Deep Active Learning for Text Classification

Bang An  
Beijing University  
Beijing, China  
No.37 XueYuan Road  
(0086)18612210095  
anbang@buaa.edu.cn

Wenjun Wu  
Beijing University  
Beijing, China  
No.37 XueYuan Road  
(0086)13641084384  
wwj@nlsde.buaa.edu.cn

Huimin Han  
Beijing University  
Beijing, China  
No.37 XueYuan Road  
(0086)13051087129  
hanhuimin@nlsde.buaa.edu.cn

ABSTRACT
In recent years, Active Learning (AL) has been applied in the domain of text classification successfully. However, traditional methods need researchers to pay attention to feature extraction of datasets and different features will influence the final accuracy seriously. In this paper, we propose a new method that uses Recurrent Neural Network (RNN) as the acquisition function in Active Learning called Deep Active Learning (DAL). For DAL, there is no need to consider how to extract features because RNN can use its internal state to process sequences of inputs. We have proved that DAL can achieve the accuracy that cannot be reached by traditional Active Learning methods when dealing with text classification. What’s more, DAL can decrease the need of the great number of labeled instances for Deep Learning (DL).

At the same time, we design a strategy to distribute label work to different workers. We have proved by using a proper batch size of instance, we can save much time but not decrease the model’s accuracy. Based on this, we provide batch of instances for different workers and the size of batch is determined by worker’s ability and scale of dataset, meanwhile, it can be updated with the performance of the workers.

CCS Concepts  
• Computing methodologies→Maching learning→Learning settings→Active learning settings

Keywords  
Active Learning; Deep Learning; Machine Learning; Artificial Intelligence; Text Classification.

1. INTRODUCTION  
In the past few years, Machine Learning has made great progress in a lot of application areas. However, as time goes by, there have been many new fields and new issues for researchers to explore. In many real-word applications, the labels are difficult to obtain because of the scarce of experts or the lack of money. In order to deal with these issues, Active Learning is proposed.

Active Learning is a setup that allows the learning algorithm to iteratively and strategically query the labels of some instances for reducing human labeling efforts.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ICVISP 2018, August 27–29, 2018, Las Vegas, NV, USA  
© 2018 Association for Computing Machinery.  
Copyright 2018 ACM ISBN 978-1-4503-6529-1/18/08 $15.00  
DOI: https://doi.org/10.1145/3271553.3271578

The model is trained on a small number of datasets firstly, and an acquisition function (often based on the model’s uncertainty) decides which instance to select. After an oracle (often a human expert) labels the selected instance, it will be added to the training dataset, and a new model will be trained on the updated training dataset. The process will be repeated again and again with the training dataset increasing in size over time Error! Reference source not found.

Text classification is a classic topic for Natural Language Processing (NLP) and has many applications in topics such as parsing, semantic analysis, information extraction and web searching Error! Reference source not found. Nowadays the core task in text classification is how to present features. In spite of the fact that many researchers have developed some more complex methods in order to extract more contextual information and accurate word order, but there still exist few issues such as data sparseness or data insufficiency which impact the classification accuracy deeply.

It’s well known that the key idea lying behind active learning is a machine learning algorithm can achieve greater accuracy with fewer training labels if it is allowed to choose data from which it learns Error! Reference source not found. Although traditional AL has been applied in the domain of text classification successfully, also by definition fails to address some important questions:

First, it doesn’t solve with the feature extraction issue in text processing. It means researchers still need to compare with different methods in order to select which one is best, because the choice of method will impact the result of the model training seriously.

Second, it always costs too much time to finish the task and is not effective to assign label works to many workers. The size of train dataset will be bigger and bigger as the acquisition function selects instance one by one, and a bigger train dataset means it would take more time for the model to train one time. Meanwhile the labeling task cannot be assigned to many workers at the same time efficiently. On the one hand, the workers have to wait until the acquisition function selects the instance, on the other hand, it will be a big challenge for the model to response if too many workers label at the same time.

In recent years, the rapid development of word embedding and deep neural networks has made a great progress to various NLP tasks especially for text classification. RNN can analyze a text word by word and store the semantics of all the previous text in hidden layer, with this advantage it can capture the contextual information easily. Which means it may have the ability to achieve higher accuracy in classification than traditional method. What’s more, with the help of word embedding, some composition-based methods are proposed to capture the semantic representation of texts Error!
By this way, researchers don’t need to spend time on feature extraction any more. However, it is obvious that deep neural networks depend on a big number of labeled instances and too much computational resource.

Based on these issues, we consider to combine Deep Learning with Active Learning which we call it Deep Active Learning. We select RNN because it has the advantage of capturing semantics of variable-length texts. Based on the advantage of RNN, we don’t need to spend time on comparing different algorithms of feature extraction and can achieve lower error rates. With the advantages of Active Learning, we can save around 50% labor to get the labeled instances Error! Reference source not found.. What’s more, we improve the efficiency of traditional Active Learning, we select batch-mode for Active Learning and design a new strategy that can distribute label work to different workers.

This paper is organized as follows: section 2 introduces the related work for traditional Active Learning, text classification and deep learning. In section 3 we propose two models, one is traditional active learning model for text classification and the other is DAL model. In section 4 we design two parts of experiments, one for demonstrating our DAL can achieve higher accuracy and the other for demonstrating that our distribute strategy can not only save time but also maintain the performance of traditional AL methods. Finally, we conclude in section 5 with some discussion.

2. Related Work

2.1 Active Learning Strategies

There are two setups of active learning for multiclass classification: stream-based and pool-based. For stream-based setup, the instances come in sequence and the algorithm has to decide whether to query the label of the instance or discard it. By contrast, the pool-setup is more flexible when we try to deal with practical problems, because in real world we often gather all unlabeled data at once instead of one by one. There are two pools for pool-based setup, one called the labeled pool and the other called the unlabeled pool. We treat the whole data pool including n instances by using $D = \{x_1, x_2, \ldots, x_n \cup \{(x_{n+1}, y_{n+1}), \ldots, (x_m, y_m)\}\}$, where the input instance $x_i \in \mathbb{R}^d$ and $y_i$ is the label of $x_i$. The $D$ is the combination of the unlabeled pool $D_u$ which contains the first $n_u$ instances, and the labeled pool $D_l$ which contains other instances. The algorithm will select the most informative instance from the $D_u$ to query its label, then the instance will be moved to the $D_l$. The next model will be trained with the updated $D_l$ and finally select the new instance circularly. In the widely used pool-based approach, we start with a small labeled training set $D_1$ and a large pool of unlabeled data $D_u$.

The choice of instance to be labeled is done through an acquisition function, which ranks points based on their potential informativeness. There are some frameworks for measuring informativeness such as uncertainty sampling and query-by-committee (QBC).

The main idea of uncertainty sampling is to select the instance which is the least certain to label. Probabilistic model will be used in this algorithm, for example, a more general uncertainty sampling variant might query the instance whose prediction is the least confident as equation (1):

$$x_{LC}^* = \arg \max_{x \in \mathbb{R}^d} P(y|x)$$

where $y = \arg \max_{y \in \mathbb{R}^d} P(y|x)$, or the class label with the highest posterior probability under the model $\theta$ Error! Reference source not found.. Another general active learning framework is QBC. We use a committee of models $c = \{\theta^{(1)}, \ldots, \theta^{(C)}\}$ to represent $C$ different hypotheses that are consistent with $D_l$. The most informative query is the instance over which the committee is in most disagreement about how to label.

Vote Entropy (VE) is the measure to select the instance in QBC and the formula is as equation (2):

$$x_{VE}^* = \arg \max_{x \in \mathbb{R}^d} \frac{1}{C} \sum_{c=1}^{C} \log \frac{P(y|x)}{P(y|x)}$$

where $V(y, m)$ is the number of “votes” label $m$ receives from all the committee member’s label work at sequence position $t$ Error! Reference source not found.. Compared with these two popular frameworks, although in some scenarios the QBC is proved more effective than uncertainty sampling, we have to commit the time it needs is a few times longer because it will go through many “votes” from multiple committee members. As we know Deep Learning is time consuming and dependent on computing resources, so it is not easy to combine with QBC.

2.2 Text classification

Text classification is very important nowadays with the recent explosion of available text data on the Internet, it mainly focuses on three topics: feature engineering, feature selection and using different types of machine learning algorithms. For feature engineering, the most widely used feature is the bag-of-words. In addition, some more complex features have been designed such as part-of-speech tags, noun phrases and tree kernels Error! Reference source not found.. Compared with bag-of-words models, a text is often represented as the bag of its words, which means disregard sentence grammar and even word order but only keep multiplicity.

Term Frequency Inverse Document Frequency (TF-IDF) is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus Error! Reference source not found.. Higher TF-IDF values words means that have imply they have a stronger relationship in the document which they appear. And it has been proved that the combination of bag-of-words and TF-IDF can outperform either bag-of-words or TF-IDF Error! Reference source not found.. After feature extraction the machine learning algorithms often use classifiers such as logistic regression (LR), naive bayes (NB) and support vector machines (SVM). Compared with logistic regression, SVM can analyze data used for multiclass classification, linear classification and non-linear classification using what is called the kernel trick. Besides, different experimental results have proved that SVM performs better than NB in general classification tasks, and average improvement is +6.36% or even +28.78% by using SVM Error! Reference source not found.. Although SVM is useful in many cases of classification, it is very sensitive to outliers and noises in the training dataset, meanwhile, cannot realize the feature selection. And Deep Learning can solve with these issues.

2.3 Deep Learning for text classification

Deep Learning is part of a broader family of machine learning methods based on learning data representations. The famous architectures such as Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) have been successfully applied to fields including speech recognition, computer vision, text classification and so on.
CNN has made a great success in the domain of text classification, it has been trained with one layer of convolution on top of word vectors obtained from a neural language model. Compared with CNN, the advantage of RNN is the ability to better capture the contextual information. This could be beneficial to capture semantics of variable-length texts. Gating mechanisms have been developed for basic RNN, resulting in two prevailing types one called long short-term memory (LSTM) and the other called gated recurrent unit (GRU).

It has been proved RNNs perform well and robust in a broad range of tasks except when the task is essentially a key-phrase recognition task as in some sentiment detection and question-answer matching settings. But current literature does not support such a clear conclusion which is better between LSTM and GRU.

There is no doubt that Deep Learning depends on a larger number of dataset than traditional method. What’s worse, obtaining more labeled instances is expensive and even unrealistic in practical applications. So we consider to take advantage of the concept of Active Learning. By this way, the RNN will select most instances to train every batch and just need a smaller number of labeled instances to achieve final accuracy than traditional ways.

2.4 Batch-Mode for Active Learning

In traditional Active Learning, only one instance will be selected at a time which lead train speed of model is very slow. By contrast, batch-mode active learning allows the learner to query instances in groups, which is better suited to parallel labeling environments or models with slow training procedures. It is more feasible because it does not require model to be re-trained after an oracle finish one selection, by this way the batch-mode can make good use of human labor.

There are some approaches for batch-mode. And a simple strategy is to select the top k most informative examples called batch-greedy. Which means the approach will greedily select examples with a batch, and assemble batches in a greedy manner. Other methods such like using the Fisher information matrix for the measurement of model uncertainty or optimal batch-model policy have been proved using batches only incurs a bounded increase of cost as compared to allowing fully sequential selection and more effective. However, all the algorithms only focus on how to select instances or which instance to select. They fail to answer how many instances to select one batch.

3. Proposed Model

3.1 Traditional Active Learning

3.1.1 Feature Extraction

To deal with the raw input from the train dataset, we use the method which combines of bag-of-words and TF-IDF. First we select the top 3,000 words that shows most frequent in the bag-of-words, and then calculate TF-IDF.

As the term implies, TF-IDF calculates values for each word in a document through an inverse proportion of the frequency of the word in a particular document to the percentage of documents the word appears in. So it is very useful in text mining. There are some different methods for TF-IDF such as boolean frequencies, term frequency adjusted for document length and raw count, and the formula we select is as equation (3):

\[
tf idf(t, d, D) = tf(t, d) \times idf(t, D) = \frac{f_{t,d}}{\sum_{i \in D} f_{i,d}} \times \log \frac{N}{|\{d \in D : t \in d\}|}
\]

where N is the total number of documents in the corpus \(N = |D|\), and \(|\{d \in D : t \in d\}|\) number of documents where the term \(t\) appears. After dealing with the dataset we select SVM as the classifier.

3.1.2 Support Vector Machines as the classifier

For text classification, we will always receive the training dataset \(\{x_1, x_2, x_3, ..., x_n\}\) and their labels \(\{y_1, y_2, y_3, ..., y_n\}\) at the same time. If all the training data are vectors in some space \(X \subseteq \mathbb{R}^d\), then all the training instances lie close to the hyperplane called support vectors. And more generally, SVM allow one to project original training instance in space \(X\) to a higher dimensional feature space \(\mathcal{F}\) via a kernel operator \(K\). In a word, the set of classifiers is equation (4)

\[
f(x) = \left(\sum_{i=1}^{n} \alpha_i K(x_i, x)\right)
\]

The kernel we select is Radial Basis Function (RBF), it uses the equation (5) to represent the feature vectors in input space for two samples \(x\) and \(x'\):

\[
K(x, x') = \exp \left(-\frac{||x-x'||^2}{2\sigma^2}\right)
\]

3.1.3 Measures to select instances

Because we aim to deal with multiclass classification, the least confidence method has a noticeable disadvantage. Imagine there are three classes, the probability that instance \(a\) belongs to three classes is 0.5, 0.5, 0.0 and instance \(b\) 0.25, 0.25, 0.25. The least confidence will give the same value for instance \(a\) and instance \(b\), however, it’s obvious that instance \(a\) is more uncertain than instance \(b\). Which means the instance with highest score may not be the most informative one.

In order to address this problem, we select to use the method called margin sampling, as equation (6) show:

\[
x_{MS} = P_0(\hat{y}|x) - P_0(\hat{\hat{y}}|x)
\]

where \(\hat{\hat{y}}\) is the second most probable class. Every turn we calculate informativeness among all instances and use the batch-greedy method to select instances in batch.

3.2 Deep Active Learning

Based on these related works we mentioned above, we propose a new model which uses RNN as the acquisition function in AL, and the structure is shown in figure 1:

![Figure 1. The structure of DAL.](image-url)
3.2.1 Word Embedding to deal with the input data
To deal with the input data, we use the word embedding and set the embedding size 800 in all cases. When given a sentence fragment \( F = \{w_1, w_2, ..., w_T\} \), where \( T \) is the sentence length, we need to get its one-hot vector representation \( v_t \) first and then transform each word in fragment into a real-valued vector \( e_t \) via looking up the embedding matrix \( W^{\text{word}} \in \mathbb{R}^{d_w \times V} \). That is equation (7):
\[
e_t = W^{\text{word}} v_t
\]
where \( d_w \) is the size of word embeddings and \( V \) is the size of vocabulary. Moreover, the embedding matrix \( W^{\text{word}} \) needs to be learned. With the help of word embedding, some composition-based methods are proposed to capture the semantic representation of texts.

3.2.2 The structure of Deep Neural Networks
RNN are sequence-based models of key importance for natural language understanding, language generation and many other tasks. The input of the model is seen as a sequence of symbols, where at each time step a simple neural network is applied to a single symbol, as well as to the network’s output from the previous time step.

By this way, we don’t need to consider how to extract features from the input data. For example, given input sequence \( x = [x_1, x_2, ..., x_n] \) of length \( N \), a simple RNN is formed by a repeated application of a deterministic function \( f_k \). This generates a hidden state \( h_t \) for time step \( t \) as shown:
\[
h_t = f_k(x_t, h_{t-1}) = \sigma(x_t W_h + h_{t-1} U_h + b_h)
\]
where \( W \) is the weight matrix. For some non-linearity \( \sigma \). The model output can be defined such as equation (9):
\[
f_y(x) = h_T W_y + b_y
\]
As we mentioned in section 3.2, there is no such a clear conclusion which is better between LSTM and GRU. So we set two models for DAL, one core is LSTM and the other is GRU.

LSTM use different gates within the RNN units. The LSTM maintains a separate memory cell inside that updates and exposes its content only when necessary. The input at time \( t \) is \( x_t, h_{t-1}, c_{t-1} \), and output is \( h_t, c_t \), all of them can be updated by equations (10-15) as shown:
\[
i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)
\]
\[
f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)
\]
\[
o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)
\]
\[
g_t = \tanh(W_g x_t + U_g h_{t-1} + b_g)
\]
\[
c_t = f_t \odot c_{t-1} + i_t \odot g_t
\]
\[
h_t = o_t \odot \tanh(c_t)
\]
At the same time GRU models the word sequence \( x \) as follow equations (16-19):
\[
z = \sigma(x_t U_z + h_{t-1} W_z)
\]
\[
r = \sigma(x_t U_r + h_{t-1} W_r)
\]
\[
s_t = \tanh(x_t U_s + (h_{t-1} \odot r) W_s)
\]
\[
h_t = (1 - z) o_t + z o_{t-1}
\]
where \( \sigma \) denotes the logistic sigmoid function and \( \odot \) denotes the element-wise vector product. And \( U \in \mathbb{R}^{d_h}, W \in \mathbb{R}^{h \times d_h} \).

Next, we add a dropout layer after each RNN layer so as to avoid over-fitting problem. Through many experiments the value of the probability of retaining a unit \( p \) we select is 0.8 as shown.

Finally, there are two fully connected layers in our structure to deal with the high-level reasoning. The first one has connections to the output of RNN layer. Connected with the first fully connected layer is another dropout layer and relu layer. After relu layer applies the non-saturating activation function \( f(x) = \max(0, x) \), the second fully connected layer connects with the modified Softmax layer.

3.2.3 Method to select instances
As we mentioned above, uncertainty sampling is a very straightforward probabilistic learning model. So we decide to combine Softmax with the uncertainty sampling (equation 20) to select the instances with most informativeness based on what we mentioned in section 2.1 and section 3.1.3:
\[
[p_0(y = 1|x), ..., p_0(y = C[x, \theta])] = \text{Softmax}(f^\theta(x))
\]
Through this function we will get the probability that an instance belongs to each class. The least confident instance is the one whose probability in each category is closest. The calculate formula is as equation (21):
\[
x_{softmax}^\theta = p_0(y|x) - p_0(\bar{y}|x)
\]
where \( \bar{y} \) belongs to \( \{1, 2, ..., C\} \) as equation (22) show,
\[
p_0(\bar{y}|x) = \text{Softmax}(f^\theta(x))_c
\]
By this way, we can get the most informative instances in batch and then add them into the labeled pool, the RNN model will be retrained with the updated train dataset iteratively.

4. Experiment
We conducted two parts of experiments to validate the proposed framework. For the first experiment we planned to prove that DAL can achieve the higher accuracy with fewer labeled instances. We selected the public datasets named THUCNews which contains 10 class and 10000 instances as shown. We compared LSTM core and GRU core in the beginning and selected the better one to join the comparative experiments.

In the second experiment we planned to validate that by using the proper batch size for AL we can save much time but not decrease the accuracy. And base on the experiment we want to find relation between time-consuming and batch number in order to design the distribute strategy. We also used the same dataset but decreased the number of data to 3000 because the size of data was not key factor.

4.1 The experiment of DAL
We designed three sets of experiments:

For Traditional AL, we used bag-of-words and TF-IDF to deal with the dataset, we selected SVM as the classifier and uncertainty sampling with margin sampling. The structure was shown in section 3.1.

For Random, we used bag-of-words and TF-IDF to deal with the dataset, we selected SVM as the classifier but selected the instances randomly.

For DAL, we didn’t need to extract features as we mentioned above, so we just used word embedding and finally modified the uncertainty sampling with Softmax. The structure was shown in section 3.2.
As we mentioned in section 2.1, both the DAL and Traditional AL first selected the same labeled training set $D_l$ to pre-train. And then selected the most informative instances in the same batch size and the size is 64 from $D_u$ iteratively. The procedure of experiments is as figure 2:

**Figure 2. The procedure of experiments.**

First of all, just as what we said in section 2.3, we need to compare which one is better between LSTM core and GRU core in DAL, all experiment conditions are the same except the RNN core. The result is as table 1:

### Table 1. Comparison between LSTM and GRU.

<table>
<thead>
<tr>
<th></th>
<th>LSTM core</th>
<th>GRU core</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1 value</td>
<td>85.6</td>
<td>90.3</td>
</tr>
<tr>
<td>Time-consuming</td>
<td>19h 57mins</td>
<td>21h 32mins</td>
</tr>
</tbody>
</table>

As we can see from table 1, although GRU takes more than 1 hour than LSTM, GRU behaves better with the dataset in DAL. So we choose GRU as the RNN core and start to do the comparison experiment finally.

And the result of experiments is as figure 3:

**Figure 3. The result of the first experiment.**

From figure 3 we can see the model can not only achieve higher accuracy after each label turn, but also can achieve the final accuracy that traditional models cannot reach. What’s more, it’s noticeable that after 40 rounds both the traditional AL and the DAL have reached a high stable accuracy, which means DAL can reach final accuracy by using only 45% of the initial dataset with the advantage of Active Learning.

However, we have to admit the DAL will spend three times as much time as traditional AL, besides it depends on large number of data to pre-train. Time-consuming of three sets of experiments is shown in table 2.

### 4.2 The experiment of Different Batch Size

First we wanted to prove that using different batch size would save time but not decrease the model’s accuracy so we designed three sets of experiments:

The first one we selected batch size for only one instance, the second one 16 instances and the third one 64 instances. And because the first experiment would cost too much time, we decreased the size of dataset to 3,000 instances.

As we can see from the figure 4, there is no significant difference in accuracy among the three experiments. However, the difference in time-consuming is very obvious which is shown in the table 3:

### Table 2 Time-consuming of the first experiment

<table>
<thead>
<tr>
<th></th>
<th>TAL</th>
<th>Random</th>
<th>DAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time-consuming</td>
<td>9h 9mins</td>
<td>3h 10mins</td>
<td>21h 55mins</td>
</tr>
</tbody>
</table>

### Table 3. Time-consuming of the second experiment.

<table>
<thead>
<tr>
<th></th>
<th>Batch 1</th>
<th>Batch 16</th>
<th>Batch 64</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time-consuming</td>
<td>18h 50mins</td>
<td>1h 14mins</td>
<td>19mins</td>
</tr>
</tbody>
</table>

It’s obvious that it will save 99% time by using batch size 64 instead of only one instance. Although there is no significant difference among accuracy results, it doesn’t mean the bigger the batch size, the better. On the one hand, it will not reflect the advantages of active learning if the size is too big (consider to return all unlabeled instances in one turn). On the other hand, the label job of batch size needs to fit the label worker’s ability in real application. In order to discuss the relation between time-consuming and batch size, we also record the time-consuming in each turn which is shown as figure 5:
We propose a novel framework for active learning which we call it Deep Active Learning. In DAL we don’t need to consider how to extract features because RNN can capture the contextual information, meanwhile it can achieve a higher accuracy than traditional AL when dealing with the text classification problem. What’s more DAL can reach the high stable precision by only using 45% of the initial dataset. But we have to commit DAL will consume much more time and need more labeled instances to pre-train than traditional AL method.

Refer to DAL, we also modify the traditional framework of AL, we design a strategy that assign label work to different workers. We select the batch size of instances from the unlabeled pool and the batch size is determined by the worker’s ability and the scale of unlabeled pool. What’s more it can be updated as the performance of workers. By this way we can save much time but not decrease the model’s accuracy.

**REFERENCES**


Authors’ background

<table>
<thead>
<tr>
<th>Your Name</th>
<th>Title*</th>
<th>Research Field</th>
<th>Personal website</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bang An</td>
<td>Master student</td>
<td>Active Learning</td>
<td></td>
</tr>
<tr>
<td>Wenjun Wu</td>
<td>Full professor</td>
<td>Machine Learning, Big data, Artificial Intelligence</td>
<td></td>
</tr>
<tr>
<td>Huimin Han</td>
<td>Master student</td>
<td>Active Learning</td>
<td></td>
</tr>
</tbody>
</table>

*This form helps us to understand your paper better, the form itself will not be published.

*Title can be chosen from: master student, Phd candidate, assistant professor, lecture, senior lecture, associate professor, full professor