OpenRE endpoint API that uses BERT-large-cased as the pre-trained language model (Devlin et al., 2019). For each page, it may produce about 40 to 100 noisy relations. Thus, the annotator needs to review and confirm all outputs in the summarization panel.

This case study demonstrates how to use MIC to accomplish multiple tasks of annotation, starting from sentence labeling, then named entity detection and finally extract important financial semantic relationships from text. All annotated entities and relationships are associated with three different positional indices: (1) sentence index, (2) token position index, and (3) character position index. This allows precise identifications and visualizations of extracted entities and relationships. Users can save these annotations and visualize them directly in MIC, or export them as JSON format for general machine learning model training and validation outside of MIC.

6 Evaluation of Annotation Efficiency and Accuracy

We conducted a benchmark study to investigate the efficiency and accuracy of MIC in real annotations. Three different data sets were used for task preparation: (1) CoNNL2003 (Tjong Kim Sang and De Meulder, 2003); (2) NYT Open Relation Extraction Benchmark (Mesquita et al., 2013); (3) Proprietary fintech customer support call transcripts. The first and second data sets are publicly accessible and widely used as NER and Relation Extraction benchmarks. The third data is a proprietary data set and the goal is to obtain three levels of annotations. The first level is sentence tagging: the annotator needs to extract the main customer complaint sentences from the call transcript if the complaint is related to buy/sell financial product (stocks/funds), denoted as a buy/sell relevant sentence. All other sentences are irrelevant. The second level is NER: Among the relevant sentences, tag any mentioned financial products (stock/fund tickers, bank names, and others) as named entities. The third level is to annotate any unary relationship, if related to buy or sell, of annotated entities if mentioned in the same sentence. For example, **Sell** Apple Stock, **Buy** NVDA, **Exchange** Money Market Funds, and so on. We compared four annotation tools, including free version of Prodigy, YEDDA (Yang et al., 2018), GATE(Bontcheva et al., 2013) and the proposed MIC tool. Four annotators selected from the

2Tool	CoNNL		NYT		Call	
	Avg Time	F1-score	Avg Time	F1-score	Avg Time	F1-score
Prodigy	63	0.75	94	0.52	208	0.86
YEDDA	85	0.76	101	0.60	189	0.82
GATE	84	0.75	118	0.62	150	0.78
MIC-NM	45	0.74	74	0.64	60	0.84
MIC-M	38	0.78	95	0.59	44	0.88

Table 2: Comparison of Average Annotation Time (integers as minutes) for different tasks and the Average F1-scores.

master internship program were trained to perform annotations. Each annotator spent about 2 hours on each tool using labeled data to get familiar with a tool's specific annotation mechanism. Then, a random sample of 50-sentence corpora from data sets (1) and (2), and 20 random transcriptions of data set (3) were assigned for annotation. For each annotator, sentences/transcriptions were stratified samplings by different annotation tools, so there was no occurrence of seen sentences across different tools. In this setting, each task had four samples, assigned to four annotators in parallel and all annotated once using the same tool. Because data in all tasks comes with ground truth labels, we measure annotation performance in this step through evaluating the micro-entity precision, recall, and then calculate the F1 scores of the annotated labels against the ground truth labels. The average annotation time and F1-scores of four annotators spent on this task-tool combination were recorded and compared in Table 2. Notice that CONLL and NYT are popular data sets studied in literature, and the best F1-score achieved by ML models on CONLL is around 0.76(Parker and Yu, 2021), and 0.59 for NYT (Sun and Wu, 2019).

We compared MIC in two configurations: (1) MIC-NM only allowed manual annotations, so no pre-trained model was used. (2) MIC-M included pre-trained annotation models so annotators could confirm final labels using auto-annotations. In the MIC-M setting, MIC included four NER models (FLAIR, FINBERT_HMM, EN_CORE_WEB_MD, SNIPS), one MaMa RE model (Wang et al., 2020), and a proprietary intent classification model to classify sentences. The average annotation time of four annotators, and the F1-score of their annotated results evaluated by ground truths, are reported in Table 1. As shown, on almost all tasks, MIC significantly reduced annotation time and obtained comparable performance. One exception was for the NYT task using MIC-M, where the MAMA (Wang et al., 2020) model was slow in execution, and results were very noisy. Thus annotators spent extra effort filtering the results and accidental misses caused performance drop. In particular, annotators found MIC very helpful in annotating long call transcripts because it provided a friendly interface filtering irrelevant sentences and allowed smooth switches among sentence/entity/relation annotations. In contrast, in other tools, annotators were overwhelmed by a dominant number of unrelated sentences, which caused serious distractions. Another advantage annotator liked MIC most was the stacked investigation of multiple auto-annotations tagged by pre-trained models, especially on CONNL task where pre-trained NER models were domain-homogeneous. In contrast, when annotating financial product entities, the four pre-trained models were not very helpful, mainly because those models had never adapted to the Financial NER domain.

7 Scalability and Deployment

For budget reasons, the demo system of MIC hosted at www.textmic.com is deployed on a single AWS T3.xlarge instance. However, RESTFUL APIs can be distributed to different physical instances for better performance and richer model capacity. Thus MIC could host a wide range of large-scale pre-trained models in its library and allow easy adaption of relevant models in specific annotation tasks.

8 Future Development

Our roadmap to enhance MIC for the future lies ahead in several directions. We are interested in connecting MIC to advanced processes cloud-based APIs such as zero-shot learning, few-shot learning, and textual entailment models to provide annotators access to more SoTA NLP models.

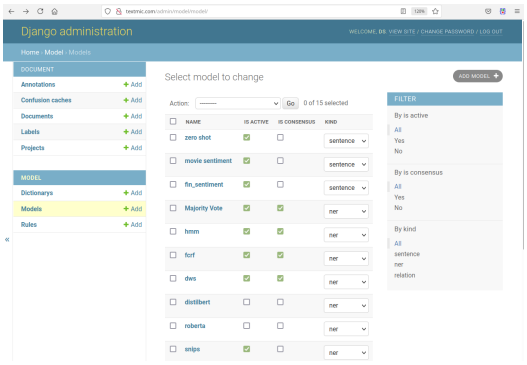

Figure 3: Django backend management of pre-trained annotation models

Additionally, we'll implement automated training pipelines for several weak supervision algorithms including (Ratner et al., 2017; Shang et al., 2018; Parker and Yu, 2021) to allow automatic denoising of conflicting human or machine labels.

References

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. Flair: An easy-to-use framework for state-of-the-art nlp. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59.

Marcel Bollmann, Florian Petran, Stefanie Dipper, and Julia Krasselt. 2014. CorA: A web-based annotation tool for historical and other non-standard language data. In *Proceedings of the 8th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (LaTeCH)*, pages 86–90, Gothenburg, Sweden. Association for Computational Linguistics.

Kalina Bontcheva, Hamish Cunningham, Ian Roberts, Angus Roberts, Valentin Tablan, Niraj Aswani, and Genevieve Gorrell. 2013. Gate teamware: a web-based, collaborative text annotation framework. *Language Resources and Evaluation*, 47:1007–1029.

Wei-Te Chen and Will Styler. 2013. Anafora: A web-based general purpose annotation tool. In *Proceedings of the 2013 NAACL HLT Demonstration Session*, pages 14–19, Atlanta, Georgia. Association for Computational Linguistics.

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.

Volker Gast, Lennart Bierkandt, and Christoph Rzymski. 2015. Creating and retrieving tense and aspect annotation with graphanno, a lightweight tool for multi-level annotation. In *ACL 2015*.

Xiaochuang Han and Jacob Eisenstein. 2019. Unsupervised domain adaptation of contextualized embeddings for sequence labeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4238–4248, Hong Kong, China. Association for Computational Linguistics.

Matthias Hartung, Hendrik ter Horst, Frank Grimm, Tim Diekmann, Roman Klinger, and Philipp Cimiano. 2018. SANTO: A web-based annotation tool for ontology-driven slot filling. In *Proceedings of ACL 2018, System Demonstrations*, pages 68–73, Melbourne, Australia. Association for Computational Linguistics.

Johannes Kiesel, Henning Wachsmuth, Khalid Al-Khatib, and Benno Stein. 2017. WAT-SL: A customizable web annotation tool for segment labeling. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 13–16, Valencia, Spain. Association for Computational Linguistics.

Jonathan K. Kummerfeld. 2019. SLATE: A super-lightweight annotation tool for experts. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 7–12, Florence, Italy. Association for Computational Linguistics.

Frédéric Landragin, Thierry Poibeau, and Bernard Victorri. 2012. ANALEC: a new tool for the dynamic annotation of textual data. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 357–362, Istanbul, Turkey. European Languages Resources Association (ELRA).

Dong-Ho Lee, Rahul Khanna, Bill Yuchen Lin, Seyeon Lee, Qinyuan Ye, Elizabeth Boschee, Leonardo Neves, and Xiang Ren. 2020. LEAN-LIFE: A label-efficient annotation framework towards learning from explanation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 372–379, Online. Association for 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.

Stephen Mayhew and Dan Roth. 2018. TALEN: Tool for annotation of low-resource ENtities. In *Proceedings of ACL 2018, System Demonstrations*, pages 80–86, Melbourne, Australia. Association for Computational Linguistics.

Filipe Mesquita, Jordan Schmidek, and Denilson Barbosa. 2013. Effectiveness and efficiency of open relation extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 447–457, Seattle, Washington, USA. Association for Computational Linguistics.

Jerrod Parker and Shi Yu. 2021. Named entity recognition through deep representation learning and weak supervision. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3828–3839, Online. Association for Computational Linguistics.

Alexander Ratner, Stephen H. Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. *Proc. VLDB Endow.*, 11(3):269–282.

Younes Samih, Wolfgang Maier, and Laura Kallmeyer. 2016. SAWT: Sequence annotation web tool. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*, pages 65–70, Austin, Texas. Association for Computational Linguistics.

Jingbo Shang, Liyuan Liu, Xiaotao Gu, Xiang Ren, Teng Ren, and Jiawei Han. 2018. Learning named entity tagger using domain-specific dictionary. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2054–2064, Brussels, Belgium. 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, Avignon, France. Association for Computational Linguistics.

Michael Stewart, Wei Liu, and Rachel Cardell-Oliver. 2019. Redcoat: A collaborative annotation tool for hierarchical entity typing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019 - System Demonstrations*, pages 193–198. Association for Computational Linguistics.

Changzhi Sun and Yuanbin Wu. 2019. Distantly supervised entity relation extraction with adapted manual annotations. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 7039–7046. AAAI Press.

theblackcat102. 2020. language-models-are-knowledge-graphs-pytorch. https://github.com/theblackcat102/language-models-are-knowledge-graphs-pytorch.

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*, pages 142–147.

Stephen Tratz and Nhien Phan. 2018. A web-based system for crowd-in-the-loop dependency treebanking. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018*. European Language Resources Association (ELRA).

Chenguang Wang, Xiao Liu, and Dawn Song. 2020. Language models are open knowledge graphs.

Jie Yang, Yue Zhang, Linwei Li, and Xingxuan Li. 2018. YEDDA: A lightweight collaborative text span annotation tool. In *Proceedings of ACL 2018, System Demonstrations*, pages 31–36, Melbourne, Australia. Association for Computational Linguistics.

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, Sofia, Bulgaria. Association for Computational Linguistics.