# Active Learning with Amazon Mechanical Turk

**Florian Laws Christian Scheible Hinrich Schütze**
Institute for Natural Language Processing
Universität Stuttgart
{lawsfn, scheibcn}@ims.uni-stuttgart.de

## Abstract

Supervised classification needs large amounts of annotated training data that is expensive to create. Two approaches that reduce the cost of annotation are *active learning* and *crowdsourcing*. However, these two approaches have not been combined successfully to date. We evaluate the utility of active learning in crowdsourcing on two tasks, named entity recognition and sentiment detection, and show that active learning outperforms random selection of annotation examples in a noisy crowdsourcing scenario.

## 1 Introduction

Supervised classification is the predominant technique for a large number of natural language processing (NLP) tasks. The large amount of labeled training data that supervised classification relies on is time-consuming and expensive to create, especially when experts perform the data annotation. Recently, crowdsourcing services like Amazon Mechanical Turk (MTurk) have become available as an alternative that offers acquisition of non-expert annotations at low cost. MTurk is a software service that outsources small annotation tasks – called *HITs* – to a large group of freelance workers. The cost of MTurk annotation is low, but a consequence of using non-expert annotators is much lower annotation quality. This requires strategies for quality control of the annotations.

Another promising approach to the data acquisition bottleneck for supervised learning is active learning (AL). AL reduces annotation effort by setting up an annotation loop where, starting from a small seed set, only the maximally informative examples are chosen for annotation. With these annotated examples, the classifier is then retrained to again select more informative examples for further annotation. In general, AL needs a lot fewer annotations to achieve a desired performance level than random sampling.

AL has been successfully applied to a number of NLP tasks such as part-of-speech tagging (Ringger et al., 2007), parsing (Osborne and Baldridge, 2004), text classification (Tong and Koller, 2002), sentiment detection (Brew et al., 2010), and named entity recognition (NER) (Tomanek et al., 2007). Until recently, most AL studies focused on simulating the annotation process by using already available gold standard data. In reality, however, human annotators make mistakes, leading to noise in the annotations. For this reason, some authors have questioned the applicability of AL to noisy annotation scenarios such as MTurk (Baldridge and Palmer, 2009; Rehbein et al., 2010).

AL and crowdsourcing are complementary approaches: AL reduces the number of annotations used while crowdsourcing reduces the cost per annotation. Combined, the two approaches could substantially lower the cost of creating training sets.

Our main contribution in this paper is that we show for the first time that AL is significantly better than randomly selected annotation examples in a real crowdsourcing annotation scenario. Our experiments directly address two tasks, named entity recognition and sentiment detection, but our

*Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1546–1556,
Edinburgh, Scotland, UK, July 27–31, 2011. ©2011 Association for Computational Linguistics

evidence suggests that AL is of general benefit in crowdsourcing. We also show that the effectiveness of MTurk annotation with AL can be further enhanced by using two techniques that increase label quality: *adaptive voting* and *fragment recovery*.

## 2 Related Work

### 2.1 Crowdsourcing

Pioneered by Snow et al. (2008), Crowdsourcing, especially using MTurk, has become a widely used service in the NLP community. A number of studies have looked at crowdsourcing for NER. Voyer et al. (2010) use a combination of expert and crowdsourced annotations. Finin et al. (2010) annotate Twitter messages – short sequences of words – and this is reflected in their vertically oriented user interface. Lawson et al. (2010) choose an annotation interface where annotators have to drag the mouse to select entities. Carpenter and Poesio (2010) argue that dragging is less convenient for workers than marking tokens.

These papers do not address AL in crowdsourcing. Another important difference is that previous studies on NER have used data sets for which no "linguistic" gold annotation is available. In contrast, we reannotate the CoNLL-2003 English NER dataset. This allows us to conduct a detailed comparison of MTurk AL to conventional expert annotation.

### 2.2 Active Learning with Noisy Labels

Hachey et al. (2005) were among the first to investigate the effect of actively sampled instances on agreement of labels and annotation time. They demonstrate applicability of AL when annotators are trained experts. This is an important result. However, AL depends on accurate assessments of uncertainty and informativeness and such an accurate assessment is made more difficult if labels are noisy as is the case in crowdsourcing. For this reason, the problem of AL performance with noisy labels has become a topic of interest in the AL community. Rehbein et al. (2010) investigate AL with human expert annotators for word sense disambiguation, but do not find convincing evidence that AL reduces annotation cost in a realistic (non-simulated) annotation scenario. Brew et al. (2010) carried out experiments on sentiment active learning through crowdsourcing. However, they use a small set of volunteer labelers instead of anonymous paid workers.

Donmez and Carbonell (2008) propose a method to choose annotators from a set of noisy annotators. However, in a crowdsourcing scenario, it is not possible to ask specific annotators for a label, as crowdsourcing workers join and leave the site. Furthermore, they only evaluate their approach in simulations. We use the actual labels of human annotators to avoid the risk of unrealistic assumptions when modeling annotators.

We are not aware of any study that shows that AL is significantly better than a simple baseline of having annotators annotate randomly selected examples in a highly noisy annotation setting like crowdsourcing. While AL generally is superior to this baseline in simulated experiments, it is not clear that this result carries over to crowdsourcing annotation. Crowdsourcing differs in a number of ways from simulated experiments: the difficulty and annotation consistency of examples drawn by AL differs from that drawn by random sampling; crowdsourcing labels are noisy; and because of the noisiness of labels statistical classifiers behave differently in simulated and real annotation experiments.

## 3 Annotation System

One fundamental design criterion for our annotation system was the ability to select examples *in real time* to support, e.g., the interactive annotation experiments presented in this paper. Thus, we could not use the standard MTurk workflow or services like CrowdFlower.[1]

We therefore designed our own system for annotation experiments. It consists of a two-tiered application architecture. The frontend tier is a web application that serves two purposes. First, the administrator can manage annotation experiments using a web interface and publish annotation tasks associated with an experiment on MTurk. The frontend also provides tools for efficient review of the received answers. Second, the frontend web application presents annotation tasks to MTurk workers. Because we wanted to implement interactive annotation experiments, we used the "external question"

[1] `http://crowdflower.com/`

feature of MTurk. An external question contains an URL to our frontend web application, which is queried when a worker views an annotation task. Our frontend then in turn queries our backend component for an example to be annotated and renders it in HTML.

The backend component is responsible for selection of an example to be annotated in response to a worker's request for an annotation task. The backend implements a diverse choice of random and active selection strategies as well as the multilabeling strategies described in section 3.2. The backend component runs as a standalone server and is queried by the frontend via REST-like HTTP calls.

For the NER task, we present one sentence per HIT, segmented into tokens, with a select box underneath each token containing the classes. The definition of the classes is based on the CoNLL-2003 annotation guidelines (Tjong Kim Sang and De Meulder, 2003). Examples were given for every class. Annotators are forced to make a selection for uppercase tokens. Lowercase tokens are prelabeled with "O" (no named entity), but annotators are encouraged to change this label if the token is in fact part of an entity phrase.

For sentiment annotation, we found in preliminary experiments that using simple radio button selection for the choice of the document label (`positive` or `negative`) leads to a very high amount of spam submissions, taking the overall classification accuracy down to around 55%. We then designed a template that forced annotators to type the label as well as a randomly chosen word from the text. Individual label accuracy was around 75% in this scheme.

### 3.1 Concurrent example selection

AL works by setting up an interactive annotation loop where at each iteration, the most informative example is selected for annotation. We use a pool-based AL setup where the most informative example is selected from a pool of unlabeled examples. Informativeness is calculated as uncertainty (Lewis and Gale, 1994) using the margin metric (Schein and Ungar, 2007). This metric chooses examples for which the margin of probabilities from the classifier between the two most probable classes is the smallest:

$$M_n = |\hat{P}(c_1|x_n) - \hat{P}(c_2|x_n)|$$

Here, $x_n$ is the instance to be classified, $c_1$ and $c_2$ are the two most likely classes, and $\hat{P}$ the classifier's estimate of probability.

For NER, the margins of the tokens are averaged to get an uncertainty assessment of the sentence. For sentiment, whole documents are classified, thus uncertainties can be used directly.

After annotation, the selected example is removed from the unlabeled pool and, together with its label(s), added to the set of labeled examples. The classifier is then retrained on the labeled examples and the informativeness of the remaining examples in the pool is re-evaluated.

Depending on the classifier and the sizes of pool and labeled set, retraining and reevaluation can take some time. To minimize wait times, traditional AL implementations select examples in batches of the $n$ most informative examples. However, batch selection might not give the optimum selection (examples in a batch are likely to be redundant, see Brinker (2003)) and wait times can still occur between one batch and the next.

When performing annotation with MTurk, wait times are unacceptable. Thus, we perform the retraining and uncertainty rescoring concurrently with the annotation user interface. The unlabeled pool is stored in a priority queue that is ordered according to the examples' informativeness. The annotation user interface takes the most informative example from the pool and presents it to the annotator. The labeled example is then inserted into a second queue that feeds and updates retraining and rescoring processes. The pool queue then is resorted according to the new informativeness. In this way, annotation and example selection can run in parallel. This is similar to Haertel et al. (2010).

### 3.2 Adaptive voting and fragment recovery

MTurk labels often have a high error rate. A common strategy for improving label quality is to acquire *multiple labels* by different workers for each example and then consolidate the annotations into a single label of higher quality. To trade off number of annotated examples against quality of annotations, we adopt *adaptive voting*. It uses majority

| Budget | | | NER | | | | | | Sentiment | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 5820 | | | | 6931 | | 1130 | | | | 1756 | |
| | | | #train | $F_1$ | cost/sent | w.-accuracy | #train | $F_1$ | #train | Acc | cost/doc | w.-accuracy | #train | Acc |
| RS | 1 | S | 5820 | 59.6 | 1.00 | 51.6 | – | – | 1130 | 70.4 | 1 | 74.8 | – | – |
| | 2 | 3-v | 1624 | 61.4† | 3.58 | 70.1 | – | – | – | – | – | – | – | – |
| | 3 | 5/4-v | 1488 | 63.0† | 3.91 | 71.6 | 1774 | 63.5 | 450 | 71.2 | 2.51 | 89.6 | 735 | 79.2 |
| | 4 | 5-v+f | 1996 | 63.6† | 2.91 | 71.8 | 2385 | 64.9† | – | – | – | – | – | – |
| AL | 5 | S | 5820 | 67.0 | 1.00 | 66.5 | – | – | 1130 | 74.8 | 1 | 76.0 | – | – |
| | 6 | 3-v | 1808 | 70.0† | 3.21 | 78.8 | – | – | – | – | – | – | – | – |
| | 7 | 5/4-v | 1679 | 70.4† | 3.46 | 79.6 | 1966 | 70.6 | 455 | 77.4 | 2.48 | 89.0 | 715 | 81.8 |
| | 8 | 5-v+f | 2165 | 70.5 | 2.68 | 79.3 | 2691 | 71.2 | – | – | – | – | – | – |

Table 1: For NER, active learning consistently beats random sampling on MTurk. NER $F_1$ evaluated on CoNLL test set A. #train = number of sentences in training set, S = single, 3-v = 3-voting, 5/4-voting = 5- and 4-voting for NER and sentiment resp., +f = using fragments; sentiment budget 1130 for run 1, sentiment budget 1756 averaged over 2 runs.

voting and is adaptive in the number of repeated annotations. For NER, a sentence is first annotated by two workers. Then majority voting is performed for each token individually. If there is a majority for every token that is greater than an agreement threshold $\alpha$, the sentence is accepted with each token labeled with the majority label. Otherwise additional annotations are requested. A sentence is discarded if the number of repeated annotations exceeds a discard threshold $d$ ($d$-voting).[2] We use the same scheme for sentiment; note that there is just one decision per HIT in this case, not several as in NER.

For NER, we also use *fragment recovery*: we salvage tokens with agreeing labels from discarded sentences. We cut the token sequence of a discarded sentence into several fragments that have agreeing tokens and discard only those parts that disagree. We then include these recovered fragments in the training data just like complete sentences.

**Software release.** Our active learning framework used can be downloaded at `http://www.ims.uni-stuttgart.de/˜lawsfn/active/`.

## 4 Experiments, Results and Analysis

### 4.1 Experiments

In our NER experiments, we have workers reannotate the English corpus of the CoNLL-2003 NER shared task. We chose this corpus to be able to compare crowdsourced annotations with gold standard annotations. A HIT is one sentence and is offered for a base payment of $0.01. We filtered out answers that contained unannotated tokens or were obvious spam (e.g., all tokens labeled as MISC). For testing NER performance, we used a system based on conditional random fields with standard named entity features including the token itself, orthographic features like the occurrence of capitalization or special characters and context information about the tokens to the left/right of the current token.

The sentiment detection task was modeled after a well-known document analysis setup for sentiment classification, introduced by Pang et al. (2002). We use their corpus of 1000 positive and 1000 negative movie reviews and the Stanford maximum entropy classifier (Manning and Klein, 2003) to predict the sentiment label of each document $d$ from a unigram representation of $d$. We randomly split this corpus into a test set of 500 reviews and an active learning pool of 1500 reviews. Each HIT consists of one document, valued at $0.01.

We compare random sampling (RS) and AL in combination with the proposed voting and fragment strategies with different parameters. We want to avoid rerunning experiments on MTurk over and over again, but on the other hand, we believe that using synthetic data for simulations is problematic because it is difficult to generate synthetic data with a realistic model of annotator errors. Thus, we logged a play-by-play record of the annotator interactions and labels. With this recording, we can then rerun strategies with different parameters.

We chose voting with at most $d = 5$ repetitions as

[2] It can take a while in this scheme for annotators to agree on a final annotation for a sentence. We make *tentative* labels of a sentence available to the classifier immediately and replace them with the final labels once voting is completed.

our main reannotation strategy for both random and active sampling for NER annotation. We use simple majority voting ($\alpha = .5$) for NER.

For sentiment, we set $d = 4$ and minimum agreement $\alpha = .75$ because the number of labels is smaller (2 vs. 5) and so random agreement is more likely for sentiment.

To get results for 3-voting NER, we take the recording and discard 5-voting votes not needed in 3-voting. This will result in roughly the same number of annotated sentences, but at a lower cost. This simulation of 3-voting is not exactly what would have happened on MTurk (e.g., the final vote on a sentence might be different, which then influences AL example selection), but we will assume that differences are rare and simulated and actual results are similar. The same considerations apply to single votes and to the sentiment experiments.

We always compare two strategies for the same annotation budget. For example, the *number of training sentences* in Table 1 differ in the two relevant columns, but all strategies compared use exactly *the same annotation budget* (5820, 6931, 1130, and 1756, respectively).

For the single annotation strategy, each interaction record contained only about 40% usable annotations, the rest were repeats. A comparison with the single annotation strategy over approx. 2000 sentences or 450 documents would not have been meaningful; therefore we chose to run an extra experiment with the single annotation strategy to match this up with the budgets of the voting strategies. The results are presented in two separate columns of Table 1 (budgets 6931 and 1756).

### 4.2 Results

For sentiment detection, *worker accuracy* or *label quality* – the percentage of correctly annotated documents – is 74.8. In contrast, for NER, worker accuracy – the percentage of non-O tokens annotated correctly – is only 51.6 (Table 1, line 1). This demonstrates the challenge of using MTurk for NLP annotation tasks. When we use single annotations of each sentence, NER performance is 59.6 $F_1$ for random sampling (line 1). When training with gold labels on the same sentences, the performance is 80.0 (not shown). This means we lose more than 20% due to poor worker accuracy. Adaptive voting and fragment recovery manage to recover a small part of the lost performance (lines 2–4); each of the three $F_1$ scores is significantly better than the one above it as indicated by † (Approximate Randomization Test (Noreen, 1989; Chinchor et al., 1993) as implemented by Padó (2006)).

Using AL turns out to be quite successful for NER performance. For single annotations, NER performance is 67.0 (line 5), an improvement of 7.4% compared to random sampling. Adaptive voting and fragment recovery again increase worker accuracy (lines 6–8) although total improvement of 3.5% (lines 8 vs. 5) is smaller than 4% for random (lines 4 vs. 1). The learning curves of AL vs. random in Figure 1 (top left) confirm this good result for AL. These learning curves are for tokens – not for sentences – to show that the reason for AL's better performance is not that it selects slightly longer sentences than random. In addition, the relative advantage of AL vs random decreases over time, which is typical of pool-based AL experiments.

We carried out two runs of the same experiment for sentiment to validate our first positive result since the difference between the two conditions is not as large as in NER (Figure 1, top right). After about 300 documents, active learning consistently outperforms random sampling. The first AL run performs better because of higher label quality in the beginning. The overall advantage of AL over random is lower than for NER because the set of labels is smaller in sentiment, making the classification task easier. Second, there is a large amount of simple lexical clues for detecting sentiment (cf. Wilson et al. (2005)). It is likely that some of them can be learned well through random sampling at first; however, active learning can gain accuracy over time because it selects examples with more difficult clues.

In Figure 1 (bottom), we compare single annotation with adaptive voting. The graphs show $F_1$ as a function of cost. Adaptive voting trades quantity of sampled sentences for quality of labels and thus incurs higher net costs per sentence. This results in a smaller dataset for a given budget, but this dataset is still more useful for classifier training. For NER (Figure 1, bottom left), the single annotation strategy has a faster start; so for small budgets, covering a somewhat larger portion of the sample space is beneficial. For larger budgets, however, quality of

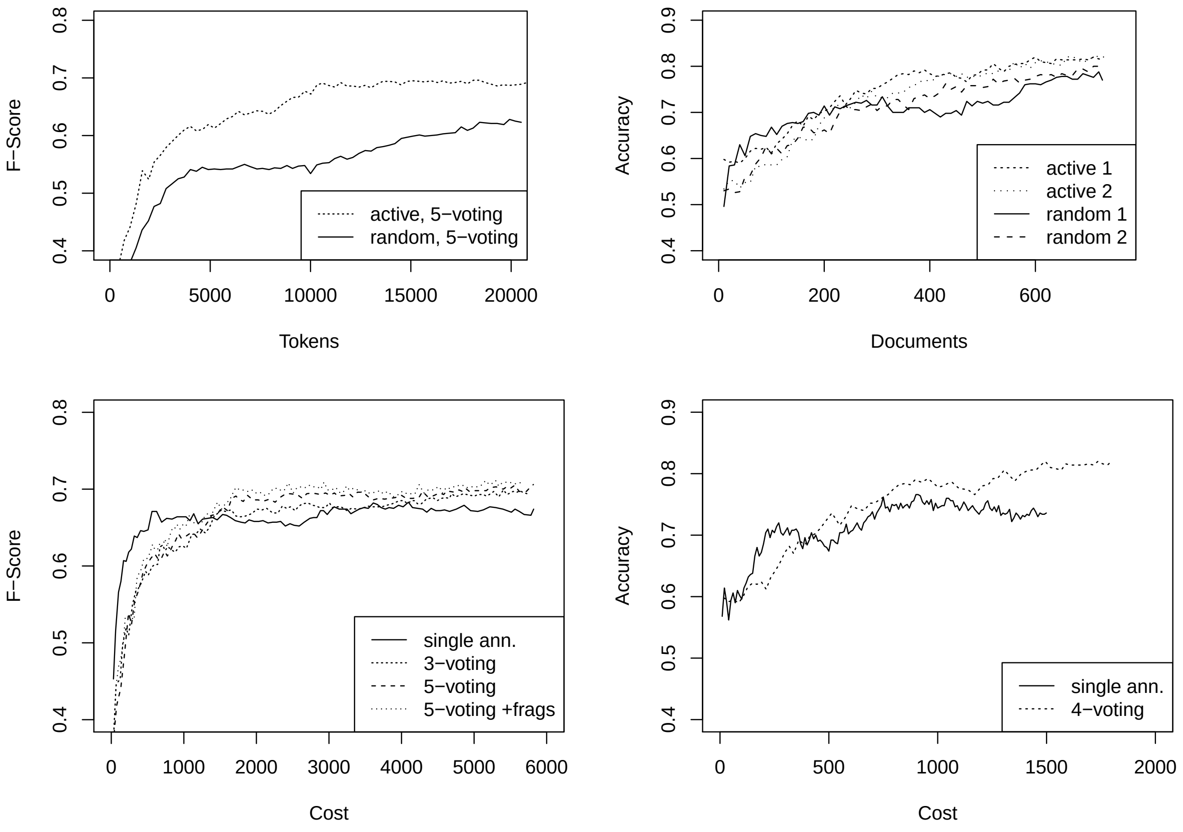

Figure 1: Top: Active learning vs. Random sampling for NER (left) and sentiment (right). Bottom: Active learning: adaptive voting vs. single annotation for NER (left) and sentiment (right).

the voted labels trumps quantity.

For sentiment (Figure 1, bottom right), results are similar: voting has no benefit initially, but as finding maximally informative examples to annotate becomes harder in later stages of learning, adaptive voting gains an advantage over single annotations.

The main result of the experiment is that active learning is better by about 7% $F_1$ than random sampling for NER and by 2.6% accuracy for sentiment (averaged over two runs at budget 1756). Adaptive voting further improves AL performance for both NER and sentiment.

### 4.3 Annotation time per token

Most AL work assumes constant cost per annotation unit. This assumption has been questioned because AL often selects hard examples that take longer to annotate (Hachey et al., 2005; Settles et al., 2008).

In annotation with MTurk, cost is not a function of annotation time because workers are paid a fixed amount per HIT. Nevertheless, annotation time plays a part in whether workers are willing to work on a given task for the offered reward. This is particularly problematic for NER since workers have to examine each token individually. We therefore investigate for NER whether the time MTurk workers spend on annotating sentences differs for random vs. AL.

We first compute median and mean annotation times and number of tokens per sentence:

| | sec/sentence | | tokens/sentence | |
|---|---|---|---|---|
| strategy | median | mean | all | required |
| random | 17.2 | 33.1 | 15.0 | 3.4 |
| AL | 17.8 | 33.0 | 17.7 | 4.0 |

We see that most sentences are annotated in a very short time; but the mean is much larger than the median because there are outliers of up to eight minutes. AL tends to select slightly longer sentences as

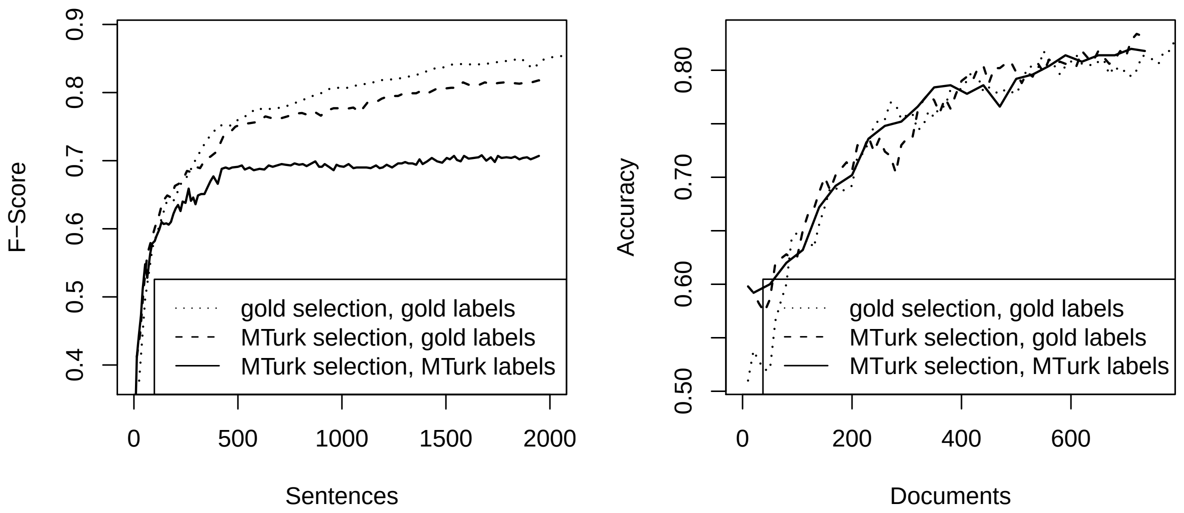

Figure 3: Performance on gold labels. Left: NER. Right: sentiment (run 1).

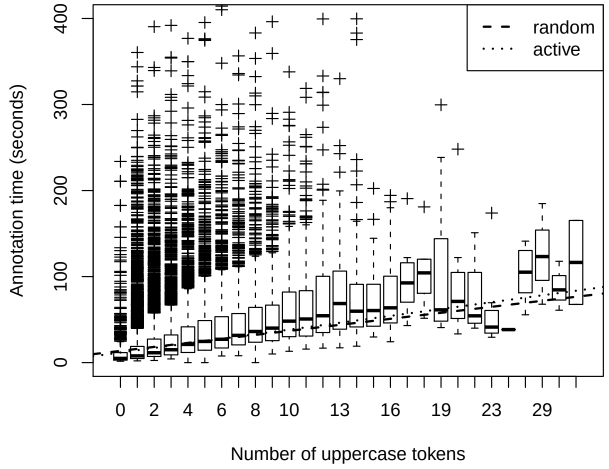

Figure 2: Annotation time vs. # uppercase tokens

well as sentences with slightly more uppercase tokens that require annotation.

In a more detailed analysis, we attempt to distinguish between (i) the effect of more uppercase ("annotation required") tokens vs. (ii) the effect of example difficulty. We fit a linear regression model to annotation time vs. the number of uppercase tokens. For the regression fit, we removed all annotation times $> 60$ seconds. Such long times indicate distraction of the worker and are not a reliable measure of difficulty.

Figure 2 shows the distribution of annotation times for both cases combined and the fitted models for each. The model estimated an annotation time of 2.3 secs for each required token for random vs. 2.7 secs for AL. We conclude that the difference in difficulty between sentences selected by random sampling vs. AL is small, but noticeable.

### 4.4 Influence of noise on the selection process

While NER performance for AL is much higher than for random sampling, it is still quite a bit lower than what is possible on gold labels. In the case of AL, there are two reasons why this happens: (i) The noisy labels negatively affect the classifier's ability to learn a good model that is used for classifying the test set. (ii) The noisy labels result in bad intermediate models that then select suboptimal examples to be annotated next. The AL selection process is "misled" by the noisy examples.

We conduct an experiment to determine the contribution of factors (i) and (ii) to the performance loss. First, we preserve the sequence of sentences chosen by our AL experiments on MTurk, with $5$-voting for NER and $4$-voting for sentiment but replace the noisy worker-provided labels by gold labels. The performance of classifiers trained on this sequence is the dashed line "MTurk selection, gold labels" in Figure 3 for NER (left) and sentiment (right).

Second, we compare with a traditional simulated AL experiment with gold labels. Here, the selection too is controlled by gold labels, so the selection has a noiseless classifier available for scoring and can perform optimal uncertainty selection. These are the

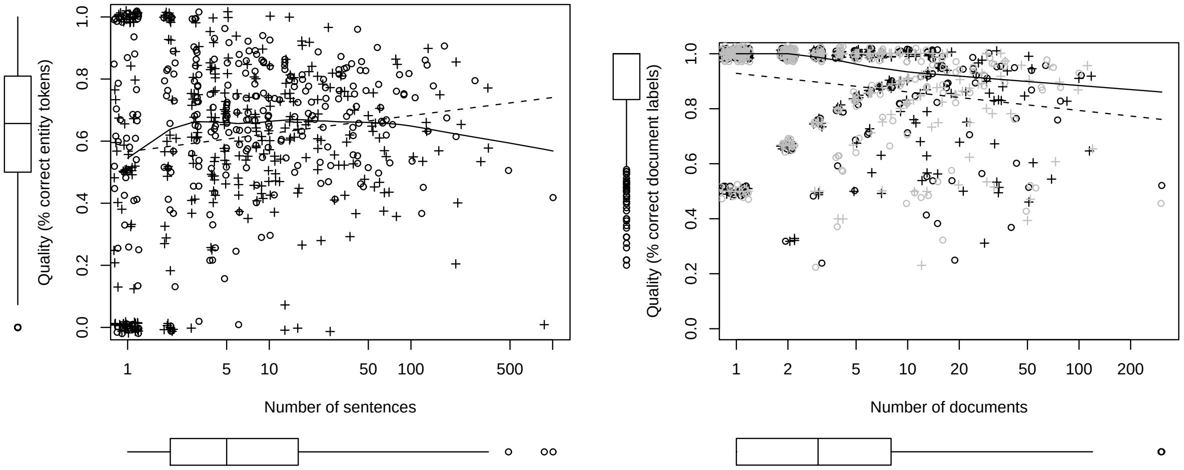

Figure 4: Worker accuracy vs. number of HITs. Each point corresponds to one worker (○ = active, + =random sampling; black and grey for different runs). Left: NER. Right: Sentiment.

dotted lines "gold selection, gold labels" in Figure 3.

We used a batch-mode AL setup for this comparison experiment. For a fair comparison, we adjust the batchsize to be equal to the average *staleness* of a selected example in concurrent MTurk active learning. The staleness of an example is defined as the number of annotations the system has received, but not yet incorporated in the computation of an example's uncertainty score (Haertel et al., 2010).

For our concurrent NER system, the average staleness of an example was about 12 (min: 1, max: 40), for sentiment it was about 2. The figure for NER is higher than the number cited by Haertel et al. (2010) because there are more annotators accessing our system at the same time via MTurk but not as high for sentiment since documents are longer and retraining the sentiment classifier is faster. The average staleness of an example in a batch-mode system is half the batch size. Thus, we set the batch size of our comparison system to 25 for NER and to 4 for sentiment.

Returning to the two factors introduced above – (i) final effect of noise on test set performance vs. (ii) intermediate effect of noise on example selection – we see in Figure 3 that (i) has a large effect on NER whereas (ii) has a noticeable, but small effect.[3] For example, at 1966 sentences, $F_1$ scores are 70.6 (MTurk-MTurk), 81.4 (MTurk-gold) and 84.9 (gold-gold). This means that a performance difference of 10 points $F_1$ has to be attributed to noisy labels resulting in a worse final classifier (effect i), and another $3.5$ points are lost due to sub-optimal example selection (effect ii).

For sentiment, the results are different. There is no clear difference between the three runs. We attribute this to the fact that the quality of the labels is higher in sentiment than in NER. Our initial experiments on sentiment were all negative (showing no improvement of AL compared to random) because label quality was too low. Only after we introduced the template described in Section 3 and used 4-voting with $\alpha = .75$ did we get positive results for AL. This leads to an overall label quality of about 90% (over all runs) which is so high that the difference to using gold labels is small if present at all.

## 5 Worker Quality

So far we have assumed that all workers provide annotations of the same quality. However, this is not the case. Figure 4 shows plots of worker accuracy as a function of worker productivity (number of annotated examples). Some workers submit only one or two HITs just to try out the task. For NER, the majority of workers submit between 5 and 10 sentences, with label qualities between 0.5 and 0.8. The chance level for correctness is around 0.25 (four

[3] Our comparison unit for NER is the sentence. We cannot compare on cost here since we do not know what the per-sentence cost of a "gold" expert annotation is.

different named entity categories for uppercase tokens). For sentiment, most workers submit 1 to 5 documents, with label qualities between 0.5 and 1. Chance level lies at around 0.5 (for two equally distributed labels).

While quality for highly productive workers is mediocre in our experiments, other researchers have found extremely bad quality for their most prolific workers (Callison-Burch, 2009). Some of these workers might be spammers who try to submit answers with automatic scripts. We encountered some spammers that our heuristics did not detect (shown in the bottom-right areas of Figure 4, left), but the voting mechanism was able to mitigate their negative influence.

Given the large variation in Figure 4, using worker quality in crowdsourcing for improved training set creation seems promising. We now test two such strategies for NER in an oracle setup.

### 5.1 Blocking low-quality workers

A simple approach is to refuse annotations from workers that have been determined to provide low quality answers. We simulated this strategy on NER data using oracle quality ratings. We chose NER because of its lower overall label quality. The results are presented in Figure 5 for random (a) and AL (b). For random, quality filtering with low cutoffs helps by removing bad annotations that likely come from spammers. While the voting strategy prevented a performance decrease with bad annotations, it needed to expend many extra annotations for correction. With filtering, these extra annotations become unnecessary and the system can learn faster. When low-quality workers are less active, as in the AL dataset, we find no meaningful performance increase for low cutoffs up to 0.4. For very high cutoffs (0.7), the beginning of the performance curve shows that further cost reductions can be achieved. However, we did not have enough recorded human annotations available to perform a simulation for the full budget.

### 5.2 Trusting high-quality workers

The complementary approach is to take annotations from highly rated workers at face value and immediately accept them as the correct label, *bypassing* the voting procedure. Bypassing saves the cost of repeated annotation of the same sentence. Figure 5 shows learning curves for two bypass thresholds on worker quality (measured as proportion of correct non-O tokens) for random (c) and AL (d). Bypassing performs surprisingly well. We find a steeper rise of the learning curve, meaning less cost for the same performance. Not only do we find substantial cost reductions, but also higher overall performance. We believe this is because high-quality annotations can sometimes be voted down by other annotations. If we can identify high-quality workers and directly use their annotations, this can be avoided.

These experiments are oracle experiments using gold data that is normally not available. In future work, we would like to repeat the experiments using methods for worker quality estimation (Ipeirotis et al., 2010; Donmez et al., 2009). For AL, the choice as to which labels are used (as a result of voting, bypassing or other) also has an influence on the selection. However, we had to keep the sequence of the selected sentences fixed in the simulations reported above. While our method of sample selection for AL proved to be quite robust even in the presence of noise, higher quality labels do have an influence on the sample selection (see section 4.4), so the improvement could be even better than indicated here.

### 5.3 Differences in quality between AL and random

The essence of AL is to select examples that are difficult to classify. As observed in our experiments on annotation time, this difficulty is reflected in the amount of time a human needs to work on examples selected through AL. Another effect to expect from difficulty could be lower annotation accuracy. We therefore examined the accuracies for each worker who contributed to both the AL and the random experiment. We found that in the NER task, the 20 workers in this group had a slightly higher (0.07) average quality for randomly selected examples. This difference is low and does not suggest a significant drop in accuracy for examples selected in AL.

## 6 Conclusion

We have investigated the use of AL in a real-life annotation experiment with human annotators instead of traditional simulations with gold labels for

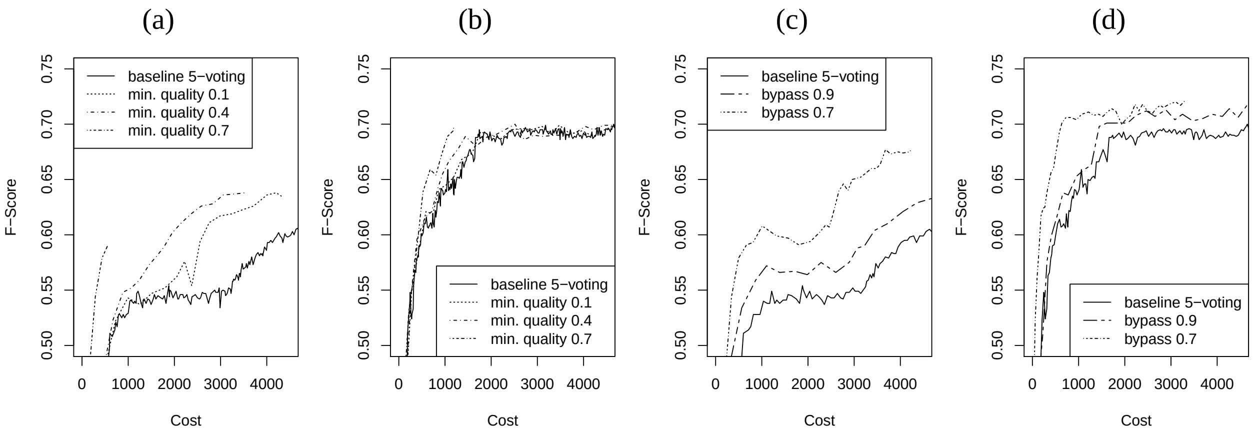

Figure 5: Blocking low-quality workers: (a) random, (b) AL. Bypass voting: (c) random, (d) AL.

named entity recognition and sentiment classification. The annotation was performed using MTurk in an AL framework that features concurrent example selection without wait times. We also evaluated two strategies, adaptive voting and fragment recovery, to improve label quality at low additional cost. We find that even for the relatively high noise levels of annotations gathered with MTurk, AL is successful, improving performance by +6.9 points $F_1$ compared to random sampling for NER and by +2.6% accuracy for sentiment. Furthermore, this performance level is reached at a smaller MTurk cost compared to random sampling. Thus AL not only reduces annotation costs, but also offers an improvement in absolute performance for these tasks. This is clear evidence that active learning and crowdsourcing are complementary methods for lowering annotation cost and should be used together in training set creation for natural language processing tasks.

We have also conducted oracle experiments that show that further performance gains and cost savings can be achieved by using information about worker quality. We plan to confirm these results by using estimates of quality in the future.

## 7 Acknowledgments

Florian Laws is a recipient of the Google Europe Fellowship in Natural Language Processing, and this research is supported in part by his fellowship. Christian Scheible is supported by the Deutsche Forschungsgemeinschaft project Sonderforschungsbereich 732.

## 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*, pages 296–305.

Anthony Brew, Derek Greene, and Pádraig Cunningham. 2010. Using crowdsourcing and active learning to track sentiment in online media. In *Proceeding of the 2010 conference on ECAI 2010: 19th European Conference on Artificial Intelligence*, pages 145–150.

Klaus Brinker. 2003. Incorporating diversity in active learning with support vector machines. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML 2003)*, pages 59–66.

Chris Callison-Burch. 2009. Fast, cheap, and creative: evaluating translation quality using Amazon's Mechanical Turk. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 286–295.

Bob Carpenter and Massimo Poesio. 2010. Models of data annotation. Tutorial at the seventh international conference on Language Resources and Evaluation (LREC 2010).

Nancy Chinchor, David D. Lewis, and Lynette Hirschman. 1993. Evaluating message understanding systems: an analysis of the third message understanding conference (muc-3). *Computational Linguistics*, 19(3):409–449.

Pinar Donmez and Jaime G. Carbonell. 2008. Proactive learning: cost-sensitive active learning with multiple imperfect oracles. In *Proceeding of the 17th ACM conference on Information and knowledge management*, pages 619–628.

Pinar Donmez, Jaime G. Carbonell, and Jeff Schneider. 2009. Efficiently learning the accuracy of labeling sources for selective sampling. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 259–268.

Tim Finin, William Murnane, Anand Karandikar, Nicholas Keller, Justin Martineau, and Mark Dredze. 2010. Annotating named entities in twitter data with crowdsourcing. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 80–88.

Ben Hachey, Beatrice Alex, and Markus Becker. 2005. Investigating the effects of selective sampling on the annotation task. In *CoNLL '05: Proceedings of the 9th Conference on Computational Natural Language Learning*, pages 144–151.

Robbie Haertel, Paul Felt, Eric K. Ringger, and Kevin Seppi. 2010. Parallel active learning: Eliminating wait time with minimal staleness. In *Proceedings of the NAACL HLT 2010 Workshop on Active Learning for Natural Language Processing*, pages 33–41.

Panagiotis G. Ipeirotis, Foster Provost, and Jing Wang. 2010. Quality management on amazon mechanical turk. In *Proceedings of the ACM SIGKDD Workshop on Human Computation (HCOMP '10)*.

Nolan Lawson, Kevin Eustice, Mike Perkowitz, and Meliha Yetisgen-Yildiz. 2010. Annotating large email datasets for named entity recognition with mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 71–79.

David D. Lewis and William A. Gale. 1994. A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12.

Christopher Manning and Dan Klein. 2003. Optimization, maxent models, and conditional estimation without magic. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Tutorials - Volume 5*, pages 8–8.

Eric W. Noreen. 1989. *Computer-intensive methods for testing hypotheses: an introduction*. Wiley.

Miles Osborne and Jason Baldridge. 2004. Ensemble-based active learning for parse selection. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 89–96.

Sebastian Padó, 2006. *User's guide to `sigf`: Significance testing by approximate randomisation*.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 79–86.

Ines Rehbein, Josef Ruppenhofer, and Alexis Palmer. 2010. Bringing active learning to life. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 949–957.

Eric Ringger, Peter McClanahan, Robbie Haertel, George Busby, Marc Carmen, James Carroll, Kevin Seppi, and Deryle Lonsdale. 2007. Active learning for part-of-speech tagging: Accelerating corpus annotation. In *Proceedings of the Linguistic Annotation Workshop at ACL-2007*, pages 101–108.

Andrew Schein and Lyle Ungar. 2007. Active learning for logistic regression: An evaluation. *Machine Learning*, 68(3):235–265.

Burr Settles, Mark Craven, and Lewis Friedland. 2008. Active learning with real annotation costs. In *Proceedings of the NIPS Workshop on Cost-Sensitive Learning*, pages 1069–1078.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Ng. 2008. Cheap and fast – but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 254–263.

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 (CoNLL 2003)*, pages 142–147.

Katrin Tomanek, Joachim Wermter, and Udo Hahn. 2007. An approach to text corpus construction which cuts annotation costs and maintains reusability of annotated data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 486–495.

Simon Tong and Daphne Koller. 2002. Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, 2:45–66.

Robert Voyer, Valerie Nygaard, Will Fitzgerald, and Hannah Copperman. 2010. A hybrid model for annotating named entity training corpora. In *Proceedings of the Fourth Linguistic Annotation Workshop*, pages 243–246.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 347–354.