Subsequence Based Deep Active Learning for Named Entity Recognition

Puria Radmard[1,2,3] pr450@cam.ac.uk
Yassir Fathullah[2] yf286@cam.ac.uk
Aldo Lipani[1,3] aldo.lipani@ucl.ac.uk

[1]University College London
[2]University of Cambridge
[3]Vector AI

Abstract

Active Learning (AL) has been successfully applied to Deep Learning in order to drastically reduce the amount of data required to achieve high performance. Previous works have shown that lightweight architectures for Named Entity Recognition (NER) can achieve optimal performance with only 25% of the original training data. However, these methods do not exploit the sequential nature of language and the heterogeneity of uncertainty within each instance, requiring the labelling of whole sentences. Additionally, this standard method requires that the annotator has access to the full sentence when labelling. In this work, we overcome these limitations by allowing the AL algorithm to query subsequences within sentences, and propagate their labels to other sentences. We achieve highly efficient results on OntoNotes 5.0, only requiring 13% of the original training data, and CoNLL 2003, requiring only 27%. This is an improvement of 39% and 37% compared to querying full sentences.

1 Introduction

The availability of large datasets has been key to the success of deep learning in Natural Language Processing (NLP). This has galvanized the creation of larger datasets in order to train larger deep learning models. However, creating high quality datasets is expensive due to the sparsity of natural language, our inability to label it efficiently compared to other forms of data, and the amount of prior knowledge required to solve certain annotation tasks. Such a problem has motivated the development of new Active Learning (AL) strategies which aim to efficiently train models, by automatically identifying the best training examples from large amounts of unlabeled data (Wei et al., 2015; Wang et al., 2017; Tong and Koller, 2002). This tremendously reduces human annotation effort as much fewer instances need to be labeled manually.

To minimise the amount of data needed to train a model, AL algorithms iterate between training a model, and querying information rich instances to human annotators from a pool of unlabelled data (Huang et al., 2014). This has been shown to work well when the queries are 'atomic'—a single annotation requires a unit labour, and describes entirely the instance to be annotated. Conversely, each instance of structured data, such as sequences, require multiple annotations. Hence, such query selection methods can result in a waste of annotation budget (Settles, 2011).

For example, in Named Entity Recognition (NER), each sentence is usually considered an instance. However, because each token has a separate label, annotation budgeting is typically done on a token basis (Shen et al., 2017). Budget wasting may therefore arise from the heterogeneity of uncertainty across each sentence; a sentence can contain multiple subsequences (of tokens) of which the model is certain on some and uncertain on others. By making the selection at a sentence level, although some budget is spent on annotating uncertain subsequences, the remaining budget may be wasted on annotating subsequences for which an annotation is not needed.

It can therefore be desirable for annotators to label subsequences rather than the full sentences. This gives a greater flexibility to AL strategies to locate information rich parts of the input with improved efficiency – and reduces the cognitive demands required of annotators. Annotators may in fact perform better if they are asked to annotate shorter sequences, because longer sentences can cause boredom, fatigue, and inaccuracies (Rzeszotarski et al., 2013).

Code is made available on: https://github.com/puria-radmard/RFL-SBDALNER

In this work, we aim to improve upon the efficiency of AL for NER by querying for subsequences within each sentence, and propagating labels to unseen, identical subsequences in the dataset. This strategy simulates a setup in which annotators are presented with these subsequences, and do not have access to the full context, ensuring that their focus is centred on the tokens of interest.

We show that AL algorithms for NER tasks that use subsequences, allowing training on partially labelled sentences, are more efficient in terms of budget than those that only query full sentences. This improvement is furthered by generalising existing acquisition functions (§ 4.1) for use with sequential data. We test our approaches on two NER datasets, OntoNotes 5.0 and CoNLL 2003. On OntoNotes 5.0, Shen et al. (2017) achieve state-of-the-art performance with 25% of the original dataset querying full sentences, while we require only 13% of the dataset querying subsequences. On CoNLL 2003, we show that the AL strategy of Shen et al. (2017) requires 50% of the dataset to achieve the same results as training on the full dataset, while ours requires only 27%.

Contributions of this paper are:

1. Improving the efficiency of AL for NER by allowing querying of subsequences over full sentences;
2. An entity based analysis demonstrating that subsequence querying AL strategies tend to query more relevant tokens (i.e., tokens belonging to entities);
3. An uncertainty analysis of the queries made by both full sentence and subsequence querying methods, demonstrating that querying full sentences leads to selecting more tokens to which the model is already certain.

2 Related Work

AL algorithms aim to query information rich data points to annotators in order to improve the performance of the model in a data efficient way. Traditionally these algorithms choose data points which lie close to decision boundaries (Pinsler et al., 2019), where uncertainty is high, in order for the model to learn more useful information. This measure of uncertainty, measured through acquisition functions, are therefore vital to AL. Key functions include predictive entropy (MaxEnt) (Gal et al., 2017), mutual information between model posterior and predictions (BALD) (Houlsby et al., 2011; Gal et al., 2017), or the certainty of the model when making label predictions (here called LC) (Mingkun Li and Sethi, 2006). These techniques ensure all instances used for training, painstakingly labelled by experts, have maximum impact on model performance. There has been exploration of uncertainty and deep learning based AL for NER (Chen et al., 2015; Shen et al., 2017; Settles and Craven, 2008; Fang et al., 2017). These approaches however, treat each sentence as a single query instead of a collection of individually labelled tokens. In these methods, the acquisition functions that score sentences aggregate token-wise scores (through summation or averaging).

Other works forgo this aggregation, querying single tokens at a time (Tomanek and Hahn, 2009; Wanvarie et al., 2011; Marcheggiani and Artières, 2014). These works show that AL for NER can be improved by taking the single token as a unit query, and use semi-supervision (Reddy et al., 2018; Iscen et al., 2019) for training on partially labelled sentences (Muslea et al., 2002). However, querying single-tokens is inapplicable in practise because, either a) annotators have access to the full sentence when queried but can only label one token, which would lead to frustration as they are asked to read the full sentence but only annotate a single token, or b) annotators only have access to the token of interest, which means that they would not have enough information to label tokens differently based on their context, leading to annotators labeling any unique token with the same label. Moreover, if the latter approach was somehow possible, we would be able to reduce the annotation effort to the annotation of only the unique tokens forming the dataset, its dictionary. Furthermore, all of these past works use Conditional Random Fields (CRFs) (Lafferty et al., 2001), which have since been surpassed as the state-of-the-art for NER (and most NLP tasks) by deep learning models (Devlin et al., 2019).

In this work we follow the approach where annotators only have access to subsequences of multiple tokens. However, instead of making use of single tokens, we will query more than one token, providing enough context to the annotators. This allows the propagation of these annotations to identical subsequences in the dataset, further reducing the total annotation effort.

3 Background

3.1 Active Learning Algorithms

Most AL strategies are based on a repeating score, query and fine-tune cycle. After initially training an NER model with a small pool of labelled examples, the following is repeated: (1) score all unlabelled instances, (2) query the highest scoring instances and add them to training set, and, (3) fine-tune the model using the updated training set (Huang et al., 2014).

To describe this further, notation and proposed training process is introduced, with details in following sections. First, the sequence tagging dataset, denoted by $\mathcal{D} = \{(\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)})\}_{n=1}^{N}$, consists of a collection of sentence and ground truth labels. The i-th token of the n-th sentence ($y_i^{(n)}$) has a label $y_i^{(n)} = c$ with c belonging to $\mathcal{C} = \{c_1, ..., c_K\}$. We also differentiate between the labelled and unlabelled datasets, $\mathcal{D}_L$ and $\mathcal{D}_U$, which initially are empty and equal to $\mathcal{D}$. Finally, we fix A as the total number of tokens queried in each iteration.

3.2 Acquisition Functions

Instances in the unlabelled pool are queried using an acquisition function. This function aims to quantify the uncertainty of the model when generating predictive probabilities over possible labels for each instance. Instances with the highest predictive uncertainty are deemed as the most informative for model training. Previously used acquisition functions such as Least Confidence (LC) and Maximum Normalized Log-Probability (MNLP) (Shen et al., 2017; Chen et al., 2015) are generalised for variable length sequences. Letting $\hat{\boldsymbol{y}}_{<i}^{(n)}$ be the history of predictions prior to the i-th input, the next output probability will be $p_{i,c}^{(n)} = P(\hat{y}_i^{(n)} = c | \hat{\boldsymbol{y}}_{<i}^{(n)}, \boldsymbol{x}^{(n)})$. Then, we define the token-wise LC score as:

$$\texttt{LC}_i^{(n)} = -\max_{c\in\mathcal{C}} \log p_{i,c}^{(n)}. \tag{1}$$

The LC acquisition function for sequences is then defined as:

$$\texttt{LC}\left(x_1^{(n)}, ..., x_\ell^{(n)}\right) = \sum_{j=1}^{\ell} \texttt{LC}_j^{(n)}, \tag{2}$$

and, for MNLP as:

$$\texttt{MNLP}\left(x_1^{(n)}, ..., x_\ell^{(n)}\right) = \frac{1}{\ell}\sum_{j=1}^{\ell} \texttt{LC}_j^{(n)}. \tag{3}$$

Note that this is similar to LC except for the normalization factor $1/\ell$. The formulation above can be applied to other types of commonly used acquisition functions such as Maximum Entropy (MaxEnt) (Gal et al., 2017) by simply defining:

$$\texttt{ME}_i^{(n)} = -\sum_{c\in\mathcal{C}} p_{i,c}^{(n)} \log p_{i,c}^{(n)}, \tag{4}$$

as the token score. Given the task of quantifying uncertainty amongst the unlabelled pool of data, both of these metrics - LC and MaxEnt - provide intuitive interpretations. eq. (1) scores highly tokens for which the predicted label has lowest confidence, while eq. (4) scores highly tokens for which the whole probability mass function has higher entropy. Both of these therefore score more highly uniform predictive distributions, which indicates underlying uncertainty.

Finally, given the similarity of performance between MNLP and Bayesian Active Learning by Disagreement (BALD) (Houlsby et al., 2011) in NER tasks (Shen et al., 2017), and the computational complexity required to calculate BALD with respect to the other activation functions, we will not compare against BALD.

4 Subsequence Acquisition

In this section we describe how we build on past works, and the core contribution of this paper. Our work forms a more flexible AL algorithm that operates on subsequences, as opposed to full sentences (Shen et al., 2017). This is achieved by generalising acquisition functions for subsequences (§ 4.1) scoring and querying subsequences within sentences (§ 4.2), and performing label propagation on unseen sentences to avoid the multiple annotations of repeated subsequences (§ 4.3).

4.1 Subsequence Acquisition Functions

Since this work focuses on the querying of subsequences, from the previously defined LC and MNLP we generalize them to define a family of acquisition functions applicable for both full sentences and subsequences:

$$\texttt{LC}_\alpha\left(x_{i+1}^{(n)}, ..., x_{i+\ell}^{(n)}\right) = \frac{1}{\ell^\alpha}\sum_{j=i+1}^{i+\ell} \texttt{LC}_j^{(n)}. \tag{5}$$

Special cases are when $\alpha = 0$ and $\alpha = 1$ which return the original definitions of LC in eq. (2) and MNLP in eq. (3). As noted by Shen et al. (2017), LC for sequences biases acquisition towards longer sentences. The tuneable normalisation factor in eq. (5) over the sequence of scores mediates the balance of shorter and longer subsequences selected. This generalisation can be applied to other types of commonly used acquisition functions such as MaxEnt and BALD by modifying the token-wise score.

4.2 Subsequence Selection

Each sentence $\boldsymbol{x}^{(n)}$ can be broken into a set of subsequences $\mathcal{S}^{(n)} = \{(x_i^{(n)}, ..., x_j^{(n)}) | \forall i < j\}$ where all elements $s \in \mathcal{S}^{(n)}$ can be efficiently scored by first computing the token scores, then aggregating as required. Once this has been done for all sentences in $\mathcal{D}_U$, a query set $\mathcal{S}_Q \subset \cup_n \mathcal{S}^{(n)}$ of non-overlapping (mutually disjoint) subsequences is found. The requirement of non-overlapping subsequences avoids the problem of relabelling tokens, but disallows simply choosing the highest scoring subsequences (since these can overlap). Instead at each round of querying, we perform a greedy selection, repeatedly choosing the highest scoring subsequence that does not overlap with previously selected subsequences. Adjustments can be made to reflect practical needs, such as restricting the length ℓ of the viable subsequences to $[\ell_{min}, \ell_{max}]$. This is because longer subsequences are easier to label, while shorter subsequences are more efficient in querying uncertain tokens, and so the selection is only allowed to operate within these bounds.

Additionally, it is easy to imagine a scenario in which a greedy selection method does not select the maximum total score that can be generated from a sentence. This scenario is illustrated in Table 1 where lengths are restricted to $\ell_{min} = \ell_{max} = 3$ for simplicity. Note that tokens can become unselectable in future rounds because they are not inside a span of unlabelled tokens of at least size ℓ_{min}. When the algorithm has queried all subsequences of this size range, it starts to query shorter subsequences by relaxing the length constraint. However in practise, model performance on the validation set converges before all subsequences of valid range have been exhausted. Nonetheless, when choosing subsequences of size $[\ell_{min}, \ell_{max}] = [4, 7]$ these will be exhausted when roughly 90% and 80% of tokens have been labelled for the OntoNotes 5.0 and CoNLL 2003 datasets.

4.3 Subsequence Label Propagation

Since a subsequence querying algorithm can result in partially labelled sentences, it raises the question of how unlabelled tokens should be handled. In previous work based on the use of CRFs (Tomanek and Hahn, 2009; Wanvarie et al., 2011; Marcheggiani and Artières, 2014) this was solved by using semi-supervision on tokens for which the model showed low uncertainty. However, for neural networks, the use of model generated labels could lead to the model becoming over-confident, harming performance and biasing (Arazo et al., 2020) uncertainty scores. Hence, we ensure that backpropagation only occurs from labelled tokens.

Our final contribution to the AL algorithm is the use of another semi-supervision strategy where we propagate uniquely labelled subsequences in order to minimise the number of annotations needed. When queried for a subsequence, the annotator (in this case an oracle) is not given the contextual tokens in the remainder of the sentence. For this reason, given an identical subsequence, a consistent annotator will provide the same labels. Therefore, the proposed algorithm maintains a dictionary that maps previously queried subsequences to their provided labels. Once a queried subsequence and its label are added to the dictionary, all other matching subsequences in the unlabelled pool are given the same, but temporary, labels.

The tokens retain these temporary labels until they are queried themselves. After scoring and ranking members of $\mathcal{S}$, the algorithm will disregard sequences that match exactly members of this dictionary, which is updated during the querying round. However, if tokens belonging to these previously seen subsequences are encountered in a different context, meaning as part of a different subsequence, they may also be queried. For example, in Table 1, if the subsequence “shop to buy” had been previously queried elsewhere in the dataset, the red subsequence will not be considered for querying, as it retains its temporary labels. Instead, the green subsequence could be queried, in which case the temporary labels of tokens 6 and 7 will be overwritten by new, permanent labels.

Therefore, the value of ℓ_{min} becomes a trade-off between the improved resolution of the acquisition function, and the erroneous propagation of shorter, more frequent label subsequences to identical ones in different contexts.

j	1	2	3	4	5	6	7	8	9	10
$x_j^{(n)}$	`Yassir`	`is`	`going`	`to`	`the`	`shop`	`to`	`buy`	`shoes`	`.`
$y_j^{(n)}$	X	O	O	O	X	X	X	X	X	X
$\texttt{lc}_j^{(n)}$	3.22	-	-	-	0.41	0.78	0.83	0.60	0.27	0.50
						$LC_1 = 0.67$			$LC_1 = 0.46$	
							$LC_1 = 0.74$			
								$LC_1 = 0.57$		

Table 1: This shows the subsequences from a sentence using $\ell_{min} = \ell_{max} = 3, \alpha = 1$. Besides the token index j, the top three rows show the tokens, labels, and the token-wise scores. If $y_j^{(n)} = \text{X}$, then the corresponding token is unlabelled, hence the score is considered when selecting the next query. After this, the subsequences constituting $\mathcal{S}^{(n)}$ are displayed with their LC$_1$ scores. In this case "`shop to buy`" will be chosen since it maximises LC$_1$, but 'traps' its surrounding tokens until ℓ_{min} is lowered to 2 and "`shoes .`" may be considered.

4.4 Subsequence Active Learning Algorithm

Finally, we summarise the AL algorithm proposed. Given a set of unlabelled data $\mathcal{D}_U$, we initially randomly select a proportion of sentences from $\mathcal{D}_U$, label them, and add these to $\mathcal{D}_L$. A dictionary $\mathcal{B}$ is also initialised. Using these labelled sentences we train a model. Then, the following proposed training cycle is repeated until $\mathcal{D}_U$ is empty (or an early stopping condition is reached):

1. Find all consecutive unlabelled subsequences in $\mathcal{D}_U$, and score them using a pre-defined acquisition function.

2. Select the top scoring non-overlapping subsequences $\mathcal{S}_Q$ that do not appear in $\mathcal{B}$, such that the number of tokens in $\mathcal{S}_Q$ is A, and query them to the annotators. Update $\mathcal{D}_L$ and $\mathcal{D}_U$. As each sequence is selected, add it to $\mathcal{B}$, mapping it to its true labels.

3. Provide all occurrences of the keys of $\mathcal{B}$ in $\mathcal{D}_U$ with their corresponding temporary labels. These will not be included in $\mathcal{D}_L$ as these are temporary.

4. Finetune the model on sentences with any label, temporary and permanent.

Repeat this process until convergence.

5 Experimental Setup

5.1 Datasets

As in previous works (Shen et al., 2017), we use the two following NER datasets:

OntoNotes 5.0. This is a dataset used to compare results with the full sentence querying baseline (Weischedel, Ralph et al., 2013), and comprising of text coming from: news, conversational telephone speech, weblogs, usenet newsgroups, broadcast, and talk shows. This is a BIO formatted dataset with a total of $K = 37$ classes and 99,333 training sentences, with an average sentence length of 17.8 tokens in its training set.

CoNLL 2003. This is a dataset, also in BIO format, with only 4 entity types (LOC, MISC, PER, ORG) resulting in $K = 9$ labels (Tjong Kim Sang and De Meulder, 2003). This dataset is made from a collection of news wire articles from the Reuters Corpus (Lewis et al., 2004). The average sentence length is 12.6 tokens in its training set.

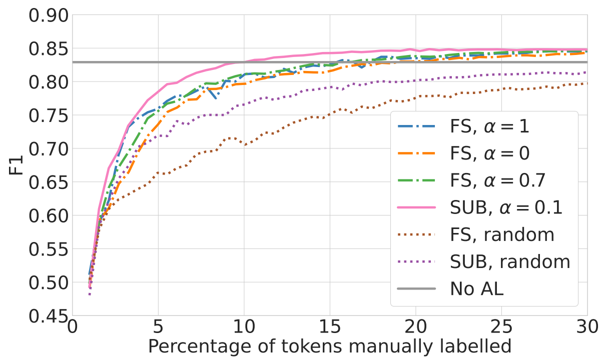
A full list of class types and entity lengths and frequencies for both datasets can be found in the Appendix.

5.2 NER Model

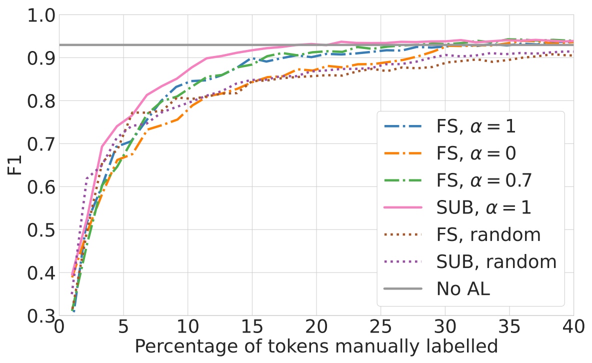
Following the work of Shen et al. (2017), a CNN-CNN-LSTM model for combined letter- and token-level embeddings was used; see Appendix for an overview of the model and hyperparameters setting and validation. Furthermore, the AL algorithm used in (Shen et al., 2017) will serve as one of the baselines following the same procedure. This represents an equivalent algorithm to that proposed, but which can only query full sentences, and does not use label propagation.

5.3 Model Training and Evaluation

As the evaluation measure we use the F$_1$ score. After the first round of random subsequence selection, the model is trained. After subsequent selections the model is finetuned - training is resumed from the previous round's parameters. In all cases, the model training was stopped either after 30 epochs were completed, or if the F$_1$ score for the valida-



(a) LC$_\alpha$ for OntoNotes 5.0 NER dataset



(b) LC$_\alpha$ for CoNLL 2003 NER dataset

Figure 1: F$_1$ score on test set achieved each round using round-optimal model parameters. All subsequence experiments here use $\ell_{min} = 4, \ell_{max} = 7$. Each curve is averaged over 10 runs.

tion set had monotonically decreased for 2 epochs. This validation set is made up of a randomly selected 1% of sentences of the original training set. After finetuning, the model reloads its parameters from the round-optimal epoch, and its performance is evaluated on the test set. Furthermore, the AL algorithms were also stopped after all hyperparameter variations using that dataset and acquisition function family had converged to the same best F$_1$, which we denote with F_1^*. For the OntoNotes 5.0 dataset, F_1^* value was achieved after 30% of the training set was labelled, and for the CoNLL 2003 dataset after 40%.

5.4 Active Learning Setup & Evaluation

We choose $\ell_{min} = 4$ to give a realistic context to the annotator, and to avoid a significant propagation of common subsequences. The upper bound of $\ell_{max} = 7$ was chosen to ensure subsequences were properly utilised, since the average sentence length of both datasets is roughly twice this size. For the OntoNotes 5.0 dataset, every round $A = 10,000$ tokens are queried, whereas for the CoNLL 2003 dataset $A = 2,000$ tokens. These represent roughly 0.5% and 1% of the available training set.

We evaluate the efficacy and efficiency of the tested AL strategies in three ways. First, model performance over the course of the algorithm was evaluated using end of round F$_1$ score on the test set. We compare the proportion of the dataset's tokens labelled when the model achieves 99% of the F_1^* score ($\hat{F_1}^* = 0.99 \times F_1^*$). We also quantify the rate of improvement of model performance during training using the normalised Area Under the Curve (AUC) score of each F$_1$ test curve. The normalisation ensures that the resulting AUC score is in the range $[0, 1]$, and it is achieved by dividing the AUC score by the size of the dataset. This implies that methods that converge faster to their best performance will have a higher normalized AUC. Second, we consider how quickly the algorithms can locate and query relevant tokens (named entities). Third, we finally evaluate their ability to extract the most uncertain tokens from the unlabelled pool.

6 Results & Discussion

6.1 Active Learning Performance

Figure 1 shows the LC$_\alpha$ performance curves for $\alpha = 0$, $\alpha = 1$ and the best performing value for each acquisition class (based on the normalised AUC score, Table 3) for full sentence querying (FS), and only the best performing α values for subsequence querying (SUB). The figure also shows the performance of training on the complete training set (No AL), and when the both sentences and subsequences are random selected by the acquisition function. The equivalent figures for MaxEnt$_\alpha$ are available in Appendix, and follow similar trends. Then, the performance of each curve, quantified in terms of the normalised AUC is summarised in Table 3.

Table 2 shows further analysis of the best results in Figure 1, with best referring to acquisition function and optimal α. These results first show that subsequence querying methods are more efficient than querying full sentences, achieving their final F_1 with substantially less annotated data, and with higher normalised AUC scores. For OntoNotes 5.0, querying subsequences reduces final proportion required by 38.8%. For CoNLL 2003, this reduction is 36.6%. Altogether, subsequence querying holds improved efficiency over the full sentence querying baseline.

	F_1 Score		Final Frac. of the Dataset	
	100%	AL	FS	SUB
ON 5.0	0.829	0.843	22%	13%
CoNLL	0.930	0.938	42%	27%

Table 2: Summary of the results of the AL strategies from Figure 1, when the models are trained using 100% of the training set and active learning (AL), with the best hyperparameter setting of the acquisition function with for full sentence and subsequence, based on normalised AUC score.

As a point of interest, full sentence querying can be easily improved by optimising α alone. For the OntoNotes 5.0 dataset, using LC_1, 24.2% of tokens are required to achieve F_1^*. This however, can be improved by 9.33% to only requiring 22.0% by choosing $\alpha = 0.7$. For CoNLL 2003, using LC_1 for full sentences, 50.0% of the dataset was required, but when using $LC_{0.7}$, it was 40.7% of the tokens.

6.2 Entity Recall

This section and the next aim to understand some of the underlying mechanisms that allow the subsequence querying methods to achieve results substantially better than a full sentence baseline. Namely, the ability of the different methods to extract the tokens for which the model is the most uncertain about. Given that the majority of tokens in both datasets have the same label - "O", signifying no entity - it is likely that tokens belonging to entities, particularly rarer classes, trigger higher model uncertainty. Querying full sentences at a time, the AL algorithm will spend much of its token budget for that round labelling non-entity tokens while attempting to locate the more informative entities. Subsequence querying methods, not faced with this wasteful behaviour, allow the AL algorithm to query entity tokens quicker, locating and labelling the majority of entity tokens faster over the course of training.

The proportion of tokens belonging to entities that the AL algorithm has queried against the round number is plotted in Figure 2 for OntoNotes 5.0. For both datasets, the random querying methods contain a distribution of token classes that reflect the dataset at large, producing roughly linear curves for this figure. Curves for all methods that employ

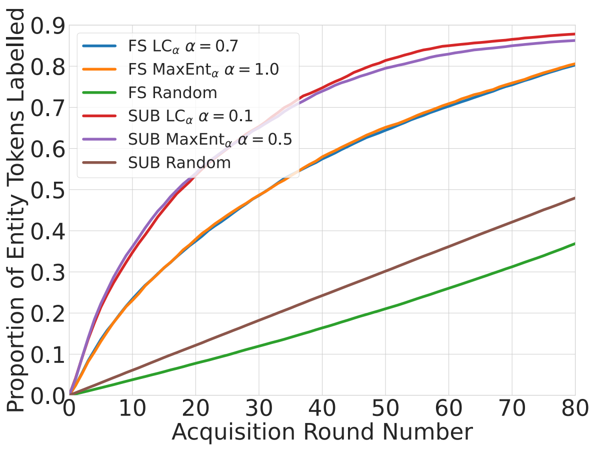


Figure 2: Proportion of tokens that belong to entities labelled, against the round number.

an uncertainty based acquisition function are concave, and the AUC reflects the ranking of model performance for each querying method. This relation suggests that shortly after initialisation, better performing algorithm variations query entity tokens faster. In later stages of finetuning this rate is reduced, likely because after labelling a large proportion of them, the remaining entity tokens cause little uncertainty for the model. In a practical setting where querying may have to be stopped before model performance has converged (i.e. due to accumulated cost of annotations), it is greatly beneficial to ensure that the model is exposed to a high number of relevant tokens, because this increases the likelihood of locating entity tokens belonging to underrepresented classes at an early stage.

6.3 Uncertainty Score Analysis

Finally, this section compares the scores of tokens in the queried set $\mathcal{S}_Q$ for each querying method. Comparing the distribution and development of these scores provides a direct insight to the core assumptions of why full sentence querying is outperformed. Figure 3 shows the difference in score distributions for sentence versus subsequence querying, against querying round number, for rounds preceding model performance convergence. First, it is seen that decreasing the individual query size (full sentence to subsequence) increases the median uncertainty extracted at the earlier rounds. Second, Figure 3 provides evidence for the mechanism suggested earlier: aggregating the token scores across full sentences means querying both the highly uncertain tokens, and the tokens that provide little uncertainty. Querying high scoring sentences like this can cause a distribution with two peaks as seen in

Dataset	Acquisition Function	Full Sentence $\alpha = 0$	Full Sentence $\alpha = 1$	Full Sentence Optimal (α)	Subsequence $\alpha = 0$	Subsequence $\alpha = 1$	Subsequence Optimal (α)
OntoNotes 5.0	LC_α	0.794	0.802	0.804 (0.7)	0.817	0.812	**0.818** † (0.1)
	MaxEnt_α	0.791	0.803	0.803 (1.0)	0.815	0.813	0.816 † (0.5)
	Random	0.734			0.769		
CoNLL 2003	LC_α	0.857	0.875	0.879 (0.7)	0.885	0.883	**0.892** † (1.0)
	MaxEnt_α	0.841	0.882	0.882 (1.0)	0.881	0.883	0.891 † (0.9)
	Random	0.824			0.859		

Table 3: Normalised AUC scores for model performance (F1 score on test set) for $\alpha = 0, 1$, and its optimal value in each case. Each pair of differences between the optimized acquisition function for full sentences and subsequences (indicated by a †) are significantly different (two-sided unpaired t-test, with p-value < 0.05).

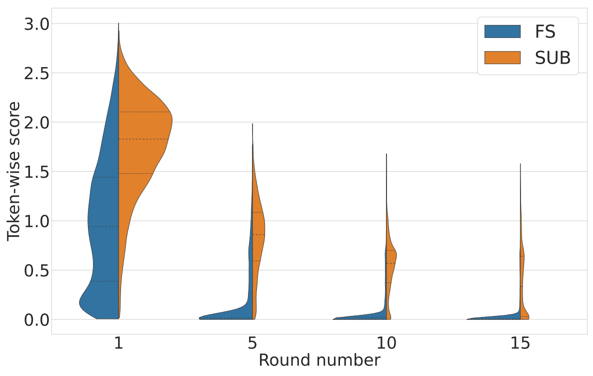


Figure 3: Distributions of the queried LC scores for the OntoNotes 5.0 dataset, made on the 1st, 5th, 10th, and 15th scoring rounds. This corresponds to scores after training on 1%, 3.2%, 6.1%, and 9.0% of the utilised training set.

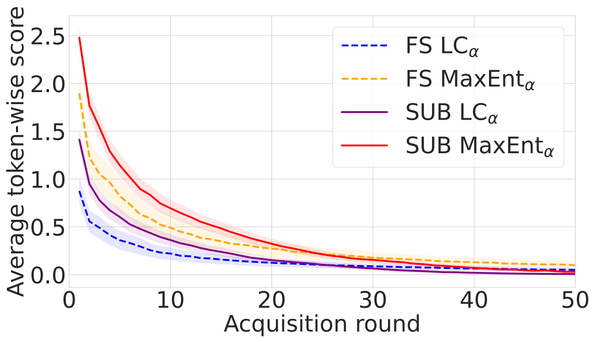


Figure 4: Average value of LC for all tokens in $\mathcal{S}_Q$ with confidence intervals, against round number. Score values are averaged over all tested values of α

the figure. As the model becomes increasingly certain about its predictions, high scores are localised within smaller subsequences, and the coarse sensitivity of full sentence querying means it forfeits all the higher scoring tokens. These differences were also observed when comparing subsequence querying methods with sub-optimal α.

This figure only analyses behaviour of up to 9% of the training set's tokens have been queried. Instead, Figure 4 show how the mean of token-wise scores evolve for different querying methods for the OntoNotes 5.0 dataset until convergence. This clearly shows that subsequence querying methods converge faster over the full course of the algorithm compared to full sentence querying. This is consistent with Figure 1 in terms of initial rate and final time of model performance convergence, namely that model performance plateaus alongside the uncertainty score.

Keeping track of query scores like this is also a reasonable idea in industrial applications. When training on a very semantically specific corpus, there may not be enough fully labelled sentences to build a test set. In that case, observing the rate progress of score convergence can be used as an early stopping method for the AL algorithm (Zhu et al., 2010).

7 Conclusion & Future Work

In this study we have employed subsequence querying methods for improving the efficiency of AL for NER tasks. We have seen that these methods outperform full sentence querying in terms of annotations required for optimal model performance, requiring 38.8% and 36.6% fewer tokens for the OntoNotes 5.0 and CoNLL 2003 datasets. Optimal results for subsequence querying (and full sentence querying) were achieved by generalising previously used AL acquisition functions, defining a larger family of acquisition functions for sequential data.

The analysis of § 6.3 suggests that a full sentence querying causes noisy acquisition functions due to the tokens in the queried sentences that were not highly scored. This added noise reduces the budget efficiency, and a subsequence querying method eliminates a large part of this effect. This efficiency also translated into a faster recall of named entities in the dataset to be queried (§ 6.2).

Limitations and future work: Limitations of this study are largely centred on the use of an oracle to provide tokens with their labels. With human annotators, the cropped context of subsequence queries may make them produce more inaccuracies than when annotating full sentences. such studies will help reveal how context affects label accuracy, how this, in turn, affects optimal hyperparameters in the subsequence selection process (such as optimal query length), further accommodations that must be made to effectively optimise worker efficiency, and how to deal with unreliable labels. We leave to future work the evaluation of these querying methods with human annotators.

There are also ways to incorporate model generated labelling methods for more robust semi-supervision into our framework that we leave to future work.

Finally, there are examples of other tasks for structured data, such as audio, video, and image segmentation, where the part of an instance may be queried. A generalisation of the strategy demonstrated for the NER case may allow for more efficient active learning querying methods for these other types of data.

References

Eric Arazo, Diego Ortego, Paul Albert, Noel E. O'Connor, and Kevin McGuinness. 2020. Pseudo-labeling and confirmation bias in deep semi-supervised learning. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.

Yukun Chen, Thomas A. Lasko, Qiaozhu Mei, Joshua C. Denny, and Hua Xu. 2015. A study of active learning methods for named entity recognition in clinical text. *Journal of Biomedical Informatics*, 58:11 – 18.

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.

Meng Fang, Yuan Li, and Trevor Cohn. 2017. Learning how to active learn: A deep reinforcement learning approach. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 595–605, Copenhagen, Denmark. Association for Computational Linguistics.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. 2017. Deep Bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1183–1192, International Convention Centre, Sydney, Australia. PMLR.

Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. 2011. Bayesian active learning for classification and preference learning.

S. Huang, R. Jin, and Z. Zhou. 2014. Active learning by querying informative and representative examples. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(10):1936–1949.

Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. 2019. Label propagation for deep semi-supervised learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Yann LeCun and Yoshua Bengio. 1998. *Convolutional Networks for Images, Speech, and Time Series*, page 255–258. MIT Press, Cambridge, MA, USA.

David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397.

Wang Ling, Chris Dyer, Alan W. Black, and Isabel Trancoso. 2015. Two/too simple adaptations of Word2Vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1299–1304, Denver, Colorado. Association for Computational Linguistics.

Diego Marcheggiani and Thierry Artières. 2014. An experimental comparison of active learning strategies for partially labeled sequences. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 898–906, Doha, Qatar. Association for Computational Linguistics.

Mingkun Li and I. K. Sethi. 2006. Confidence-based active learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1251–1261.

Ion Muslea, Steven Minton, and Craig A. Knoblock. 2002. Active + semi-supervised learning = robust multi-view learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, page 435–442, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Robert Pinsler, Jonathan Gordon, Eric Nalisnick, and José Miguel Hernández-Lobato. 2019. Bayesian batch active learning as sparse subset approximation. In *Advances in Neural Information Processing Systems*, volume 32, pages 6359–6370. Curran Associates, Inc.

Y Reddy, Viswanath Pulabaigari, and Eswara B. 2018. Semi-supervised learning: a brief review. *International Journal of Engineering Technology*, 7:81.

Jeffrey Rzeszotarski, Ed Chi, Praveen Paritosh, and Peng Dai. 2013. Inserting micro-breaks into crowdsourcing workflows. *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, 1(1).

Burr Settles. 2011. From theories to queries: Active learning in practice. In *Active Learning and Experimental Design workshop In conjunction with AISTATS 2010*, volume 16 of *Proceedings of Machine Learning Research*, pages 1–18, Sardinia, Italy. JMLR Workshop and Conference Proceedings.

Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, page 1070–1079, USA. Association for Computational Linguistics.

Yanyao Shen, Hyokun Yun, Zachary Lipton, Yakov Kronrod, and Animashree Anandkumar. 2017. Deep active learning for named entity recognition. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 252–256, Vancouver, Canada. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

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

Simon Tong and Daphne Koller. 2002. Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res.*, 2:45–66.

K. Wang, D. Zhang, Y. Li, R. Zhang, and L. Lin. 2017. Cost-effective active learning for deep image classification. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(12):2591–2600.

Dittaya Wanvarie, Hiroya Takamura, and Manabu Okumura. 2011. Active learning with subsequence sampling strategy for sequence labeling tasks. *Journal of Natural Language Processing*, 18(2):153–173.

Kai Wei, Rishabh Iyer, and Jeff Bilmes. 2015. Submodularity in data subset selection and active learning. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1954–1963, Lille, France. PMLR.

Weischedel, Ralph, Palmer, Martha, Marcus, Mitchell, Hovy, Eduard, Pradhan, Sameer, Ramshaw, Lance, Xue, Nianwen, Taylor, Ann, Kaufman, Jeff, Franchini, Michelle, El-Bachouti, Mohammed, Belvin, Robert, and Houston, Ann. 2013. Ontonotes release 5.0.

Jingbo Zhu, Huizhen Wang, Eduard Hovy, and Matthew Ma. 2010. Confidence-based stopping criteria for active learning for data annotation. *ACM Trans. Speech Lang. Process.*, 6(3).

A Model Architecture

The model architecture is built of three sections. The character-level convolutional neural network (CNN) (LeCun and Bengio, 1998) character-level encoder extracts character level features, $\mathbf{w}_j^{\text{word}}$ for each token $x_j^{(i)}$ in a sentence.

Then, a latent token embedding $\mathbf{w}_j^{\text{emb}}$ corresponding to that token is generated. The full representation of the token is the concatentation of the two vectors: $\mathbf{w}_j^{\text{full}} := (\mathbf{w}_j^{\text{char}}, \mathbf{w}_j^{\text{emb}})$. The token-label embeddings, $\mathbf{w}^{\text{emb}}$, are initialised using word2vec (Ling et al., 2015), and updated during training and finetuning, as per the baseline paper. A second, token-level CNN encoder is used to generate $\{\mathbf{h}_j^{\text{token}}\}_{j=1}^{\ell_i}$, given the token-level representations $\{\mathbf{w}_j^{\text{full}}\}_{j=1}^{\ell_i}$. The final token-level encoding is defined by another concatentation: $\mathbf{h}_j^{\text{Enc}} := (\mathbf{h}_j^{\text{token}}, \mathbf{w}_j^{\text{full}})$.

Finally, a tag decoder is used to generate the token-level pmfs over the C possible token classes: $\{\mathbf{h}_j^{\text{Enc}}\}_{j=1}^{\ell_i} \xrightarrow{\text{LSTM}} \{\hat{\boldsymbol{y}}_j^{(i)}\}_{j=1}^{\ell_i}$.

B Model & Training Parameters

Table 4 lists the hyperparameter values used to train the NER model. Note that while dropout is used during training, it is turned off when generating the probabilities that contribute to the scoring of the acquisition function. Model was developed using PyTorch, and trained on a Titan RTX.

Hyperparameter	Value
Batch size	32
Dropout rate for convolutional layers	0.5
Dropout rate for embedding layers	0.25
Gradient clipping magnitude	0.35
Character- and token-level CNN kernel size	3
Layers in character- and token-level CNNs	3
Character embedding vector size	50
Number of filters per character-level CNN layer	50
Number of filters per token-level CNN layer	300
Optimiser type	SGD
Optimiser learning rate	1.0

Table 4: Values of model and training hyperparameters used throughout the investigation.

C Dataset Analysis

Here, we cluster similar labels in the BIO format, reducing the total K classes to the $K^{(r)} = (K+1)/2$ class groups $c_1^{(r)}, ..., c_{K^{(r)}}^{(r)}$. Therefore, $c_1^{(r)}$ corresponds exactly to c_1, the empty label, while $c_k^{(r)}, k > 1$ groups the raw labels c_{2k-2} and c_{2k-1}.

Figures 5 and 7 show the distribution of these class groups for the OntoNotes 5.0 and CoNLL 2003 datasets respectively. For the former, counts range from 199 tokens for the `'LANGUAGE'` to 46698 tokens for the `'ORG'` class. The full available training set totals 1766955 tokens in 99333 sentences; this is partitioned into a train and validation set during experimentation. A further test set comprises of 146253 tokens in 8057 sentences. The latter's training set contains 172210 tokens in 13689 sentences, and its test set has 42141 tokens in 3091 sentences sentences.

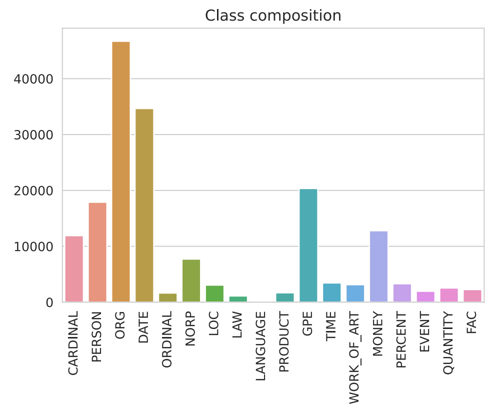


Figure 5: Composition of token classes in the Onto-Notes 5.0 English NER training set.

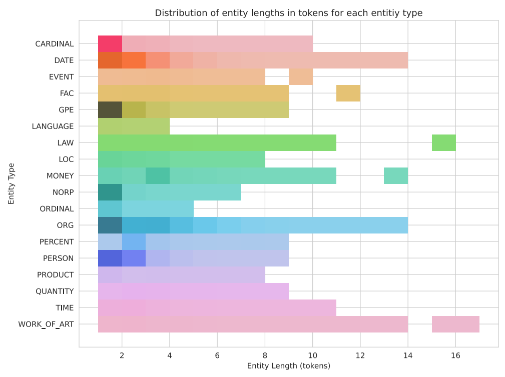


Figure 6: Lengths of entities in the Onto-Notes 5.0 training set in number of tokens, again omitting the empty class `'O'`

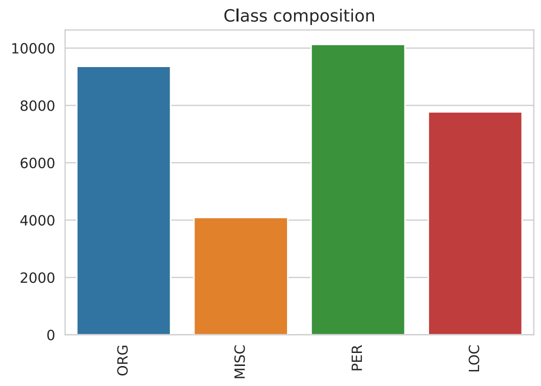


Figure 7: Composition of token classes in the CoNLL 2003 NER training set.

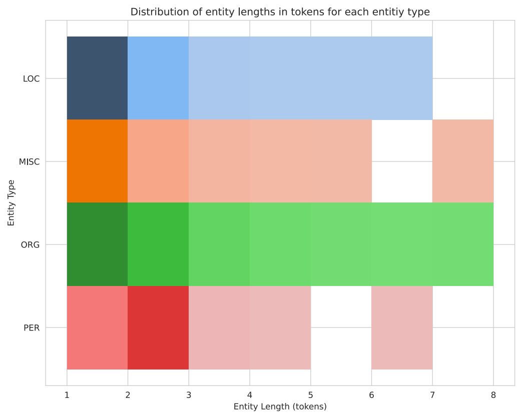
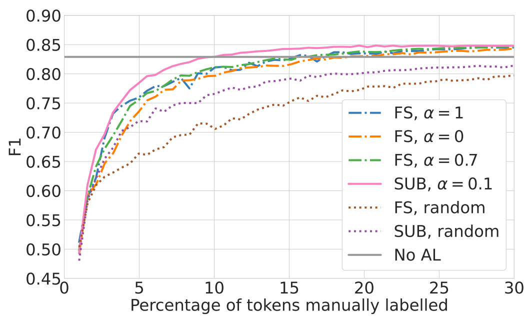


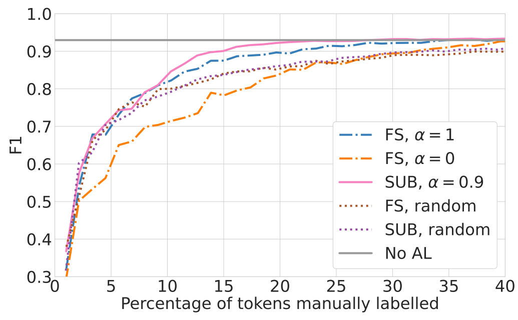
Figure 8: Lengths of entities in the CoNLL 2003 training set in number of tokens, again omitting the empty class `'O'`

D Active Learning Results for Both Datasets

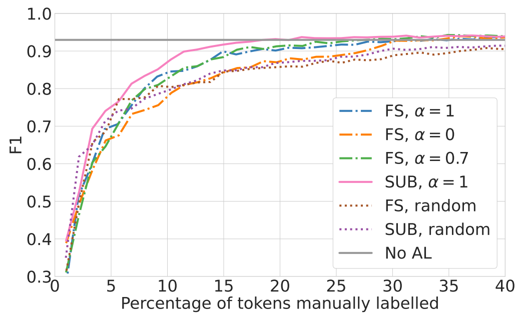
In Figure 9 we show the model performance plotted against the percentage of the tokens used as a training set for all the combinations of acquisition functions.



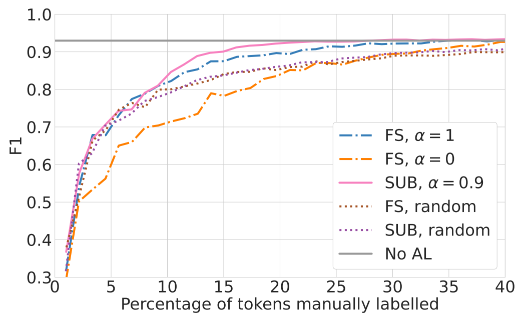
(a) LC$_\alpha$ for OntoNotes5.0 NER dataset



(b) MaxEnt$_\alpha$ for OntoNotes5.0 NER dataset



(c) LC$_\alpha$ for CoNLL 2003 NER dataset



(d) MaxEnt$_\alpha$ for CoNLL 2003 NER dataset

Figure 9: F1 score on test set achieved each round (top) and against time (bottom in each case) using round-optimal model parameters. All subsequence experiments here use $\ell_{min} = 3, \ell_{max} = 6$.