CEMA – Cost-Efficient Machine-Assisted Document Annotations

Guowen Yuan, Ben Kao, Tien-Hsuan Wu

The University of Hong Kong
{gwyuan, kao, thwu}@cs.hku.hk

Abstract

We study the problem of semantically annotating textual documents that are *complex* in the sense that the documents are *long*, *feature rich*, and *domain specific*. Due to their complexity, such annotation tasks require trained human workers, which are very expensive in both time and money. We propose CEMA, a method for deploying machine learning to assist humans in complex document annotation. CEMA estimates the human cost of annotating each document and selects the set of documents to be annotated that strike the best balance between model accuracy and human cost. We conduct experiments on complex annotation tasks in which we compare CEMA against other document selection and annotation strategies. Our results show that CEMA is the most cost-efficient solution for those tasks.

Introduction

Machine learning (ML) applications rely on the availability of large and high-quality datasets for model training. In many applications, large volumes of documents are collected from which key information is extracted via manual annotations. The extracted information can then be structured using other data models (e.g., knowledge graphs) to support information search and AI model training. Compared with other data types, document annotation can be a very expensive task. For example, it is reported in (Wu et al. 2020) that it took 11 workers with legal training 6 months to annotate a corpus of roughly 4,000 court judgments. The high cost of document annotation comes from a number of constraints and characteristics of textual data:

[Document complexity] Unlike simple annotation tasks such as image annotation, annotating a document could require substantial reading and comprehension. For example, the COLIEE2019 dataset (Shao et al. 2020), which is used in the study of legal document retrieval tasks, consists of case law articles with an average document length of 3,000 words; In (Harašta et al. 2018), 350 court decisions (documents) were manually annotated for legal reference recognition. These documents consist of 4,746 to 537,470 characters each, with an average of 36,148 characters per document. It takes much time to read and to identify salient points in such documents.

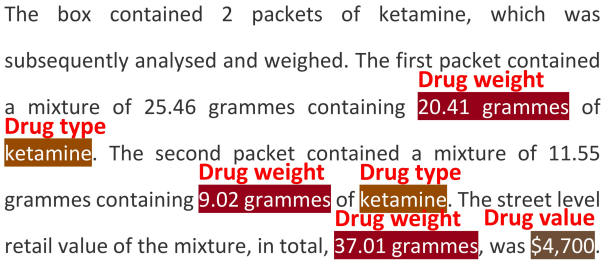


Figure 1: A fragment of an annotated judgment

[Feature complexity] Due to their non-trivial lengths, documents are usually multifaceted. Annotators are often required to remember numerous features (a.k.a. labels or tags) and to identify relevant texts that correspond to the features. For example, in (Wu et al. 2020), annotators are requested to annotate court judgments of illegal drug trafficking offenses with 82 features. Example features include *drug type*, *drug weight*, *defendant name*, and *prison term*. Figure 1 shows an example annotation where certain text segments are highlighted and their associated labels illustrated. The annotation task is thus highly complex and time-consuming, requiring workers to recall features and to associate them with relevant text segments.

[Domain knowledge] In many applications, the documents to be annotated are domain specific and are full of technical terms. For example, in court judgment annotation, one feature could be *defendant's mitigating factor*. Labeling for that feature requires a worker with legal knowledge who understands what constitutes a legitimate mitigation. In (Xia and Yetisgen-Yildiz 2012), radiology reports are annotated by radiologists and physicians to label texts that mention "critical suggestions" of follow-up actions. In such cases, annotators must possess relevant domain knowledge.

[Privacy] Often, documents contain confidential information such as business information or personal information. Medical documents, for example, may contain sensitive patient records. In such cases, the documents can only be annotated by a small group of people with access rights.

[Error-prone annotation] Due to the complexity of the annotation task, human annotations are error-prone. Given a large set of features, workers may forget to annotate some

features or mis-label text with the wrong features. In (Wu et al. 2020), each court judgment is annotated by two human workers; One provides the first annotations, and the other verifies the annotations and corrects mistakes.

The complexity of document annotations and the constraints on annotators imply that open crowdsourcing platforms are, in many cases, not suitable to carry out the annotation tasks. Instead, such tasks have to rely on small groups of trained workers, making the tasks very costly in time and money. In this paper we study how machines can assist in achieving more cost-efficient document annotation. In particular, we focus on a budget-constrained situation in which the available budget (time and money) for annotation is far less than what is needed to fully annotate the whole document corpus, which is often the case in real-world projects. The implication is that with the given budget, we strive to (1) extract as much information as possible from the annotated documents, (2) use the extracted features as training data to build the best machine annotator model. The latter can be used to help (partially) annotate documents that are not covered by the budget, and as we will see later to help improve the machine-assisted annotation task itself.

[Fully-manual Annotation (FMA) vs. Machine-assisted Annotation (MAA)] Figure 2 illustrates the processes of *fully-manual annotation* (FMA) and *machine-assisted annotation* (MAA). With FMA (Figure 2(a)), given a budget constraint, some documents are selected randomly from a corpus for annotations. For each document, the annotation task involves a human worker reading the document, highlighting important text segments, and associating each highlighted segment with the appropriate feature label. We call each "highlight and label" a *markup*. For example, Figure 1 shows 6 markups. For quality concerns, the annotated document is double-checked by a second human worker, who would confirm or correct each markup; or discover missing markups. We call the first and second workers in the process the *first annotator* and the *verifier*, respectively.

Employing two workers to annotate each document is expensive. We propose MAA (Figure 2(b)) to reduce cost. Our approach is to replace the (human) first annotator by a *machine annotator*. Initially, a small set of documents, called *seed documents*, are annotated using FMA. This seed set is used to train a machine annotator (MA). The MA is then applied to annotate *all* documents in the corpus. A selection module assesses the machine-annotated documents and selects a small batch of them to be verified and corrected by human workers. The documents with verified markups are then added to the pool of training data for refining the MA model. The process iterates until the human cost budget is exhausted. MAA is potentially more cost-efficient than FMA for two reasons. First, each document is processed by one human worker (verifier) under MAA instead of two (first annotator and verifier) under FMA. Secondly, if the first annotation done by the machine is sufficiently accurate, then most of the verifiers' job is to confirm the markups; In general, verifications are less time consuming than first annotations because the latter involve point-click-drag actions to highlight text segments and selecting appropriate feature labels from a long feature list, while the former only require simple clicks to accept the given markups.

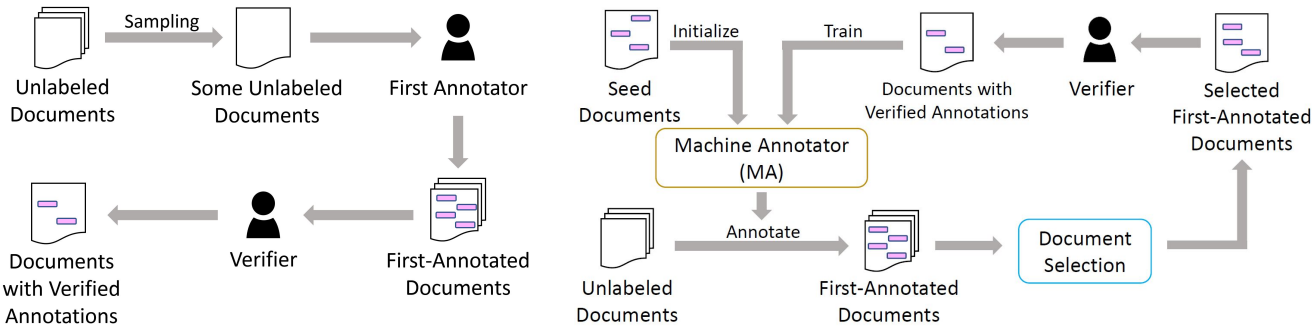
As we will show later, the document selection module is the major factor that determines the cost-efficiency of the MAA approach. On one hand, the selection module should select a document that contains many feature instances so that more examples are collected into the training pool for better MA model training. This improves the MA's accuracy resulting in less work for the human verifiers in correcting first-annotation mistakes. On the other hand, the selection module should select short documents to avoid long reading time of the human verifiers. To tackle this problem, we propose CEMA (Cost-Efficient Machine-assisted Annotation). CEMA extends the MAA approach by modeling and predicting the human cost and the machine annotator accuracy. These models are employed by CEMA in its cost-based document selection strategy. Our experiments show that CEMA is substantially more cost-efficient than other document selection methods within the MAA framework [1].

Related Work

[Machine-Assisted Annotation Tools] There are studies on designing annotation tools that facilitate efficient human annotations. BRAT (Stenetorp et al. 2012) is a tool that supports various annotation schemes for different NLP tasks, including POS tagging, semantic role labeling (Gildea and Jurafsky 2002) and dependency annotation (Nivre 2003). WAT-SL (Kiesel et al. 2017) predicts markups' highlights so that human annotators only need to perform label assignments. Tagtog (Cejuela et al. 2014) and ezTag (Kwon et al. 2018) incorporate a lexicon-based tagger and an ML annotator to pre-annotate documents. Compared with these existing tools, CEMA employs an intelligent document selection strategy that minimizes human cost in processing the selected documents and maximizes the amount of information collected from the selected documents.

[Document Selection and Active Learning] With a limited budget, it is important to select the best set of documents to be labeled that can maximize the information obtained. Works that are based on *active learning* approaches have been proposed. Many works use the *uncertainty* of the feature labels assigned by machine to documents as the document selection criteria (Berger, Della Pietra, and Della Pietra 1996; Settles and Craven 2008; Scheffer, Decomain, and Wrobel 2001; Baldridge and Osborne 2004; Hwa 2004; Culotta and McCallum 2005). The idea is to boost the accuracy of the MA by training it with documents that it may not perform well in labeling. For example, (Culotta and McCallum 2005) proposes the Least Confidence (LC) method. With LC, each word (token) of a document is assigned a feature label and a confidence score is given to each such label. The document with the smallest sum of the log of its tokens' confidence scores is selected for human annotation. In (Shen et al. 2018), the authors show that LC tends to select long documents and propose MNLP, which normalizes a document's confidence score with its length.

[1] The code of CEMA can be found in https://github.com/gavingwyuan/cema



(a) Fully-manual annotation (FMA) (b) Machine-assisted annotation (MAA)

Figure 2: Fully-manual annotation vs. machine-assisted annotation

Recently, deep learning is applied to learn a document selection policy. Examples include ALIL (Liu, Buntine, and Haffari 2018) and Dream (Vu et al. 2019). In Dream, a neural-network-based document selection policy, let us call it P, is learned by iteratively applying two steps called *wake learn* and *dream learn*. The wake-learn step is about *applying* policy P. Specifically, a batch of k documents are first shortlisted using a Least-Confidence (LC)-like method. Then, policy P is applied to the shortlisted candidates to select some documents to be labeled by human workers. The labeled documents are then used to update a machine annotator model. The dream-learn step is about *learning* policy P. Specifically, the neural network (of P) is trained by simulating an active learning process using machine-labeled documents and human-labeled documents, during which the DAGGER (Ross, Gordon, and Bagnell 2011) algorithm is used to update policy P.

There are also studies that focus on analyzing annotation costs. (Settles, Craven, and Friedland 2008) shows that incorporating cost estimates into active learning strategies can reduce the cost of training a machine annotator. In (Gao and Saar-Tsechansky 2020), more accurate human annotators are assumed to be more expensive. They propose GBAL, which determines the documents to be labeled and the human annotators to be assigned the labeling tasks.

As we will see later, CEMA differs from the above methods in that it employs a more accurate cost model to estimate human cost and a knowledge model that assesses the amount of new feature instances that a document carries to better train the MA. This results in a document selection strategy that takes into account not only the machine annotation accuracy, but also the human cost incurred. Potentially, CEMA helps select *more* and *better* documents in the MAA process.

Model and Problem Definition

Let $\mathcal{D} = \{d_1, ..., d_{|\mathcal{D}|}\}$ be a corpus of documents. Each document $d_i = [x_{i1}, .., x_{i|d_i|}]$ is modeled as a sequence of *tokens* (words) x_{ij}'s, where $|d_i|$ denotes the *length* or the number of tokens of d_i. Let $\mathcal{F} = \{f_1, ..., f_{|\mathcal{F}|}\}$ be a set of feature labels. Given a document d_i, we *annotate* it by assigning feature labels to certain text segments. Specifically, a *machine annotator* (MA) assigns a feature label $l_{ij} \in \mathcal{F}$ to each token x_{ij} of d_i, or "NIL" if the corresponding token is not part of any feature[2]. A sequence of tokens with the same feature label form a *feature instance* or a *markup*. Specifically, we abstract the annotation done by an MA on a document d_i as a sequence of markups $A_i = [m_{i1}, ..., m_{i|A_i|}]$, where $|A_i|$ denotes the number of markups in A_i. Each markup m_{ij} has the form (s_{ij}, e_{ij}, f_{ij}), which denotes that "the j-th markup in document d_i starts at (token) position s_{ij}, ends at position e_{ij} and is assigned the feature f_{ij}." For example, in Figure 1, the first markup is denoted by (24, 25, *Drug_Weight*). The annotation can also be done by a human annotator. We use A_i^H and A_i^M to denote the annotations done by human and by machine, respectively.

As we have discussed, our objective is to design machine-assisted annotation (MAA) (Figure 2(b)), particularly the *document selection module* so that (1) we minimize the human cost and (2) maximize the amount of information extracted from the annotated documents. We argued that in typical projects, the budget is not enough to annotate the complete corpus by human workers. We thus consider the optimization problem in two ways: (1) we minimize the human cost given a desired amount of extracted knowledge, or (2) we maximize the extracted knowledge without exceeding a human cost budget. In either case, we need to model human cost and knowledge. Note that the extracted knowledge is used to build a machine annotator (MA) in the MAA process. Hence, we assess the extracted *knowledge* by the accuracy of the MA built. We assess *human cost* by the following cost model that takes into account the different actions a human verifier (in MAA) would perform when he/she is presented with a document d_i and its machine first annotation A_i^M. The actions are:

Read. The verifier reads document d_i. We use t_{read} to denote the per-token reading time. **Confirm.** If the verifier agrees with a markup $m_{ij} \in A_i^M$ given by the machine, the verifier confirms m_{ij} by a simple click action. **Re-label.** If a markup has a wrong feature label, the verifier clicks on a pull-down menu and assigns it the correct label. **Delete.** If a markup does not contain any correct feature, the veri-

[2]Technically, our MA uses BIOES tagging scheme. That is, it indicates, besides a feature f, whether a token is, e.g., the start/end of f. For simplicity, we skip these details in our discussion.

fier deletes the markup by a couple of mouse clicks. **Add.** If the verifier spots a missing markup, he/she selects a text segment by a click-and-drag, and adds a label to it.

In addition, if a markup should be revised by adjusting the span of the highlight and possibly assigning it a new label, the modification can be modeled as a **Delete** followed by an **Add**. We call this a **Delete-add**.

We use t_X and n_X to denote the time cost to perform an action X and the number of X actions, respectively, where X is confirm/re-label/delete/add. The human annotation (verification) cost $c(d_i, A_i^M)$ for d_i is:

$$c(d_i, A_i^M) = t_{read} \cdot |d_i| + t_{confirm} \cdot n_{confirm} + t_{relabel} \cdot n_{relabel} + t_{delete} \cdot n_{delete} + t_{add} \cdot n_{add}. \quad (1)$$

If we employ a human annotator to first-annotate a document (instead of to verify), the cost can be modeled as $c(d_i, \emptyset)$.

CEMA

CEMA extends MAA (Figure 2(b)) by carefully crafting a document selection module that takes both human cost and amount of knowledge obtained into account. In this section we explain the various components based on which CEMA selects documents to be annotated.

[Human cost estimation] Recall that under MAA, unlabeled documents are first-annotated by a machine annotator (MA) (see Figure 2(b)). CEMA considers the human verification cost of each such unlabeled document and prefers the one with the lowest such cost in its document selection module. It assesses the cost based on Equation 1. The various t_X's (time for each action type) are obtained by observing human annotators in real annotation exercises. We estimate the n_X's (the number of each action type taken to verify a document) by an *action predictor*. Specifically, given an unlabeled document d_i, CEMA applies the machine annotator (MA) to label the tokens in d_i, i.e., it obtains the machine markups A_i^M. The action predictor employs a neural network (details to be given shortly) that outputs predicted actions (*confirm/re-label/delete/add*) on the markups. The n_X's are then tallied based on the predicted actions. Furthermore, CEMA performs *paragraph filtering*. The idea is that if the tokens of a paragraph P in d_i are all labeled "NIL" (no features) by the MA, we found that it is very unlikely that P contains any features. Hence, P is removed from d_i. This paragraph filtering saves human cost because it reduces the document's length ($|d_i|$) and hence reading time.

[Action predictor] Figure 3 illustrates the design of the action predictor. Let $d_i = [x_{i1}, .., x_{i|d_i|}]$ be a document with its machine annotation A_i^M. Recall that A_i^M assigns a feature label l_{ij} to each token x_{ij} (see Section). The predictor first obtains an embedding for each token and an embedding for its label. Specifically, the document d_i is divided into segments (such as sentences or paragraphs). Let g be a segment and x be a token in the segment. We obtain the embedding $\vec{x}$ of x using BERT by taking the final hidden states, i.e., $\vec{x} = BERT(g)$, where $\vec{x} \in \mathbb{R}^m$ and $m = 768$ is the size of BERT's hidden layer. The embedding $\vec{l}$ of the label l of token x is similarly obtained, i.e., $\vec{l} = BERT(l)$.

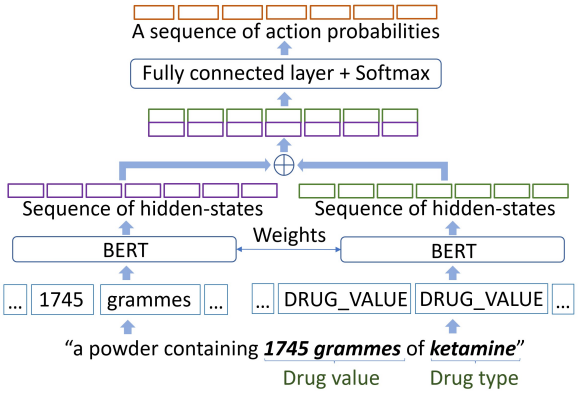


Figure 3: Action prediction

These embeddings are fed to a neural network that outputs predicted actions, one for each token. We call these token-level actions, *per-token actions* (to distinguish them from markup-level actions, see Section). The conditional probability of assigning an action a_{ij} to a token x_{ij} is defined as $p(a_{ij}|x_{ij}, l_{ij}) = SOFTMAX([\vec{x_{ij}}; \vec{l_{ij}}]W)$, where $W \in \mathbb{R}^{2m \times c}$, $a_{ij} \in \mathbb{R}^c$, and $c = 6$ is the number of per-token action types[3]. The action a_{ij} with the highest probability for token x_{ij} is taken as the predicted action on the token. Given a machine-annotated markup y, the per-token action predicted for y's first token is taken as the predicted action of markup y.

The objective of the action predictor is to maximize the probability of an action sequence a conditioned on a (token, label) sequence given in g:

$$\max \sum_{d_i \in D} \sum_{g \in d_i} \sum_{(x_{ij}, l_{ij}) \in g} \log p(a_{ij}|x_{ij}, l_{ij}), \quad (2)$$

where D is the set of documents with verified annotations. We use the cross-entropy between action predictor's output and ground truth action as the loss function in training.

[Knowledge estimation] In MAA, the (human) verified markups constitute the collected knowledge from labeled documents. During document selection, CEMA assesses each unlabeled document, say d_i. This is done by first applying the MA to d_i to obtain its machine annotation A_i^M, followed by estimating the potential knowledge that can be gained from d_i if the document is selected and (later) verified. Not all markups, however, are equal. Specifically, CEMA computes a *contribution score*, $s(d_i, A_i^M)$, of each d_i given its A_i^M. Figure 4 illustrates how the document scores are computed. In the figure, d_1 is a verified labeled document with two markups (m_1 and m_2). Documents d_2, d_3, and d_4 are unlabeled. CEMA applies MA on d_2, d_3, d_4 and obtains machine markups (m_3 to m_8). CEMA then determines if two markups are *similar*. Specifically, given two markups m_i, m_j, let e_i and e_j be the average token embeddings of the markups' highlighted texts. If the cosine similarity of e_i and e_j exceeds a threshold ρ, then markups m_i

[3]The per-token actions are *read*, *confirm*, *re-label*, *delete*, *add*, and *delete-add*.

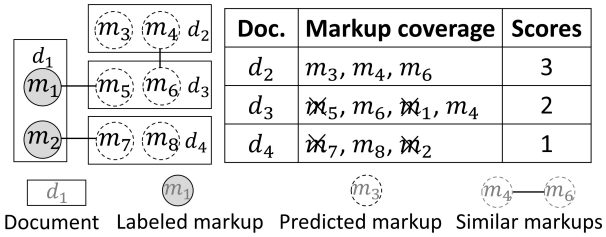


Figure 4: Contribution scores of documents

and m_j are considered similar. In Figure 4, similar markups are connected by a line. For an unlabeled document d_i, we consider the markups in A_i^M as well as the markups in other documents that are similar to those in A_i^M. For example, d_3 has a *markup coverage* of $\{m_5, m_6, m_1, m_4\}$. Note that m_5, m_6 are in d_3; m_1, m_4 are similar to some markups d_3 contains. The markup coverage of d_i reflects that, if d_i is labeled and its (verified) markups are used to train an MA, the number of markups the MA can extract if it is applied to the *whole document corpus*. The markup coverage is further modified by removing from it any markups that are similar to those that have already been collected in labeled documents. For example, m_1, m_5 are removed from d_3's markup coverage (because m_1 is in an already-labeled document and m_5 is similar to m_1). Finally, $s(d_i, A_i^M)$ is given by the cardinality of d_i's markup coverage.

In each iteration of the MAA process (Figure 2(b)), CEMA selects a batch of k unlabeled documents that give the best balance between human cost and the knowledge gained. Specifically, the document selected is given by:

$$\arg\min_{d_i \in \mathcal{D}_U} \quad w \cdot c(d_i, A_i^M) - (1-w) \cdot s(d_i, A_i^M). \qquad (3)$$

In Equation 3, $\mathcal{D}_U$ denotes the set of unlabeled documents and w is a parameter that gives relative weights to the cost and the score functions. Also, $c(d_i, A_i^M)$ and $s(d_i, A_i^M)$ are normalized using min-max normalization. In each MAA iteration, the selection is repeated k times until a batch of k documents are collected.

Experiments

We evaluate CEMA and compare it against other existing methods. In this section we report our experiment results.

[Datasets] We use the following two document datasets for our experiments. These datasets are annotated by domain experts with feature labels. The human annotations are used as ground truth for performance evaluation.

The **Drug Trafficking Judgments (DTJ)** dataset (Wu et al. 2020) consists of 4,045 court judgments on drug trafficking cases collected from the HKLII[4] website. The average document length is 940 tokens with a standard deviation of 610. We consider 5 feature labels, namely, *neutral citation*, *drug weight*, *drug type*, *drug value*, and *date of offense*.

The **German Legal (GL)** dataset (Leitner, Rehm, and Schneider 2020) consists of 66,723 sentences that are extracted from 750 legal documents written in German. We consider 7 feature labels: *person*, *location*, *organization*, *legal norm*, *case-by-case regulation*, *court decision* and *legal literature*. Since the dataset does not provide the original documents, we chunk the 66K sentences into 750 documents of 88 sentences each. (We will provide the source code of this chunking step.) The average document length is 2,931 words. For GL, we consider each sentence as a paragraph for the purpose of paragraph filtering in CEMA. We remark that the "documents" in GL have similar lengths while those in DTJ have bigger variations. This difference allows us to study the various selection methods especially those that take costs into account.

Experiments are conducted using 5-fold cross validation in which 80% of the documents are used as the set of unlabeled documents (for which the MAA process is applied), and 20% of the documents (with their ground truth markups) are used to evaluate the accuracy of the resulting machine annotator (after the MAA process terminates).

[Other settings] We adopt pre-trained English[5] and German[6] BERT models for the experiments with DTJ and GL datasets, respectively. For MAA, we use 2 human-annotated documents as seed. For this initialization, we train the machine annotator and the action predictor for 50 epochs. In each MAA iteration, the document selection module selects k = 2 documents to be verified by human workers. Practically, we could set k to be the number of human annotators so that the each batch can be obtained with parallel annotations by the workers. We observe that CEMA works reasonably well when k = 2 or above. The labeled markups are then added to the training pool to re-train the MA. We use 5 epochs in re-training. The batch size for training the MA and the action predictor are 8 and 3, respectively. We set ρ = 0.8 and w = 0.5. In all trainings, we use AdamW (Loshchilov and Hutter 2019) optimizer and set the learning rate to 2×10^{-5}. Based on observing real human annotation tasks, we set the time costs (in seconds) of verifier actions to t_{read} = 0.3s; $t_{confirm}$ = 1s; $t_{relabel}$ = 4s; t_{delete} = 1.5s; t_{add} = 10s as our default setting.

[Other methods and evaluation metrics] We compare CEMA against the following document selection strategies:

- **Random (Rand)**: select a document randomly.
- **Min Length (MinL)**: select the shortest document.
- **Least Confidence (LC)**: (Culotta and McCallum 2005).
- **Dream (Dream)**: (Vu et al. 2019).
- **MNLP (MNLP)**: (Shen et al. 2018).

LC , Dream and MNLP are described in Section .

Recall that the MAA process is iterative. In each iteration, a batch of k documents are selected, which are first-annotated by an MA. These (machine-annotated) documents are then presented to human workers for verification, and the verified markups are added to the training pool to re-train the MA. Hence, as MAA iterates, we accumulate more human costs (in verification) and knowledge (in verified markups). We compare the various methods by reporting their cumulative costs (using Equation 1) and the F1 score of the MA (which is trained by the cumulated verified markups). For

[4]https://www.hklii.org/

[5]https://huggingface.co/bert-base-uncased

[6]https://huggingface.co/dbmdz/bert-base-german-cased

computing human costs, the numbers of the various action types (the n_X's in Equation 1) on a selected document d_i are determined by comparing the machine annotations (A_i^M) with the ground truth markups of d_i.

In addition, we report the costs and knowledge obtained using fully-manual annotation (FMA, see Figure 2(a)) as a reference. For FMA, the human cost includes both first-annotation cost and verification cost. We estimate the (human) first-annotation cost by assuming a *perfect* human annotator. Specifically, the cost includes a document's reading time plus all the *add* actions of all the (ground-truth) markups in the document. The verification cost would then include the document's reading time plus all the *confirm* actions of the markups. We remark that this is an *optimistic estimate* of FMA's cost because in practice human annotators do make mistakes and so verification usually involves other actions that are more costly than simple confirms. To evaluate the knowledge FMA obtains, we compute the F1 score of a machine annotator that is built using the verified markups collected in the FMA process.

[Results] We first compare the methods in terms of the cost accrued for reaching a certain level of accumulated knowledge; and the amount of saving MAA achieves compared against FMA. Specifically, we record the cumulative human costs until the F1 score of the MA trained using the verified markups collected by each of the various methods is at least 0.8. Table 1 shows the results for the DTJ and GL datasets. For each dataset, we show the FMA cost (1st column) as reference; The first row shows the human cost (in bold) of each MAA document selection method. Each cost is also expressed (in brackets) as a fraction of the corresponding FMA cost. From the results, we make a few observations:

• While FMA employs two workers to annotate each selected document, MAA replaces the first-annotator by a machine annotator (MA). Intuitively, since MAA uses only 1 human worker instead of 2, MAA should save 50% of human cost compared against FMA. There are a few factors that affect this saving. First, the first-annotation done by the machine annotator could be less accurate than a human annotator, which results in more human time in verification, reducing the saving. Secondly, MAA could use a "*smart*" document selection strategy that selects short documents (e.g., MinL) or documents that contain important markups (e.g., LC); This reduces the *total amount* of document contents that need to be read and verified (by human verifiers), increasing the saving. The exact savings depend on the strategy and the characteristics of the document corpus. For example, the costs of MAA using Rand are 48.7% and 45.3% of those of FMA for DTJ and GL, respectively. Note that Rand does not perform smart document selections. The fact that the savings (1-48.7% = 51.3% and 1-45.3% = 54.7%) it registers are higher than 50% implies that the machine annotator (MA) is doing reasonably well. (The over-50% savings come from the fact that verification generally costs less than first-annotation.) MAA, which uses machine to first-annotate documents, is thus a viable option.

• The performance of MinL is surprisingly poor. In particular, its cost for GL (40,413) is 81.8% of FMA (49,376). The saving (1-81.8%=18.2%) is far less than the 50% mark, showing that MAA with MinL is hugely counterproductive (compared with simple random selection). For each dataset in Table 1, the second row shows the average length of selected documents under each selection strategy. We see that MinL's choices of documents are much shorter than Rand. (For DTJ: 304 tokens (MinL) vs. 893 tokens (Rand); for GL: 2,051 tokens (MinL) vs. 2,886 tokens (Rand).) The reason why MinL incurs very high cost even though it processes short documents is that the documents it picks are generally of low knowledge quality and thus MinL needs to process more documents. The third row in Table 1 shows the total number of markups collected by each strategy. We see that for GL, MinL collects a large number of markups (2,671) compared with others (which are all < 1700). Recall that in this experiment, the MAA process stops when the MA's F1 score reaches 0.8. The fact that MinL needs to collect such a big number of markups (from many documents) before the process terminates indicates significant redundancy and biases in the collected markups. This result shows that a selection strategy that considers only document length does not work well; Other criteria, such as the knowledge contents given by the markups, should also be taken into account.

• Another surprising result is that smart selection strategies, such as LC, MNLP and Dream, do not have clear advantages over simple random selection. The costs of LC are 46.1% (DTJ) and 46.5% (GL); the costs of MNLP are 45.6% (DTJ) and 53.5% (GL); the costs of Dream are 82.8% (DTJ) and 45.2% (GL). While the costs of LC and MNLP are somewhat comparable to those of Rand, Dream performs poorly for the DTJ dataset with a high cost (82.8%). We remark that LC and Dream select documents with the objective of improving the MA model, they do not consider documents' length or the human costs involved in processing the selected documents. Given that long documents tend to contain more markups, as a result, the documents LC and Dream select are (on average) the longest among all methods. For example, an average document selected by Dream in DTJ has 2,575 tokens. That is way bigger than an average document selected by Rand, which is just 893 tokens in length. For both DTJ and GL datasets, LC and Dream have worse performance than Rand because they ignore the cost factor. In contrast, MNLP considers documents' length in addition to their contents. For DTJ, where documents vary much in their lengths, MNLP tends to select much shorter documents (741 tokens/doc) while targeting those with useful markups. It thus performs better than Rand, LC and Dream. For GL, however, there is less variation in document lengths. In this case, MNLP is less cost-efficient than Rand, LC and Dream.

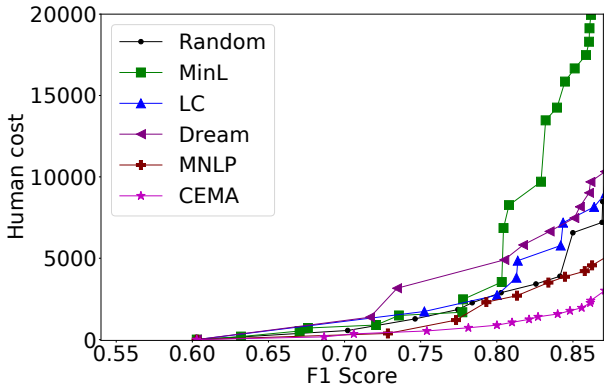
• CEMA performs the best among all methods. Its costs are 15.0% (DTJ) and 24.4% (GL). In particular, CEMA is 6.7 times more cost-efficient than FMA for DTJ. The good performance of CEMA comes from three factors. First, it uses a more precise human cost estimation to evaluate the verification cost of each candidate document. This is especially effective for DTJ because the documents have big variations in their lengths and markup counts. Secondly, CEMA analyzes the knowledge contents of documents to determine their contribution scores. This effectively identifies documents that can best improve the MA model. Thirdly, CEMA

		Rand	MinL	LC	Dream	MNLP	CEMA
DTJ (FMA cost = 5,917)	**MAA cost (as fraction of FMA cost)**	**2,884 (48.7%)**	**3,537 (59.8%)**	**2,731 (46.1%)**	**4,899 (82.8%)**	**2,700 (45.6%)**	**887 (15.0%)**
	Average number of tokens per selected document	893	304	1,934	2,575	741	200*
	Total number of markups in selected documents	51	123	72	47	106	70*
GL (FMA cost = 49,376)	**MAA cost (as fraction of FMA cost)**	**22,384 (45.3%)**	**40,413 (81.8%)**	**22,961 (46.5%)**	**22,307 (45.2%)**	**26,423 (53.5%)**	**12,037 (24.4%)**
	Average number of tokens per selected document	2,886	2,051	3,438	3,253	2,996	1,498*
	Total number of markups in selected documents	1,340	2,671	1,390	1,397	1,653	958*

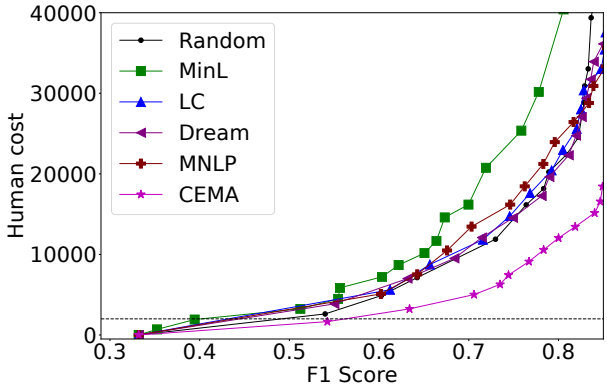
Table 1: Human cost incurred for a method's MA to achieve at least 0.8 F1. (*Counted after paragraph filtering.)

performs paragraph filtering, which removes paragraphs that are unlikely to contain any markups. This reduces document lengths and hence human verifiers' reading time. The last point is evidenced by Table 1, where we see that the numbers of tokens/doc under CEMA, after paragraph filtering, are 200 (DTJ) and 1,498 (GL). Both are much lower than those of other methods. On closer inspection, we find that CEMA's paragraph filtering is highly accurate and effective. For example, for DTJ, CEMA removes 86.3% of the paragraphs in its selected documents. Among the removed paragraphs, more than 99% of them do not contain any markups in the ground-truth data.

Figures 5(a) and 5(b) show the methods' accrued human costs and their accumulated knowledge (expressed in terms of MA's F1 scores) as the MAA process iterates. We can draw similar conclusions as in our previous discussion: (1) MinL, LC, Dream, and MNLP do not exhibit clear advantage over Rand. (2) Among those four existing methods, MNLP gives better performance for the DTJ dataset. This is because documents in the DTJ dataset have a bigger variation in length and MNLP considers document lengths in its selection strategy. (3) CEMA outperforms all methods significantly, especially for DTJ. We remark that in a limited-budget scenario, it is very important that we employ the most cost-efficient MAA method. This is because without spending over the budget, an efficient method, such as CEMA, can collect the best set of markups to train the most accurate MA; and an accurate MA gives us the best shot at (machine) annotating the rest of the documents that cannot be covered by human annotation due to budget concerns.



(a) DTJ



(b) GL

Figure 5: Human costs and MA F1 scores under different MAA document selection strategies

Conclusion

In this paper we propose machine-assisted annotation (MAA) as the solution to annotate a document corpus under a stringent budget constraint. Specifically, we replace human first-annotators in FMA by a machine annotator (MA), and employ human workers only for annotation verification. We also propose CEMA, which extends the basic MAA with a document selection module. The core idea of CEMA is to strike a balance between the verification cost and the amount of knowledge gained in document selection. Specifically, we propose a carefully engineered cost estimator model and a knowledge estimator model. Through experiments that are based on real document annotation tasks, we show that our cost estimator model, which employs an action predictor, is highly accurate. In particular, the cost estimator is very effective in identifying low-cost documents for selection during the initial stage of the MAA process. We also compare CEMA against a number of other document selection strategies. Our results show that CEMA achieves significantly larger savings compared with other methods over a wide range of annotation environments and scenarios.

Acknowledgments

This project is supported by Innovation and Technology Fund (ITS/234/20) and the WYNG Foundation (HKU KG210018).

References

Baldridge, J.; and Osborne, M. 2004. Active learning and the total cost of annotation. In *EMNLP*, 9–16.

Berger, A.; Della Pietra, S. A.; and Della Pietra, V. J. 1996. A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1): 39–71.

Cejuela, J. M.; McQuilton, P.; Ponting, L.; Marygold, S. J.; Stefancsik, R.; Millburn, G. H.; Rost, B.; Consortium, F.; et al. 2014. tagtog: interactive and text-mining-assisted annotation of gene mentions in PLOS full-text articles. *Database: The Journal of Biological Databases & Curation*.

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

Gao, R.; and Saar-Tsechansky, M. 2020. Cost-accuracy aware adaptive labeling for active learning. In *AAAI*, volume 34, 2569–2576.

Gildea, D.; and Jurafsky, D. 2002. Automatic Labeling of Semantic Roles. *Computational Linguistics*, 28(3): 245–288.

Harašta, J.; Šavelka, J.; Kasl, F.; Kotková, A.; Loutockỳ, P.; Míšek, J.; Procházková, D.; Pullmannová, H.; Semenišín, P.; Šejnová, T.; et al. 2018. Annotated Corpus of Czech Case Law for Reference Recognition Tasks. In *International Conference on Text, Speech, and Dialogue*, 239–250. Springer.

Hwa, R. 2004. Sample Selection for Statistical Parsing. *Computational Linguistics*, 30(3): 253–276.

Kiesel, J.; Wachsmuth, H.; Al Khatib, K.; and Stein, B. 2017. WAT-SL: a customizable web annotation tool for segment labeling. In *Proceedings of the Software Demonstrations of the 15th Conference of EACL*, 13–16.

Kwon, D.; Kim, S.; Wei, C.-H.; Leaman, R.; and Lu, Z. 2018. ezTag: tagging biomedical concepts via interactive learning. *Nucleic Acids Research*, 46(W1): W523–W529.

Leitner, E.; Rehm, G.; and Schneider, J. M. 2020. A Dataset of German Legal Documents for Named Entity Recognition. In *LREC*, 4478–4485.

Liu, M.; Buntine, W.; and Haffari, G. 2018. Learning how to actively learn: A deep imitation learning approach. In *ACL*, 1874–1883.

Loshchilov, I.; and Hutter, F. 2019. Decoupled weight decay regularization. In *ICLR*.

Nivre, J. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the Eighth International Conference on Parsing Technologies*, 149–160.

Ross, S.; Gordon, G.; and Bagnell, D. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 627–635. JMLR Workshop and Conference Proceedings.

Scheffer, T.; Decomain, C.; and Wrobel, S. 2001. Active hidden markov models for information extraction. In *International Symposium on Intelligent Data Analysis*, 309–318. Springer.

Settles, B.; and Craven, M. 2008. An analysis of active learning strategies for sequence labeling tasks. In *EMNLP*, 1070–1079.

Settles, B.; Craven, M.; and Friedland, L. 2008. Active learning with real annotation costs. In *Proceedings of the NIPS workshop on cost-sensitive learning*, volume 1.

Shao, Y.; Mao, J.; Liu, Y.; Ma, W.; Satoh, K.; Zhang, M.; and Ma, S. 2020. BERT-PLI: Modeling Paragraph-Level Interactions for Legal Case Retrieval. In *IJCAI*, 3501–3507.

Shen, Y.; Yun, H.; Lipton, Z. C.; Kronrod, Y.; and Anandkumar, A. 2018. Deep Active Learning for Named Entity Recognition. In *ICLR*.

Stenetorp, P.; Pyysalo, S.; Topić, G.; Ohta, T.; Ananiadou, S.; and Tsujii, J. 2012. BRAT: a web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of EACL*, 102–107.

Vu, T.; Liu, M.; Phung, D.; and Haffari, G. 2019. Learning how to active learn by dreaming. In *ACL*, 4091–4101.

Wu, T.; Kao, B.; Cheung, A. S. Y.; Cheung, M. M. K.; Wang, C.; Chen, Y.; Yuan, G.; and Cheng, R. 2020. *Integrating Domain Knowledge in AI-Assisted Criminal Sentencing of Drug Trafficking Cases*, volume 334 of *Frontiers in Artificial Intelligence and Applications*, 174–183. IOS Press.

Xia, F.; and Yetisgen-Yildiz, M. 2012. Clinical corpus annotation: challenges and strategies. In *Proceedings of the Third Workshop on Building and Evaluating Resources for Biomedical Text Mining (BioTxtM'2012) in conjunction with the International Conference on Language Resources and Evaluation (LREC), Istanbul, Turkey*.