J. Hlaváčová (Ed.): ITAT 2017 Proceedings, pp. 144–152
CEUR Workshop Proceedings Vol. 1885, ISSN 1613-0073, © 2017 T. Šabata, T. Borovička, M. Holeňa

# K-best Viterbi Semi-supervized Active Learning in Sequence Labelling

Tomáš Šabata[1], Tomáš Borovička[1], and Martin Holeňa[2]

[1] Faculty of Information Technology,
Czech Technical University in Prague,
Prague, The Czech Repubic
[2] Institute of Computer Science,
Czech Academy of Sciences,
Prague, The Czech Republic

*Abstract:* In application domains where there exists a large amount of unlabelled data but obtaining labels is expensive, active learning is a useful way to select which data should be labelled. In addition to its traditional successful use in classification and regression tasks, active learning has been also applied to sequence labelling. According to the standard active learning approach, sequences for which the labelling would be the most informative should be labelled. However, labelling the entire sequence may be inefficient as for some its parts, the labels can be predicted using a model. Labelling such parts brings only a little new information. Therefore in this paper, we investigate a sequence labelling approach in which in the sequence selected for labelling, the labels of most tokens are predicted by a model and only tokens that the model can not predict with sufficient confidence are labelled. Those tokens are identified using the k-best Viterbi algorithm.

## 1 Introduction

Hidden Markov models (HMMs) and conditional random fields (CRFs) are very popular models in sequence labelling tasks such as handwriting recognition, speech recognition, DNA analysis, video analysis, information extraction or natural language processing (NLP). They achieve good results if a high quality and fully annotated dataset is available. Unfortunately, in these tasks, obtaining labels for data may be expensive. The annotation cost is a motivation for using active learning. Active learning usually begins with a small labelled set $\mathcal{L}$ and in each iteration, the most informative instance of an unlabeled set $\mathcal{U}$ is chosen, annotated by an oracle and added to the set $\mathcal{L}$. The model is retrained using the extended set $\mathcal{L}$ and the whole process repeats till a stopping criterion is met. This approach is valuable in tasks where unlabeled data are easily available but obtaining their labels is expensive. In this case, it aims at achieving higher accuracy with minimal cost.

Nevertheless, labelling long sequences can be troublesome, in particular for a human annotator who is prone to create labels of lower quality. To address the problem, we can combine active learning with semi-supervised learning. Semi-supervised active learning in sequence labelling means that a model labels those parts of a sequence that are easy to predict and let the annotator to focus only on parts of sequences that are the most uncertain.

In this paper, we propose a semi-supervised active learning approach that uses the k-best Viterbi algorithm to detect candidates for manual labelling. The proposed approach was experimentally evaluated on an NLP task, part-of-speech tagging.

In the second section, we provide an overview of related work in active and semi-supervised learning. The third section recalls some basics of hidden Markov models that are necessary for understanding of the proposed approach which is introduced in the fourth section. An experiment description, its result and analysis are given in the fifth section. The paper is concluded by a discussion of the results and possible future work.

## 2 Related work

While active learning has been studied for classification and regression tasks [1], less attention has been given to the task of sequence labelling. Despite this, the most of the algorithms developed for the task of classification can also be adapted for the task of sequence labelling [2].

Active learning can be applied in three different scenarios: *pool-based sampling*, *stream-based selective sampling* and *membership query synthesis*. The most commonly used scenario is pool-based sampling originaly proposed in [3]. It has been studied for many real-world problem domains with sequence labelling included. For example, speech recognition [4], information retrieval [5] or named entitiy recognition [6]. The main idea of pool-based active learning is using a *query strategy framework* to find the most informative sample (sequence) from the unlabeled set (pool) of samples. This selected sample is annotated and added to the labelled set. The model is retrained, and the whole process repeats. The second scenario, stream-based selective sampling, is also possible to use in sequence labeling [7] but it is used less commonly. The difference against pool-based sampling is that samples are coming in a stream and the framework decides to annotate the sample or to discard it. The discarded samples are never later used in training. The main idea of the third scenario, membership query synthesis, is that a learner can query any unlabeled instance, usually generated de novo.

Active learning can use one of six different query strategy frameworks [1]. The most commonly used frameworks are *Uncertainty Sampling* [8] and *Querry-by-Committee* [9]. Uncertainty Sampling selects sample in which the model is least confident. Query-by-Committee maintains a committee of predictors, and the sample on which the predictors disagree most regarding their predictions is considered to be the most informative. Other query strategies applicable to sequences are *Expected Gradient Length*, *Information Density*, *Fisher Information* and *Future Error Reduction* [2]. The *Future Error Reduction* framework is not commonly used due to its high computational complexity.

Semi-supervised learning methods were developed with the same motivation of a partly unlabeled dataset. *Self-Training* is a commonly used technique where the predictor is firstly trained on a small labelled dataset and then used to annotate data. The most confident labels are added to the training set, and the predictor is retrained. Self-training has found application in several tasks of natural language processing [10, 11, 12]. Another technique, *Co-training*, is a multi-learner algorithm where learners have independent, complementary features of the dataset and produce labelled examples separately [13]. Semi-supervized learning was also applied to sequence modelling tasks [14, 15].

In tasks where a large amount of labelled data is required (for example, NLP tasks), the semi-supervised learning does not perform well due to the propagation of many tagging errors through the learning dataset. The problem of the data pollution was partially solved in [16], where a human was put into training loop to correct labelled examples. However, correction of labelled data can be time-consuming and is similar to labelling the data from scratch. To address the problem, a semi-supervised active learning which does not need any human inspection was proposed in [17]. The approach uses active learning to find the most informative sequences. The model labels the most informative sequences and uses a marginal probability of each sequence token to decide if the prediction is confident. The method contains two parameters, a delay of running semi-supervised approach and a confidence threshold. A proper setting of parameters is necessary to achieve the desired results.

Inspired by the semi-supervised method in [17], we proposed a method that does not need the confidence threshold parameter due to using the k-best Viterbi paths.

## 3 Preliminaries

In the paper, we focus on a task of part of speech tagging. For the simplicity, our approach is shown by means of HMM but can be extended to CRF as well. In this section, the principles of an HMM will be recalled.

### 3.1 Hidden Markov Models

With each HMM, a random process indexed by time is connected, which is assumed to be in exactly one of a set of $N$ distinct states at any time. At regularly spaced discrete times, the system changes its state according to probabilities of transitions between states. The time steps associated with time changes are denoted $t = 1,2,3,\ldots$ . The actual state at a time step $t$ is denoted $q_t$.

The process itself is assumed to be a first-order Markov chain which is described as a matrix of transition probabilities $A = \{a_{ij}\}$, defined

$$a_{ij} = P(q_t = y_j | q_{t-1} = y_i), \quad 1 \leq i,j \leq N. \qquad (1)$$

A simple observable Markov chain is too restrictive to describe the reality. However, it can be extended. Denoting Y the variable recording the states of the Markov chain, an HMM is obtained through completing Y with a random variable X. In the context of that HMM, $X$ is called 'observable variable' or 'output variable', whereas $Y$ is called 'hidden variable'. The hidden variable $Y$ takes values in the set $\{y_1, y_2, \ldots, y_N\}$ and the observable variable $X$ takes values in a set $\{x_1, x_2, \ldots, x_M\}$.

We assume to have an observation sequence $O = o_1 o_2 \ldots o_T$ and a state sequence $Q = q_1 q_2 \ldots q_T$ which corresponds to the observation sequence. HMM can be characterised using three probability distributions:

1. A state transition probability distribution $A = \{a_{i,j}\}$.
2. A probability distribution of observable variables, $B = \{b_i(x_k)\}$, where $b_i(x_k)$ is the probability of $o_t$ assuming the value $x_k$ if $q_t$ is in the state $y_i$ and it is defined
$$b_i(k) = P(o_t = x_k | q_t = y_i). \qquad (2)$$
3. An initial state distribution $\pi = \{\pi_i\}$ is defined by
$$\pi_i = P(q_1 = y_i).$$

With these three elements, HMM is fully defined and denoted $\theta = (A, B, \pi)$.

The parameters of an HMM can be learned either in a semi-supervised way with the Baum-Welch algorithm [18] or in a fully-supervised way with the maximum-likelihood estimation (MLE). In the fully-supervised way, values of both observable and hidden variables are known.

In the MLE, we assume a training set $D = \{(o^1, q^1), \ldots, (o^n, q^n)\}$ of a size $n$ whose elements are independent. The MLE consists in taking those parameters $\theta^*$ that maximize the probability of the training set:

$$\theta^* = \operatorname{argmax}_\theta P(D|\theta). \qquad (3)$$

Due to (1) and (2), the probability in (3) turns to:

$$P(D|\theta) = \prod_{i,j} a_{i,j}^{T_{i,j}} \prod_{i,k} [b_i(k)]^{E_i(k)}, \sum_j a_{i,j} = 1, \sum_k b_i(k) = 1$$

where $T_{i,j}$ stands for number of transitions from state $y_i$ to state $y_j$ in the training set and $E_i(k)$ stands for number of emissions of value $x_j$ in state $y_i$. Then, parameters $A$ and $B$ can be obtained by following formulas:

$$a_{i,j} = \frac{T_{i,j} + r_{i,j}}{\sum_{j'}(T_{i,j'} + r_{i,j'})} \text{ and } b_i(k) = \frac{E_i(k) + r_i(k)}{\sum_{k'}(E_i(k) + r_i(k'))}, \quad (4)$$

where $r_i, j$ and $r_i(k)$ are our prior beliefs. The prior beliefs are used in the case of an insufficiently large dataset, where the estimate would lead to zero probabilities of events which never occurred in $D$.

To simplify the notation, we define variables $\alpha$ and $\beta$ as follows:

$$\alpha_t(i) = p(o_1, ..., o_t, q_t = y_i | \theta), \quad (5)$$
$$\beta_t(i) = p(o_t + 1, ..., o_T, q_t = y_i | \theta). \quad (6)$$

These variables are computed using the following *forward-backward* algorithm [18]:

$$\alpha_1(i) = \pi_i b_i(o_1),$$
$$\alpha_{t+1}(i) = \Big(\sum_{j=1}^{N} \alpha_t(j) a_{j,i}\Big) b_i(o_{t+1}),$$

respectively,

$$\beta_T(i) = 1,$$
$$\beta_t(i) = \sum_{j=1}^{N} a_{i,j} b_j(o_{t+1}) \beta_{t+1}(j).$$

### 3.2 Marginal probability

Once, the model is learned, it can be used for the prediction of a sequence of hiddden states given an observable sequence. In the task of finding the most likely states sequence, it is possible to find the sequence that maximises the expected number of correctly assigned states. From (5) follows that the *marginal probability* of being in a specific state $i$ at a particular time $t$ is:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{n} \alpha_t(j)\beta_t(j)} \quad (7)$$

Then, maximising the expected number of correctly assigned states can be achieved through applying $q_t = \arg\max_{y_i \in Y} \gamma_t(y_i)$ to the whole sequence. However, the approach can find a sequence with very low or even zero probability in case the sequence is not feasible.

### 3.3 Viterbi algrithm

Viterbi algorithm is a dynamic programming algorithm that finds the most likely state sequence as a whole by maximising of $P(Q, O|\theta)$. It gradually counts the maximal probability of the state chain from its beginning till the state in time $t$ with the state $q_t$ being $y_i$ represented by a variable $\delta_t(i) = \max_{q_1,...,q_{t-1}} P(q_1, ..., q_t = y_i, o_1, ..., o_t)$. The algorithm is initialized as follows:

$$\delta_1(i) = \pi_i b_i(o_1), \quad (8)$$

and for each $2 \leq t \leq T$ and each $y_i$ from $Y$, the algorithm calculates the variable $\delta_t(i)$:

$$\delta_t(i) = (max_{1 \leq j \leq N} \delta_{t-1}(y_j) a_{i,j}) b_i(o_t). \quad (9)$$

In each time $t$ and for each node $i$, the algorithm stores the link to one of all predecessor nodes with which it forms the best path. These links are stored in the additional two-dimensional array $\psi_t(i)$, where:

$$\psi_1(i) = 0,$$
$$\psi_t(i) = \arg\max_{1 \leq j \leq N} \delta_{t-1}(j) a_{ji}.$$

The probability of the most probable sequence can be found by $\max_{1\ leqi \leq N} \delta_T(i)$ and the most probable state path $Q^* = (q_1^*, q_2^*, ..., q_T^*)$ can be found by backtracking:

$$q_T^* = \arg\max_{1 \leq i \leq N} \delta_T(i),$$
$$q_t^* = \psi_{t+1}(q_{t+1}^*).$$

The Viterbi algorithm has a similar structure as the forward-backward algorithm, and both have complexity $\mathcal{O}(N^2 T)$.

## 4 Proposed approach

Our proposed approach is an adaptation of the semi-supervised active learning method (*SeSAL*), originally proposed in [17]. Both SeSAL and our adaptation are based on a standard fully-supervized active learning algorithm (*FuSAL*). The concept of FuSAL algorithm is decribed by pseudocode in Algorithm 1.

An *utility function* $\phi_M(x)$ represents an informativness of the sample $x$ given the model $M$. In the algorithm, any utility function can be used to find the most informative sequence [2].

In the SeSAL, the most informative instance is annotated by a model $M$ and only the tokens whose predicted labels have a confidence smaller than a given threshold are given to a human annotator (oracle). Finding the optimal threshold value is an optimisation task minimising the dataset pollution and the number of queried labels. If the threshold is too high, a human annotates labels in which the model is well confident. On the other hand, if the threshold is too low, the algorithm accepts incorrectly labelled tokens which may result in a polluted training set.

In the SeSAL, they use a parameter called *delay* that represents a number of iterations of the FuSAL before the algorithm is switched to SeSAL. This helps to avoid producing errors coming from incorrect labels comming from an insufficiently converged model.

**Algorithm 1** FuSAL algorithm

```
Given:
    $\mathcal{L}$: set of labeled examples
    $\mathcal{U}$: set of unlabeled examples
    $\phi_M$: utility function
Algorithm:
 1: while stopping criterion is not met do
 2:     learn model M from $\mathcal{L}$
 3:     for all $x_i \in \mathcal{U}$:$u_{x_i} \leftarrow \phi_M(x_i)$
 4:     select $x^* = \arg\max_{x_i} u_{x_i}$
 5:     query an oracle for labels $y$ of $x^*$
 6:     remove $x^*$ from $\mathcal{U}$
 7:     insert $< x^*, y >$ into $\mathcal{L}$
 8: end while
```

In our approach, the confidence of labels is replaced by calculating the k best Viterbi paths to find tokens where predictions of the model differ in the k most likely sequences. The number of paths affects the behaviour of the algorithm, however, we assume this parameter to be less data dependent than confidence threshold. We call the approach *k-best Viterbi SeSAL*. The pseudocode of it is described in Algorithm (2).

**Algorithm 2** k-best Viterbi SeSAL algorithm

```
Given:
    $\mathcal{L}$: set of labeled examples
    $\mathcal{U}$: set of unlabeled examples
    $\phi_M$: utility function
    k: number of paths
Algorithm:
 1: while stopping criterion is not met do
 2:     learn model M from $\mathcal{L}$
 3:     for all $x_i \in \mathcal{U}$:$u_{x_i} \leftarrow \phi_M(x_i)$
 4:     select $x^* = \arg\max_{x_i} u_{x_i}$
 5:     find the k best Viterbi paths $\{v_1, ..., v_k\}$
 6:     for t in length($x^*$) do
 7:         if $v_i(t)$ for all $i = 1, \dots, k$ are equal then
 8:             label $x^*(t)$ with $y(t) = v_1(t)$
 9:         else
10:             query an oracle for a label $y(t)$ of $x^*(t)$
11:         end if
12:     end for
13:     remove $x^*$ from $\mathcal{U}$
14:     insert $< x^*, y >$ into $\mathcal{L}$
15: end while
```

The proposed approach uses the approach from the FuSAL active learning framework to find the most informative instance (lines 2-4). Then, the semi-supervised learning is applied in order to label the instance. The algorithm computes the k best Viterbi sequences that are used to detect not likely labels (line 5).

The Viterbi algorithm described in section 3.3 provides only one best sequence. To produce k best sequences it is not enough to store only one best label per node. The simplest way how to modify Viterbi algorithm is to store up to k best predecessors that can form k best sequences. Unfortunately, with this modification, the algorithm has the computational complexity of $\mathcal{O}(kTN^2)$. This computational overload can be lowered by the iterative Viterbi A* algorithm which has the complexity of $\mathcal{O}(T + kT)$ in the best case and $\mathcal{O}(TN^2 + kTN)$ in the worst case [19].

With the k-best Viterbi paths found, the algorithm loops trough the decoding (lines 6-12). The label is accepted only if all sequences produced it. Otherwise, a human annotator (oracle) is called to label the instance.

## 5 Experiment and results

In this section, we describe an experiment used for the evaluation of the proposed method. The method is evaluated on an NLP task called part-of-speech tagging (POS). The input to the POS is a set of meaningful sentences. The output is a set of tag sequences, one tag for each word. Word classes (noun, verb, adjective, etc.) or their derivates are the most often used tagsets. The number of tags is not limited.

POS is a difficult task for two reasons. First, the number of possible words in the text can be very high, and it may contain words that occur rarely. Second, some words can have assigned several tags, and to find the correct tag, the context of the sentence is needed. CRFs can take a wide context into account and thus is the most commonly used in the POS. However, though it is impossible to take a wide context into account in HMM, it is a sufficiently good performing model for our experiment.

In our experiment, we used data from the Natural language toolkit [20], which provides data for many NLP tasks such as POS, chunking, entity recognition, information extraction, etc. A few statistics for the employed benchmark datasets are provided in Table 1. Each dataset contains its proper tagset and a simplified tagset with 12 tags representing ten basic word classes, a dot and the rest.

Table 1: Benchmark datasets.

| Dataset | #sentences | #words | #tags |
|---|---|---|---|
| Brown | 57340 | 56057 | 472 |
| CoNLL2000 | 10948 | 21589 | 44 |
| Treebank | 3914 | 12408 | 46 |

In order to compare the datasets, HMMs were trained using supervised learning on the full dataset with all labels available. Accuracy and the $F_1$ score measures were used for the performance comparison. The performance was measured for both the original tagset (Acc 1 and F-score 1) and the simplified tagset (Acc 2 and F-score 2). The data was randomly split into training and testing sets in a 7:3 ratio. The performance of the supervised learning is shown in Table 2. Due to the results in the table, we consider

HMM to be sufficiently well performing in the experiment. The worse F-score in the case of Brown dataset with all tags is caused an approximately ten times higher number of possible hidden values.

Table 2: Prediction performance learned on the full dataset. Training and testing data were randomly split in a 7:3 ratio. Acc 1 and F-score 1 represent results based on all tags, whereas, Acc 2 and F-score represent results based on simplified tags.

| Dataset | Acc 1 | Acc 2 | F-score 1 | F-score 2 |
|---|---|---|---|---|
| Brown | .9421 | .9572 | .4520 | .9245 |
| Conll2000 | .9508 | .9546 | .9080 | .9408 |
| Treebank | .9189 | .9307 | .8205 | .9291 |

## 5.1 Experimental setup

For most of the experiments we used the following settings. In the base model, HMM, tags were considered to be hidden state values and words were considered to be observable variable values. The parameters of the model were estimated using MLE. To handle words that have not occurred in the training set, we added uniformly distributed pseudo-counts to both matrices $A$ and $B$. Prior beliefs were set to be uniformly distributed, therefore, each word has the probability of $1/|words|$.

In order to simulate a standard situation in active learning, the original dataset was randomly split into training and testing sets in a 7:3 ratio and then, the training set was randomly split into labelled and unlabeled sets in a 3:7 ratio.

In each iteration of the experiment, the most informative instance was selected, annotated and put into the labelled training set. As most informative were considered instances maximizing the employed one of the following four uncertainty measures:

- least confidence

$$\phi_{LC}(x) = 1 - P(y_1^*|x;\theta),$$

- margin

$$\phi_M(x) = -(P(y_1^*|x;\theta) - P(y_2^*|x;\theta)),$$

- total token entropy

$$\phi_{TE}(x) = -\sum_{t=1}^{T}\sum_{n=1}^{N} P(y_t = n|x;\theta)\log P(y_t = n|x;\theta),$$

- k-best sequences entropy

$$\phi_{SE}(x) = -\sum_{\hat{y}\in V} P(\hat{y}|x;\theta)\log P(\hat{y}|x;\theta),$$

where $V$ is set of k-best Viterbi sequences and $y_k^*$ is the k-th most probable sequence of labels. The behaviour of different uncertainty measures is investigated in the experiment in Section 5.2.

After finding them most informative sequence, semi-supervised learning was applied. The sequence was labelled according to Algorithm 2. The algorithm has one parameter, the number of k best sequences. The effect of the parameter on the performance of the proposed approach is described in the experiment in Section 5.3.

## 5.2 Uncertainty measure

At first, we study effects of uncertainty measures on the proposed method. The measures were evaluated on the TreeBank dataset with 30% labeled instances. The parameter k was set to 100.

The experiment has shown that the computational complexity of the k-best sequence entropy measure and the margin measure is too high for practical usage due to the calculation of the k best Viterbi paths (two best Viterbi paths respectively) for each unlabeled instance. Moreover, active learning that uses as a measure the k-best sequences entropy had a tendency to choose short sentences. In that case, active learning had a lower accuracy than the random sampling method.

The computational complexity of least confident and total token entropy measures were reasonable even for datasets with a big number of unlabeled samples. The performance comparison is shown in Figures 1 and 2. According to the experiment results, FuSAL with the least confident measure achieved higher accuracy after 50 iterations. However, the total token entropy measure achieved the certain level of accuracy in less queried tokens which can be preferable for some tasks.

Taking into account the computational complexity of the methods, the least confidence measure is used in the rest of the experiment.

## 5.3 Parameter settings

In semi-supervised learning, a well performing model is crucial to produce good quality labels. In SeSAL algorithm, the parameter delay controls how many iterations of FuSAL algorithm is used before semi-supervised approach is applied. The goal of this experiment was an analysis of the relationship between the parameter delay and the parameter k. Since the proposed method does not use the delay parameter, it has been simulated using datasets with a different number of labelled samples. The experiment was evaluated on the biggest dataset, Brown, with three initial settings a) 10% of labelled samples, b) 30% of labelled samples, c) 60% of labelled samples.

It has been shown that the value of the parameter k is highly correlated with the number of labelled samples in the dataset. In the dataset with 10% of labelled samples, the high value of the parameter k has shown to be crucial

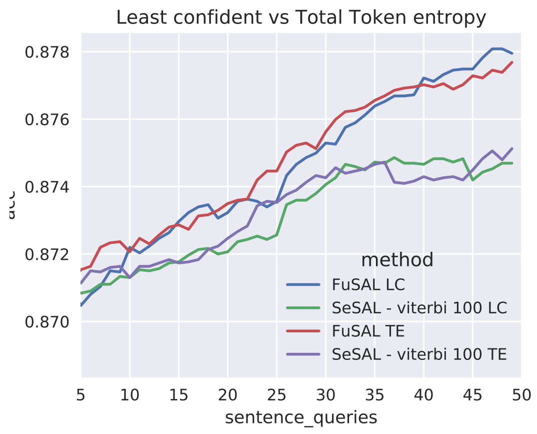

Figure 1: Comparison of the least confident measure (LC) and the total token entropy measure (TE) for different numbers of queries in connection with FuSAL and Viterbi SeSAL.

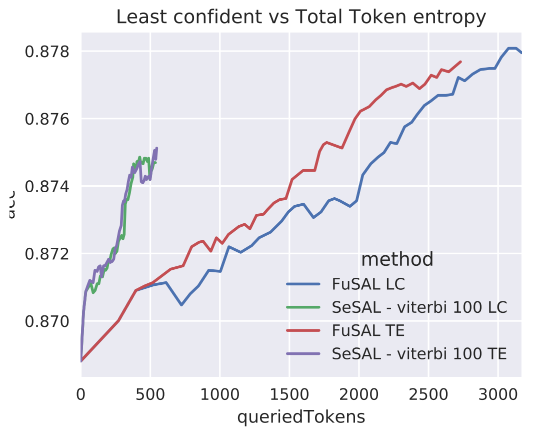

Figure 2: Comparison of the least confident measure (LC) and the total token entropy measure (TE) for different numbers of queries in connection with FuSAL and Viterbi SeSAL.

to reduce the number of errors propagated to the training dataset (Figure 3). With increasing number of labelled samples, a high value of the parameter k becomes less effective. In Figure 4 the difference between parameter k=100 and k=200 almost vanished. Moreover, regarding the number of queried labels, the settings k=100 becomes more efficient (Figure (5). The same trend was also observed in the case where 60% of instances were labelled.

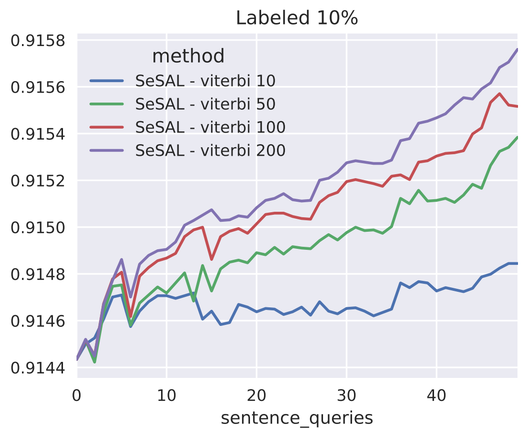

Figure 3: An accuracy regarding a number of queried sentences where 10% of the training set is labeled

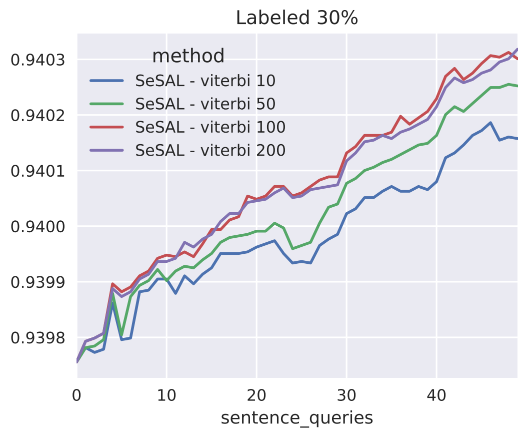

Figure 4: An accuracy regarding a number of queried sentences where 30% of the training set is labeled

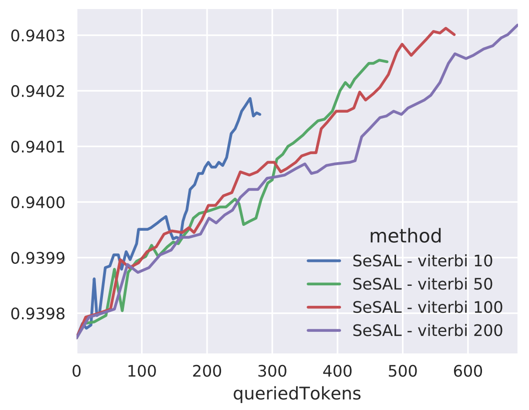

Figure 5: An accuracy regarding a number of queried tokens where 30% of the training set is labeled

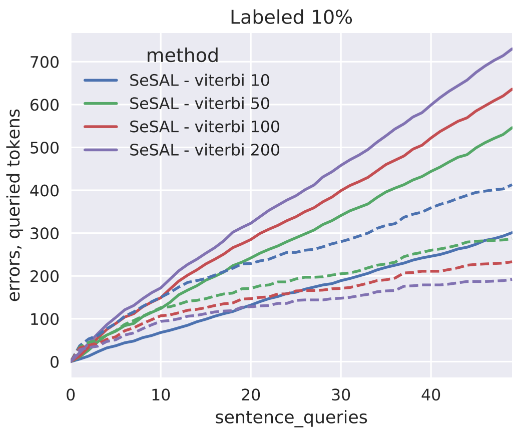

Figure 6: A number of queried tokens (solid line) and errors (dashed line) regarding a number of sentences where 10% of the training set is labeled.

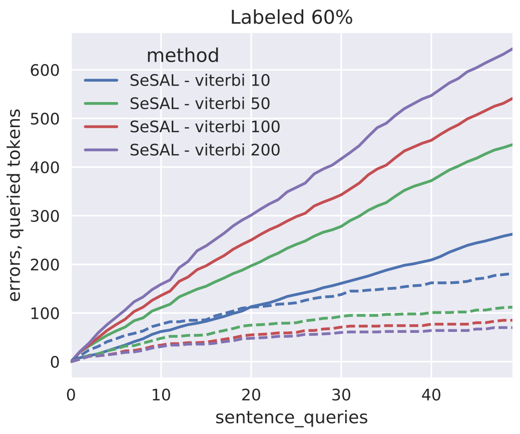

Figure 7: A number of queried tokens (solid line) and errors (dashed line) regarding a number of sentences where 60% of the training set is labeled.

## 5.4 Number of queried tokens and errors propagation

The parameter k affects the number of queried tokens and the number of errors propagated to the learning set. The optimal setting of the parameter minimises both. The experiment in this section analyses the relationship between tokens and errors.

One should consider the number od labelled samples in setting of the parameter k. In the case of less labelled samples, the parameter k should be set to a higher number to avoid production of errors (Figure 6). After several iterations, when the base model is more accurate, higher values of parameter k become less effective (Figure 7).

However, even with an almost labelled dataset and the settings k = 200 we were not able to avoid errors in labelling. From all 2402 annotated tokens, 57 were annotated wrongly. We consider the complicated control of an acceptable error rate as one of the biggest disadvantages of the proposed method.

## 5.5 Comparison with other methods

To evaluate the performance of the proposed method in comparison with other methods an accuracy was measured regarding the number of queried sentences and the number of queried tokens. Furthermore, the number of errors propagated to the learning set was measured. All experiments were evaluated on the Brown dataset with the simplified tagset.

The SeSAL with uncertainty threshold and the proposed method can be compared only if the parameters are set such that the methods produce an approximately same number of errors. In the experiment, confidence threshold was set to 0.48 and parameter of the number of paths k was set to 100.

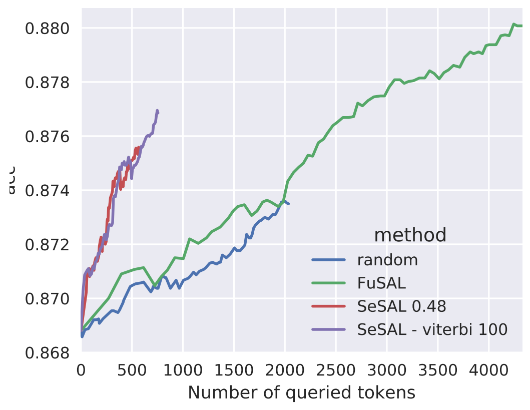

Figure 8: Achieved accuracy over the number of queried tokens.

As expected, the FuSAL method achieved the highest accuracy because all labels were annotated manually, thus correctly. In the number of queried tokens, Viterbi SeSAL achieved bigger accuracy in more queried tokens (Figure 8). The explanation can be seen in Figure 9 where the number of errors and the number of queried tokens was measured. In the given settings, the number of errors propagated to the learning set was lower in Viterbi SeSAL at the expense of the number of queried tokens. Although, after several iterations, the error rate of the proposed method has been lower than in the SeSAL method.

## 6 Conclusion and future work

We proposed a semi-supervised active learning method that is easy to setup for the sequence labelling and is suf-

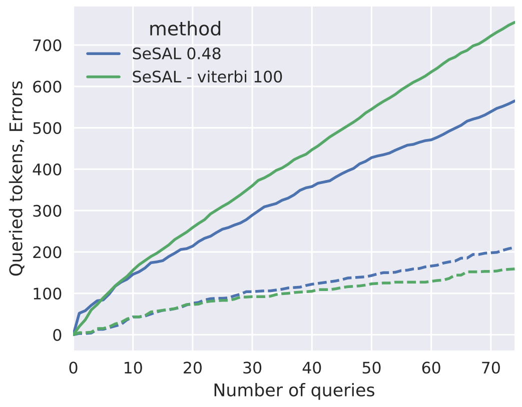

Figure 9: The number of queried tokens (solid line) and the number of errors (dashed line) over the number of queries.

ficiently well performing in comparison with the semi-supervised active learning method that use an uncertainty threshold and a marginal probability. The proposed method uses k best Viterbi paths to find the tokens in which the model is not sufficiently confident.

The number of errors, the number of queried tokens and the computational complexity are controlled by the parameter k. In order to reduce the number of errors propagated to the labelling set, the parameter k should be set as high as it is reasonable in terms of the computational time. The computational complexity of k-best Viterbi path algorithm can be partially reduced using iterative Viterbi A* algorithm. In addition to a high computation complexity, a complicated control of the number of propagated errors is disadvantage of the proposed method.

An area for further research is the exploration of Co-training in combination with the Query-by-Committee active learning framework where both approaches consider several different views of the data. Furthermore, the semi-supervised active learning method that can be applied to both probabilistic and deterministic sequential models should be more studied to find a general solution for them.

## Acknowledgements

The reported research was supported by the CTU grant nr. SGS17/210/OHK3/3T/18 and by the Czech Science Foundation grant nr. 17-01251.

## References

[1] Burr Settles. Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11, 2010.

[2] Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1070–1079. Association for Computational Linguistics, 2008.

[3] David D Lewis and William A Gale. 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. Springer-Verlag New York, Inc., 1994.

[4] Gokhan Tur, Dilek Hakkani-Tür, and Robert E Schapire. Combining active and semi-supervised learning for spoken language understanding. *Speech Communication*, 45(2):171–186, 2005.

[5] Cynthia A Thompson, Mary Elaine Califf, and Raymond J Mooney. Active learning for natural language parsing and information extraction. In *ICML*, pages 406–414, 1999.

[6] Lin Yao, Chengjie Sun, Shaofeng Li, Xiaolong Wang, and Xuan Wang. Crf-based active learning for chinese named entity recognition. In *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pages 1557–1561. IEEE, 2009.

[7] Ido Dagan and Sean P Engelson. Committee-based sampling for training probabilistic classifiers. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 150–157. The Morgan Kaufmann series in machine learning,(San Francisco, CA, USA), 1995.

[8] David D Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the eleventh international conference on machine learning*, pages 148–156, 1994.

[9] H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294. ACM, 1992.

[10] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 189–196. Association for Computational Linguistics, 1995.

[11] Ellen Riloff, Janyce Wiebe, and Theresa Wilson. Learning subjective nouns using extraction pattern bootstrapping. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 25–32. Association for Computational Linguistics, 2003.

[12] Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. Semi-supervised self-training of object detection models. 2005.

[13] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM, 1998.

[14] Andrew M. Dai and Quoc V. Le. Semi-supervised sequence learning. *CoRR*, abs/1511.01432, 2015.

[15] Shi Zhong. Semi-supervised sequence classification with hmms. *International Journal of Pattern Recognition and Artificial Intelligence*, 19(02):165–182, 2005.

[16] David Pierce and Claire Cardie. Limitations of co-training for natural language learning from large datasets. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 1–9, 2001.

[17] Katrin Tomanek and Udo Hahn. 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: Volume 2-Volume 2*, pages 1039–1047. Association for Computational Linguistics, 2009.

[18] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[19] Zhiheng Huang, Yi Chang, Bo Long, Jean-Francois Crespo, Anlei Dong, Sathiya Keerthi, and Su-Lin Wu. Iterative viterbi a* algorithm for k-best sequential decoding. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 611–619. Association for Computational Linguistics, 2012.

[20] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, 2009.