Active learning to rank for assessing the linguistic quality of sentences

Alexander Kuhnle
Queens’ College

UNIVERSITY OF CAMBRIDGE

A dissertation submitted to the University of Cambridge in partial fulfilment of the requirements for the degree of Master of Philosophy in Advanced Computer Science

University of Cambridge
Computer Laboratory
William Gates Building
15 JJ Thomson Avenue
Cambridge CB3 0FD
UNITED KINGDOM

Email: aok25@cam.ac.uk

— June 11, 2015 —
Declaration

I, Alexander Kuhnle of Queens’ College, being a candidate for the M.Phil. in Advanced Computer Science, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

Total word count: 12,544

Signed:

Date:

This dissertation is copyright © 2015 Alexander Kuhnle. All trademarks used in this dissertation are hereby acknowledged.
Abstract

The traditional way of teaching a foreign language involves long-standing practice and receiving feedback under the guidance of a teacher. However, a large part of the coaching process does not necessarily require the employment of a human teacher, but rather can be automated to facilitate self-practice and to increase the scale of foreign language education by teaching a large number of students without requiring a correspondingly large educational staff. Consequently, automated tutoring and assessment systems for non-native speakers of English are becoming increasingly important. Andersen et al. [2013] present a novel approach to this, in employing a discriminative ranking model as a feedback system on the linguistic quality of individual sentences, in addition to feedback on the quality of an entire text. By its non-specific identification of problematic parts of the text, this can have pedagogic advantages, e.g. by giving the learner the opportunity to come up with a better version himself.

We extend their work by using active learning to rank, which aims to reduce the amount of training data required by a ranking model to achieve a certain level of performance by selectively choosing the most informative training instances to annotate and train upon. Annotated data is a major problem in this setting, since there are no publically available datasets with sentence-level annotations, and creating one is time-consuming and expensive. We investigate a number of active selection techniques that are appropriate for a ranking model and systematically compare them to a random learner. More specifically, we demonstrate that active learning can reduce the amount of labeled training data. Moreover, we perform a detailed analysis of the results obtained. To the best of our knowledge, this is the first attempt to investigate the effectiveness of active learning to rank in the domain of learner data.
Chapter 1

Introduction

More and more people from all around the world begin learning English as a foreign language, particularly since nowadays a certificate of proficiency in English is a basic requirement for many things, including various professions as well as admission to university. To handle this vast amount of learners and examinations, automated text assessment systems are becoming increasingly important and necessary as they facilitate large-scale teaching and assessment, while simultaneously reducing costs. Furthermore, they allow the application of consistent assessment criteria and therefore reduce inconsistency, which may arise when assessment is performed by multiple assessors. Finally, automated assessment platforms give learners the valuable opportunity of self-tutoring and prompt feedback, which has been shown to increase learning efficiency, particularly for advanced learners.

While simple question-focused exercises like multiple-choice can easily be automated, free-text answers are more difficult to evaluate. Williamson [2009] reviews twelve different automated free-text assessment systems, including \textit{e-Rater} [Attali and Burstein, 2006], \textit{Intelligent Essay Assessor} [Landauer et al., 2003], and \textit{IntelliMetric} [Rudner et al., 2006]. The following thesis is based on previous work [Yannakoudakis et al., 2011, Andersen et al., 2013] on a system which scores texts written by English learners on an intermediate level in response to prompts within an examining context. Their system provides feedback at three different levels of granularity, the first covering the entire written text, the second focusing on individual sentences, and the third looking at the word-level. Herein, we focus on extending their sentence-level model, and more specifically on using active learning to rank to reduce the amount of training data needed for assessing the linguistic quality of sentences, independently of their context and appropriateness to the prompt. By directly rating the linguistic quality, it is hoped that the system is easier to deploy, as it is more likely to generalise across different genres and topics. Moreover, the marking criteria for many such free-text exercises are mainly based on the appropriate and correct use of various difference linguistic constructions.

More precisely, our work aims to extend the sentence-level evaluation system of the online \textit{SAT system} presented by Andersen et al. [2013]. This novel approach of assessing not the entire script\footnote{A \textit{script} is the entire text written by an English learner as answer to a prompt.}, but parts of the text, aims to give indirect feedback by ranking the sentences of a document according to their linguistic quality. In doing so, the learner can identify...
problematic parts of his answer and himself think about how to rephrase the respective sentences. On an intermediate language level such a way of giving feedback is believed to be advantageous from a pedagogic viewpoint over a system which detects and corrects grammatical, punctuation, spelling, etc. mistakes, particularly if its recall is relatively low.

The system uses a discriminative ranking machine learning model which compares sentences and hence constructs a quality ranking of them. Discriminative ranking has been shown to achieve state-of-the-art results in assessing free-text writing [Yannakoudakis et al., 2011]. The reason therefore is that ranking is a more generic approach to the task, since it explicitly captures the scoring relationship between items and does not necessarily attempt to fit items to a scoring scale. Moreover, it exploits the labeling information efficiently with respect to this task by basing its input on the combinatorially richer set of pairwise comparisons of the items (see section 2.2).

We integrate active learning into this rank learning system and investigate its use to reduce the amount of pre-labeled training data. Since there are no publically available datasets which include scoring annotations at the sentence-level\(^2\), reducing the dependence on large amounts of data is of great advantage for the deployment of an automatic quality assessment of sentences. This thesis presents an active learning to rank approach and investigates a wide range of selection techniques. These are based on and extend ideas commonly found in the literature about determining the potential relevance of an previously unlabeled instance to a maximal performance gain of the learning model by looking at the local and global uncertainty of this instance within the set of all instances. Various such techniques are systematically compared on the basis of different ways of measuring the quality of a linguistic ranking of sentences, and selection strategies are identified that show superior performance in this task compared to a random learner baseline, which is typically employed to assess the superiority of active learning. To the best of our knowledge, this is the first attempt to investigate the effectiveness of active learning to rank in the domain of learner data.

Furthermore, we perform an analysis of the results obtained in our active learning setup. On the one hand, we observe that a random learner is a baseline which is very hard to beat, while on the other hand, the learning curves show a less continuously increasing and more noisy fluctuating behaviour with an already high initial performance. An investigation of these factors reveals, among others, that a few of the employed features provide the machine learning model with enough information to learn an almost \textit{"optimal"} – i.e. passive learner performance level – ranking system with only a small fraction of the training data. This observation implies that a more simple rank learning model with a reduced feature set and without actively selecting training instances still allows to substantially reduce the amount of training data while yielding similar results.

\(^2\)In fact, we use a publically available dataset with answer-level scores, and another private set of only a small number of scripts with explicit sentence-level scores.
Chapter 2

Background and Related Work

2.1 Classification

In supervised machine learning, the aim of classification is learning to automatically assign the correct class $y \in \mathcal{C}$ of a finite set of classes $\mathcal{C}$ to an instance of the problem domain, represented by a feature vector $x \in \mathcal{F}$ of a finite-, high-dimensional vector space $\mathcal{F} = \mathbb{R}^n$, the so called feature space\(^1\). It is assumed that every instance belongs to a single class. Given a set of class-labeled training instances $\{(x_i, y_i)\} \subseteq \mathcal{F} \times \mathcal{C}$, a (discriminative) classifier $f_{\text{class}}: \mathcal{F} \rightarrow \mathcal{C}$ is trained according to an underlying machine learning model. The resulting classifier is hoped to assign the correct class $f_{\text{class}}(x) = y \in \mathcal{C}$ to a previously unseen instance $x$.

A special case hereof are binary classifiers with a linear learning model, meaning that there are only two classes $\mathcal{C} = \{-1, 1\}$ to discriminate, and that the model assumes the problem data to be linearly separable, i.e. the vectors of the two classes can be separated by a hyperplane $H \subset \mathcal{F}$. In the following we only consider unbiased hyperplanes, which can be represented by a weight vector $w \in \mathcal{F}$ via $H = \{h \in \mathcal{F} \mid w \cdot h = 0\}$. For such a binary linear classifier, classification is done via determining on which side of its hyperplane an instance $x$ is located\(^2\):

$$f_w^{\text{class}}(x) = \text{sign}(w \cdot x) \in \{-1, 1\}.$$  

The training procedure, on the other hand, essentially tries to find a hyperplane $w$, which is consistent with the given labeled training data $\{(x_i, c_i)\}$, i.e. it satisfies the following constraints\(^3\):

$$f_{w}^{\text{class}}(x_i) = w \cdot x_i \overset{!}{=} y_i \quad \text{or equivalently} \quad (w \cdot x_i) \cdot y_i \overset{!}{>} 0.$$  

\(^1\)Every $n$-dimensional vector space can be embedded into a vector space $\mathbb{R}^n$ for $n \in \mathbb{N}$.
\(^2\)The orientation of the weight vector $w$ can always be adjusted to the label classes by changing its sign.
\(^3\)We use the superscript exclamation mark to emphasise that an (in)equality has to hold, and a value (in this case $w$) needs to be found accordingly.
A **large margin** binary linear classifier is additionally required to maximise the overall margin of the training data, which for an instance \((x, y)\) is the distance from the hyperplane, given by the absolute value of its scalar product with the weight vector \(|w \cdot x|\):

\[
\text{Maximise} \quad \sum_i |w \cdot x_i| = \sum_i (w \cdot x_i) \cdot y_i.
\]

The margin of an instance \(|w \cdot x|\) can be understood as the **classification certainty**. In that way, a binary linear classifier is not only a **discriminative** model, enabling to distinguish between two classes, but also reports the **certainty** of the classification output.

### 2.2 Rank Learning

Rank learning comprises the task of learning an ordering \(R\) of instances of the problem domain, again represented by vectors \(x \in \mathcal{F}\) of a feature space \(\mathcal{F}\). Although in general this does not necessarily have to be the case, we assume \(R\) to be a total order (i.e. it is antisymmetric, transitive, and total), which can be isomorphically embedded in the real numbers, meaning that each instance \(x_i\) can be associated with a number \(r_i\) such that \(x_i \leq x_j\) if and only if \(r_i \leq r_j\).

Provided with rank-labeled training instances \(\{(x_i, r_i)\} \subset \mathcal{F} \times \mathbb{R}\), the goal is to learn a **ranking function** \(f^{\text{rank}}: \mathcal{F} \rightarrow \mathbb{R}\) according to an underlying machine learning model, which best captures the respective ordering \(R\), i.e. for a ranking pair \((x_i \leq x_j) \in R\) the function is supposed to give \(f^{\text{rank}}(x_i) \leq f^{\text{rank}}(x_j)\).

A **linear** rank learning model assumes that the problem data is **linearly rankable**, requiring the existence of a valid ranking function \(f^{\text{rank}}_w\) which can be expressed solely in terms of a weight vector \(w \in \mathcal{F}\), \(f^{\text{rank}}_w(x) = w \cdot x\). We refer to \(w \cdot x\) as the assigned **ranking score**. During training, a linear ranking function is learned by finding a weight vector \(w\) which is **consistent** with the labeled training data \(\{(x_i, r_i)\}\) in satisfying the following constraints:

\[
f^{\text{rank}}_w(x_i) = w \cdot x_i \leq f^{\text{rank}}_w(x_j) = w \cdot x_j \quad \text{if and only if} \quad r_i \leq r_j.
\]

By rearranging the inequation using the linearity of the dot product, we alternatively end up with

\[
f^{\text{rank}}_w(x_j) - f^{\text{rank}}_w(x_i) = w \cdot x_j - w \cdot x_i = w \cdot (x_j - x_i) = f^{\text{rank}}_w(x_j - x_i) \geq 0.
\]

This formulation of the constraint reveals how a ranking problem can always be associated with a corresponding classification problem: Its instances are the **pairwise difference vectors** \([x_j-x_i]_{i,j} \in \mathcal{F}\) and its labels are given by \([\text{sign}(r_j-r_i)]_{i,j} \in \{-1, 1\}\). Consequently, the underlying binary decision such a classifier tries to make is to determine whether the

\[4\text{Technically, it is the euclidean distance scaled by the length of }w, \text{ but this does not matter here.}\]
first instance is ranked lower than or equal to the second instance. Moreover, a linearly rankable ranking problem implies a linearly separable associated classification problem.

This approach to rank learning via transformation to a classification problem was first taken by Joachims [2002], employing an SVM-based model. However, such an approach comes with the problem of a fast growing training set size $0.5 \cdot n \cdot (n - 1)$, bounded by $O(n^2)$ in the number of instances $n$, since for each new instance, $n$ pairwise difference vectors are added. In this work we use the **Timed Aggregate Perceptron (TAP)** [Briscoe et al., 2010], a variant of the batch perceptron algorithm. Essentially, a TAP is also a linear rank learning model, which differs from a rank SVM model mainly in the way the optimisation problem is computationally solved. The SVM model uses efficient techniques to solve the dual quadratic optimisation problem emerging from the training procedure described above [Joachims, 1999], whereas the TAP model iteratively corrects the weight vector according to a timing variable, which controls both the magnitude of the correction effect and when the process stops. Furthermore, it reduces the number of involved training difference vectors by sampling with an upper bound. All in all, we found that the TAP model performs substantially faster than the SVM model in our experiments.

In the literature one can often find a different version of a ranking problem, which is more closely related to multi-class classification. In this case, the goal is to provide a ranking of a fixed finite set of labels for each instance. Har-Peled et al. [2002] gives an overview of different sub-problems, including binary classification as a ranking problem with two labels and a focus on the maximum of the resulting ranking. Their approach of *constraint classification* subsumes all these sub-problems as special cases of a single problem, which can be solved by a linear model. A different approach to this kind of rank learning problem [Fürnkranz and Hüllermeier, 2003] is to divide the problem of ranking $n$ labels into $0.5 \cdot n \cdot (n - 1)$ classification problems for pairwise label preference. Brinker [2004] reviews both methods in the context of active learning.

Another different formulation of a ranking problem is closely related to regression, where one explicitly assigns values of a ranking scale to instances [Herbrich et al., 2000]. While for linear rank models (as presented earlier in this section) one obtains ranking scores as well, the model focuses on the *internal* ranking structure amongst the instances. In particular, the ranking scores assigned by such a model do not carry any external *absolute* meaning, but only their *relative* order within the model is of relevance, which is contrary to the task of regression.

### 2.3 Active Learning

Active learning underlies the assumption that the process of acquiring raw, unannotated data is much less problematic than annotating instances with their corresponding label. In many domains the latter involves trained experts to ensure consistent high-quality annotations, which makes it a time-consuming and costly effort.

Active learning\(^5\) [Settles, 2010] tries to reduce the labeling effort by giving the learning model

\(^5\)Sometimes this technique is also called *selective sampling*. 

5
the control over iteratively choosing training instances which it deems most informative
with respect to a maximal improvement of the model. The underlying hypothesis of active
selection is that random selection for annotation might lead to annotating instances that
are already similar to those in the training set and therefore may not benefit the machine
learning model. Starting with only a small seed of initially labeled training data, the
active learner is trained on this set and subsequently is given the opportunity to select
a batch of unlabeled instances from a pool, which then gets annotated and added to
the training set. This process of consecutive training and selection phases is repeated for
multiple rounds, while the model’s parameters and features are considered constant and
do not change. Active learning is usually contrasted to passive learning, which refers
to the “maximum” performance obtained by the machine learning model when trained
on the whole training/pool set\(^6\), and random learning, in which the learner randomly
(not actively) chooses instances from the pool.\(^7\) A schematic working of an active learning
algorithm can be found in figure 2.1.

1. Select \(s\) seed instances \(\text{Seed} = \{(x_i, y_i)\}\) from the instance pool and annotate them.
   They form the initial training set \(\text{Train}_0 = \text{Seed} = \{(x_i, y_i)\}\), whilst the remaining
   instances give the pool \(\text{Pool}_0 = \{x_i\}\).
2. For each round \(r = 0, 1, 2, \ldots, R\), until a stopping criterion is met:
   (a) Train a model \(f_r\) on the training data \(\text{Train}_r\).
   (b) Select a batch of \(b_r\) instances \(\text{Batch}_r = \{x_i\} \subset \text{Pool}_r\) from the pool data
       \(\text{Pool}_r\) and annotate them. This gives the new training set \(\text{Train}_{r+1} = \text{Train}_r \cup
       \text{annotate(}\text{Batch}_r\text{)}\), and pool \(\text{Pool}_{r+1} = \text{Pool}_r - \text{Batch}_r\).
3. Return the final model \(f_R\).

Figure 2.1: Active learning algorithm.

Some of the first attempts on active learning can be found in [Seung et al., 1992, Cohn
et al., 1994, Lewis and Gale, 1994, Cohn et al., 1996]. The crucial part of this iterative
procedure is the method of selecting the instances in each round, which form the most
informative batch set for the learning process. In the next section we give an overview of
selection techniques found in the literature.

### 2.4 Selection Techniques

An optimal active learner [Roy and McCallum, 2001] incrementally chooses at each round
the instances that maximise the expected final performance\(^8\), that is, after the whole

\(^6\)We note that in our experiments we simulate active learning on the basis of an already fully pre-
annotated dataset. In general, however, annotating the entire training set obviously contradicts the
purpose of active learning to reduce annotation effort.

\(^7\)We point out that random learning is sometimes also referred to as passive learning in the literature.
However, we understand the passivity of a learner as making no attempt at all to learn iteratively, but
instead directly queries the annotation of all instances.

\(^8\)The performance is often expressed in terms of a loss function, which measures the deviation from an
optimal solution. Thus, maximising performance becomes minimising the loss.
learning procedure terminates (see figure 2.1):

\[
\text{Select } \arg \max_{B \subset \text{Pool}_r} E\left[ \text{Perf} \left( f_R \mid \text{Batch}_r = B \right) \right].
\]

However, the performance maximisation potential of an instance might depend on how many subsequent learning rounds are still to come, and might affect the selection choice of training instances in subsequent training rounds. For most of the real-world applications, it is impossible to predict the influence of adding instances in a specific round on the final results. Consequently, although optimal, such a technique is usually infeasible. There are attempts to ensure various forms of a weaker optimality which we present in the following.

Instead of optimising with respect to the whole learning procedure, one could focus only on the current round, i.e. select the instances which maximise the expected performance of the model after re-training on the augmented training set:

\[
\text{Select } \arg \max_{B \subset \text{Pool}_r} E\left[ \text{Perf} \left( f_{r+1} \mid \text{Batch}_r = B \right) \right].
\]

This is often seen as an appropriate theoretically optimal selection technique [Cohn et al., 1996, Roy and McCallum, 2001], but in most of the cases it is still infeasible due to the fact that for many learning models (re-)training is an expensive process and hence cannot easily be done multiple times. However, this would be necessary for such a method, at least once for every instance in the pool. An interesting attempt to tackle this problem for classification is presented by Roy and McCallum [2001]. They are able to greatly reduce time consumption by using Monte Carlo sampling methods to get a reliable approximation of the expected performance after re-training the model.

To avoid elaborate and expensive re-training, instead of properly calculating the performance resulting from adding a batch of instances to the training set, a common simplification is to estimate the expected performance gain:

\[
\text{Select } \arg \max_{B \subset \text{Pool}_r} E\left[ E\left( \text{Perf}(f_{r+1}) \mid \text{Batch}_r = B \right) \right].
\]

One way to estimate the effect of instances on the performance of the model is to base the analysis on the likelihood of these instances changing the current model significantly, as presented by Donmez and Carbonell [2008] for rank learning. Other selection strategies do not directly optimise an evaluation metric, but rather choose instances according to a “non-optimal” criterion. One such approach is uncertainty sampling, which selects the instances for which the model is most uncertain [Lewis and Gale, 1994]. For probabilistic classifiers this is the case for an instance \( x \) with \( P(f_r(x) = y) \approx \text{const} \) across \( y \in C \).

The method of basing the selection criterion on the uncertainty of the current learning model’s decision is especially popular in the context of binary linear discriminative classification using e.g. SVMs. These models do not naturally enable the application of probabilistic concepts like estimation to uncertainty sampling\(^9\), but instead they explicitly output a certainty value for their classifications, defined as the distance to the hyperplane\(^{10}\).

\(^9\)However, Platt [1999] describes a way to assign meaningful probabilities to classification results of a binary linear model.

\(^{10}\)An informal correspondence to the optimisation criteria above can be described as follows: Selecting the least certain instance is similar to selecting the instance with the highest probability of being incorrectly classified, and hence its inclusion to the training set is expected to change the model “most significantly.”
With the theoretic foundation of the concept of a *version space* [Mitchell, 1982] representing all weight vectors resp. hyperplanes which are consistent with the data, Tong and Koller [2002] show how under perfect conditions (e.g. noise-free data), selecting the least certain instances maximises the reduction of the version space and consequently the performance gain. This method is sometimes referred to as *simple margin*:

\[
\text{Select } \arg \max_{x \in \text{Pool}} |w \cdot x|^{-1}.
\]

Vlachos [2004] uses, among others, this approach and extends it to multi-class classification. Scholn and Coln [2000] describe the same optimisation strategy, but with a more geometric motivation of how a highly uncertain instance has the greatest potential to change the linear model. The *query-by-committee* approach [Seung et al., 1992] is motivated by information theory and closely related to the aforementioned active selection strategies. Here, a *committee* of (binary linear) models are randomly sampled from the version space as its approximation and used to evaluate the pool instances. The instances with the maximal disagreement in the committee, i.e. half of the models classify them as positive and half as negative, get selected.

Building on the *simple margin* method, more elaborate techniques are presented, particularly addressing the fact that in real-world applications the data is not perfectly noise-free. Tian and Lease [2011] include additional relative locational information about an instance amongst the others in the feature space – the *local structure* – to avoid choosing outliers, while Ho et al. [2011] select more uniformly across the whole range of occurring certainty values and not only the minimal, i.e. most uncertain ones.

A lot of previous work on active learning has been done on classification; however, the underlying principles often cannot be easily transferred to rank learning. Apart from the more technical fact that the evaluation metrics are fundamentally different, the two main challenges in learning to rank are:

- The instances cannot be treated independently from each other, but their position in the ranking interrelates with every other instance.
- The notion of uncertainty defined as the classification certainty resp. margin is not explicitly defined for an instance (but only for pairwise difference vectors, see section 2.2).

One attempt to integrate active selection for rank learning is presented by Donmez and Carbonell [2009], where they use a different performance assessment based on the *hinge rank loss* rather than the typical loss functions (squared loss, 0-1 loss) in the context of classification.

A first approach to *uncertainty sampling* for linear rank learning – in parallel to the linear binary classification case – is described by Brinker [2004]. It uses the minimum ranking score difference, i.e. the minimal certainty \( w \cdot (x_j - x_i) \) of the pairwise difference vector in the corresponding classification problem, as an approximation of the potential to improve the current model. The underlying hypothesis here is that closely ranked instances are the ones for which the model is most uncertain. Yu [2005] extends this idea to a greater batch size by selecting the *most ambiguous* set of pool instances and hence choosing the set with
the smallest inner overall certainty about their ranking\footnote{Note that $x_i \leq x_j$ implies $w \cdot (x_j - x_i) = w \cdot x_j - w \cdot x_i \leq 0.$}:

$$\text{Minimise } \sum_{(x_i \leq x_j) \in \text{Batch}} w \cdot (x_j - x_i).$$

Many approaches capture the idea of selecting pairwise difference vectors by explicitly adding pairs $x_i$ and $x_j$ of instances from the pool and using the ranking uncertainty of this pair $|w \cdot (x_j - x_i)|^{-1}$ as the main maximisation criterion of selection. However, this basic idea is often integrated as part of a more elaborate selection procedure, taking into account the interdependence of instances, possible existence of outliers, and the fact that a small ranking score difference can be caused by a de facto close position in the target ranking.

To incorporate the position of a pair of instances within the current ranking of instances and relations to the already annotated instances in the training set, the selection strategy of Qian et al. [2013] calculates a \textit{global uncertainty} value for each of the two instances on the basis of all pairwise difference vectors resulting from adding this instance to the current training set. Arens [2011] presents a method which simply selects the instances with the highest ranking score in the pool. This reflects the idea that one either wants to particularly improve the ranking quality of top instances, or one considers those as the most informative for improving the learning model. Other approaches, e.g. Dzyuba et al. [2013] for rank learning – or also Tian and Lease [2011] for classification – involve the location of a pool instance relative to the train instances or the other pool instances in the form of either their cosine similarity or their euclidean distance. In general, an instance is considered more informative if it is located close to many other pool instances, but farther away from the already labeled train instances.
Chapter 3

Experimental Setup

We extend an existing Java project for sentence-level linguistic quality assessment of English learner scripts by integrating active learning. The project was previously used to obtain the results presented by Andersen et al. [2013] and its fundamental setup\(^1\) is assumed to be appropriate for the task. Scripts provided in a certain XML format (see section 3.1) are read and a variety of linguistic features (see section 3.2) are extracted. The core of the learning process, the machine learning model, is a Timed Aggregate Perceptron [Briscoe et al., 2010], which is integrated in an active learning procedure with flexible parameters (see section 3.4) and selection technique (see section 3.5). The rankings produced in our experiments are evaluated by four different measures (see section 3.3). In the following sections we go into more detail about various aspects of our implementation, the used datasets, and the evaluation measures.

3.1 Datasets

3.1.1 FCE Dataset

The FCE dataset [Yannakoudakis et al., 2011] is extracted from the Cambridge Learner Corpus, a large collection of texts written by English learners as part of the English for Speakers of Other Language (ESOL) examination. More specifically, it consists of exam scripts written by learners of English sitting the First Certificate in English (FCE) exam that assesses the upper-intermediate proficiency level. The answers to two free-text tasks are used, which ask the examinee to write a total of 200-400 word letter, report, article, or short story, etc. They are marked by the examiners and an overall grade from 1 to 40 for the whole script, i.e. both answers, is assigned. The marking criteria primarily assess linguistic competence.

The dataset is publicly available in XML format\(^2\). Apart from the provided meta-data

\(^1\)That is, in particular the available feature types, the implementation of feature extraction from the XML file datasets, the rank learning model and its integration, and the employed evaluation measures.

\(^2\)The dataset can be found under the following link: http://ilexir.co.uk/applications/clc-fce-dataset/.
such as the learner’s native language or age, the scripts have also been manually annotated with respect to the linguistic errors committed. Errors are categorised using a taxonomy of around 80 types, and both the incorrect (<i>i</i>) and the correct (<c>c</c>) version is given. For instance, the following error indicates that the verb (V) “presented” needs to be replaced (R) with “put on”:

```xml
<e type="RV">presented</e><c>put on</c></e>
```

Furthermore, the parser output for both the incorrect and the corrected version of each sentence using the Robust Accurate Statistical Parser (RASP) [Briscoe et al., 2006] is provided for our experiments. All in all, an exemplary script is encoded as follows:

```xml
<script id="TR666" exam="FCE 0102" date="2000-06">
  <learner age="26-30" l1="ja"/>
  <results g="25.0"/>
  <answer q="1">
    <eval src="ex" s="3.3"/>
    <text>
      Dear Sir/Madam
      I am writing to express how <e type="W">I was dissatisfied</e> with OVER THE RAINBOW, which was <e type="MP">presented</e> by the Circle Theatre. First of all, the reason why I bought the ticket was Danny Brook, who is my <e type="SA">favorite</e> star, and was going to <e type="RV">play for</e> the show. However, there was a different actor instead of him, and this was <e type="R">most my disappointment</e>. Moreover, <e type="R">as your advertisement</e>, we would get <e type="MD">a</e> discount price for the ticket, but it was not available.
      ...
      </text>
      <rasp> ...
      </rasp>
      <crasp> ...
      </crasp>
    </answer>
    <answer q="4"> ...
    </answer>
  </answer>
</script>
```

Of the information provided, we use the RASP parser output for feature extraction, the overall script score ranging from 1 to 40 (in example: <results g="25.0"/>), and the number of errors annotated per sentence, regardless of their type. Since the scripts are not annotated with scores at the sentence-level, we assign a pseudo-gold score to each sentence S by taking the script-level score divided by the number of errors for this sentence S to train our sentence-level model:

\[
\text{Pseudo-gold score}(S) = \frac{\text{Script score}}{\text{Number of errors in } S}.
\]
As described by Andersen et al. [2013], this most closely reflects what we are trying to predict, i.e. a linguistic quality score for sentences.

The FCE dataset is divided into a training set of 1141 and a test set of 97 scripts (each script contains two answers). We create a development set of \(~10\%\) from the training set for model selection and batch size tuning as well as for our stopping criterion, while the test set was retained for the final test results.

<table>
<thead>
<tr>
<th>FCE Dataset Details</th>
<th>Scripts</th>
<th>Sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>1027</td>
<td>27798</td>
</tr>
<tr>
<td>Development</td>
<td>114</td>
<td>3155</td>
</tr>
<tr>
<td>Test</td>
<td>97</td>
<td>2720</td>
</tr>
</tbody>
</table>

3.1.2 WI Dataset

The Write & Improve (WI) dataset\(^3\) is a collection of answers written by English learners using the SAT system, the online self-learning platform described by Andersen et al. [2013]. Essentially, both the language level (upper-intermediate) and the tasks (free-text prompts) are similar to the FCE dataset. The dataset is provided in XML format, with similar annotations as for the FCE dataset; however, the marking scale ranges from 0 to 13. Moreover, these scripts have been annotated with gold scores at the sentence-level (in example: `<eval s="7" g="B2.i"/>`, `<eval s="8" g="B2.ii"/>`, ...). These gold scores represent the overall quality of a sentence with respect to aspects not only related to its linguistic quality, but also its lexical variety, coherence, etc. Herein, we investigate the usefulness of these scores for predicting the quality of sentences. An example script is given below (each “answer” is a single sentence):

```xml
<script id="2843" date="2014-01-20">
  <eval s="9" g="C1.i"/>
  <answer>
    <eval s="7" g="B2.i"/>
    <text>
      Dear Mr. Leader,
    </text>
    <rasp> ... </rasp>
    <crasp> ... </crasp>
  </answer>
  <answer>
    <eval s="8" g="B2.ii"/>
    <text>
      I’m glad I met you and your group of overseas students.
    </text>
    <rasp> ... </rasp>
    <crasp> ... </crasp>
  </answer>
  <answer>
  </answer>
</script>
```

\(^3\)The dataset is not (yet) publically available.
I’m sure that you won’t be disappointed with the town I live in because there is a lot to see!

The WI dataset which has been annotated with sentence scores consists of 79 scripts. Similar to the FCE dataset, we divided it into ~80% for training and ~10% for development and test set respectively.

<table>
<thead>
<tr>
<th>WI Dataset Details</th>
<th>Scripts</th>
<th>Sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>65</td>
<td>824</td>
</tr>
<tr>
<td>Development</td>
<td>7</td>
<td>88</td>
</tr>
<tr>
<td>Test</td>
<td>7</td>
<td>104</td>
</tr>
</tbody>
</table>

### 3.2 Features

We use all of the following feature types (except where otherwise mentioned) to appropriately capture the relevant characteristics for the linguistic quality of a sentence and its words, similar to Andersen et al. [2013]:

- Affixes
- Part-of-speech tag counts
- Use of clausal subjects and modifiers
- Phrase structure rules (as output by the RASP parser)
- Rule-based error counts & error words: Rules which have been automatically extracted from the Cambridge Learner Corpus (incorrect word uni-, bi-, and trigrams) and a dictionary (incorrect derivational and inflectional constructions).
- Corpus error-rate: If a word trigram is not present in a large corpus of English texts (more than 2 billion tokens), it is counted as an error.

### 3.3 Evaluation Measures

#### 3.3.1 Pearson Correlation Coefficient

The Pearson correlation coefficient is a measure of the linear dependence between two values, where 1 is absolute correlation, 0 is no correlation, and −1 is perfect negative
correlation. Given two set of values \{x_i\} and \{y_i\}, the Pearson correlation coefficient is defined as

\[ r = \frac{\sum_i (x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2} \cdot \sqrt{\sum_i (y_i - \bar{y})^2}} \]

where \(\bar{x} = \frac{1}{n} \cdot \sum_i x_i\) is the average. Essentially, the coefficient indicates to which degree the data points \((x_i, y_i)\) can be fitted to a line. The measure is not robust, i.e. it is sensitive to single outlier points.

We calculate the correlation between ranking scores of the sentences assigned by the rank learning model and the gold scores for the WI dataset. In case of the FCE dataset we do not have gold scores, hence we calculate the correlation between the assigned ranking scores and the (negated\(^4\)) error count of a sentence as an indicator of its linguistic quality, which is assumed to be significantly impaired by grammatical errors.

### 3.3.2 Spearman’s Rank Correlation Coefficient

A similar idea of dependence is captured by Spearman’s rank correlation coefficient. It measures the degree to which the data points \((x_i, y_i)\) can be described as a monotonic function. Here, 1 means a perfect monotonic relation, 0 is no monotonically increasing correlation, and \(-1\) is a total monotonically decreasing correlation. Consequently, it is a more relaxed version of the Pearson correlation coefficient, for which the monotonic function additionally has to be linear. It is defined as

\[ \rho = 1 - \frac{6 \cdot \sum_i \left(\iota(x_i) - \iota(y_i)\right)^2}{n \cdot (n^2 - 1)} \]

where \(\iota(x_i)\) is the index of \(x_i\) in an ordered list of \(\{x_i\}\), ties are broken by assigning their averaged index, and \(n\) is the number of samples.

The Spearman coefficient is preferable to the Pearson coefficient, since it only looks at the ranking characteristics of two sample sets, and not their actual values. Similarly, the ranking scores themselves are less important for a linear rank model as we use it (see section 2.2), but instead their pairwise differences get maximised in order to optimise the implicitly represented ranking structure. Furthermore, the coefficient is less sensitive to outliers.

### 3.3.3 Average Precision

Average precision is a measure used in information retrieval which evaluates the quality of a ranking by looking at the position of presumably high-ranked items in the actual ranking. We calculate precision at each correct, i.e. error-free sentence in the ranking, and average over all correct sentences:

\[ \text{AP} = \frac{1}{|\text{correct sentences}|} \cdot \left( \sum_i \delta(\text{sentence at index } i \text{ is correct}) \right) \]

\(^4\)We negate the error count so that a high rank corresponds to a small number of mistakes, and thus a correlation of 1 (and not \(-1\)) is optimal.
\[ \frac{1}{i} \cdot |\{\text{correct sentences up to index } i\}|, \]

where \( \delta(\text{proposition}) \) equals 1 if the proposition is correct and 0 otherwise. It is expected that correct sentences are higher up in the ranking. An average precision of 1 is achieved when the set of correct sentences constitute the top part of the ranking, while in the reverse case the smallest, but still positive value is attained.

Similar to the two correlation coefficients, average precision is a global ranking measure, which looks at the entire ranking and evaluates it. However, contrary to the former two, which assess the actual assigned ranking scores, average precision is a binary concept which only discriminates a sentence as being correct or incorrect. Thus, while a good Pearson correlation often comes with a good Spearman correlation, it does not necessarily imply a good average precision, and vice versa.

### 3.3.4 Pairwise Accuracy

Another reasonable assumption about correct sentences is that they should always be ranked higher than their incorrect counterpart. This idea is captured by the pairwise accuracy measure. Since the datasets we use provide corrections of all errors in a sentence, we can compare the learner’s incorrect sentence to the corrected version. We do this for every occurring incorrect sentence, ignoring sentences without errors and resolving ranking ties by discounting them with factor 0.5 (\( S_c \) refers to the correct and \( S_{inc} \) to the incorrect version of a sentence \( S \)):

\[
\text{Pairwise} = \left| \frac{\{f_{\text{rank}}(S_c) > f_{\text{rank}}(S_{inc})\} + 0.5 \cdot \{f_{\text{rank}}(S_c) = f_{\text{rank}}(S_{inc})\}}{|\{\text{incorrect sentences}\}|} \right|.
\]

Pairwise accuracy is the only relative ranking measure, as it assesses the ranking by comparing the ranking scores of the two versions for each sentence in isolation, i.e. independent from any other item in the ranking. This is why a ranking with high pairwise accuracy does not necessarily correspond to a high global ranking quality.

### 3.4 Learning Procedure

The following pseudo code sketches our basic learning procedure. Here, \textbf{Seed}, \textbf{Train}, and \textbf{Pool} are the seed, training, and pool set respectively (as in section 2.3), while \textbf{Test} is an independent set for evaluation of the performance of the model in our final experiment. The development set \textbf{Dev} is another evaluation set, which is also used for our stopping criterion.

```plaintext
Seed = Train_0 := read(seedDataset)
Pool_0 := read(poolDataset)
Dev := read(developmentDataset)
Test := read(testDataset)
r := 0
repeat
```
This procedure entails various parameters, which can be chosen according to the task. We briefly present the most fundamental and common of these. However, information about specific values is deferred to the evaluation section 4.

**Seed Size and Selection Method** The concept of active learning does not specify how the start training set is obtained. On the one hand, the seed size is a fundamental quantity. A seed should contain enough, but not more instances than necessary to learn a decent enough initial model. On the other hand, in our experiments we try both the option of a randomly selected and of an actively selected seed. In the latter case we use the sentence length as a simple utility criterion for selection – the longer a sentence, the more structure and possibly errors it contains, which should provide the learner with more informative initial data for the task of predicting the linguistic quality of sentences.

**Batch Size** Experiments have shown that the choice of the batch size can have a big effect on the performance of the active learning method, see e.g. Dzyuba et al. [2013]. A smaller batch size is motivated by the fact that the ranking model is updated relatively more often with respect to the size of the training set. Consequently, the learning procedure is more adaptive and reaches a better performance faster (w.r.t. the training set size). On the contrary, a greater batch size allows for more information per learning step, and also reduces the number of overall rounds required to achieve desired results.

**Stopping Criterion** Since active learning aims to reduce the required training data, the learner needs to decide when to stop to actually reduce the annotation effort. We experiment with two different stopping criteria:

- **Fixed Rounds**: Active learning is repeated for a fixed number of rounds.
- **Variance Threshold**: The learner stops as soon as its performance on the development set Dev has stabilised sufficiently, i.e. the variance of performance values (see
section 3.3) in a window of the most recent rounds has fallen below a pre-defined threshold.

3.5 Selection Technique

Most notably though, we investigate the problem central to the active learning concept: How to best choose new training instances in each round to maximise the learning effect and consequently the performance gain. In the following we introduce our active learning setup for rank learning. The selection techniques are based on the notion of a relevance measure – a term that aims to subsume methods which select the batch of instances per round in a greedy fashion. We first introduce the concept of relevance measures, then describe how they are used in our greedy selection algorithm, discuss performance issues, and finally contrast selection methods found in the literature to our own.

3.5.1 Relevance Measures

A relevance measure is a mapping $\text{rel} : F \to \mathbb{R}$, assigning a relevance value to an instance with regard to its estimated uncertainty under the current trained model and its future performance gain potential. Our greedy selection strategies use relevance measures to choose batch instances according to the objective of maximising the relevance to the ranking model. We distinguish between comparison-based measures which determine relevance relative to other instances, and instance-based measures which assess instances individually. Most similar to our definitions are the different criteria presented by Dzyuba et al. [2013], from where our terminology is inspired.

Comparison-Based Measures

This type of relevance measure assesses the relevance of an instance in the pool by comparing it to the instances of either the current training or pool set. We consider three ways of comparing two instances $x_i$ and $x_j$ and calculating their difference $\text{diff}(x_i, x_j) \in \mathbb{R}_{\geq 0}$:

- The euclidean distance $\text{dist}(x_i, x_j)$ captures the common understanding of a distance between two points. It incorporates both the length and the orientation of the vectors. However, length-dependent measures are often seen as less appropriate in the context of feature vectors. The euclidean distance is calculated as
  \[
  \text{dist}(x_i, x_j) = \| x_j - x_i \| = \sqrt{(x_j - x_i) \cdot (x_j - x_i)} \in \mathbb{R}_{\geq 0}.
  \]

- The cosine dissimilarity $\text{cosd}(x_i, x_j)$ captures the difference in orientation between two vectors. As it only focuses on the orientation, it is a length-independent measure, and consequently often preferred over the euclidean distance. It is calculated as
  \[
  \text{cosd}(x_i, x_j) = \frac{1}{\pi} \cdot \left( \pi - \arccos \left( \frac{x_i \cdot x_j}{\| x_i \| \cdot \| x_j \|} \right) \right) \in [0, 1].
  \]
The ranking score difference \( \text{rankdiff}(x_i, x_j) \) is the difference of the ranking scores for \( x_i \) and \( x_j \), which is equal to the classification certainty of its pairwise difference equivalent:

\[
\text{rankdiff}(x_i, x_j) = f^\text{rank}_w(x_j) - f^\text{rank}_w(x_i) = |w \cdot x_j - w \cdot x_i|
\]

These difference measures allow us to compare a given instance \( x \in \text{Pool} \) to the training or the pool set \( A \in \{\text{Train}, \text{Pool}\} \). The resulting set of difference values can then be used to calculate the relevance of this instance \( x \) with respect to the set \( A \). We introduce three different ways to do this (based on an underlying comparison function \( \text{diff}(\cdot, \cdot): \mathcal{F} \times \mathcal{F} \rightarrow \mathbb{R}_{\geq 0} \)):

- The diversity (inverse: similarity) of an instance \( x \in \text{Pool} \) from the respective set \( A \) is defined as the minimum distance to this set:

\[
\text{div}(\text{diff}, A)(x) = \min \{ \text{diff}(x, x') \mid x' \in A \}.
\]

- The marginality (inverse: centrality) of an instance \( x \in \text{Pool} \) from the respective set \( A \) is defined as the maximum distance to this set:

\[
\text{marg}(\text{diff}, A)(x) = \max \{ \text{diff}(x, x') \mid x' \in A \}.
\]

- The sparsity (inverse: density) of an instance \( x \in \text{Pool} \) from the respective set \( A \) is defined as the average distance to this set:

\[
\text{spars}(\text{diff}, A)(x) = \frac{1}{|A|} \cdot \sum_{x' \in A} \text{diff}(x, x').
\]

The first is a local measure considering only instances of minimal difference, while the latter two are global measures taking the entire ranking into account. A general guideline for selecting the most relevant instances – which is often found in the literature – is to select the ones which are similar to other unknown instances (i.e. small local difference from the pool set), but dissimilar to already labeled instances (i.e. great global difference from the training set).

### Instance-Based Measures

These measures assess the relevance of instances individually, ignoring their relation to other instances. We use the following two non-comparison-based relevance measures:

- An informativeness score \( \text{inform}(x) \) is assigned to an instance \( x \in \text{Pool} \), that represents an estimate of the likelihood of being informative and improving the ranking. Like in 3.4 for seed selection, we use the sentence length as an indicator of how much information a sentence is likely to hold, that is, the longer a sentence, the more linguistic structure and possibly errors it contains.

- The ranking score \( \text{rank}(x) \) of an instance \( x \in \text{Pool} \) is given by \( \text{rank}(x) = f^\text{rank}(x) = w \cdot x \). Maximising this measure means that only top-ranked pool instances are selected.
Manipulators & Combinators

Moreover, we use the inverse operation to invert the output of a relevance measure (e.g. to change the diversity measure to a similarity measure, see comparison-based relevance measures), addition and multiplication to add and multiply the result of different measures respectively, and difference to subtract two measures. In the literature one often finds a combination of two (or more) relevance measures to fuse an estimate for the local informativeness of an instance for the ranking model with an estimate for the global informativeness.

3.5.2 Three Selection Methods

The fact that instances in a ranking are dependent on each other – i.e. adding an instance results in many pairwise difference vectors being used during training – makes it often infeasible to non-greedily choose an optimal batch set per iteration, since one would have to evaluate this large set of difference vectors for each instance. We present three greedy selection techniques. The first is based solely on the concept of a relevance measure and chooses instances greedily according to their relevance. The second selects the pair of instances with the highest rank uncertainty, i.e. for which their pairwise difference vector is least certainly classified by the current model, optionally incorporating a relevance measure to form a more complex pairwise relevance assessment. The third method selects the entire batch at once, trying to maximise the inner uncertainty of all batch instances.

Single Selection

The first method, single selection, adds the instances with the highest relevance according to the measure used. This resembles the approach commonly used in active learning for classification models, which underlies the argument that the relevance of every instance should be evaluated individually. In the following we informally sketch the algorithm for our single selection technique:

```plaintext
Batch_r := new array of size b_r
M := new array of size b_r initialised to −∞
for x in Pool_r // iterate over instances in pool
    m := R(x) // calculate relevance of instance x
    if m > M[0] // does instance need to be added to the batch?
        for i := 0, 1, 2, ..., b_r − 1 // search index for new instance in batch
            if m > M[i] // is i the correct index for the new instance?
                for j := b_r − 2, b_r − 3, ..., i // shift other instances and relevance
                    M[j + 1] := M[j] // values accordingly
                    Batch_r[j + 1] := Batch_r[j]
                M[i] := m // insert new instance and relevance value
                Batch_r[i] := x
                break
        Train_r+1 := Train_r ∪ Batch_r // update training and pool set
    Pool_r+1 := Pool_r − Batch_r
```

20
Pairwise Selection

The second method, on the other hand, implements a pairwise selection technique. Here, pairs of instances are greedily selected according to maximising their ranking score uncertainty, resembling the argument that for the underlying model the uncertainty of the corresponding difference vector is most informative. Hence, it is essentially a local uncertainty measure as it uses the closest neighbours in the ranking. Moreover, this selection method optionally integrates an additional relevance measure, which is used to calculate the relevance value of both instances individually, and either the sum or product of the two values will then be multiplied with the ranking score uncertainty. We refer to these two modes as additive and multiplicative pairwise selection respectively. All in all, for two instances \( x_i \) and \( x_j \) of the pool and a relevance measure \( R: \mathcal{F} \rightarrow \mathbb{R} \), the value getting maximised by this greedy selection is:

\[
\frac{1}{f_{w}^{\text{rank}}(x_j) - f_{w}^{\text{rank}}(x_i)} \cdot \left( R(x_i) + /\cdot R(x_j) \right).
\]

In other words, here we select the pair of instances that maximises the above formula, which is based on a combination of their ranking uncertainty and their individual relevance. Below is an overview of the pairwise selection algorithm (we go into more detail about the sorted pool set and the next valid instance in the next section):

```
Batch_r := new array of size \( b_r \)
M := new array of size \( b_r \) initialised to \(-\infty\)

for \( x_i \) in Pool_r sorted by ranking score  // iterate over instances in pool
    \( x_j := \) next valid (minimum distance required) instance in sorted Pool_r
    \( m := (f_{w}^{\text{rank}}(x_j) - f_{w}^{\text{rank}}(x_i))^{-1} \)  // calculate ranking score difference
    \( m_1 := R(x_i) \)  // calculate relevance of individual instances
    \( m_2 := R(x_j) \)
    if multiplicative pairwise selection
        \( m := m \cdot m_1 \cdot m_2 \)
    else
        \( m := m \cdot (m_1 + m_2) \)
    if \( m > M[b_r - 1] \)  // does instance need to be added to batch?
        for \( i := 0, 2, 4, \ldots, b_r - 2 \)  // search index for new instance in batch
            if \( m > M[i] \)  // is \( i \) the correct index for the new instance?
                for \( j := b_r - 4, b_r - 6, \ldots, i \)  // shift other instances and relevance
                    \( M[j + 2] := M[j] \)  // values accordingly
                    \( M[j + 3] := M[j + 1] \)
                    \( \text{Batch}_r[j + 2] := \text{Batch}_r[j] \)
                    \( \text{Batch}_r[j + 3] := \text{Batch}_r[j + 1] \)
                \( M[j] := m \)  // insert new instances and relevance value
                \( M[j + 1] := m \)
                \( \text{Batch}_r[j] := x_i \)
                \( \text{Batch}_r[j + 1] := x_j \)
        continue outermost iteration after \( x_j \)
Train_{r+1} := \text{Train}_r \cup \text{Batch}_r  // update training and pool set
Pool_{r+1} := \text{Pool}_r - \text{Batch}_r
```
Batch Selection

The batch selection method is a direct implementation of the technique presented by Yu [2005] (see section 2.4), i.e. the batch set $\text{Batch}_r$ gets selected according to the following optimisation:

$$\text{Minimise } \sum_{(x_i \leq x_j) \in \text{Batch}_r} w_r \cdot (x_j - x_i).$$

3.5.3 Performance Issues

Due to the generally very large number of instances in the training and particularly the pool set, it is not possible to calculate the values of the comparison-based relevance measures in a feasible computing time. Evaluating these measures involves iterating over all the instances in training/pool set, and the evaluation has to be done for every pool instance in both the single and pairwise selection algorithm. This leads to a worst-case time requirement of $O(n^2)$ in the number of instances $n$. For most of the presented comparison-based measures this issue cannot be efficiently resolved, which is why we do not calculate the relevance measures on the entire training/pool set (if it contains too many instances), but on basis of a subset of 1000 randomly sampled instances.

However, the ranking scores take a special position in this setting. While both euclidean distance and cosine dissimilarity are based on using feature vectors of a high-dimensional vector space in their calculation, ranking score difference uses values of the 1-dimensional space $\mathbb{R}$. Consequently, these values can be ordered. The resulting sorted list of feature vectors gives rise to the following two improvements:

- The diversity relevance measure for ranking score differences can be computed efficiently in $O(\log n)$: We apply binary search to the sorted list to find the instance in question in $O(\log n)$, then calculate the difference to the nearest valid lower- and higher-ranked neighbour (see below), which takes constant time $O(1)$, and finally output the minimum difference. Similarly, the marginality measure for ranking score differences can be calculated in $O(1)$ by comparison with the lowest- and highest-ranked instance. Consequently, we do not use the randomly sampled subset of 1000 instances for those two measures.

- Since the pairwise selection technique is based on ranking score uncertainty, which is maximised by minimising its inverse – i.e. the corresponding ranking score difference – we do not need to consider every instance pair (leading to $O(n^2)$). Instead, we can iterate over the sorted list of instances and only need to form pairs with valid neighbours at each position, which gives a computation time of $O(n)$.

By a valid pair of instances we refer to a pair $x_i$ and $x_j$ for which the euclidean distance satisfies $\text{dist}(x_i, x_j) \geq 0.01$. For all the comparisons in our algorithms we apply this pruning threshold of having at least an euclidean distance of 0.01. It represents the idea that almost equal feature vectors are naturally ranked close together, since the represented sentences are linguistically very similar (with respect to the used feature set), and a rank
learning model does not benefit from focusing on discriminating them. The use of such a pruning technique is reported, for instance, by Dzyuba et al. [2013].

At the beginning of each active learning round, we sort both the current training and pool set once, and randomly sample the subset of 1000 instances for both sets, which is hypothesised to be representative of each set respectively.

3.5.4 Contrasting relevance measures to previous work

We conclude by presenting some selection techniques found in the literature, and contrast them to examples of techniques based on relevance measures, which are similar or even identical.

The various criteria of Dzyuba et al. [2013] can be almost directly transferred to a single selection method. inform + rank resembles their quality assessment, div(dist, Train) corresponds to their definition of diversity, and spars(dist, Pool) captures their density criterion. For the alternative uncertainty-based approach they describe we can formulate a similar measure as a multiplicative pairwise selection technique with the underlying relevance measure div(cosd, Train). Ours differs in that it does not calculate the cosine dissimilarity of pairwise difference vectors, but of the two feature vectors separately, combined multiplicatively, and that it does not combine the uncertainty of the pair additively with the cosine dissimilarity value, but multiplicatively.

The selection strategy of Qian et al. [2013] can be rephrased in terms of relevance measures as the additive pairwise selection technique, which captures the local uncertainty, with an appropriate global uncertainty relevance measure. While our system essentially allows to implement their entropy formula as a comparison-based measure (using entropy in the same way as, for instance, the diversity measure is using the minimum operation), we note that the presented entropy formula has some flaws, most seriously that the resulting value is constant:

\[
D(x_i, x_j) = \frac{w^T(x_i - x_j) - \min_{x_k} w^T(x_i - x_k)}{\max_{x_k} w^T(x_i - x_k) - \min_{x_k} w^T(x_i - x_k)}
= \frac{w^T x_i - w^T x_j - w^T x_i - \min_{x_k} - w^T x_k}{w^T x_i + \max_{x_k} - w^T x_k - w^T x_i - \min_{x_k} - w^T x_k}
= \frac{-w^T x_j - \max_{x_k} w^T x_k}{\min_{x_k} w^T x_k - \max_{x_k} w^T x_k}
= \text{constant in } x_i.
\]

Since the aim of their criterion is to select instances in the dense part of the pool, and not outliers, we think that using spars(rankdiff, Pool)\(^{-1}\) instead is more in the spirit of this idea.

The top- and mid-sampling methods of Arens [2011] are essentially similar to using rank for top-sampling and marg(rankdiff, Pool)\(^{-1}\) for mid-sampling in a single selection strategy.

Although not concerned with rank learning, the local structure measure of Tian and Lease [2011] for classification can be simulated in terms of relevance measures by defining their
$SL$ as $\text{div}(\cosd, \text{Train})$, $SN$ as $\text{div}(\cosd, \text{Pool})$ (setting $m = 1$ and only considering pool instances), combining them to $SN - SL$ (reverse order, since in our setting measures get maximised). We interpret the simple margin in the context of rank learning as the additive pairwise selection method with the aforementioned measure as inner relevance measure. The only difference is the multiplicative combination of simple margin and difference instead of an additive. We think that a multiplicative combination is more appropriate, since the value range of both measures is fundamentally different.

The popular approach of using the distance to the hyperplane, the simple margin, as a measure of the model’s uncertainty [Tong and Koller, 2002, Schohn and Cohn, 2000, Vlachos, 2004] can be represented in two ways using our setup. This ambiguity is due to the fact that, contrary to classification, adding an instance in our model implies adding multiple pairwise difference vectors to the training data. If one wants to focus on the uncertainty of a single instance with respect to the whole pool set, the appropriate relevance measure is $\text{div}(\text{rankdiff}, \text{Pool})^{-1}$. If, on the other hand, one intends to select the pair of instances which form the pairwise difference vector with the greatest uncertainty, ignoring the other difference vectors implicitly involved, the basic pairwise selection method (without any inner relevance measure) should be used.
Chapter 4

Experiments & Evaluation

In the following experiments we compare the active learning selection techniques described in section 3.5.1, including the relevance measures extracted from the literature (see section 3.5.4), to a passive learner, which is trained on the whole training set, and a random learner, which randomly samples instances of a batch size equal to the active learner’s one. The division of the datasets in training, development, and test set is fixed throughout this section. If not stated otherwise, results are obtained by running the same experiments four times with different seeds (using the same four seeds for all the different experiments) and averaging the results. This is done to remove noise and observe the average, i.e. “true”, efficiency of the relevance measures.

4.1 FCE Dataset Experiments

We begin by presenting the results on the FCE dataset. Since our active learning implementation allows for a broad range of different selection strategies, first we conduct a preliminary experiment to identify the most promising set of methods. For this we choose a seed size of 100 random instances and apply a batch size of 100 instances. The experiment is run for ten rounds and uses the FCE development set for evaluation. The results for the final performance after ten rounds can be seen in table 4.1. We also include the various selection methods (i.e. their adaptation to relevance measures) found in the literature as described in section 3.5.4. In the following, we refer to the active learners by their letter (e.g. (a)) assigned in this table 4.1.

We make two surprising observations: On the one hand, the pairwise accuracy values with almost entirely less than 0.5 are relatively low, given that simply always assigning “true” for any pairwise comparison of an erroneous sentence and its corrected version yields an accuracy of around 0.5\(^1\). On the other hand, the performance values for most of the methods, including the random learner, are close to the passive learner’s, while they use only a small fraction of less than 4% of the available pool data (1100 of 27798 instances). We defer a more thorough discussion of this observation to the next section.

\(^1\)It is assumed that the order of the comparison is randomly chosen, i.e. approximately half of the times it has the form “[correct] > [incorrect]” and the other half “[incorrect] > [correct]”.

25
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) Multiplicative Pairw.:</td>
<td>0.714</td>
<td>0.682</td>
<td>0.782</td>
<td>0.477</td>
</tr>
<tr>
<td>div(cosd, Train) [Dzyuba et al., 2013]</td>
<td>0.708</td>
<td>0.672</td>
<td>0.770</td>
<td>0.484</td>
</tr>
<tr>
<td>(b) Additive Pairwise:</td>
<td>0.710</td>
<td>0.677</td>
<td>0.771</td>
<td>0.471</td>
</tr>
<tr>
<td>spars(rankdiff, Pool)(^{-1}) [Qian et al., 2013]</td>
<td>0.697</td>
<td>0.672</td>
<td>0.769</td>
<td>0.479</td>
</tr>
<tr>
<td>(c) Additive Pairwise:</td>
<td>0.703</td>
<td>0.668</td>
<td>0.775</td>
<td>0.468</td>
</tr>
<tr>
<td>div(cosd, Pool) − div(cosd, Train) [Tian and Lease, 2011]</td>
<td>0.709</td>
<td>0.671</td>
<td>0.764</td>
<td>0.470</td>
</tr>
<tr>
<td>(d) Basic Pairwise:</td>
<td>0.709</td>
<td>0.671</td>
<td>0.764</td>
<td>0.470</td>
</tr>
<tr>
<td>[Tong and Koller, 2002]</td>
<td>0.707</td>
<td>0.670</td>
<td>0.775</td>
<td>0.461</td>
</tr>
<tr>
<td>(e) Additive Pairwise:</td>
<td>0.701</td>
<td>0.668</td>
<td>0.760</td>
<td>0.467</td>
</tr>
<tr>
<td>div(rankdiff, Pool)(^{-1}) [Tong and Koller, 2002]</td>
<td>0.713</td>
<td>0.667</td>
<td>0.747</td>
<td>0.462</td>
</tr>
<tr>
<td>(f) Additive Pairwise:</td>
<td>0.701</td>
<td>0.668</td>
<td>0.764</td>
<td>0.455</td>
</tr>
<tr>
<td>marg(rankdiff, Pool)(^{-1}) [Arens, 2011]</td>
<td>0.707</td>
<td>0.668</td>
<td>0.764</td>
<td>0.468</td>
</tr>
<tr>
<td>(g) Additive Pairwise:</td>
<td>0.700</td>
<td>0.656</td>
<td>0.759</td>
<td>0.453</td>
</tr>
<tr>
<td>marg(rankdiff, Pool)(^{-1}) [Arens, 2011]</td>
<td>0.689</td>
<td>0.655</td>
<td>0.750</td>
<td>0.482</td>
</tr>
<tr>
<td>(h) Inform:</td>
<td>0.697</td>
<td>0.656</td>
<td>0.759</td>
<td>0.453</td>
</tr>
<tr>
<td>inform + rank + div(dist, Train) + spars(dist, Pool) [Dzyuba et al., 2013]</td>
<td>0.691</td>
<td>0.646</td>
<td>0.747</td>
<td>0.413</td>
</tr>
<tr>
<td>(i) Basic Pairwise:</td>
<td>0.669</td>
<td>0.616</td>
<td>0.660</td>
<td>0.517</td>
</tr>
<tr>
<td>[Yu, 2005]</td>
<td>0.664</td>
<td>0.628</td>
<td>0.736</td>
<td>0.415</td>
</tr>
<tr>
<td>(j) Basic Pairwise:</td>
<td>0.657</td>
<td>0.617</td>
<td>0.731</td>
<td>0.419</td>
</tr>
<tr>
<td>[Arens, 2011]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.1: Performance of various selection strategies after ten rounds, starting with a randomly selected seed of 100 instances (0.36%) and applying a batch size of 100 instances over ten rounds, overall using 1100 instances (3.96%). The rows are sorted according to the average of the four evaluation scores, so that the selection strategies with a good performance in all four evaluation measures are high up in the table.
Apart from that, the results in table 4.1 suggest that pairwise selection performs better than the single selection method and definitely better than batch selection, since learner (f) is the only single selection method that on average manages to slightly outperform the lowest of the pairwise selection strategies. Still, some of the single selection methods excel for individual evaluation measures. Most notably, (h) leads with respect to the Pearson correlation coefficient (only 0.001 lower than the passive learner) and (o) has the best pairwise accuracy (even exceeding the passive learner by 0.04).

However, the apparent tendency towards pairwise selection confirms that there is no single criterion to best optimise relevance (contrary to classification, where instances are independent of each other), but that a synergy of different aspects is advantageous. Pairwise selection implicitly minimises the ranking difference by choosing the pair which is closest to each other in rank, and hence can be seen as a local, pool-focused relevance measure. In addition, it can explicitly integrate other measures which assess relevance training-set-based, e.g. (a), (c), (e), or globally, e.g. (b), (g). But even without integrating an inner relevance measure, i.e. solely basing instance-wise or pairwise selection on locally minimising the ranking difference to the pool set, (d) and similarly (f) are superior to the random learner. Since these are the only measures in average exceeding the random baseline, we conclude that classification certainty of their pairwise difference vector captures the relevance of a pair of instances for ranking best, and can be improved by including additional aspects like global or training-set-based information.

We select the six best selection techniques for a more detailed analysis of their active learning behaviour in the subsequent experiments. First we investigate the learning curves of the previous experiment, which can be seen in figure 4.1. In general, the random learner is outperformed across all the evaluation measures by active learning, most clearly in the case of pairwise accuracy, where (a) and (c) even exceed the passive learner. Furthermore, active learning performs almost consistently better with respect to Spearman correlation, and in the beginning also for Pearson, with (e) here being the best strategy. In the average precision diagram, the situation is less clear because of a strong random learner, but in the second half of the experiment the active learners become superior. Overall, strategy (a) is the best for average precision. Note that in all these cases the differences are not very large, at most around 0.01 to 0.02, or 0.03 for pairwise accuracy.

In a series of experiments we assessed the effect of varying the batch size. The results for a smaller batch size of 50 are presented in figure 4.2. We find that decreasing the number of instances per round negatively affects performance across evaluation measures. The performance of many active learners gets more unstable and their superiority over the random learner is less clear, with the exception of pairwise accuracy.

The case of a batch size of 200 (see figure 4.3) essentially shows similar results to the ones observed in figure 4.1. However, the advantage of active learning over the random baseline is less clear, particularly for Pearson correlation and average precision. Regarding pairwise accuracy, (a), (b) and (f) reach and occasionally outperform the passive learner. Further increasing the batch to 500 instances (see figure 4.4) confirms the observed tendency that active learning is clearly superior for pairwise accuracy and Spearman correlation, but with increasing batch size this is not the case for Pearson correlation and average precision.

We proceed by investigating the use of a more carefully selected seed. For this we use
Figure 4.1: Performance curves of several learners, starting with a randomly selected seed of 100 instances (0.36%) and applying a batch size of 100 instances over ten rounds, overall using 1100 instances (3.96%). The learners are: passive (gray), random (red), (a) (blue), (b) (green), (c) (violet), (d) (gold), (e) (brown), (f) (cyan).
Figure 4.2: Performance curves of several learners, starting with a randomly selected seed of 100 instances (0.36%) and applying a batch size of 50 instances over ten rounds, overall using 600 instances (2.16%). The learners are: passive (gray), random (red), (a) (blue), (b) (green), (c) (violet), (d) (gold), (e) (brown), (f) (cyan).
Figure 4.3: Performance curves of several learners, starting with a randomly selected seed of 100 instances (0.36%) and applying a batch size of 200 instances over ten rounds, overall using 2100 instances (7.55%). The learners are: passive (gray), random (red), (a) (blue), (b) (green), (c) (violet), (d) (gold), (e) (brown), (f) (cyan).
Figure 4.4: Performance curves of several learners, starting with a randomly selected seed of 100 instances (0.36%) and applying a batch size of 500 instances over ten rounds, overall using 5100 instances (18.35%). The learners are: passive (gray), random (red), (a) (blue), (b) (green), (c) (violet), (d) (gold), (e) (brown), (f) (cyan).
a simple way to assess the potential informativeness of an instance via its sentence length, similar to the informativeness relevance measure (see section 3.5.1). Apart from that, we use the same setup as in the original experiment (figure 4.1), i.e. 100 seed and batch instances over ten rounds. The learning curves in figure 4.5 show that initially the performance is very low, but in the first three rounds increases a lot. The performance of the active learners is much more unstable in this experiment, and across all performance measures apart from pairwise accuracy the random learner is superior to most of them, except when using selection strategy (e). However, for the random learner and for (e), such an actively selected seed is leading to better results and hence is advantageous. Looking at pairwise accuracy, the performance of the active learners are definitely superior and reach the highest values we reported so far after the first three rounds. In the last three rounds performance decreases again, with (d) being less effective than the random learner.

We conclude our experiments on the FCE dataset with a final experiment on the test set, using our original setup of both a randomly selected seed and a batch size of 100 instances each, which we found to be the overall best configuration. In this experiment, our alternative, more sophisticated stopping criterion is applied (see section 3.4), as opposed to just running an experiment for a fixed number of rounds. It uses the variance of the preceding performance values on the development set as a measure of stabilisation of a learner’s performance. More precisely, in each round (excluding the first nine) we calculate the variance of the values of the previous ten rounds for each performance measure, and check whether the result is below a threshold. If this is the case, the learning procedure terminates and the current model is used as the learner’s final ranking model. The threshold values are $10^{-6}$ for Pearson and Spearman correlation, $10^{-5}$ for average precision, and $5 \cdot 10^{-6}$ for pairwise accuracy, all of them chosen very roughly based on the observations in the preceding experiments. The results are shown in figure 4.6.

It can be seen that all learners terminate in round 24 to 34 (8.99% to 12.59% of the pool instances), except for (c). For both Spearman correlation and pairwise accuracy the active selection strategies are clearly superior to the random learner, with values 0.005 to 0.01 greater than random selection for Spearman correlation, and 0.005 to 0.017 greater for pairwise accuracy. In both cases, almost all active learners, except (e) for Spearman correlation, show a slightly (for Spearman correlation) or clearly (for pairwise accuracy) better performance than the random learner. In the case of Pearson correlation, the active learners except for (f) with an improvement of 0.005 to 0.01 are clearly better than the passive learner’s performance, but only at most around 0.005 superior compared to the random learner. Finally, the diagram for average precision shows that only the selection techniques (a) and (c) clearly improve over the random learner with an around 0.007 greater performance value. Moreover, these are the only two learners who achieve a performance slightly superior to the passive learner.

To show that our stopping criterion does not lead to a too early termination of the learning process, we compare the learning curves to the same experimental setup without stopping criterion, presented in figure 4.7. It becomes apparent that the performance would not have increased substantially after the stopping criterion ended the learning procedure. We calculate the mean for each learning curve after the termination point in an overall observation window of 50 rounds and compare it with the final performance of the learner with stopping criterion, which indeed gives us a deviation of at most around 0.0055 (in
Figure 4.5: Performance curves of several learners, starting with a seed of the 100 longest sentences (0.36%) and applying a batch size of 100 instances over ten rounds, overall using 1100 instances (3.96%). The learners are: passive (gray), random (red), (a) (blue), (b) (green), (c) (violet), (d) (gold), (e) (brown), (f) (cyan).
Figure 4.6: Performance curves of several learners with stopping criterion, starting with a randomly selected seed of 100 instances (0.36%) and applying a batch size of 100 instances over 50 rounds, overall using 5100 instances (18.35%). The learners and their respective data usage are: passive (gray) with 100%, random (red) with 12.59%, (a) (blue) with 11.15%, (b) (green) with 11.51%, (c) (violet) with 15.47%, (d) (gold) with 12.59%, (e) (brown) with 8.99%, (f) (cyan) with 8.99%. 
most cases still much smaller, though). Hence our stopping criterion provides an effective way for active selection methods to achieve a passive learner level of performance with only around 9% to 15% of the training data.

Our results imply that active learning helps to reduce the amount of training data required for a desired performance level, and is clearly superior over random selection for pairwise accuracy and Spearman’s rank correlation coefficient. A batch size of 100 instances is optimal; a smaller batch size destabilises performance, while a greater batch size compromises the superiority of active learning. The best selection strategy depends on the evaluation measure one wants to optimise, but our final experiment (see figure 4.6) suggests that the multiplicative pairwise selection method with integrated relevance measure $\text{div}(\cos d, \text{Train})$ belongs to the best for all evaluation measures except for Pearson correlation. Moreover, we identified a reliable stopping criterion, which leads to final performances close to or even better than the passive learner’s performance.

4.2 Analysis of the FCE Results

In this section, we investigate the task of learning to assess the linguistic quality of sentences with respect to the experiments on the FCE dataset in more detail. In particular, we want to find an explanation for the, in the context of active learning, “untypical” shape of most performance diagrams. In the literature, such diagrams commonly begin with a performance of around 0 for the initial seed, and subsequently increase continuously up to, or even exceeding, the passive learner’s results. At this, the active selection strategy usually shows an initially superior performance gain over random selection, which later on catches up and finally reaches the active learner.

Our diagrams differ in two major points from the above described “common” behaviour of active learning. On the one hand, our different selection techniques all either already start with a relatively high performance value or attain it very soon after, while the actual value range of the subsequent performance development is rather small. On the other hand, after almost promptly entering a high performance level, the behaviour of the curves does not show a clear monotone ascent, but a more noisy up and down tendency. These two facts in combination might question the clear superiority of the active learning method over the random learner, especially for Pearson correlation and average precision.

First, we are going to investigate whether the amount of initial training data matters for the high performance in the beginning, or whether this effect is independent of the number of seed sentences. For this, we run an experiment of ten rounds where we look at the output of a random learner starting with only two seed instances and incrementing its training set by two further instances per round. This is the most extreme case with only one pairwise difference vector as an initial set and the smallest possible pairwise-selected batch size. The resulting diagrams (based on the FCE test set) in figure 4.8 show that the performance starts surprisingly good (given a training set of only two instances), particularly the Spearman coefficient of 0.542 and the average precision value of 0.507 (passive learner: 0.700 and 0.747 respectively). After a severe drop (increase for pairwise accuracy) performance seems to stabilise on a level of around 0.65 for the Pearson coefficient.
Figure 4.7: Performance curves of several learners, starting with a randomly selected seed of 100 instances (0.36%) and applying a batch size of 100 instances over 50 rounds, overall using 5100 instances (18.35%). The learners are: passive (gray), random (red), (a) (blue), (b) (green), (c) (violet), (d) (gold), (e) (brown), (f) (cyan).
A relatively good level of performance indeed seems to be almost independent of the amount of training data. Instead, the stability of the performance seems to be affected by it, observing that the performance drop/increase in this example in the first two rounds is considerably larger than the noisy up and down of the curves presented in the previous section. This instability can be explained by the fact that such a small set of data might be a poor representation of the dataset, giving the ranking model a “wrong impression” of the situation and how to weight the features. However, the good initial performance when training on only the seed of two instances can be caused by a very small number of highly informative features which are present in every sentence, since such a small amount of training instances cannot possibly be very useful.

We claim that the good performance is mainly due to the two feature types corpus error-rate and rule-based errors. The former consists of one feature indicating the error-rate,
Table 4.2: Comparison of the performance of a passive learner for various feature sets.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>All features</td>
<td>0.763</td>
<td>0.700</td>
<td>0.747</td>
<td>0.487</td>
</tr>
<tr>
<td>Corpus</td>
<td>0.723</td>
<td>0.628</td>
<td>0.594</td>
<td>0.508</td>
</tr>
<tr>
<td>Rule-based</td>
<td>0.610</td>
<td>0.605</td>
<td>0.686</td>
<td>0.444</td>
</tr>
<tr>
<td>Corp. &amp; rule-based</td>
<td>0.746</td>
<td>0.687</td>
<td>0.728</td>
<td>0.480</td>
</tr>
<tr>
<td>All except former two</td>
<td>0.698</td>
<td>0.654</td>
<td>0.730</td>
<td>0.475</td>
</tr>
</tbody>
</table>

while the latter consists of features for a set of 20 error types found. Together they represent a very small number of virtually always occurring features, and consequently these two presumably are responsible for the behaviour in this experiment, where only a tiny fraction of the training data is used. We next assess the passive learner when trained on only these two features, and for comparison the case of training without these two. Table 4.2 summarises the results.

It becomes apparent that both feature types do comparatively well, corpus error-rate more for Pearson, Spearman, and pairwise accuracy, rule-based errors for average precision. However, especially their combination yields almost “optimal” values, with only at most 0.02 deviation from the passive learner’s performance when trained on all feature types. On the contrary, excluding these two feature types from the entire feature set results in similar values for average precision and pairwise accuracy, but decreases Pearson and Spearman correlation more severely by around 0.065 resp. 0.046.

Subsequently, we take the best two feature types (i.e. corpus- and rule-based errors) and investigate a random learner on the test set in an experiment of ten rounds with ten batch instances each, starting with a seed of ten sentences. The diagrams in figure 4.9 show that the high level of performance of a passive learner based on these two features can be achieved with only a very small number of randomly selected sentences (110 of 27798 instances, i.e. less than 0.5%). This explains the strong performance of the random learner in general, which turns out to be a hard baseline.

On the contrary, if we run the same experiment, but use all of our other feature types except these two, the performance is much lower, as can be seen in figure 4.10. In fact, the correlation coefficients are only less than 0.02 higher than the performance of a very simple non-machine-learning approach using sentence length as a rough approximation for estimating the linguistic quality of sentences, following the idea that longer sentences are likely to contain more errors and hence have a less overall linguistic quality. We get a Pearson/Spearman correlation of 0.616 and 0.569 respectively between error count and sentence length on the FCE test data. Moreover, sorting the FCE test data according to increasing sentence length and applying the average precision formula to the obtained ranking results in a value of 0.669, which again is only around 0.03 lower than the average precision of the rank learning model in the former experiment (see table 4.2).

Furthermore, we analysed the performance of a fully unsupervised approach of assigning ranking scores by counting the number of errors found in a sentence by the two feature types corpus error-rate and rule-based errors. Here, we obtain results (see table 4.3) considerably lower than the ones of the machine learning model using these two features.
Figure 4.9: Learning curves of a random learner (green) with reduced underlying feature set (corpus error-rate, rule-based errors), starting with a randomly selected seed of ten instances and using a batch size of ten instances over ten rounds. For comparison the performance of the passive learner (red) with the entire feature set is shown.

While the Pearson correlation coefficient is virtually the same, the Spearman coefficient is around 0.04 lower than the former value, average precision even more seriously with around 0.11, and pairwise accuracy around 0.12. The inferior results are explained by the fact that the learning model additionally weights the different error features.

We demonstrated how the initial high performance is due to the two explicitly error-focused features corpus error-rate and rule-based errors and rather independent from the actual amount of used data. A small number of around 100 instances is sufficient to achieve a performance very close to the passive learner trained on all instances and using the entire feature set (see figure 4.9. Even though these feature types have such a high impact mainly because they are less based on information extracted from instances and more on external knowledge like an error corpus or error rules, we have shown that the machine learning model is still essential to achieve this performance by optimising the weighting of the various error types found in a sentence.
4.3 Analysis of the WI Dataset

Herein, we explore the usefulness of the WI gold sentence-level scores. When training a passive model on these gold scores and evaluating it on the test set, we observe a very low Pearson/Spearman correlation coefficient of only 0.162 resp. 0.180 with respect to these gold sentence scores, while the resulting ranking has an average precision value of 0.368. In another experiment we evaluated the learner’s performance on the training set itself. However, it showed that the performance is still quite los (with 0.365, 0.355, and 0.344 respectively). Below we present an attempt to explain why the results are so low.

Such a poor performance indicates that our machine learning approach has difficulties to systematically learn these gold scores. Possible reasons for these difficulties are:

- The annotation of the dataset cannot be used to learn absolute rankings, but rather only relative ones that depend on the script from which the sentences come.
Table 4.3: Comparison of the performance of the passive learner with various unsupervised approaches based on counting corpus- and rule-based errors.

<table>
<thead>
<tr>
<th></th>
<th>Pearson</th>
<th>Spearman</th>
<th>Avg. Prec.</th>
<th>Pairwise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passive Learner</td>
<td>0.763</td>
<td>0.700</td>
<td>0.747</td>
<td>0.487</td>
</tr>
<tr>
<td>Corpus</td>
<td>0.723</td>
<td>0.628</td>
<td>0.605</td>
<td>0.379</td>
</tr>
<tr>
<td>Rule-based</td>
<td>0.513</td>
<td>0.408</td>
<td>0.383</td>
<td>0.394</td>
</tr>
<tr>
<td>Corp. &amp; rule-based</td>
<td>0.746</td>
<td>0.648</td>
<td>0.616</td>
<td>0.358</td>
</tr>
</tbody>
</table>

- The representation of sentences by the employed feature set is insufficient\(^2\).

Regarding the first point, we observed that there seems to be a correlation between the sentence-level and the script-level scores (Pearson/Spearman correlation coefficient: 0.563 and 0.656). This suggests that the sentence-level scores are best understood as relative \textit{intra-script} scores, i.e. inside the scope of a script, but are less useful when aiming for \textit{inter-script} comparability (which is the way we try to use them). The distribution of deviations in figure 4.11 shows that the majority of sentence-level scores are \textit{very close} to the script score, with only 169 of 1016 differing more than 2 from the script score (in a range from 0 to 13). Furthermore, we observe a tendency towards sentence scores that are lower than the respective script score.

We also analysed the sentence ranking given by the gold scores, and report a Pearson/Spearman coefficient of 0.209/0.247 to error counts per sentence and an average precision of 0.514\(^3\). These low values show that the gold scores do not directly capture aspects of

\(^2\)We remark, though, that we experimented with various other feature combinations available in our implementation (e.g. lemma n-grams, presence of unique words) without success.

\(^3\)Note that the lowest possible value is 0.225 when placing all the correct sentences at the low end of a
linguistic quality.

These observations strengthen our assumption that the sentence-level gold scores represent a different ranking problem, which is concerned about the score of sentences within scripts, and not the linguistic quality of a sentence in isolation. In the case of this new task, sentence scores seem to be *locally* dependent on the context of the containing script and thus are not *globally* comparable amongst scripts. Perhaps training a model with pairwise constraints per script will be better suited on this dataset.
Chapter 5

Conclusion and Future Work

In this work we presented a generic active learning system which is able to capture many of the approaches for an active selection strategy found in the literature. We applied this system to the problem of assessing the linguistic quality of sentences in the context of automated English learner examination, building on an existing machine learning approach to the problem, as presented by Andersen et al. [2013]. Experiments are evaluated by four performance measures capturing different aspects of sentence quality.

Our experiments show that it is possible to reduce the training data substantially, while still resulting in virtually the same performance. The effect of incorporating active learning is positive, although small, and its use depends on the performance evaluation measure one focuses on and on the selection technique applied. We find that it is particularly useful for the two ranking performance measures Spearman’s rank correlation coefficient and pairwise accuracy, where some selection techniques show better results while simultaneously reducing the amount of training data by around 90%.

The reason for an only small positive effect is due to a surprisingly strong performance of the random selection baseline. A more thorough investigation of the matter revealed that randomly selecting less than 0.5% of the entire training set, and applying a rank learning model with a feature set reduced to corpus- and rule-based error features yields performance values close to the non-reduced setup using the whole training set. Consequently, the amount of training data can be dramatically reduced when one is willing to accept a resulting performance slightly lower than might be possible. Whether the relatively small additional improvement coming with the integration of active learning is useful and practically relevant, depends on the application context. We leave it for future work to test the significance of this improvement.

Furthermore, we assessed the usefulness of a small dataset with gold annotations at the sentence-level. However, we observed that the sentence gold scores of this dataset exhibit a non-negligible correlation within a script and are only weakly correlated to the number of errors per sentence. This suggests that linguistic quality is not an aspect modeled by these gold scores. Moreover, we observed that they are not suitable for our machine learning setup and evaluation methods. It is worth investigating in future work, whether a different model which takes script constraints into account is more suitable here, and how the results can be used in the context of an automated assessment and feedback system.
Bibliography


