Predicting User Competence from Text

Yonas Woldemariam

Department of Computing Science
Umeå University, Sweden
yonasd@cs.umu.se

Abstract. We explore the possibility of learning user competence from a text by using natural language processing and machine learning (ML) methods. In our context, competence is defined as the ability to identify the wildlife appearing in images and classifying into species correctly. We evaluate and compare the performance (regarding accuracy and F-measure) of the three ML methods, Naive Bayes (NB), Decision Trees (DT) and K-nearest neighbors (KNN), applied to the text corpus obtained from the Snapshot Serengeti discussion forum posts. The baseline results show, that regarding accuracy, DT outperforms NB and KNN by 16.00%, and 15.00% respectively. Regarding F-measure, K-NN outperforms NB and DT by 12.08% and 1.17%, respectively. We also propose a hybrid model that combines the three models (DT, NB and KNN). We improve the baseline results with the calibration technique and additional features. Adding a bi-gram feature has shown a dramatic increase (from 48.38% to 64.40%) of accuracy for NB model. We achieved to push the accuracy limit in the baseline models from 93.39% to 94.09%.

1 Introduction

We evaluate and compare machine learning (ML) based models that predict the competence level of users from the text they have written. The main purpose of this study is to identify an effective model and integrate into a cross-media analysis framework as a meta data extractor. In general the model can be used in other platforms where the proficiency evaluation of users is needed from their posted texts.

We consider texts obtained from Snapshot Serengeti\(^1\) (SNS) discussion forum posts. The selected ML models are Decision Trees (DT), Naive Bayes (NB), and K-nearest neighbors (KNN). They have been proposed to be studied in our earlier work \([12]\), we studied two sentiment analysis methods, namely the lexicon-based and recursive neural tensor network by applying them on the SNS forum posts. As part of the extension of \([12]\), we study the problem of competence analysis as an advanced form of simple sentiment (positive/negative) detection and analysis by using the same dataset. The methods resulted from these studies are intended to support text analysis tasks in MICO\(^2\). MICO is an emerging solution for analyzing, annotating and publishing media resources.

\(^1\) https://www.snapshotserengeti.org
\(^2\) https://www.mico-project.eu
We analyze the texts generated by volunteer users of the SNS. Users are provided with randomly selected images of the wildlife in the Serengeti National Park, Tanzania, and expected to classify each image into one of 48 species [4]. Users then discuss what they observe in each image with their own texts in the forum. Administrators of the SNS project are interested to assess how well their users perform in classifying images, to understand and manage the users better. As textual contents constitute a big part of the profiles of users, new methods are needed for the exploitation of this content to detect and identify the competence of users. Thus, our main objective is to propose an ML-based model that predicts a competence level of users from their text.

The remainder of this paper is organized as follows: Section 2 discusses the notion of competence in the context of this study. Section 3 presents related literature review. Section 4 describes the corpus used in this study and 5 discusses the ML methods selected to be studied. Section 6 discusses the evaluation and comparison of the selected ML methods. Section 7 discusses the result. The final section describes future works.

2 The Notion of Classification Competence

The users of SNS classify a number of images and discuss the respective classifications. Competence is defined as the ability to identify the animals appearing in the images and classifying them into species correctly. For example, the animal in an image looks like a Weasel of some kind, one user might classify it as a Mongoose, whereas another user might confuse it with a Zorilla as the two species have some characteristics in common. Obviously, experts in the field tend to make more accurate classifications than non-expert volunteers. However, there is no any record in the SNS database showing the expertise of users.

To assess the expert level of the users, a majority vote scheme has been applied to the classifications of each image. That means that, each image is shown to multiple volunteer users then majority votes are taken as an accurate classification of the animal appeared on the image. Depending on the number of correctly classified images carried out by individual users, a weight is assigned to each user as an overall performance using a rating scale from 0 (the least competent users) to 1 (the most competent users) is used.

Obviously, relying on the majority votes as a ground truth has a number of problems, for example an image might contain an animal which is a bit hard to be discerned by non-expert users, but it probably receive majority votes for inaccurate classification than expert users. Thus, we need to come up with a better approach, for example by adapting previously proposed algorithms [4]. Authors in [4], carried out a quite detailed analysis to develop a weighted majority voting method for combining users annotations into an overall classification. A mechanism has also been devised for handling blank classifications, where some very blurry images are reported as blank by some users, but those images might contain animals.
3 Related Work

The most related works in the area of competence analysis from textual contents include [1, 5]. In [1], the use of ML and NLP methods to evaluate the competence of medical students from their clinical portfolio has been discussed. Specific competence goals have also been defined according to the competence-based curriculum practiced in the medical schools in USA. That allows them to identify the potential features associated with competence and extract those features from students notes. Moreover, they make use of available resources such as unified medical language system (UMLS) and knowledgeMap concept indexer (KMCI), that make the modeling a bit easier. On the contrary, we do not have such domain-specific resources. Thus, we make use of methods that automatically extract useful patterns representing competence from annotated training data without using external knowledge sources, e.g bag-of-words models. However, then we end up with a very large number of features, as each word in the corpus is a feature. Compared to other supervised learning methods, DT, NB and KNN are better alternatives due to their efficiency for data in high dimensional space.

A preliminary study has been carried out in [1] to apply ML-based methods to identify student experiences in different competence medical domains. They use a well-defined set of competence goals recommended by the two national accreditation bodies in USA including the Accreditation Council for Graduate Medical Education and the American Association of Medical Colleges. The proposed approach utilizes the medical domain specific models such as UMLS and KMCI to detect biomedical concepts from students notes. In their experiments, they trained three ML-based classifiers, NB, SVM and LR on an annotated corpus consists of 399 clinical notes. Their result shows that the performance of the ML methods vary across the competence domains identified for medical students.

A comprehensive survey about the existing state of the art approaches for automatic essay scoring has been presented in [5]. Various learning techniques used in earlier studies have also been discussed in the survey, such as classification and regression. Regression-based approaches include support vector machine (SVM). Where as, the classification-based approach includes NB and KNN.

As argued in Section 1, we apply the selected ML methods to the preprocessed (with natural language processing (NLP) tools) and annotated dataset of the SNS. We train the selected ML methods on labeled text and evaluate their prediction accuracy through cross-validation technique. We also experiment with strategies that could potentially enhance the prediction performances of the chosen ML methods. These strategies are feature engineering, and scale calibration. Then we run the corresponding comparisons.

4 Corpus Description

We use a corpus of comments collected from the SNS forum for both training and testing ML based models. All comments written by individual user have been aggregated into a document so that each document is labeled with competence
values, ranging from 0.00 to 1.00. The main idea is to predict the competence level for the new users based on their comments using learned models. The distribution of the corpus over the competence scale is provided in Figure 1. The corpus contains the comments generated by a total of 5,243 distinct users. We have mainly two types of experimental settings baseline and calibrated setting. In a baseline setting we apply fine-grained competence labels to divide the users into 5 categories very incompetent, incompetent, average, competent, very competent based on their competence values, ranging [0.00, 0.2], (0.20, 0.40], (0.40, 0.60], (0.60, 0.80], and [0.80, 1.00] respectively. Most of the users fall into very competent, competent categories. To reduce this class-imbalance problem and improve the performance of the models, we attempted to calibrate the competence scale to have three categories competent, average, incompetent, ranging [0.00, 0.33], (0.33, 0.67] and (0.67, 1.00] respectively. We attempted to analyze the behaviour of the selected models by doing so and got a dramatic improvement of the prediction accuracy.

Fig. 1: Distribution of the corpus over competence scale.

5 Text Analysis

The NLP tasks for the text corpus have been performed by the Rapidminer tool 3, which is an open source software for data mining. Rapidminer provides several

3 https://rapidminer.com
text analysis modules and ML algorithms. Rapidminer is a plausible choice to approach our problem because it supports the extraction of bag-of-word features from a raw unstructured text corpus. Moreover, Rapidminer provides Java API support for the integration of the resulting ML-based models into MICO, to which we implement text analysis components. We run the following main text processing operations on the text corpus to produce a featured dataset, so that it can be used to train the selected ML models.

**Tokenization** splits the comments posted by each user into a sequence of tokens, a token for example, might be a word. There are several ways of doing that by using regular expressions, specially to control tokens containing non-standard characters, but to keep the originality of the contents we used the default setting of the Rapidminer tokenizer module. Avoiding any kind of exclusion of such characters also potentially contribute to the ML models to learn the actual fact of the contents.

**Stemming** takes the word tokens returned during the tokenization phase and generate a morphological base form of the words by stripping the word suffixes. Rapidminer uses the Porter stemming algorithm [8], which is considered as a de facto standard algorithm for English. As it applies linguistic rules, it suffers from exceptions, which in its turn affects the quality of our training set.

**Generating n-grams** an n-gram is a sequence of tokens of length n. Here we generated bi-grams (n=2) and tri-grams (n=3) to use them as additional features to the baseline bag-of-word feature set. Since all possible sequences of tokens have to be generated for each document and since they also turn out to be a part of a feature set, it becomes computationally expensive. Due to this problem, we could only apply the bi-gram and tri-feature to the NB model.

**Extraction of number of tokens** returns the total number of tokens in each document and is another important feature intended to be included in characterizing the text written by the users. All types of tokens have been counted regardless of their lengths. In principle, it does not seem to make sense to consider tokens having a length less than 4, also some NLP applications ignore their importance in practice. However, users might prefer discussing their idea very briefly using informal words, and discarding their short tokens might affect the modeling in a way such that it fails to learn their features.

**Extraction of aggregate tokens length** it is a computation of the aggregate length of all tokens in a text.

6 **Method Description**

We give a formal and brief description for the three ML models used in this study, naive bayes [2], decision trees [7] and K-nearest neighbour [13]. In addition to their bag-of-words features support, we chose these models because they are easy to understand and interpret, and implement.
6.1 Naive Bayes

Naive Bayes (NB) is a probabilistic classifier and applied to several text classification problems [5]. Once trained with a corpus of documents, the NB model returns the most probable class for the input text based on the Bayes rule of conditional probability. First, the text (a document) needs to be defined and represented with a set of features and we also assume that $T$ is a set of training samples. Then the NB takes a feature vector $\vec{d} = (f_1, \ldots, f_n)$ of the document and applies the following equation to predict the most likely class:

$$ \arg\max_C P(C|\vec{d}) $$

(1)

$$ P(C|\vec{d}) = \frac{P(f_1, \ldots, f_n|C)P(C)}{P(f_1, \ldots, f_n)} . $$

(2)

Here the term $P(C|\vec{d})$ is a probability of $\vec{d}$ being in a class $C$, defined as:

$$ P(C|\vec{d}) = \frac{P(C) \prod_{i=1}^{n} P(f_i/C)}{P(f_1, \ldots, f_n)} . $$

(3)

Here the term $P(C)$ is a prior probability of a class $C$ and $(f_i/C)$ is a conditional probability of $f_i$ given a class $C$. Since the $P(f_1, \ldots, f_n)$ is the same for all classes. Then, the above equation can be reduced to:

$$ P(C|\vec{d}) = P(C) \prod_{i=1}^{n} P(f_i/C) $$

(4)

A probability $P$ over $T$ is estimated based on word/token and class counting as follows:

$$ P(C) = \frac{\text{count}(C)}{|T|} . $$

(5)

$$ P(f_i/C) = \frac{\text{count}(f_i, C)}{TC} . $$

(6)

Here $\text{count}(C)$ returns the number of times that the class $C$ is seen in $T$, and $|T|$ is the total number of samples in the training corpus, $TC$ is the Total number of (word/token) in a class $C$. In a bag-of-words model each feature $f_i$ for $i=1 \ldots n$, represents a word/token, therefore $\text{count}(f_i, C)$ returns the number of times the word/token $f_i$ seen in the class $C$. To avoid zero probabilities, laplace correction (add-one smoothing) has been used. That is a commonly used parameter smoothing technique which adds one to each count.
6.2 Decision trees

Decision trees (DT) is extensively used in a wide range of NLP applications for building tree structured predictive models. For example, it has been shown to be effective in a segmental duration prediction model for a text to speech system [3]. Decision trees built by the DT algorithm consist of a root node, which represents the most discriminatory feature in the training feature set, edges, that represent answers to questions asked by internal nodes, and leaf nodes that represent decisions [7]. To split training samples \((T)\) with \(n\) number of classes of the form, \((f_1, \ldots, f_n, C)\) into subtrees, the DT algorithm computes the Entropy \((H)\), which is a measure of homogeneity of \(T\), and the Information Gain \((IG)\), which is a measure of a decrease in \(H\).

Here are the equations for \(H\) and \(IG\) respectively:

\[
H(T) = - \sum_{i=1}^{n} P(C_i) \log_2 P(C_i), \tag{7}
\]

where the term \(P(C)\) is a probability of a class \(C_i\). The IG for any \(f_i\) in a feature set characterized the \(T\) defined as:

\[
IG(T, f_i) = H(T) - \sum_{x \in X} P(x) \sum_{i=1}^{n} P(C_i|x) \log_2 P(C_i|x), \tag{8}
\]

where \(X\) is a set of values of the feature \(f_i\) in \(T\), and the term \(P(x)\) is a probability (see equations 4 and 5 for its estimation) of the value \(x \in X\).

During a decision tree construction, the feature yielding the highest \(IG\) taken by the DT algorithm to split the samples recursively.

7 K-Nearest Neighbour

K-Nearest Neighbour (KNN) is a non parametric classifier. In the KNN algorithm, \(K\) represents the number of samples in a training set that are closest to an input sample. Those samples belong to the class predicted by the algorithm. The nearest neighbours to the input samples are obtained by using, for example, Euclidean distance. KNN has been used in many applications such as search engines [6], and pattern matching [13].

The Euclidean distance between the two feature vectors, \((f_1^1, \ldots, f_n^1)\) and \((f_1^2, \ldots, f_n^2)\) representing two documents \(\vec{d}_1\) and \(\vec{d}_2\) respectively can be estimated as:

\[
D(\vec{d}_1, \vec{d}_2) = \sqrt{\sum_{i=1}^{n} (f_i^1 - f_i^2)^2}. \tag{9}
\]

During the prediction phase, each class in the training set suggest their \(K\) number of vectors with the least value of \(D\) as the nearest neighbour(s) to a new unseen vector representing a document \(\vec{d}\). When \(K\) is greater than 1 then the class with a majority vote will be assigned to the \(\vec{d}\), otherwise the class with a smallest \(D\) is assigned.
8 Training and Testing

To train and test the models, we have randomly split up the whole corpus composed of 5,242 samples and 10,062 features, into 70% training and 30% test set. We make use of particularly a shuffled sampling, where samples are chosen with random orders. Before the split, the target label accuracy has been discretized from a numeric type into a nominal type to meet the requirement posed by the ML algorithms implementation in the Rapidminer ML environment.

We apply the same learning process and settings to the ML models DT, KNN and NB selected for this study, in terms of the size and types of the training set. During the training phase, the models parameters have been optimized to reduce model over-fitting through a held-out data set.

To evaluate how well the trained models perform against the training set regarding prediction accuracy, precision recall and F-measure, we run a number of tests. Since the test set has been taken from the labeled corpus it does not require a manual annotation for both baseline and calibrated settings. The evaluation results has been shown in Table 1.

9 Comparison Across Models

The following commonly used standard metrics [14] (i.e., their values range between 0 and 1) have been used to measure the performance of the models:

Accuracy \((A)\) as it is defined in equation 10, that tells how many of the documents (here, the single document represent the text written by each user) are correctly classified out of the test set.

Precision \((P)\) as it is defined in equation 11, that indicates the sensitivity of the models towards true predictions.

Recall \((R)\) as it is defined in equation 12, it shows how the models performs for each class on the basis of the size of their test set.

F-measure is a harmonic mean of \(P\) and \(R\). For a binary classification its estimation is straightforward. For multi-class problems, there are two common approaches, micro-averaging and macro-averaging of F-measure. Micro-averaging takes the global values of \(P\) and \(R\) for the F-measure estimation, whereas the macro-averaging takes the local values of \(P\) and \(R\). In micro-averaging the F-measure has the same value as accuracy unless a bias is estimated which is mostly applied to a cross-model analysis. Because of that, we used the macro-averaging to compare the models investigated in this study.

Given that each instance represents a text written by a user and every \(C_i\) for \(i=1,...,N\) is a subset of a test set \(T\), where \(N\) is the number of classes, we define the following equations for the metrics, \(A\), \(P_i\), \(R_i\) and F-measure, respectively:

\[
A = \frac{\sum_{i=1}^{N} TP_i}{|T|} \tag{10}
\]
Predicting User Competence from Text

\[ P_i = \frac{TP_i}{(TP_i + FP_i)} \]  
(11)

\[ R_i = \frac{TP_i}{(TP_i + FN_i)} \]  
(12)

\[ F-measure_i = \frac{2P_iR_i}{(P_i + R_i)} \]  
(13)

\[ F-measure = \frac{\sum_{i=1}^{N} F-measure_i}{N} \]  
(14)

Where:

- \( TP_i \) (true positive) is the number of instances accurately predicted to class \( C_i \)
- \( FP_i \) (false positive) is the number of instances wrongly predicted to class \( C_i \)
- \( FN_i \) (false negative) is the number of instances belong to class \( C_i \), but not accurately predicted to that class

Table 1: Cross-validation and comparison results

<table>
<thead>
<tr>
<th>Models</th>
<th>Accuracy (%) Baseline Features</th>
<th>Accuracy (%) Added Features</th>
<th>F-measure (%) Baseline Features</th>
<th>F-measure (%) Added Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT</td>
<td>79.34</td>
<td>80.04</td>
<td>19.51</td>
<td>32.62</td>
</tr>
<tr>
<td>NB</td>
<td>48.38</td>
<td>48.38</td>
<td>21.71</td>
<td>21.71</td>
</tr>
<tr>
<td>KNN</td>
<td>63.19</td>
<td>69.42</td>
<td>21.74</td>
<td>33.79</td>
</tr>
<tr>
<td>Hybrid</td>
<td>73.74</td>
<td>74.00</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>After calibration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DT</td>
<td>93.39</td>
<td>94.09</td>
<td>34.42</td>
<td>52.47</td>
</tr>
<tr>
<td>NB</td>
<td>68.28</td>
<td>68.28</td>
<td>33.18</td>
<td>33.18</td>
</tr>
<tr>
<td>KNN</td>
<td>88.94</td>
<td>91.61</td>
<td>33.60</td>
<td>45.58</td>
</tr>
<tr>
<td>Hybrid</td>
<td>91.86</td>
<td>93.26</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

10 Discussion

The baseline validation results shown in Table 1 tell us that the DT outperforms the other two models, NB, and KNN, by 16.00% and 15.00% of accuracy, respectively. However, the DT has got the lowest value of F-measure with the baseline features, due to its smallest recall value. We also attempted to build a hybrid model that combines the three models (DT, NB and KNN). However the hybrid model has relatively a poor performance regarding F-measure, given that it has much better accuracy value than NB. These figures indicate the possibility of predicting user classification competence using the selected ML models based
on their comments to a certain degree of effectiveness. These models have been studied with other supervised learning models in [9, 10] for text classification problem, it has been shown that DT and KNN outperform NB regarding both accuracy and F-measure metrics. One of the possible reasons for the poor performance of the NB model may be its strong independent interaction assumption between features, words/tokens which act as unigram features of the model [2].

We added more features (see the description below) to enhance the performance of the models and we achieved a substantial prediction accuracy increase with only the DT and KNN models. The performance of the NB has been improved by exclusively adding a bigram feature, however, we do not consider that improvement in the comparison with other models to avoid unfair comparisons (see the details below). We have also achieved even more increase in performance regarding accuracy and F-measure with a calibration technique applied to all the three models.

We tried to observe and analyze evaluation results in various possible conditions. Having those conditions provide us different perspectives of the problem being studied. That is also important to ensure the reliability of our experiments through observable consistencies of invariant parameters in all conditions. The first condition in our experimental settings is the models training with five different classes and baseline bag-of-words features. These classes represent five categories of users in the scale of from the least competent users to the most competent ones. The second condition followed by adding more features such as a number of classification (NoC), a number of tokens (NoT), and aggregate token length (LoT) to the models trained in the first condition. Here, the combined effect of the NoT and LoT has been studied independently as well as the NoC. In the third condition, the models have been trained to have three different classes to capture three categories of users, namely, less competent, competent, and high competent. Then we applied the first two conditions as the subconditions. The next paragraph discusses how the added features impact the performance of the models.

![Fig. 2: Summary of the experimental results of this study](image.png)
**NoC** represents the total number of images classified by each user of SNS. This feature has improved the accuracy of DT (from 79.34% to 80.04%) and KNN(63.19% 68.21%), which indicates their sensitivity to the numerical features as compared to the NB model. Specially, the DT used this feature as a root node during the construction of its decision tree, which means the NoC has been taken as the best predicator even from the basic features. Obviously, it is natural to assume that a user with a high number of classifications to fall in the very competent category, despite it is not always true. Moreover, the corpus used in this study also reflect this fact as well, for example, there are users who made classifications of more than 18,000 images, and they are in the very competent category. This shows that how experiences affect the classification proficiency of the users.

**NoT and LoT** these features represent the size of the comments posted by the users in terms of tokens/words. We are interested in observing their combinational effect on the prediction of competence, they are closely related and assumed to be the good indicators of competence as the most competent users tend to write long comments. Unfortunately, these features have no any effect on neither of the models.

**Bi-gram** represents a sequence of two words. This feature has dramatically improved on (from 48.38% to 64.40%) the accuracy of the NB model. Having the more contextual information in texts always improves the efficiency and accuracy of its classification. Generally, it is also an evident that n-gram features have impact on several text classification applications. Due to the more memory requirement, we could not see the effect of the bigram feature in the DT and KNN models. One possible approach to face this challenge is reduce the size of the training data to meet, but which might cause unfair comparison due to a different setting.

We attempted to analyze the performance of the selected ML models and make a generalization with a limited number of experimental conditions. However, still different results might be obtained and have new perspectives to the problem if we had approached it differently. For example, as we mentioned in section 2, we followed a heuristic approach to set the ground truth for computing user competence, but there are other possible ways to try that could potentially produce better results.

We assumed that a written text is probably a good indicator about the classification competence of users, but this assumption does not hold for some cases, for example some competent users (given that they made accurate classifications) may not be interested or have time to discuss their classifications as much as incompetent users. In this case, our models could fail to detect the real competence of such users.

On the one hand, leveraging the NLP components available in the Rapidminer tool has been useful for general text analysis and bag-of-words feature extraction in our study. But on the other hand, considering additional tools dealing with more advanced aspects of text, such as noisiness [11] could potentially improve our results. Noisiness is a problem commonly associated with the
text obtained from social media as well as citizen science project media, that makes a parsing a bit hard.

10.1 Calibration Technique

As it is noted from Table 1, we have a class-imbalanced data problem. This causes bias to the test dataset to be classified into these classes regardless of their actual classes, and thereby severely affects the prediction accuracy of the models. So, one of the possible approaches to this problem is to take the distribution of the target feature into account and divide the corpus into three classes based on equally sized and partitioned ranges of the accuracy values. By doing so, we achieved pushing the accuracy limit in the baseline models to 94.09%. As it is shown in the Figure 2 (b) all the three models got improved with the application of the calibration technique.

11 Conclusion and Future Work

In this study, we achieved three main goals, applying the selected ML models DT, KNN and NB, effectively to learn a user competence from the text obtained from the SNS posts, evaluating and comparing the performance (regarding accuracy, precision, recall and F-measure) of these models and improving the baseline results through additional features and scale calibration. Regarding accuracy, DT outperforms NB and KNN by 16.00%, and 15.00% respectively. Regarding F-measure, K-NN outperforms NB and DT by 12.08% and 1.17%, respectively.

The learned models in this study can be applied to other citizen science projects such as the galaxy project supported by an online platform where images of galaxies are posted to be classified by volunteer users. Moreover, the model performance improving techniques proven to be effective in this study could be useful in other related areas such as text classification problems.

Our next steps to further improve our results are to consider and experiment with other ML and Ontology based models such as SVM and Neural Nets. We will also attempt to work on implementing a new ground truth for competence estimation. Then the resulting model has been intended to be integrated into a cross-media analysis framework MICO.

References


4 https://www.galaxyzoo.org


