DELFT UNIVERSITY OF TECHNOLOGY

MASTERS THESIS

Automated Writing Feedback

Author:
Paul VAN DER LAAN

Supervisors:
Prof. dr. Marcus SPECHT and
Manuel VALLE TORRE

A thesis submitted in fulfillment of the requirements
for the degree of Master of Science

in the

Web Information Systems Group
Software Technology

September 24, 2021

DELFT UNIVERSITY OF TECHNOLOGY

Abstract

Electrical Engineering, Mathematics and Computer Science
Software Technology

Master of Science

Automated Writing Feedback

by Paul VAN DER LAAN

The rising number of students challenges the teacher's time-consuming task to provide consistent and high-quality feedback for all students. To address the traditional education's challenges, researchers refer to online educational tools to assist teachers. Although large writing tools (for instance, Grammarly and Microsoft Editor) assist students to write effectively, their primary objective is not to educate students. Therefore, we propose *RevisionCoach* – an automated writing feedback system that iteratively constructs educational, localized feedback to assist students to learn how to write. Clear and effective writing is important for students to succeed in academic endeavors and allows teachers to focus on feedback for the assignment's primary task. *RevisionCoach*'s objective is to educate, and for that reason, the design considers learning by deliberate practice, differentiated learning, and self-regulated learning. In addition, *RevisionCoach*'s feedback has four layers: a sentence-level mistake highlight, an assessment category (coherence, cohesion, readability, and formality), a correction category (rewrite, reword, and rephrase), and a correction suggestion.

RevisionCoach's categorized feedback allows for convenient evaluation that addresses *RevisionCoach*'s capability to predict the writing mistake's importance. In the evaluation, we ask writing experts, students, Grammarly, and Microsoft Editor to find writing mistakes in text and to rate the mistake's importance. Compared to the experts' importance predictions, *RevisionCoach* predicts the mistake importances more accurately (1.89 mean square error (MSE)) than a random baseline (2.70 MSE), student baseline (2.50 MSE), Grammarly (2.52 MSE), and Microsoft Editor (2.02 MSE). Furthermore, the study illustrates the experts' challenge to provide localized feedback for writing skills because the experts have merely 71.2% agreement about the top 2 most severe mistakes. At the same time, *RevisionCoach* achieves an average agreement of 72.9% with the experts' mistake predictions.

Acknowledgements

This thesis project depended on several people who supported me with ideas, feedback and who made the entire thesis process more enjoyable. First, I like to thank my supervisors at TU Delft, prof. dr. Marcus Specht and Manuel Valle Torre for the interesting discussions about education, automated systems, and the thesis project in general. Furthermore, the educational technology company FeedbackFruits for the resources and support, particularly Joost Verdoorn and Francisco Morales from the research and development team for introducing me to large-scale educational systems, ideas, resources, and interesting discussions. Last, I would like to thank everyone who participated in the user studies to evaluate the automated writing feedback systems.

Contents

Abstract **iii**

Acknowledgements **v**

1 Introduction **1**
- 1.1 Problem Statement . . . 2
- 1.2 Research Questions . . . 3
- 1.3 Proposed Solution . . . 3
- 1.4 Contributions . . . 4
- 1.5 Report Organization . . . 5

2 Related Work **7**
- 2.1 Automated Essay Scoring . . . 7
- 2.2 Automated Writing Evaluation . . . 9
- 2.3 Feedback Systems . . . 10

3 Requirement Analysis **15**
- 3.1 Discussions with Experts . . . 16
  - 3.1.1 FeedbackFruits . . . 16
  - 3.1.2 Professor with Education System Experience . . . 17
  - 3.1.3 Professor with NLP Experience . . . 17
  - 3.1.4 Writing Expert . . . 18
  - 3.1.5 Educational Systems Expert . . . 18
- 3.2 Feedback Data Exploration . . . 18
- 3.3 Final Requirements . . . 19

4 Conceptual Design **21**
- 4.1 Explored Automated Feedback Systems . . . 21
  - 4.1.1 Feedback Linking . . . 21
  - 4.1.2 Feedback Generation . . . 23
    - Data Preprocessing . . . 23
    - Feedback Generation . . . 24
    - Feedback Generation Evaluation . . . 25
- 4.2 RevisionCoach . . . 26
  - 4.2.1 Revision Models . . . 28
    - Rephrase Revision . . . 28
    - Rewrite Revision . . . 29
    - Reword Revision . . . 29
  - 4.2.2 Scoring Models . . . 29
    - Readability . . . 30
    - Coherence . . . 30
    - Cohesion . . . 30
    - Formality . . . 30

4.2.3 Feedback Model . . . 31
4.2.4 User Interface Design Prototype . . . 31

5 Implementation **33**
5.1 Revision Models . . . 33
5.1.1 Background of Transformer Models . . . 34
BERT and GPT-2 . . . 35
T5 . . . 36
5.1.2 Rephrase Revision . . . 37
5.1.3 Rewrite Revision . . . 38
Text Generation Architecture . . . 38
Fine-Tuning T5 . . . 39
5.1.4 Reword Revision . . . 41
5.2 Scoring Models . . . 42
5.2.1 Readability . . . 42
5.2.2 Coherence . . . 43
LSA and Word2Vec . . . 43
ELMo and BERT . . . 44
5.2.3 Cohesion . . . 45
5.2.4 Formality . . . 45
5.3 Feedback Controller . . . 45
5.4 User Interface Design . . . 46

6 Evaluation **47**
6.1 Method . . . 47
6.2 Study . . . 48
6.2.1 Study Biases . . . 48
6.2.2 Study Setup . . . 49
6.3 Limitations . . . 50

7 Results **51**

8 Discussion **57**
8.1 Interpretation of *RevisionCoach*'s Evaluation Results . . . 57
8.2 Answers to the Research Questions . . . 60

9 Conclusion **63**

10 Future Work **65**

A Data Exploration **67**
A.1 Dataset Preprocessing . . . 68
A.2 Exploration . . . 69

B T5 Text Generation Fine-Tuning Predictions **73**

Bibliography **75**

Chapter 1

Introduction

The rising number of students confronts teachers with the time-consuming challenge of providing consistent and high-quality feedback. During the learning process, feedback is an essential step for students to learn how to write well (Bernius, Krusche, and Bruegge, 2021). Students that know how to write are more likely to persuade others and are more likely to succeed in academic endeavors (McNamara et al., 2015). Unfortunately, writing assessment is labour intensive (Woods et al., 2017); and therefore, there is an opportunity for automated feedback systems. Feedback systems need to comprehend text quality and construct educational feedback for recognized text patterns in order to support teachers with writing assessment. Consequently, teachers and students gain significant benefits in terms of time consumption, feedback availability, and feedback consistency.

Online peer reviews are a common technology to address the teacher's time-consuming challenge to provide high-quality and consistent feedback. In the best scenario, students receive high-quality feedback on their work right away. Literature supports the usefulness of peer reviews and recognizes that the ability to learn with peers is a key skill for lifelong learning (Thomas, Martin, and Pleasants, 2011; Nicol, Thomson, and Breslin, 2014; Ibarra-Sáiz, Rodríguez-Gómez, and Boud, 2020). However, peer reviews do not replace the teacher's valuable expert-level feedback to enhance the student's learning process. To analyse the student's learning process, we consider Bloom's Taxonomy (Kennedy, Hyland, and Ryan, 2006), that indicates that teaching and assessing students should support the student's learning process to move the student's abilities to a higher stage, from knowledge, comprehension, application, analysis, synthesis, to evaluation. Although a byproduct of peer reviewing is that students receive feedback (from other students that are learning the material), the primary purpose is to assist students to master mainly the top level of Bloom's taxonomy - the ability to evaluate. To this end, feedback is essential to assist students to acquire skills required to learn how to evaluate, because targeted high-quality feedback enhances learning in three significant ways (Boud and Falchikov, 2007): (1) accelerates the learning process, (2) optimizes the quality, and (3) raises the achievement standards.

To address the peer reviews' issues, automated essay scoring (AES) and automated writing evaluation (AWE) construct automated feedback for students. Automated feedback systems that address time, cost, reliability, and generalizability issues in writing assessment introduce new concerns related to vulnerability to cheating and the requirement of a large dataset (Dikli, 2016). Although AES systems achieve promising results (Rudner and Liang, 2002; Lonsdale and Strong-Krause, 2003; Dikli, 2016; Taghipour and Ng, 2016; Ke and Ng, 2019), the systems output general and holistic scores that are less useful in a classroom setting, where detailed feedback help students more to improve their essays (Ke and Ng, 2019). In addition, part of the AES methods requires significant feature engineering work and data annotation (Li et al., 2018) that challenges the model's generalizability. AES methods that circumvent feature engineering rely on neural-network-based or transformer-based methods that analyze text directly, where a significant trade-off is the explainability of the system's reasoning.

On the other hand, AWE research addresses the challenge to transform a holistic essay-level score into localized feedback. In order to provide localized feedback, the essay-level score is transformed into a sentence-level score, which is a procedure that received relatively little attention (Andersen et al., 2013). The obvious challenge concerns the dataset restrictions because a dataset that requires significant manual effort to annotate is impractical to generalize to unexplored assignments. In addition to the dataset restrictions, AWE methods need to address the challenge of presenting feedback in an educative manner and taking the student's interaction with the system into account to stimulate positive learning effects for novice to advanced learners. The role of feedback is essential for learning, and in this thesis, we explore automated feedback systems and we propose a new automated feedback system that assists students with learning how to write high-quality text.

1.1 Problem Statement

This thesis focuses on accurate feedback automation for essay assignments that encourages long-term learning effects for students. Ideally, an automated writing feedback system uses educational principles to encourage long-term learning effects. In addition, the automated feedback system needs to address generalizability to other assignments because feedback data, even holistic essay scores, is expensive and difficult to annotate (Hellman et al., 2020).

To start, the assignment choice. A perfect feedback system that works for the rarest assignment type is not the goal of this thesis, but instead a feedback system that works for a common assignment type to drastically reduce the teacher's feedback time consumption. Essay assignments are common and appear in most programs, where an essay is any text between 100 to 800 words, excluding figures and tables. On the contrary to text, figures and tables are outside this thesis' scope since automatically understanding figures and tables is a challenging problem that falls beyond the scope of this thesis.

Students are more likely to increase learning efforts with clear goals and clear beliefs of eventual success (Hattie and Timperley, 2007). Furthermore, (automated) feedback helps students to clarify goals and to enhance commitment to reach the goals (Hattie and Timperley, 2007). Although the teacher defines the essay's goal, we define the clear goal to learn how to write high-quality text, and students can recognize effective writing as a tool for eventual success. When the feedback system focuses on common writing mistakes, teachers can focus on feedback that requires a deeper understanding of the essay.

Moreover, feedback accuracy is important because an inaccurate system will be challenging and frustrating for students, counteracting the potential learning effects. In addition to accurate feedback, effective feedback is targeted at students at an appropriate level (Hattie and Timperley, 2007) to reduce the gap between current and desired understanding effectively.

The last challenge is to evaluate the proposed automated feedback system. Natural text is challenging to evaluate because of its open-ended nature (Celikyilmaz, Clark, and Gao, 2020). Therefore, the proposed system needs to be structured in a way that is possible to evaluate effectively. In the following chapter, we define the research questions.

1.2 Research Questions

Given the problem statement, we define the following research questions:

1. How to support learners with feedback on essays?
2. How can feedback be presented to improve the student's learning process?

The first research question is the most important given the current state-of-the-art systems, what is effective, and how to build a better system that assists students to write better essays. On the other hand, the second research question raises an important question: how to visualize feedback effectively to encourage long-term learning effects for students. We propose the following solution for these research questions.

1.3 Proposed Solution

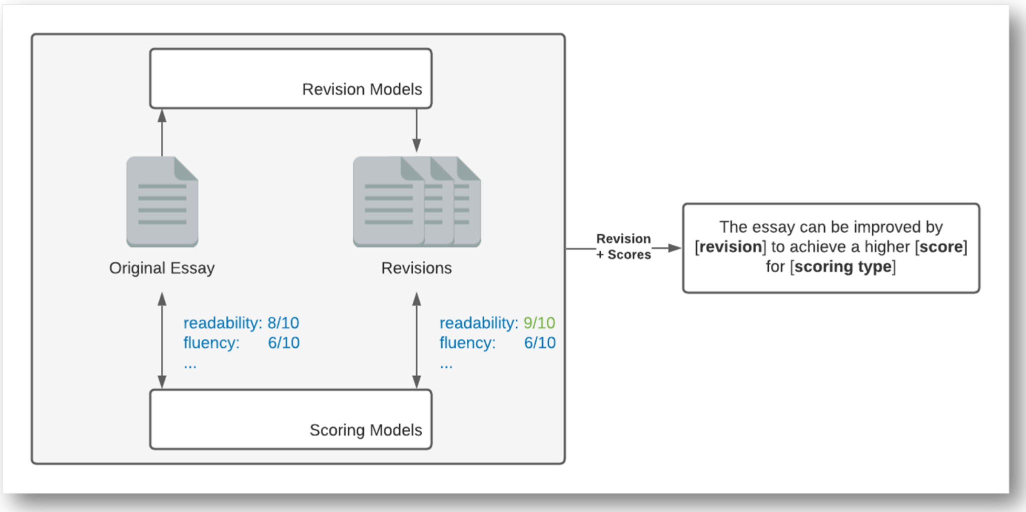


FIGURE 1.1: *RevisionCoach* system flow

In this thesis, we propose *RevisionCoach*, a feedback system for essays that focuses on writing issues between lower-order concerns (for instance, the grammar and spelling) and higher-order concerns (the 'big picture', for example, the structure, development, and reasoning). *RevisionCoach* has two main design goals: (1) assist students how to improve their essays by providing feedback, to help teachers with time-consuming feedback iterations, and (2) the ability to differentiate and to provide helpful feedback for novice to advanced learners.

To achieve this, RevisionCoach focuses on cohesion, coherence, formality, and readability. In contrast to Grammarly[1] and Microsoft Word[2], RevisionCoach disregards spelling and grammar feedback and offers a learning-oriented environment to

[1] https://www.grammarly.com/
[2] https://www.microsoft.com/en-us/microsoft-365/word

encourage students to develop long-term essay writing skills. Moreover, instead of correcting mistakes immediately, learning is enhanced when students process information on a deeper level (Kirschner and Hendrick, 2020).

RevisionCoach aims to provide structure and tips for students to think about possible textual improvements, to encourage students to process information on a deeper level. Intuitively, whenever an advanced writer writes text, the writer revises the text until the text reaches an acceptable quality level. *RevisionCoach* follows this concept by generating revisions for individual sentences and by predicting scores that reflect the essay's quality. For each revision, *RevisionCoach* calculates formality, readability, cohesion, and coherence scores. Revisions that score significantly higher in one of the scoring categories are used as feedback for the student. The process to iteratively focus on one writing quality element at a time and to encourage students to make gradual improvements relates to deliberate practice (Anders Ericsson, 2008), that improves the student's learning performance. In addition, *RevisionCoach* encourages self-regulated learning (Pintrich, 1995) because students set their own learning goals (when is the quality acceptable), and the students are in control of their learning process.

In addition to learn by deliberate practice and to encourage self-regulated learning, *RevisionCoach* supports differentiated learning by providing layers of feedback. The feedback layers that *RevisionCoach* presents rely on the revisions and scoring strategies. First, *RevisionCoach* highlights sentences that can be improved. Second, the score category indicates what aspect of the sentence can be improved. Predefined tips and suggestions per scoring category are helpful to aid the student in improving the sentence. Third, *RevisionCoach* can provide an improvement suggestion that relates to the revision model – for example, to reword, rephrase, or rewrite. And finally, the complete revision can be shown to the student as the sentence's possible correction to inspire the student. These four layers of feedback options support differentiated learning because advanced learners can consider feedback highlights as useful feedback, while more novice learners potentially need more feedback layers.

Finally, the system is evaluated in a study that compares *RevisionCoach*'s feedback to expert feedback in order to investigate *RevisionCoach*'s feedback accuracy. In the comparison the inter-rater agreement between *RevisionCoach* and the expert is used as evaluation metric.

1.4 Contributions

RevisionCoach embodies the main contribution of this research project. The following list summarizes all contributions of this project:

1. ***RevisionCoach***: a feedback system on essay writing style. The system uses a combination of existing methods and models to expand the methodology of Woods et al., 2017 to find sentences that need more work ('highlights'). On top of that, *RevisionCoach* informs the user of the highlight's mistake category and proposes potential highlight corrections and considers educational theories (learning by deliberate practice, self-regulated learning, and differentiated learning). While other feedback systems require expensive, time-consuming manual annotation that reduces scalability and generalizability to other assignments, *RevisionCoach* merely needs a dataset of domain-dependent essays without additional manual annotations.

2. **Exploration of possible feedback systems**: we explored three versions of feedback systems in total, and *RevisionCoach* uses components of the first two versions. Instead of covering only the final system, we discuss the advantages and disadvantages of other intuitive automated feedback systems.

3. **Google Colabs Notebook User Interface Prototype**: the benefit of a notebook UI is that the revision and scoring models are loaded once in the notebook and are stored in the notebook's memory. The colab notebook allows for fast development cycles and removes the need to deploy code to an external server in order to hold the revision and scoring models in memory. Any code revision is automatically deployed and the UI can be updated immediately. *RevisionCoach*'s UI prototype showcases the potential of *RevisionCoach* as effective writing assistant and provides the opportunity for long-term learning effects studies.

4. **Expert evaluation**: a formative first study that assesses *RevisionCoach*'s feedback accuracy and compares *RevisionCoach*'s feedback predictions to Grammarly, Microsoft Editor, computer science students (MSc), and the random baseline.

1.5 Report Organization

In this thesis, we discuss the related work in chapter 2 and use that to define the initial system requirements in chapter 3. After the requirements, we discuss the conceptual design in chapter 4, which shows the design ideas to construct an education automated writing feedback system. Furthermore, in chapter 5 we elaborate upon the conceptual design and we provide the implementation details. The last part of this thesis covers the evaluation (chapter 6), results (chapter 7), discussion (chapter 8), conclusion (chapter 9), and future work (chapter 10).

Chapter 2

Related Work

Automated feedback research on essay writing relies on automated essay scoring (AES) and automated writing evaluation (AWE) literature. In this chapter, we discuss how AES methods are used in AWE to construct formative feedback. In addition to the AES and AWE background, we discuss several deployed feedback systems in this chapter's last section.

2.1 Automated Essay Scoring

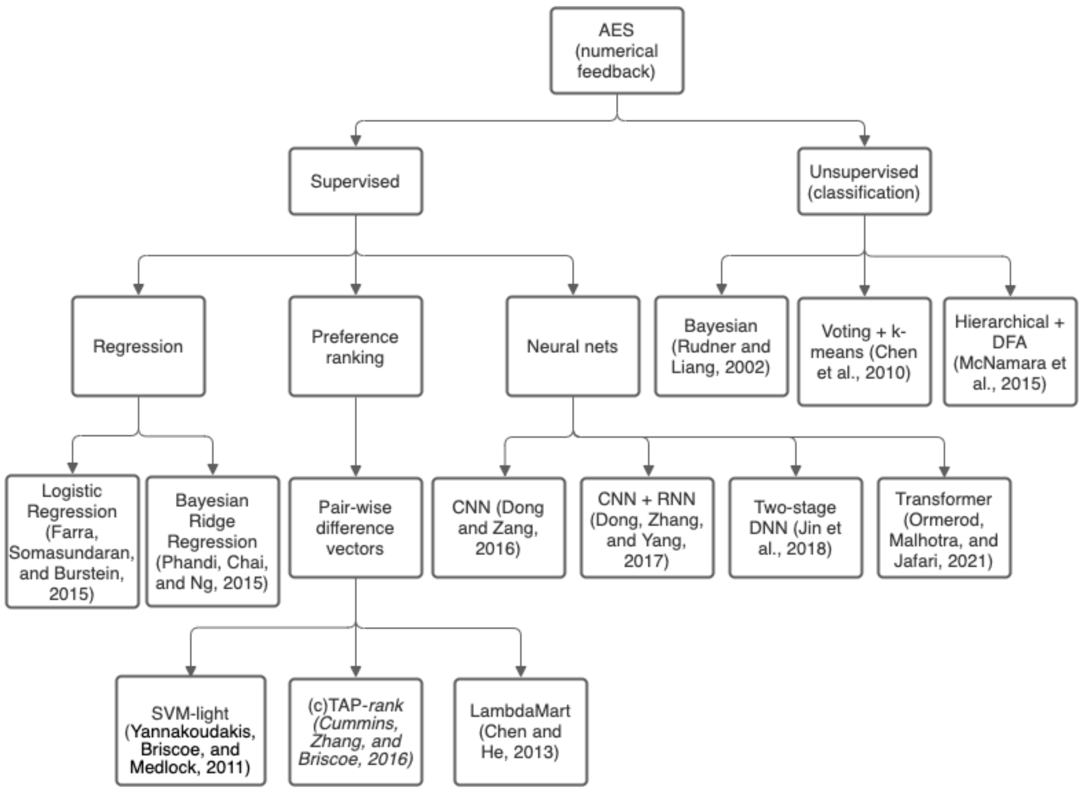

AES (numerical feedback)
Supervised
Unsupervised (classification)
Regression
Preference ranking
Neural nets
Bayesian (Rudner and Liang, 2002)
Voting + k-means (Chen et al., 2010)
Hierarchical + DFA (McNamara et al., 2015)
Logistic Regression (Farra, Somasundaran, and Burstein, 2015)
Bayesian Ridge Regression (Phandi, Chai, and Ng, 2015)
Pair-wise difference vectors
CNN (Dong and Zang, 2016)
CNN + RNN (Dong, Zhang, and Yang, 2017)
Two-stage DNN (Jin et al., 2018)
Transformer (Ormerod, Malhotra, and Jafari, 2021)
SVM-light (Yannakoudakis, Briscoe, and Medlock, 2011)
(c)TAP-rank (Cummins, Zhang, and Briscoe, 2016)
LambdaMart (Chen and He, 2013)

FIGURE 2.1: Taxonomy of the automated essay scoring methods.

AES research investigates essay scoring models, for instance, related to the general quality or coherence. Although AES research predicts numerical (holistic) feedback, that is considered a drawback in the automated feedback setting because novice learners need more structure to learn quick and to avoid getting stuck. Intuitively, a score is less structured and informative than localized and formative feedback. On the other hand, AES research is relevant because it requires text quality understanding to score a text - and all automated feedback systems share the challenge to evaluate text. To distinguish between AES methods, we separate supervised methods (regression, preference ranking, neural nets) from unsupervised (clustering) methods. Figure 2.1 shows the AES methods' taxonomy that are discussed in this section and that are grouped by similar methodology. In the following paragraphs, we analyse several AES methods and we explain the AES methods in abstracted terms.

Supervised methods guide the training process with annotated essay scores, and a part of the methods depend upon feature engineering. In Table 2.1 we show examples of AES features that are grouped into task-independent and task-dependent features, where task-dependent features depend upon the assignment. In literature, regression models use AES features to predict essay scores, for example using logistic or Bayesian ridge regression (Farra, Somasundaran, and Burstein, 2015; Phandi, Chai, and Ng, 2015). One of the main advantages is that regression is explainable in terms of the most important features that determine the predicted essay score.

Other supervised methods rely upon preference ranking (Yannakoudakis, Briscoe, and Medlock, 2011; Chen and He, 2013; Cummins, Zhang, and Briscoe, 2016). Preference ranking uses two data items to train the model, whereas regression uses one data item. Frequently, preference ranking uses two data items to calculate difference vectors, where each difference vector represents the difference between two texts' feature vectors. The goal is to maximize the number of correctly ordered pairs of the difference vectors, and therefore, to model the quality relationships between essays.

In general, feature engineering requires experts with experience and knowledge to construct informative features, while neural networks can take the text as input directly removing the need for feature engineering. Common architectures are a convolutional neural network (Dong and Zhang, 2016), a combination of convolutional neural network and recurrent neural networks (Dong, Zhang, and Yang, 2017), a deep neural network (Jin et al., 2018), and a transformer (Ormerod, Malhotra, and Jafari, 2021). Although these methods achieve high accuracy, the methods have a downside: the model's decision process becomes hard to explain and requires a significant amount of data to train.

Task-independent	Task-dependent
Variation of text structure	Relevance of the essay's text
Presence of formal words	Similarity to source(s) or plagiarism
Transitional moves or rhetorical moves	
Grammar and spelling mistakes	
Essay length, if there is a time limit	

TABLE 2.1: Examples of task-dependent and task-independent automated essay scoring feature descriptions Farra, Somasundaran, and Burstein, 2015; Zesch, Wojatzki, and Scholten-Akoun, 2015

In contrast to supervised methods, unsupervised methods rely on classification models and require less or no data annotation. Rudner and Liang, 2002 use a Bayesian model to classify with two scoring classes. On the contrary, Chen et al., 2010 uses six scoring classes and a voting algorithm combined with k-means clustering to classify essays. The intuition behind that approach is that voting enforces similar essays to be clustered together, where the number of shared terms is used as similarity metric. The final score is assigned to all essays per cluster using historical data about scoring distributions. Similarly, McNamara et al., 2015 uses hierarchical clustering to cluster assignments to a 6-score scale based on a wide range of textual features.

2.2 Automated Writing Evaluation

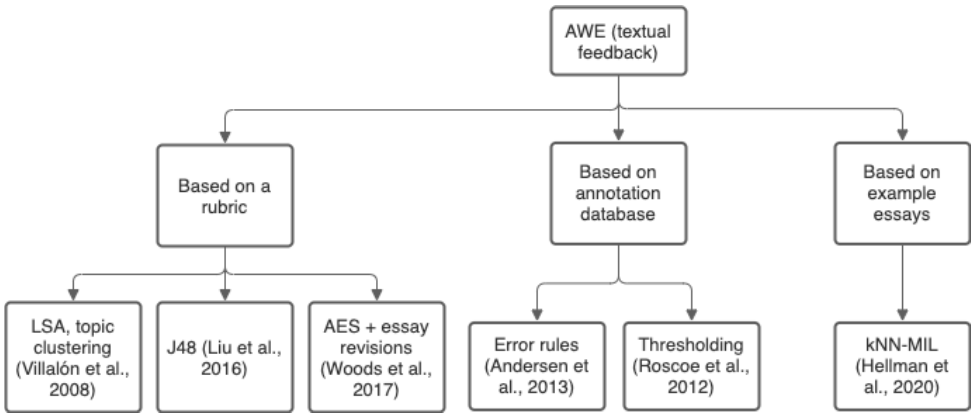

AWE (textual feedback)
Based on a rubric
Based on annotation database
Based on example essays
LSA, topic clustering (Villalón et al., 2008)
J48 (Liu et al., 2016)
AES + essay revisions (Woods et al., 2017)
Error rules (Andersen et al., 2013)
Thresholding (Roscoe et al., 2012)
kNN-MIL (Hellman et al., 2020)

FIGURE 2.2: Taxonomy of the automated writing evaluation methods.

Although AES effectively provides numerical feedback, textual feedback adds more context and detailed feedback on the student's writing skills. For instance, a score of 4 out of 6 for *structure* is less informative than feedback that informs the student that the introduction is missing. Formative feedback is a more complex problem to solve because the search space increases significantly. In AES literature it is common to predict a rating on a 6-point scale, and therefore, AES has a smaller search space than predicting a single word of formative feedback. The system needs to understand the work itself to provide valuable formative feedback, for instance, related to the text's readability or content. Automated Writing Evaluation (AWE) tries to formulate formative feedback and is commonly used as a backbone to construct formative feedback. Figure 2.2 shows a taxonomy of the AES methods that are analysed and explained in this section.

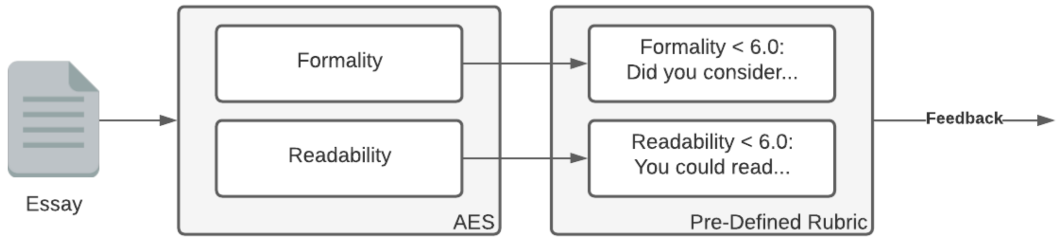

Essay
Formality
Readability
AES
Formality < 6.0: Did you consider...
Readability < 6.0: You could read...
Pre-Defined Rubric
Feedback

FIGURE 2.3: Visualization of how automated essay scoring can be used to fetch relevant and formative feedback.

A common way to transform AES models' essay scores to formative feedback is to score models in several assessment categories (for instance, general or readability scores), and subsequently, to use the predicted scores to fetch relevant feedback from a predefined feedback rubric (Villalón et al., 2008; Liu et al., 2016; Woods et al., 2017). Figure 2.3 illustrates the process to transform AES models' predicted scores into formative feedback. Although it is intuitive to combine AES with a predefined feedback rubric to extract relevant feedback, the feedback variety is small. An annotation database prevents this problem by allowing for more annotations in a large database (Roscoe et al., 2012). Unfortunately, a large expert-level feedback database is time-consuming and expensive to construct and leads to generalizability challenges to propose feedback for unexplored assignments. Apart from fetching predefined feedback based upon scores, another option is to detect common error patterns based on a large corpus of text and to use that to fetch appropriate annotations from the database (Andersen et al., 2013). Moreover, AWE can consider example essays as feedback sources. For these methods, the essay dataset requires manual feedback annotations to gather similar feedback for semanticly similar essays. Hellman et al., 2020 use essays as feedback source in a system that clusters essay sentences and that subsequently applies available feedback to sentences of a new essay.

AES and AWE research is the backbone for deployed automated feedback systems. In the next section, we discuss several deployed automated feedback systems.

2.3 Feedback Systems

This section discusses how online educational feedback systems address automating writing feedback, and we elaborate upon the advantages and disadvantages. Table 2.2 shows an overview of the covered systems and the system's related educational properties. As Dikli, 2016 mentions, there have been substantial outcomes from the use of educational technology, and the educational tools' designs influence the initial requirement for this thesis project.

	Technique	Main Focus	Dataset Restrictions
AcaWriter	Rules	Rhetorical moves	predefined rules
Criterion	Rules & ML	Grammar, word usage, mechanics	Expert annotations
Revision Assistant	Revision-based	Rubric specific feedback	Expert feedback & annotations
CoFee	Text Similarity	General feedback	Detailed expert feedback
Grammarly	?	Spelling, grammar, tone, fluency	?

TABLE 2.2: Overview of feedback system properties. Rules refers to predefined rules and ML refers to machine learning methods in the technique column.

To start, AcaWriter[1] is an open-source tool that targets the proper usage of rhetorical moves in text. The following quote from Knight et al., 2020 explains their perception of effective writing: "Effective writing incorporates instantiations of particular text structures – rhetorical moves – that communicate intent to the reader." To summarize, AcaWriter uses predefined rules to analyse the correct usage of rhetorical moves. AcaWriter analyses text with syntactic parsing and dependency trees to assign rhetorical move labels to sentences (such as context, challenge, or own opinion), based on a predefined lexicon - a list of expressions that may instantiate a rhetorical move. The usage of predefined rules has advantages and disadvantages. An advantage is that the rules are transparent and interpretable, while on the other hand, it requires significant manual effort to find and define effective rules. A text has many forms, and defining rules that capture all variants is challenging. One of the main contributions of AcaWriter is the thorough evaluation in classroom environments, including some evidence of impact that students who use AcaWriter are more likely to improve their text (Knight et al., 2020). The experiments are performed in several domains, for instance, in law and accounting.

The following system we discuss is Criterion, an educational system that assesses the essay's text quality. Criterion predicts scores using 12 features, for instance, the number of grammar errors and the average length of words. In addition, Criterion

[1] acawriter.uts.edu.au

detects errors concerning grammar, word usage, and mechanics. Most of the error detection is done similarly to AcaWriter with predefined rules and text analysis. Contrary to AcaWriter, Criterion uses machine learning to detect and highlight text repetition (Burstein, 2004) using a dataset that contains 300 manually annotated essays. The annotation process involves two judges to label words that interfere with smooth reading. A decision-based algorithm predicts the repetitive words using the manually annotated corpus. The drawback of this approach is the strict dataset constraints because when 300 annotated essays are needed for each domain, it restricts the system's scalability. A second drawback is that most of the corrective feedback concerns words, disregarding an overall assessment of individual sentences. Students learn more from understanding writing mistakes in a context than to correct word-level mistakes directly.

Woods et al., 2017 scores essays similarly to Criterion - although different features and models are used to predict the scores - and extends the system with sentence-level scores and feedback using a revision-based system called Revision Assistant. One of the challenges is to convert essay scores into sentence-level scores to achieve localized sentence-level feedback. The obvious solution is to use sentence-level score annotations and to train a simple scoring model to predict sentence-level scores, but unfortunately, sentence-level annotations are time-consuming and expensive (Hellman et al., 2020). Revision Assistant scores the essay, removes a single sentence, scores the essay again, and calculates the score difference to transform the essay-level scores into sentence-level scores. This technique allows the model to calculate the impact of a single sentence that conveniently does not require additional sentence-level score annotations. The last step is to link feedback to the sentence-level scores based on a predefined rubric containing feedback. The revision-based method effectively predicts localized sentence scores because it prevents the requirement of sentence-level annotations. Last, Revision Assistant uses text quality scores to retrieve relevant predefined feedback texts. Although this method effectively controls the system's feedback output (to prevent feedback that does not make sense), it takes significant manual effort to provide a wide range of educational feedback.

In contrast to Revision Assistant's limited predefined feedback, the Intelligent Essay Assistor (IEA) uses an approach based on essay similarity (Foltz, Laham, and Landauer, 1999) to incorporate a wide range of possible feedback. Foltz, Laham, and Landauer, 1999 assign the essay's semantic similarity to the cosine similarity between the essays' latent semantic analysis text embeddings (Landauer, Foltz, and Laham, 1998). IEA uses a dataset of essays with overall grades. Given a new essay X, the weighted average of the scores of the 10 most similar essays is used to predict an overall score, where the weights are the semantic similarities between essay X and the essay from the dataset.

Most feedback systems' challenge is to determine text quality and useful feedback, while the text similarity approach's challenge is to determine similar mistakes properly. New essays are matched with the most similar essay in the essay dataset for a specific domain. IEA provides numerical feedback using the most similar essay's feedback. Although this approach is useful for assignments that result in similar essays, the main challenge is that the predicted feedback becomes inaccurate when the essays' content vary too much.

Contrary to IEA's numerical feedback, Bernius, Krusche, and Bruegge, 2021 propose CoFee that uses a similar approach to construct textual feedback. CoFee uses more advanced text embeddings than IEA (ELMo, Peters et al., 2018), while at the same time, the similarity computation remains the same. The text pre-processing step of CoFee includes a simple keyword matching approach to identify 10 topics.

First, stopwords are removed and words are lemmatized, then the topics are identified. The second pre-processing step is to break the text into clauses and to merge all clauses that share the same topic into segments. Clauses with an unidentified topic start a new segment. After pre-processing, ELMo transforms the text segments into text embeddings. While IEA computes the cosine similarities directly, CoFee clusters the embeddings with the HDBSSCAN clustering algorithm using the cosine similarity as distance metric (McInnes, Healy, and Astels, 2017). An important note is that CoFee uses a pre-trained ELMo model trained on complete sentences of Wikipedia and news articles, while CoFee uses that ELMo model with pre-processed inputs, resulting in a mismatch with the training input. This can cause a decrease in performance of embedding prediction. For each new text, CoFee searches for the closest cluster, and suggests the feedback that is linked to the cluster. Specifically, feedback that was given on other embeddings in the cluster is suggested as feedback for the new text. Intuitively, when only the most similar text is used, the items in the clusters do not provide additional benefits.

CoFee's data collection setup is different from other feedback systems because it collects the dataset during a course. For each student answer, either a teacher provides feedback or CoFee suggests feedback for the teacher to approve. Feedback that is given to other items in the cluster is suggested as feedback. One of the advantages of this approach is that it does not rely on a strict dataset, and therefore, the system is easily deployable, scalable and usable in any domain, given the required assignment setup. At the same time, starting with an empty dataset is a disadvantage since it has a cold start problem (when no answers are processed yet) and it suggests relatively few feedback (26% on average during the experiment of Bernius, Krusche, and Bruegge, 2021). Although collecting data during a course takes time, the resulting feedback that CoFee suggests is expert feedback about the text's content and is not limited to a predefined feedback rubric. Furthermore, CoFee suggests feedback based on the similarity of text answers, while the feedback text is disregarded. This is a problem because the feedback addresses a specific mistake. Therefore, CoFee suggests predefined feedback to new texts, while the model cannot determine whether the text contains the mistake that the feedback targets, because CoFee has no information about what mistake the feedback addresses. On the other hand, CoFee does know that the text is similar (in some way) to the original sentence that the feedback targets. As a result, CoFee potentially has issues dealing with sentences that are similar in content that contain different mistakes. The main benefit of CoFee's use case is the accuracy. Even if the feedback accuracy is not perfect (85% on average in Bernius, Krusche, and Bruegge, 2021), the teacher will disregard erroneous feedback since the system acts as a computer assistant for human teachers.

Finally, systems such as Grammarly and Microsoft Editor focus less on educating users and more on correcting mistakes directly. To the best of our knowledge, these systems do not share their algorithms and models. Grammarly[2] is a popular online writing tool that helps people write text. The tool focuses on corrective feedback by showing the suggested corrections immediately. In particular, Grammarly targets context-specific mistakes concerning grammar, spelling, wordiness, style, punctuation, and plagiarism. These categories address lower-order concerns of writing. Grammarly and Microsoft Editor do not focus on educating people how to write but instead on improving texts directly to help people communicate more effectively. Grammarly and Microsoft Editor are effective in helping people to quickly update their texts with an intuitive user interface. On top of that, corrective feedback can be

[2] www.grammarly.com

educational in the correct setting (Guénette, 2007), and Ghufron and Rosyida, 2018 show that Grammarly contributes positively in reducing writing errors while stimulating self-regulated learning. However, contrary to Ghufron and Rosyida, 2018, a meta-analysis of writing interventions (Graham and Perin, 2007) shows that grammar and spelling are least effective to improve the essay's quality.

Chapter 3

Requirement Analysis

This chapter covers the requirements. As discussed in the related work section, providing reliable, high-quality, educational and formative feedback is challenging. To address that challenge, a range of feedback systems focuses on essay-level numerical feedback that can be paired with pre-defined feedback rubrics to provide students with formative feedback. Although essay scoring models combined with pre-defined methods is an interesting method to construct formative feedback, the main drawback is that experts need to define feedback for each assignment.

This thesis goes one step further and focuses on educational, formative, and localized feedback for essays to encourage students' long-term learning effects, and at the same time, to minimalize the need for manual annotations. Following other research we consider sentence-level feedback as localized feedback (Higgins et al., 2004; Andersen et al., 2013; Woods et al., 2017; Hellman et al., 2020). The main requirement is to develop a reliable educational system that educates students to write high-quality essays, and more importantly, to assist students to learn from common writing mistakes. It is important to note that the feedback system's intent is not to be used in a stand-alone fashion but instead to be used alongside existing educational practices. Experts or students can focus on more advanced feedback that requires a deeper understanding of the text, while automated feedback focuses on common writing mistakes. In doing so, the feedback system supports teachers (and students) with the time-consuming feedback task.

The main goal is to assist students how to write better texts; and therefore, we start with defining effective learning (Anders Ericsson, 2008):

1. Provide a clear task with clear goals
2. Motivate improvement
3. Provide feedback
4. Provide iterative and repeated opportunities for gradual refinement and improvement

These four conditions fall under Deliberate Practice that is identified by Anders Ericsson, 2008 to achieve high levels of learning performance.

A closely related education practice is self-regulated learning that requires a student to have learning goals and motivation, to have the ability to monitor their performance, and to be in control (Pintrich, 1995). In the context of assisting students to learn how to write, it is essential to encourage self-regulated learning because self-regulated learning prepares students for learning outside the university in formal and informal settings (Ibarra-Sáiz, Rodríguez-Gómez, and Boud, 2020).

Another key factor of automated feedback systems is differentiated learning to tailor educational methods to the needs of differently skilled learners. Differentiation in learning is important because novice learners require more guidance than advanced learners (Kirschner and Hendrick, 2020). Kirschner and Hendrick, 2020 suggest to utilize differentiated learning at an early stage in the learning process. Although differentiated learning is time-consuming and infeasible in large classroom settings (to identify each learner's skills and to tailor the educational methods for their needs), an automated feedback system can show tailored feedback visualizations. The following sections refine the requirements based on literature, discussions with experts, and feedback dataset exploration.

3.1 Discussions with Experts

To refine the initial requirements further, we interview several experts. Each discussion focuses on a different part of automated feedback systems: data, models, academic writing mistake categories, and user interface. The topics to discuss are based on the related work research about deployed automated feedback systems.

3.1.1 FeedbackFruits

The first discussion is with FeedbackFruits[1]; a company that provides online educational tools to educational institutions, and we collaborate throughout this thesis project. Similar to this thesis, FeedbackFruits researches feedback automation for students. As an online educational company, FeedbackFruits has access to rich feedback datasets containing peer review feedback (of students) and expert feedback (of teachers).

FeedbackFruits has two main layers of feedback: (1) peer feedback by students and (2) feedback by teachers. To support students in their educational process, FeedbackFruits searches for a layer in between, consistent automated feedback with reasonable quality, preferably using the common peer review feedback data.

At the start of this project, FeedbackFruits depended mainly on rule-based systems to construct writing feedback, and as a consequence, FeedbackFruits' feedback datasets are not optimally used. FeedbackFruits' main requirement is to use the common peer review feedback data to construct high-quality, generalizable, explainable, and formative feedback for essay assignments to use the untouched peer review data as a knowledge source. Any attempt at feedback automation is considered a contribution - the solution is either functional and educational or the research discovers the drawbacks of potential feedback systems.

On top of that, the feedback system's generalizability - the ability to use the systems for other assignments - relies on each assignment's required manual annotations. It is essential to consider an automated feedback system's generalizability because a completely generalizable system requires no extra effort to deploy the system to unexplored assignments. For example, a system that requires hundreds of expert annotations per assignment is impractical to generalize to other assignments. In the setting of FeedbackFruits, there are many assignments to consider.

System	Dataset Size
Revision Assistant	280 to 1000 essays
The Intelligent Essay Assessor	100 essays
Criterion	465 essays

TABLE 3.1: Dataset sizes for automated feedback systems

3.1.2 Professor with Education System Experience

The first expert has extensive experience in online learning and educational systems. For automating feedback on writing, the expert recognizes that text generation models are promising to generate feedback directly. According to the expert, the main issue of peer review data is the feedback quality. Although students can provide high-quality feedback, the dataset contains noise, and the feedback will not be expert-level quality. Expert feedback has a broader application and is more valuable in terms of expertise. Unfortunately, the obvious solution to use expert data presents new issues in data collection regarding scalability and generalizability of the feedback system because expert data for specific assignments is hard to construct. The expert estimates that a dataset of around 100 essays with annotated feedback is sufficient for a feedback system. Second, the expert indicates that including additional information (for instance, from DBPedia or Wikipedia) potentially supports models to construct better feedback.

From this conversation, we recognize that data exploration is essential, and to consider to generate feedback text directly. The automated feedback model needs a solid foundation in terms of data, and preferably, the automated feedback model needs to have a wide application. Therefore, we focus on easy-to-construct datasets to increase the feedback system's generalizability. Based on the expert's 100 essay estimation and the sizes of other feedback systems (Table 3.1), we add the requirement to target datasets of more than 100 essays. The following discussion concerns text generation and potential directions for automated feedback systems.

3.1.3 Professor with NLP Experience

The second expert has extensive NLP experience and knowledge about state-of-the-art models, such as recent transformer models. During this discussion, we discussed potential feedback systems in detail. Common and intuitive directions to automate feedback are semantic similarity search, rule-based feedback mapping, and essay scoring. Although all options are potentially viable, the expert mentioned that generative models (to generate text) are promising to generate educational feedback directly. In addition, the expert stresses the importance of the model's ability to reason about specific writing mistakes, and therefore, it is important to know what writing mistakes to address. When the system provides general feedback, it becomes challenging to evaluate and to compare the results with other systems (Woods et al., 2017, Bernius, Krusche, and Bruegge, 2021).

[1] feedbackfruits.com

3.1.4 Writing Expert

To have a more in-depth discussion about possible writing mistakes, we discuss common writing mistakes with the third expert, who is active in research on learning how to write high-quality academic text. The expert shares what problems frequently occur in essays - for example subject-verb agreement mistakes such as 'people is', incorrect usage of connective words, and missing links in argumentation. A surprising insight is that the expert indicates that researchers extensively tried to categorize academic writing feedback without success, and that we should not expect to properly categorize all possible writing mistakes in detail in a couple of months. On top of the discussion, the expert sent several documents with an example grading rubric, an academic writing checklist, and a scientific writing terms glossary.

Based on the discussion with the expert, we extend the requirements considering writing mistake categorization. As the expert mentions, categorizing academic writing mistakes is challenging, and therefore, we intend to focus on common writing mistakes. Moreover, based on the expert's example rubrics and writing checklists that all contain a writing style category, and because capable systems (Grammarly, Microsoft Editor) address lower-order concerns (spelling, grammar) and FeedbackFruits addresses higher-order concerns (essay structure), we conclude that writing style is a promising and achievable direction to take.

3.1.5 Educational Systems Expert

The last discussion is about possible effective user interface designs for educational systems. We recognize that an effective user interface is important for the requirements because that determines how learners interact with the automated feedback models. The expert has successfully developed and deployed a feedback system for a faculty at TU Delft. In contrast to this thesis project, the expert's feedback system mainly focuses on peer reviews and not on automated feedback. One of the focus points is to let students actively think about mistakes in assignments to stimulate self-regulated learning and to encourage long-term learning effects. The expert's system achieves that by including students in the decision-making process and providing iterative feedback on assignment revisions. The expert recognizes that automated formative feedback systems can be effective in education, although automated feedback models are hard to construct.

3.2 Feedback Data Exploration

Based on the conversations with experts, we recognize that data exploration (for at least 100 essays for a single assignment) is an essential step in the requirement analysis. To this end, this section discusses the data exploration to finalize the requirement analysis.

Although we partnered with FeedbackFruits for their expertise and feedback datasets, we considered other feedback sources. Following the requirement to consider feedback datasets containing more than 100 essays per assignment, we search for an essay dataset with feedback on sentence-level for a single assignment. First, the peer review website Peer[2] that is used throughout TU Delft for online peer reviews. Although Peer's database is considerable, the database is smaller than FeedbackFruits' database and the data is less structured than FeedbackFruits' dataset.

[2] https://peer.tudelft.nl/

The main difference is that all FeedbackFruits' peer reviews contain text highlights that specifically indicate writing mistakes in texts, which is intuitively useful in order to provide localized feedback. The second consideration is the fourth expert's feedback system, but unfortunately, the assignments contain images (for instance diagrams or flow charts) that the feedback addresses, while FeedbackFruits has essay assignments with feedback on writing. Last, we contacted professors at TU Delft, but these datasets do not exceed 100 essays for a single assignment. After considering the alternatives, we decide to focus on FeedbackFruits' peer review datasets.

Based on the peer feedback similarity clustering (for more details, we refer to Appendix A), we discover that the peer feedback data is challenging to clean. The main challenge is to filter on educational feedback, and at the same time, to make sure that enough feedback samples are left after cleaning. Although there are clearly common feedback topics, a reliable and generalizable educational feedback system that automatically detects topics and selects relevant (or generates) high-quality educational feedback is challenging to construct. Consequently, feedback on essay content becomes challenging to address considering the manual effort to find feedback categories for every assignment, which contradicts the earlier requirement to use the least amount of manual annotations to improve the system's generalizability to other assignments. For these reasons, we decide not to focus on using peer review feedback knowledge to construct content-related feedback, and instead, we decided to focus on automated feedback approaches that target the text's writing quality.

Another finding from the data exploration is that text embeddings and textual semantic similarity is a promising direction to take. As a result, we add the requirements to focus on pre-defined writing mistake categories and textual semantic similarity.

3.3 Final Requirements

Requirement	Significant Literature	FbF	E1	E2	E3	E4
1. Easy-to-collect dataset		X				
2. Consider text generation			X	X		
3. Multiple levels of feedback	Kirschner and Hendrick, 2020					
4. Deliberate practice	Anders Ericsson, 2008					X
5. Sentence-level feedback	Andersen et al., 2013					
6. Pre-defined assessment categories				X	X	
7. Easily extendible and generalizable		X	X			

TABLE 3.2: Overview of the most significant related work's and discussions' influence on the final requirements. FbF refers to FeedbackFruits and systems refers to automated feedback systems discussed in the related work chapter. Furthermore, 'E' stands for expert and an 'X' means that the requirement follows from literature or from a discussion.

Combining the data exploration conclusions with the expert discussions, we present the following list that contains the final requirements. In addition to the final requirements list, Table 3.2 shows a summary of the main consideration for each requirement. Although we included more arguments for each requirement throughout this chapter, Table 3.2 includes only the most significant arguments.

1. Use an easy-to-collect dataset that requires the least amount of additional manual annotations

2. Consider text generation (or text comprehension) models, textual semantic similarity, and automated essay scoring models
3. Provide different levels of feedback (highlight mistake, show tips, and suggest corrections), to encourage differentiated learning and to offer guidance for novice to advanced learners
4. An iterative feedback system that encourages students to think about writing style mistakes and to stimulate gradual refinements and improvements of their performance (self-regulated learning by deliberate practice)
5. Provide reliable, sentence-level, and textual feedback, for instance, on text coherence or text cohesion
6. Provide feedback for pre-defined writing assessment categories
7. Support extension (adding writing assessment categories)

In the next chapter, we address all requirements and propose a reliable, generalizable, educational, iterative, localized, and formative feedback system for specific writing style mistake categories that assists students in learning how to write high-quality essays.

Chapter 4

Conceptual Design

In this chapter, we propose *RevisionCoach*, an automated feedback system for writing style. Before *RevisionCoach*, we explored automated feedback systems that use feedback matching and feedback generation to construct formative feedback, and because *RevisionCoach*'s design is inspired by these other feedback systems, we discuss the other feedback systems first.

4.1 Explored Automated Feedback Systems

First, we discuss the feedback system that searches for the best matching feedback in a database and a feedback system that generates feedback directly. *RevisionCoach* follows the requirements of the previous chapter, but the previously explored automated feedback systems do not. However, the explored system's findings inspired the current requirement definitions. For the explored systems, we elaborate upon the advantages and disadvantages.

4.1.1 Feedback Linking

The first feedback system uses text similarity to match essay sentences to sentences in the peer review database. For the sentence's match in the database, we retrieve the attached feedback. Therefore, this system relies on an additional dataset requirement that essays have sentence-level mistake annotations with attached feedback. The data preprocessing process is described in Appendix A.

This similarity-based approach is used in research before (Louis and Higgins, 2010; Bernius and Bruegge, 2019; Bernius, Krusche, and Bruegge, 2021). While this method seems deceptively simple, it addresses important challenges regarding the feedback variety and quality; in contrast to other research where pre-defined feedback is used (for instance in Villalón et al., 2008; Andersen et al., 2013; Liu et al., 2016; Woods et al., 2017). Similarity-based feedback linking has the potential to provide specific content-related feedback, given that a large dataset of feedback samples is available. Sources for content-related feedback can be peer reviews (student-based) and expert feedback (expert-based). Unfortunately, finding an expert-level feedback dataset is challenging, and each assignment type requires a pre-defined highlight-feedback dataset. Apart from the ability to provide content-related feedback, a similarity-based approach has other advantages. A system that works with peer reviews or expert feedback is practical in courses with previous editions. Given that the course's assignments are similar, previous year data can be used to suggest feedback for the current year, making it a practical solution to reduce the time-consumption of teachers for giving feedback. A second benefit is the explainability potential because the ability to explain the predicted feedback helps to understand the functionality and reliability of the feedback system.

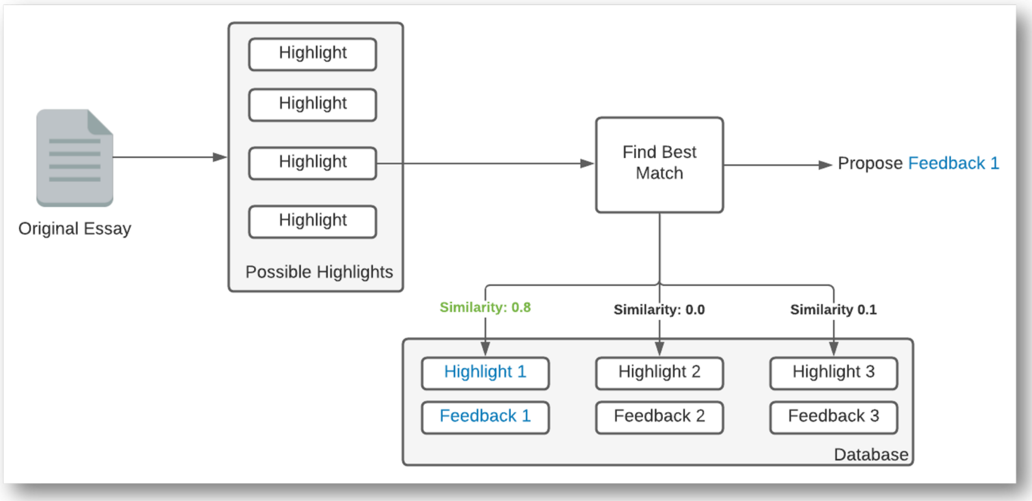


FIGURE 4.1: Model structure

Figure 4.1 visualizes the similarity-based feedback system. As depicted in the figure, the system consists of three main parts. The first part constructs possible highlights from an input essay using a moving window (a window with a fixed length that moves one word at a time). After that, the second part finds the best matching highlight in the dataset for each possible highlight. The best matching highlight's feedback is used as suggested feedback for each constructed highlight in the system's last part. This approach uses Augmented SentenceBERT (Thakur et al., 2020) to calculate sentence embeddings, that we compare using the cosine distance to measure the semantic similarity between sentences. In order to do this efficiently, we precompute the peer review dataset's sentence highlight embeddings and define this as the embedding corpus. While the embedding corpus is calculated once, we compute the embeddings for highlights in new essays (highlight embeddings) each time. Finally, for each highlight embedding, we compute the semantic similarity with all corpus embeddings.

The system performed worse than expected during manual evaluation, which is hypothetically caused by the fact that the system does not consider the feedback's information because it only focuses on matching essay highlights. The second issue is the data noise - students often take shortcuts and provide low-quality data. Determining which feedback is educational is a challenging problem to solve. Although the preprocessing stage cleans the data, the feedback is not perfect, and we consider to use expert feedback to solve that issue. To address the issues, we redesigned the feedback linking model to include the feedback's information. *RevisionCoach* addresses the missing feedback information issue by considering pre-defined assessment categories that inform the feedback models about what writing problems to look for. Secondly, *RevisionCoach* removes the need for a feedback dataset that removes the noisy feedback data concerns. In the following section, we discuss a feedback model that generates feedback directly, and in contrast to feedback linking, the generation model takes the feedback's information into account.

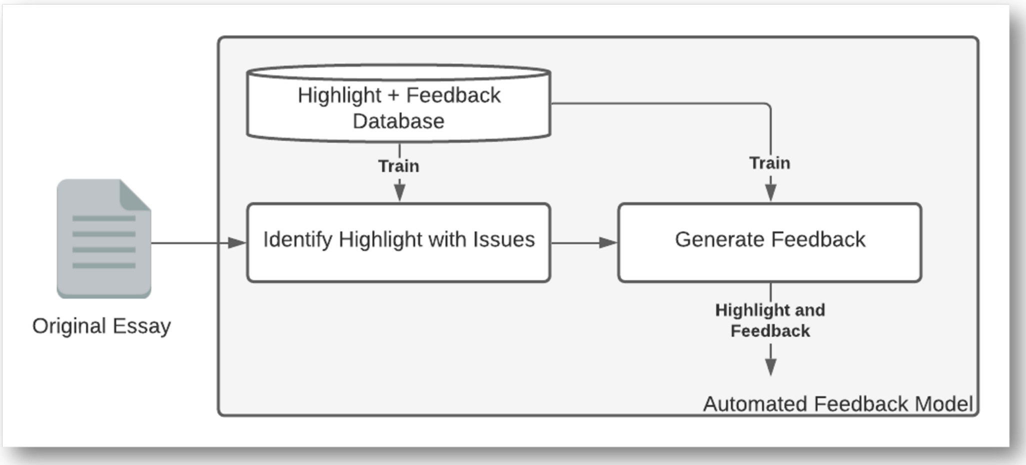


FIGURE 4.2: Model structure

4.1.2 Feedback Generation

The second system generates feedback directly, which addresses the feedback linking model's issue of disregarding the feedback information. The feedback generation model's design includes two main modules (Figure 4.2), both include a transformer architecture and are implemented with HuggingFace[1]. Although feedback linking considers all essay sentences as potential feedback highlights, this option is inconvenient for feedback generation considering the text generation time. Therefore, the first module predicts potential sentence-level highlights in the input essay to use as input for the feedback generation module.

To predict potential sentence-level highlights, we use a T5 transformer model in a text classification setting. In the fine-tuning process, the training data considers essay sentences as input and a boolean that is true when the sentence has feedback attached as label. In practice, the training dataset contains sentences that received feedback and sentences that did not receive feedback. In the end, the model takes a highlight as input and predicts if the highlight needs feedback. Besides the prediction, the model offers a prediction confidence score between zero and one.

The second part of the system concerns feedback generation using a second T5 transformer model. This time, the inputs are highlights, and the labels are corresponding feedback texts. In the following paragraphs we describe the data preprocessing process.

Data Preprocessing

In addition to the preprocessing stage described in Appendix A, this iteration extracts additional information (*extra context*) to provide the model with more information, as shown in Figure 4.3. First, we use YAKE (Campos et al., 2020) to extract important keyphrases from the highlight to perform a Wikipedia search for additional information. If the Wikipedia[2] search has no results, we extract named entities from the keyphrases with Spacy[3], and filter on the location, event, work of art, product, and organization entities. Furthermore, if none of the extracted named entities

[1]`huggingface.co`
[2]Python package: `https://pypi.org/project/wikipedia/`
[3]Python package: `https://spacy.io/usage/linguistic-features#named-entities`

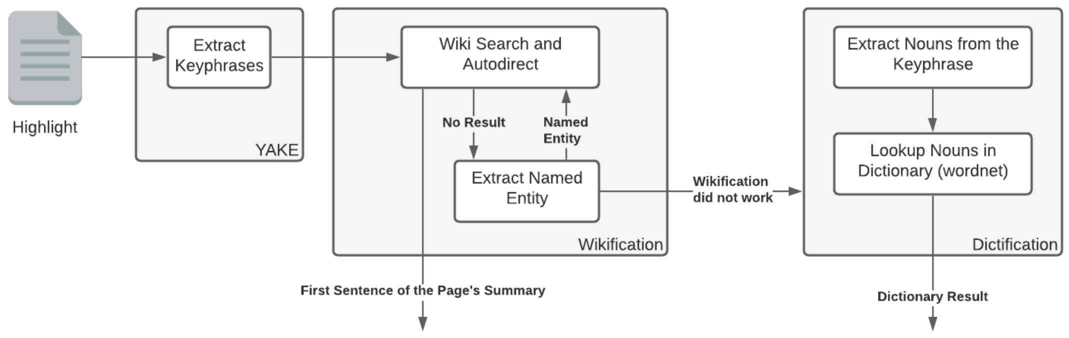

Highlight
Extract Keyphrases
YAKE
Wiki Search and Autodirect
No Result
Named Entity
Extract Named Entity
Wikification
First Sentence of the Page's Summary
Wikification did not work
Extract Nouns from the Keyphrase
Lookup Nouns in Dictionary (wordnet)
Dictification
Dictionary Result

FIGURE 4.3: Flow diagram that describes extra context extraction.

have a corresponding Wikipedia article - with at least 85% similarity to the article's title - we extract nouns from the keyphrase and perform a dictionary lookup using WordNet[4].

Secondly, we augment the original feedback dataset to construct a larger dataset containing more variation in highlight text. All augmentation methods are similar to the revision models of *RevisionCoach* (we refer to subsection 4.2.1 for more details).

On top of extracting extra context and data augmentation, we experimented with adding custom tokens to reduce the transformer's encoder vocabulary. This process maps numbers to specific tokens, such as ([NUM], [YEAR], or [CURRENCY]), which does not result in an information loss but does reduce the vocabulary size. Since the resulting loss did not decrease significantly compared to a model without this preprocessing step, we decided not to include this step. It complicates the system's design because the custom tokens are generated in the feedback, which need to be mapped to their original form.

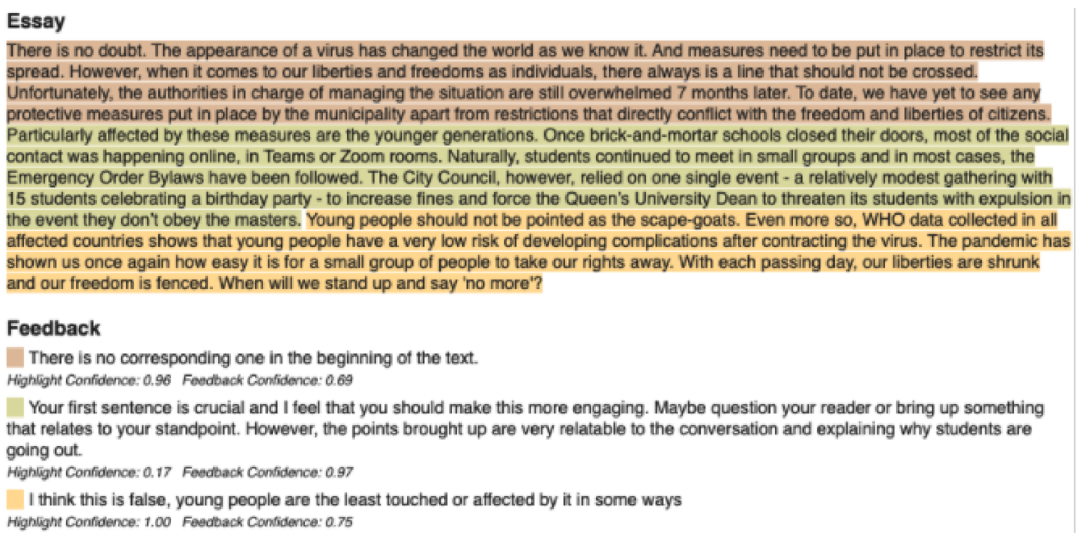
Feedback Generation

After the input is preprocessed and the models are trained (as depicted in Figure 4.2), the model is capable of generating feedback for an input essay. There are several options to generate text from the transformer model's output. In itself, the output consists of probabilities for each token in the vocabulary for each token in the generated output, meaning that an output sentence of 10 tokens with a vocabulary of 10.000 has a shape of [10, 10.000]. There are several options to decode the output probabilities, including Greedy Search, Beam Search, Top-K sampling, and Top-p sampling (supported by the transformer library called HuggingFace[5]). Each option tries to generate the most natural-looking text and tries to prevent text repetition. Although some options can generate multiple variations of the text, the feedback system requires one output text only. We use beam search with five beams and greedy decoding, one return sequence, and early stopping to produce a single feedback text. We hypothesize that the final feedback confidence is more reliable with greedy decoding, where the highest probability is preferred. By considering more beams - or paths in a search tree - we encourage the model to search beyond the next best token.

The output is not only feedback text but also the model's highlight and feedback confidence. To calculate the confidences, we take the average of the individual token probabilities that the transformer model outputs.

[4]https://www.nltk.org/howto/wordnet.html
[5]https://huggingface.co/blog/how-to-generate

Feedback Generation Evaluation



Essay

There is no doubt. The appearance of a virus has changed the world as we know it. And measures need to be put in place to restrict its spread. However, when it comes to our liberties and freedoms as individuals, there always is a line that should not be crossed. Unfortunately, the authorities in charge of managing the situation are still overwhelmed 7 months later. To date, we have yet to see any protective measures put in place by the municipality apart from restrictions that directly conflict with the freedom and liberties of citizens. Particularly affected by these measures are the younger generations. Once brick-and-mortar schools closed their doors, most of the social contact was happening online, in Teams or Zoom rooms. Naturally, students continued to meet in small groups and in most cases, the Emergency Order Bylaws have been followed. The City Council, however, relied on one single event - a relatively modest gathering with 15 students celebrating a birthday party - to increase fines and force the Queen's University Dean to threaten its students with expulsion in the event they don't obey the masters. Young people should not be pointed as the scape-goats. Even more so, WHO data collected in all affected countries shows that young people have a very low risk of developing complications after contracting the virus. The pandemic has shown us once again how easy it is for a small group of people to take our rights away. With each passing day, our liberties are shrunk and our freedom is fenced. When will we stand up and say 'no more'?

Feedback

🟫 There is no corresponding one in the beginning of the text.
Highlight Confidence: 0.96 Feedback Confidence: 0.69

🟩 Your first sentence is crucial and I feel that you should make this more engaging. Maybe question your reader or bring up something that relates to your standpoint. However, the points brought up are very relatable to the conversation and explaining why students are going out.
Highlight Confidence: 0.17 Feedback Confidence: 0.97

🟨 I think this is false, young people are the least touched or affected by it in some ways
Highlight Confidence: 1.00 Feedback Confidence: 0.75

FIGURE 4.4: Example of generated feedback during the evaluation session.

In the feedback generation study, we asked ten PhD students and professors from the Centre of Education and Learning (CEL)[6] and TU Delft to write an essay and to evaluate the feedback. Because the feedback generation model requires sentence-level mistake highlights and textual feedback, we use a peer review database from FeedbackFruits, and consequently, the study participants write an essay for the peer-reviewed assignment about a Covid-19 news article. Finally, the study participants rate the generated feedback in terms of naturalness, relevance, and informativeness.

In Figure 4.4 we show an example of feedback that was generated during the evaluation session, where the highlight confidence shows the model's confidence that the highlight contains a mistake, and the feedback confidence shows the model's confidence of the generated feedback. Unfortunately, the evaluation shows that the model's feedback accuracy is low because most feedback is not educational. Although some feedback is informative (such as the yellow feedback item in Figure 4.4), other feedback is irrelevant and distracting (such as the green feedback item in Figure 4.4). The other opinion texts show similar results, where more distracting than informative feedback is generated.

Text generation is challenging, and feedback generation is arguably more challenging, because of the educational aspect of feedback. In feedback generation it is not sufficient to have natural text because of the educational aspect to assist students in writing better essays. Moreover, it is challenging to assess the feedback generation model's feedback accuracy automatically. For example, the model can generate random feedback with high confidence for a random input that does not require feedback, although it has little educational benefit. In practice, it is challenging to filter out non-educational feedback.

Another issue with the feedback generation model's direction is the explainability and evaluation difficulty. It is challenging to reason about the model's feedback generation decision process. Although it is possible to analyse the model's attention scores and to discover the most important highlight words that the model considers, it is challenging to explain and it is challenging to evaluate the model's accuracy and effectiveness. Furthermore, we recognize that using more feedback data will

[6]https://www.educationandlearning.nl/home

eventually lead to an accurate feedback system, while it introduces a more significant challenge to evaluate because the number of feedback possibilities increases. In general, open-ended text generation systems are challenging to evaluate.

To address the challenge to evaluate open-ended feedback systems, *RevisionCoach* considers a limited number of feedback categories to construct categorized feedback. In chapter 10, we discuss a hybrid approach that combines *RevisionCoach* and the feedback generation model to generate categorized feedback text. In the following section, we discuss *RevisionCoach*'s design, which reuses the discussed feedback system models' components.

4.2 RevisionCoach

RevisionCoach is built following the requirements and the discovered disadvantages of the previously discussed systems. Therefore, *RevisionCoach* has access to the feedback's information and does not predict open-ended feedback. Before we explore *RevisionCoach's* implementation details, we discuss the system's conceptual design that is based on the requirement analysis of the previous chapter.

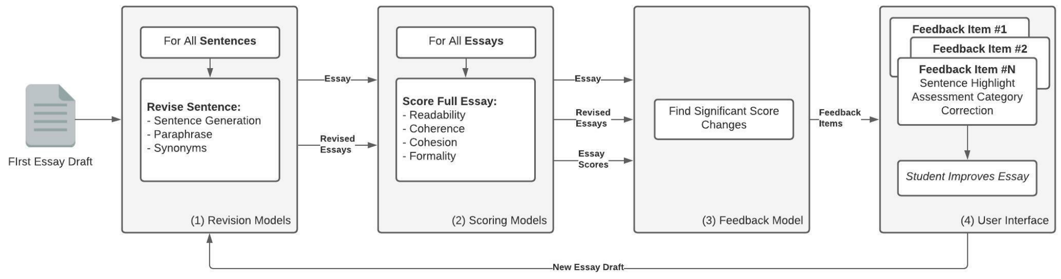


FIGURE 4.5: *RevisionCoach* system flow

The intuitive reasoning behind the design of *RevisionCoach* is based upon the writing process of an author. Writing is an iterative process, and at the end of each iteration the writer assesses the essay's quality. After the initial quality assessment, the writer revises the low-quality sentences and subsequently assesses the essay's quality again. This process is repeated until the essay's quality reaches the author's writing quality standards. Following the requirements, *RevisionCoach* needs only essays data without additional annotations. As a result, *RevisionCoach* can be used for any essay assignment as long as *RevisionCoach* has access to relevant texts (for instance, an essay dataset or related Wikipedia pages). To the best of our knowledge, other AES and AWE systems require manual annotation to automate feedback. Furthermore, *RevisionCoach* predicts structured feedback that is not open-ended to make the system easier to evaluate.

An overview of RevisionCoach is displayed in Figure 4.5 that follows the intuitive and iterative writing process: revise the essay, assess the essay's quality, present feedback, improve the essay, and repeat. Component (1) generates possible corrections for individual essay sentences. In each revision exactly one sentence is revised to support sentence-level feedback. Second (2 in Figure 4.5), the scoring component assesses the essay's and the revisions' quality. In the second component, the scoring models relate directly to *RevisionCoach*'s writing quality assessment categories. For example, the readability scoring model represents the essay's readability assessment category. All scores are used in (3) where *RevisionCoach* uses the scores to find significant changes in writing quality scores, and since all revisions contain exactly one revised sentence, the significant change is a result of a single sentence-level change. The significant changes represent the essay's sentence that can be improved with respect to a specific writing quality assessment category. Last (4), *RevisionCoach* visualizes four layers of feedback to support differentiated learning: a sentence highlight, a writing assessment category (cohesion, coherence, readability, formality), a revision category (rephrase, rewrite, or reword), and a possible correction. At the end of each cycle, the student uses the feedback to improve the essay on sentence-level.

RevisionCoach uses the iterative writing intuition to offer an iterative writing assistant that encourages students to think about common writing style mistakes and to learn by deliberate practice. Once the student considers the essay's quality to reach the requested quality standards, *RevisionCoach* supports the student to improve the essay's quality for pre-defined writing assessment categories. Students can focus on one category at a time and iteratively improve their essay's quality with sentence-level improvements. The impact of sentence-level improvements is directly visible alongside pre-defined tips and suggestions. This way, *RevisionCoach* motivates the learner to make improvements, allows the learner to have clear goals, and directs the learner's attention to sentences that can be improved to improve the overall essay quality.

Furthermore, *RevisionCoach* supports and enables self-regulated learning, since students can set and monitor their own goal (when are the quality scores sufficient), check their work, receive tips, and improve their work. The perfect answer is rarely suggested to students, pressuring them to come up with their own ideas. To support this process, *RevisionCoach* guides this improvement process and prevents students from getting stuck. An advantage of *RevisionCoach* is that it can provide instant feedback on request, and as research indicates, providing regular and frequent feedback enables effective self-regulation by students (Gibbs and Simpson, 2005). In addition, *RevisionCoach* can show the improvement percentage for students' sentence-level improvements to motivate students to improve their own work.

In addition to learning by deliberate practice and self-regulated learning, *RevisionCoach* supports differentiated feedback because it constructs for levels of feedback. For advanced learners it can be sufficient to present feedback that highlights low-quality sentences, while novice learners need more structure to prevent getting stuck. Students can immediately view sentence highlights and the corresponding writing assessment categories. In addition, students can request writing tips - predefined tips that are linked to the assessment categories. When students are stuck after the students received the sentence highlight, the writing tip, and the assessment category, *RevisionCoach* offers a possible sentence correction as inspiration for the student. Furthermore, the feedback levels can be combined to construct the following feedback template:

```
The [highlight]'s [assessment category] can be improved by [
    correction category] the sentence.

For example: The [first sentence's] [readability] can be
    improved by [rewording] the sentence.
```

In addition to the feedback template, *RevisionCoach*'s technique to find sentence-level mistakes can be used to find word-level mistakes to provide word-level hints for the student to improve the sentence. To not overcomplicate the design, and because of time considerations, we leave this for future work.

In the remainder of this chapter, we explain the four main components: the revision component, the scoring component, the feedback component, and the user interface component (Figure 4.5). Before *RevisionCoach* is able to coach students on writing style, *RevisionCoach* needs to learn how to assess writing quality (the scoring models) and it needs to learn how to write high-quality text (the revision models), while limiting the number of required manual annotations.

4.2.1 Revision Models

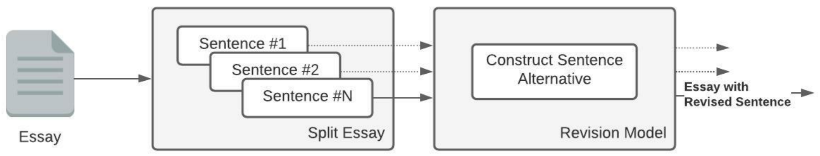


FIGURE 4.6: The flow of a revision model in *RevisionCoach*. N is the total number of sentences.

The purpose of the revision component is to generate a high-quality and context-dependent essay sentence that ideally improves the original sentence. To achieve that, *RevisionCoach* needs to learn how to write an appropriate essay sentence, using three revision strategies to revise individual sentences in an essay: rephrasing, rewriting, rewording. Figure 4.6 visualizes an overview of the revision model's design. To start, the essay is split into sentences. The revision model generates a revision for every sentence. Subsequently, the original sentence is replaced with the revised sentence to construct a revised essay. Since each essay revision contains exactly one revised sentence, *RevisionCoach* can provide sentence-level feedback because when the revised essay is compared to the original essay, the score change is caused by the single revised sentence. As a consequence, for every essay *RevisionCoach* generates $N * M$ revisions, where N is the number of sentences and M is the number of revision models.

Revising sentences is similar to data augmentation in natural language processing (NLP, a research field that focuses on natural language understanding, extraction, and analysis techniques), which expands the training data with additional text samples to make models more robust. Examples are to consider text samples that contain typing mistakes or alternative sentence representations in the training data. Researchers found that appropriate data augmentation techniques are helpful for reducing the generalization error for deep learning models (Zhang and LeCun, 2016). The challenge of data augmentation in NLP is that text is not continuous; and therefore, text augmentation can not be done by adding a number to an embedded - or tokenized - sentence. The semantics of a sentence matter, and the revision model's purpose is to revise sentences while keeping a valid English sentence. To this end, we use data augmentation techniques as a backbone to *RevisionCoach*'s revision models. In the following paragraphs we discuss the conceptual design of *RevisionCoach*'s revision models: rephrase, rewrite, and reword.

Rephrase Revision

The first revision allows *RevisionCoach* to construct a revision by restructuring and rewording sentences while keeping the original sentence's semantic meaning. To achieve this, we rely on paraphrasing methods that are a common technique for NLP data augmentation (Anaby-Tavor et al., 2019; Krishna, Wieting, and Iyyer, 2020; Gao et al., 2020).

Rewrite Revision

The second revision model's purpose is to write a sentence that is different in meaning compared to the original sentence. Text generation is the most obvious option to generate natural text, a solution that is used in question answering systems (Yu et al., 2020) and summarization systems (Verma and Om, 2019). A similar task is story generation, where a language model needs to complete a story based on some context (Xu et al., 2020), that shows promising results. Furthermore, text generation techniques are effective in text translation and reading comprehension (Raffel et al., 2019; Radford et al., 2020).

The intuition behind this approach is that for sentence X_i, *RevisionCoach* uses the sentences X_{i-1} and X_{i+1} as context in order to generate a context-dependent revision. In practice, the model learns from similar essays to construct natural text. Although the paraphrase's semantics are similar to the original sentence, the generated paraphrase's semantic meaning can be entirely different - as long as the sentence fits in the original sentence's context.

Reword Revision

The last revision techniques uses the sentence's part-of-speech (POS) elements to replace specific words in the sentence with the most similar synonym. To replace words with their related synonyms is a common data augmentation technique (Zhang and LeCun, 2016). Using synonyms to change a sentence is a form of paraphrasing; however, we separate the two methods because synonyms are replaced at word-level while *RevisionCoach*'s paraphrases are done at sentence-level. Therefore, we expect that the two revision methods result in different sentence revisions.

4.2.2 Scoring Models

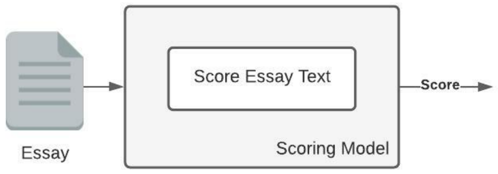


FIGURE 4.7: The flow of a scoring model in *RevisionCoach*.

RevisionCoach's second component scores the original essay and the revisions to assess the essay's writing quality (Figure 4.7), and because *RevisionCoach* considers multiple scoring methods, each scoring method represents a pre-defined assessment category. To this end, we use existing writing quality categories as scoring methods with small alterations. Based on the related work (chapter 2), the requirements, and the discussions with experts, *RevisionCoach* uses readability, coherence, cohesion, and formality as writing style assessment categories. Other writing style categories can be considered to extend *RevisionCoach*'s ability to assess text. First, we provide the reasoning for the chosen pre-defined writing quality assessment categories, and second, we explain the general approach to compute the writing quality scores for each assessment category.

Readability

The first assessment category of *RevisionCoach* relates to the readability, a common feature for AES systems (for instance by Graesser et al., 2004; Dikli, 2016; Zesch, Wojatzki, and Scholten-Akoun, 2015; McNamara et al., 2015; Ke and Ng, 2019; Uto, Xie, and Ueno, 2020). Readability refers to the reading difficulty of a text and depends on the word usage (Ke and Ng, 2019). In addition to AES systems that consider readability as an important feature, the writing expert in section 3.1 provided example rubrics for assessing text quality that indicate the importance of text readability (with a grading weight of 15%).

Coherence

Readable text that does not make sense results in poor text quality, and for that reason, we consider text coherence as the second assessment category. A coherent text is logical and easier to understand (McNamara and Kintsch, 1996). Witte and Faigley, 1981 mentions the condition that a coherent text should provide only information relevant to the text's topic and a coherent text should be understandable in a real-world setting. Furthermore, coherence relates to the logical structure and to the reasoning of text, and because of that, coherence is considered an important feature for AES models (Zesch, Wojatzki, and Scholten-Akoun, 2015; Li et al., 2018). In addition, the writing expert in section 3.1 noted that the flow and reasoning in text commonly goes wrong; and therefore, coherence is an important indicator of text quality. For these reasons, we decide to include text coherence as writing assessment criterion.

Cohesion

In contrast to coherence that requires the text to be understandable in a real-world setting, cohesion is about the mechanisms used to connect sentences in the text. A cohesive text is not always coherent, and the following example illustrates the difference:

1. The player kicked the ball out of the stadium.
2. However, the tree is green.
3. Company X uses primarily green containers.

While the sentences are cohesive, it is difficult to construct a real-world scene for the sentences. The sentences have no clear purpose, and it is difficult to find a common topic that the sentences share. Although cohesive text does not imply coherent text, cohesion is still an important quality indicator of text (Witte and Faigley, 1981). In fact, cohesive tie analysis (the way in which pieces of text are linked together) is the backbone of AcaWriter (Knight et al., 2020). Knight et al., 2020 that shows how a system, that mostly focuses on cohesion to express writing quality, can be effective in guiding students to write better texts. Therefore, *RevisionCoach* includes cohesion as assessment category.

Formality

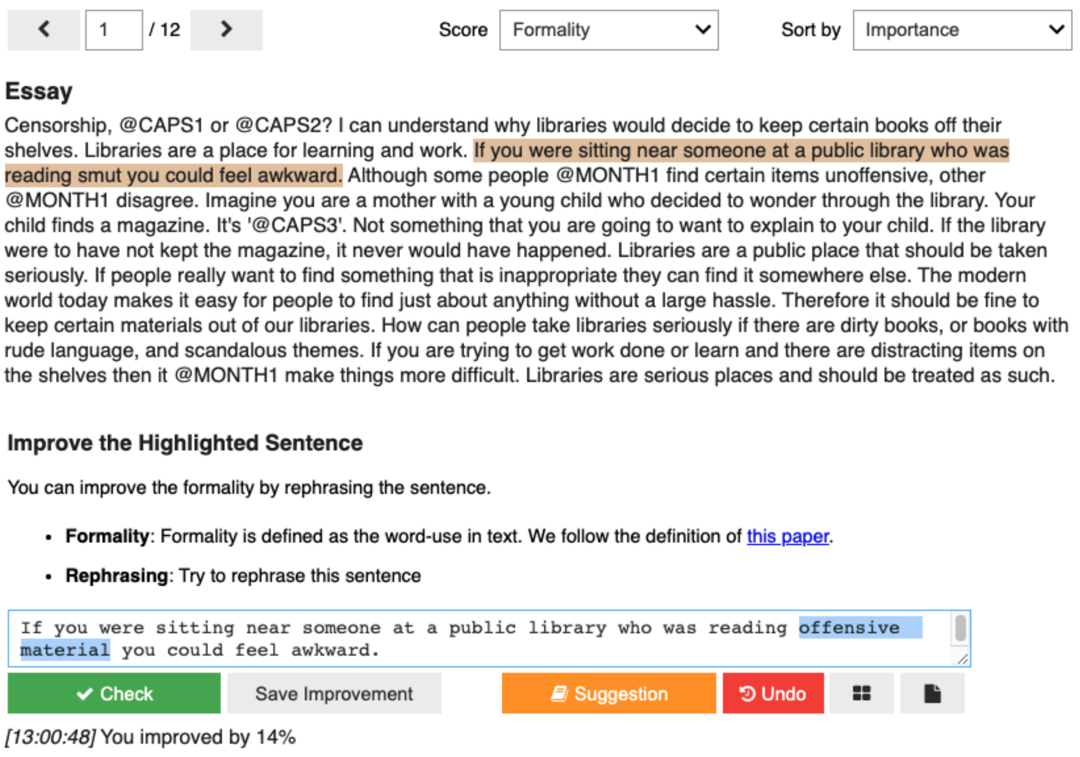
The last assessment category is formality that focuses on proper word usage. Similar to readability, formality is used in AES research (Zesch, Wojatzki, and Scholten-Akoun, 2015). Also, the writing expert's example rubric (that is discussed in section 3.1) include formality as grading criteria.

4.2.3 Feedback Model

Since essay scoring results in an essay-level score and the requirements indicate to focus on localized feedback, *RevisionCoach* transforms the essay-level score into a sentence-level score. For this, we follow an approach similar to Woods et al., 2017 that removes individual sentences and measures the impact. A significant impact on the essay score as a consequence of a single sentence removal indicates the importance of that sentence. In the same way, a significant impact on the essay score as a consequence of a single sentence revision indicates the importance of the revision. This technique can be used with multiple scoring models to assess the essay's quality for multiple writing assessment categories.

RevisionCoach considers an improvement as significant when the improvement is larger than 5% compared to the original score. To not overload students with possible improvements, which is possible since all three revision models can suggest an improvement for the same sentence and for the same assessment category, *RevisionCoach* only suggests the highest score improvement for each sentence for each assessment category. There is no feedback left when the revisions do not improve the essay's quality in any of the assessment categories. In *RevisionCoach*'s user interface prototype, we display the discovered feedback to the student.

4.2.4 User Interface Design Prototype



Essay

Censorship, @CAPS1 or @CAPS2? I can understand why libraries would decide to keep certain books off their shelves. Libraries are a place for learning and work. If you were sitting near someone at a public library who was reading smut you could feel awkward. Although some people @MONTH1 find certain items unoffensive, other @MONTH1 disagree. Imagine you are a mother with a young child who decided to wonder through the library. Your child finds a magazine. It's '@CAPS3'. Not something that you are going to want to explain to your child. If the library were to have not kept the magazine, it never would have happened. Libraries are a public place that should be taken seriously. If people really want to find something that is inappropriate they can find it somewhere else. The modern world today makes it easy for people to find just about anything without a large hassle. Therefore it should be fine to keep certain materials out of our libraries. How can people take libraries seriously if there are dirty books, or books with rude language, and scandalous themes. If you are trying to get work done or learn and there are distracting items on the shelves then it @MONTH1 make things more difficult. Libraries are serious places and should be treated as such.

Improve the Highlighted Sentence

You can improve the formality by rephrasing the sentence.

- **Formality**: Formality is defined as the word-use in text. We follow the definition of this paper.
- **Rephrasing**: Try to rephrase this sentence

```
If you were sitting near someone at a public library who was reading offensive
material you could feel awkward.
```

[13:00:48] You improved by 14%

FIGURE 4.8: *RevisionCoach*'s user interface.

As mentioned in the requirements (chapter 3), the user interface follows the conditions of deliberate practice (Anders Ericsson, 2008) to encourage effective learning. The main purpose of *RevisionCoach*'s user interface prototype is to visualize the potential of *RevisionCoach*'s feedback. Because the user interface is not tested or evaluated in the interest of time, we consider the user interface as a prototype.

First, the learner has access to a clear user interface that displays clear tasks and goals, namely to improve individual sentences based on a writing quality assessment category. Students select individual sentences to improve and students gradually move through the text to implement improvements. This process stimulates iterative learning and provides repeated learning opportunities. Furthermore, to support differentiated learning, the user interface supports the four layers of feedback (sentence highlight, mistake category, correction category, and suggested correction), and students can request the scoring rubric at any time. More structured feedback is available since the writing assessment categories are linked to pre-defined tips and suggestions. All actions are conveniently logged for evaluation purposes in local storage.

In Figure 4.8 (a screenshot of *RevisionCoach*'s user interface), the header allows students to iterate between sentences, select an assessment category, and select a sorting option (importance or sentence index). For a given selection, the UI shows the sentence that is under consideration and related pre-defined tips. In the current UI, the pre-defined tips are short and provide general explanations for the assessment category, while it should be defined by a writing expert. Moreover, the bottom part of Figure 4.8 allows the student to change the sentence and to examine the improvement, where the improvement shows the quality increase for the selected assessment category (100% refers to *RevisionCoach*'s predicted best revision; therefore, it is possible to improve more than 100% when the student's improvement is better than *RevisionCoach*'s). Other than checking the improvement, the student can save the improvement (and the improved sentence will replace the original sentence in the UI), ask for a suggestion (show *RevisionCoach*'s best revision), undo the previous improvement (reverts the 'save improvement' action), view all assessment quality scores, and view the action history. The following chapter discusses *RevisionCoach*'s implementation details.

Chapter 5

Implementation

This chapter covers the implementation details of the revision models, the scoring models, the feedback model, and the user interface. Although we consider three revision models and four scoring models, extending the system with additional models is possible.

Package	Version
nltk	3.2.5
spacy	2.2.4
spacy-readability	1.4.1
transformers	4.5.1
sentencepiece	0.1.96
ipython	5.5.0
pandas	1.1.5
torch	1.9.0+cu102
sklearn	0.0
numpy	1.19.5

TABLE 5.1: The most important Python packages for *RevisionCoach*.

To reduce the *RevisionCoach*'s waiting time between development and deployment, we use a Colab Notebook[1] provided by Google. All Python code - back-end and front-end - resides in the same notebook, and therefore, the notebook code execution loads all revision models, scoring models, feedback model, and the user interface. The code is accessible here[2]. Table 5.1 contains the most important Python packages and their version numbers. In the notebook, the components follow the model-view-controller design pattern. Furthermore, code updates to one of the components are directly available in the user interface (UI). Another advantage of the colab notebook is that the notebook is shareable with test users for evaluation purposes. First, we discuss the revision models' implementation details.

5.1 Revision Models

RevisionCoach's revision models are written in Python and rely on a range of tools and resources, including transformer models, a linguistic corpus, a part-of-speech (POS) tagger, a paraphrase dataset, and an essay dataset. For a new assignment, the only required data is essay texts without additional annotations. Other sources can be considered, such as relevant academic papers or Wikipedia articles, although the

[1] `colab.research.google.com`

[2] GitHub: `https://github.com/PJvanderLaan/Automated-Writing-Feedback`

original revision models rely on similar essay texts. This section starts with background information of transformer models to provide arguments for the used transformer architecture. Afterward, we discuss the revision models.

5.1.1 Background of Transformer Models

The revision models depend on text comprehension models that are capable of generating natural language. Currently, state-of-the-art natural language processing (NLP) models use a transformer architecture (Vaswani et al., 2017) for text comprehension tasks, such as open-domain question-answering or summarization. Transformer architectures make use of the attention mechanism that existed already at the time that Vaswani et al., 2017 was published. On the other hand, the breakthrough of Vaswani et al., 2017 relies upon two crucial advancements: multiple attention heads and positional encoding of tokens.

Before diving into a brief explanation of transformer models, it is important to introduce the purpose of transformers. In particular, the core machine learning task that transformers in NLP consider is called sequence transduction. Most tasks that consider sequences as input and that require a transformation fall in the sequence transduction category (Graves, 2012). Sequence transduction models commonly include an encoder that encodes the sequence into a vector representation, and a decoder that maps the numerical representation back to the original form. The two parts work together to support memorization and pattern discovery, and as a result, the model learns to model sequence transduction tasks.

In contrast to the common recurrent neural networks (RNN) and convolution neural networks (CNN) architectures, transformers rely on the attention mechanism. During the training phase, three independent matrices are learned to construct the query, key, and value vector based on the input - a word sequence transformed into word embeddings. An attention function maps a query vector to an output using key-value vector pairs, and the query and key vectors calculate the attention score using the dot-product. Instead of the dot-product, additive attention scores using a feed-forward network are an alternative. As a result, additive attention scores are significantly slower to compute because the dot-product calculation is based on fast, optimized matrix multiplication code (Vaswani et al., 2017). After the attention score calculations, the scores are multiplied with the value vector to compute the final attention. In practice, multiple query, key, and value vectors are combined into respective matrices to parallelize the computation. The intuition is that the query, key, and value matrices learn to find patterns in sequences using the attention scores and learn to determine the word patterns' importances with the value vectors.

Instead of calculating the attention scores of a single sequence, the model learns more when attention is calculated for multiple representations of the sequence. In Vaswani et al., 2017, the query, key, and value vectors are linearly projected h times with different learned projections. The projections are used as input to the attention function and are computed in parallel. All final outputs are concatenated. This process, where multiple attention functions are used on projected inputs, is called multi-head attention. Transformer architectures commonly have multiple multi-head attention (sub-)layers - Vaswani et al., 2017 uses 6 - to capture low- and high-level relationships. According to Vaswani et al., 2017, the lower layers of multi-head attention capture mostly linear word order, and the upper layers carry more syntactic information of the sequence.

Using multiple multi-head attention functions, a transformer has four main advantages over the RNN and CNN architectures (Vaswani et al., 2017):

1. **Long memory.** The attention functions can attend to any position of the input query with $O(1)$ path length, whereas other model structures, such as RNN, require $O(n)$ as path length where n is the sequence length. For instance, this is important when words at the start of the first sentence are important for the words at the end of the last sentence. In contrast to recurrent neural networks, the attention function do not suffer from a vanishing gradient.

2. **Parallelization and interpretability.** Attention can be computed in parallel, in contrast to recurrent neural networks that are processed sequentially. In addition, attention weights are transparent and can be visualized for improved interpretability.

3. **All input is considered.** CNN use filters to find patterns that frequently do not consider the entire input. The transformer consistently considers the entire input and allows connections between all input pairs.

4. **Partly explainable.** In contrast to black-box neural networks, the attention scores of multi-head attention layers can be visualized. As a result, the transformer models are better interpretable and explainable.

The original implementation of Vaswani et al., 2017 uses an encoder-decoder architecture with a translation task as training data. Since then, authors have proposed improved architectures and training to generalize the applicability and improve the performance. We discuss three relevant architectures: BERT, GPT-2, and T5.

BERT and GPT-2

Devlin et al., 2019 focuses on transfer learning to generalize the applicability of the model and introduces the pre-training task. Transfer learning depends on a pre-training and fine-tuning step. The intuition is that the model is pre-trained on a task and fine-tuned on a downstream task. As a result, the downstream task does not train all parameters from scratch, which is time-consuming and expensive. Another benefit is that the same BERT pre-training architecture is possible for numerous NLP downstream tasks. An additional step between pre-training and fine-tuning is called intermediate training. Pruksachatkun et al., 2020 show that the intermediate training task can improve the transformer's performance.

Although the original transformer architecture uses an encoder and decoder, BERT works with only the encoder part. Unlike other transformer architectures, the pre-training task of BERT is bidirectional. The original architecture is unidirectional, which means that each token can attend only to the previous tokens in the attention layers during training. Therefore, all tokens to the right are excluded from learning the language relationships. BERT notes that both contexts are important, especially when considering NLP tasks such as question answering and summarization (Devlin et al., 2019).

BERT introduces masked language modeling (MLM) as a pre-training task to support bi-directional training. Tokens are randomly masked with a random or unique mask token. Subsequently, the training goal is to predict the vocabulary ids of the masked tokens. Furthermore, BERT trains on the next sentence prediction (NSP) task to model the relationships between sentences. MLM does not directly capture the relationships between entire sentences, whereas NSP is designed for that purpose. However, according to Rogers, Kovaleva, and Rumshisky, 2020, removing the NSP pre-training task does not decrease the performance.

Other transformer architectures are more suitable for text generation tasks due to BERT's bidirectional nature and the lack of a decoder. Text generation is auto-regressive (unidirectional), meaning that each generated token uses all previous tokens as input. Although bidirectional learning is hypothetically better to model language, text generation remains a unidirectional task, since only previous tokens are known during text generation.

In contrast to BERT, GPT-2 (Radford et al., 2018) is a transformer-based language generation model that trains unidirectionally, has an autoregressive nature, and uses only a decoder. GPT-2 can generate human-like text and GPT-2 supports the auto-regressive property, but intuitively, missing bidirectional contexts during pre-training results in a less powerful understanding of language. On the contrary to BERT and GPT-2, T5 (Raffel et al., 2019) is autoregressive and includes bi-directional language modeling.

T5

The authors of T5 state that the original work is primarily a survey of the existing techniques (Raffel et al., 2019). Nonetheless, the T5 model is trained on different data and includes an adaptation to MLM as a pre-training task. The architecture is similar to the original transformer architecture (Vaswani et al., 2017) and contains an encoder and decoder stack.

T5 is designed to be a text-to-text transfer transformer that is usable for numerous NLP tasks by transforming the task into a text-to-text format. To support regression tasks, the authors use string representation of the numbers. In practice this transformers the regression task in a multi-classification task.

Raffel et al., 2019 uses a new dataset that consists of clean text – the Collossal Clean Crawled Corpus (C4). To summarize, C4 consists out of cleaned web extracted text using several preprocessing steps. As a result, the C4 dataset is large (750GB) compared to other datasets.

During pre-training, T5 uses unidirectional masking in the decoder and bidirectional masking in the encoder in contrast to BERT (only an encoder and bidirectional) and GPT-2 (only a decoder and unidirectional). Masked tokens are excluded from the attention calculation in the attention heads. The unidirectional auto-regressive property, which is important for text generation, is kept while holding on to bidirectional language modeling in the encoding step. The transformer can reason about bidirectional information in the input text as a result. This architecture achieved the best performance in the study of Raffel et al., 2019.

Although T5 depends on existing techniques and architectures, it has important differences with other architectures such as BERT. Whereas BERT includes byte-level tokenization, T5 uses only SentencePiece to encode text as WordPiece tokens. The benefit of byte-level tokenization is that it generalizes better to unknown words. As a result, byte-level tokenization supports a larger vocabulary; however, the drawback is that the search space becomes larger. The second important difference is related to pre-training. T5's pre-training objective uses a small adaptation to BERT's MLM training task. Whereas BERT masks individual tokens, T5 masks spans of tokens as denoising objective, using the same dropout percentage for tokens - 15% is masked. To conclude, the last difference is that BERT is an encoder-only model and is trained to predict the masked tokens. On the contrary, T5 also includes a decoder and is trained to reconstruct the entire input. Raffel et al., 2019 finds that MLM with masking spans of 3 tokens performs the best as a pre-training objective.

To conclude, we use the background information to select the best transformer architecture for each task. Apart from BERT, GPT-2, and T5, there are numerous other transformer architectures. However, we consider BERT, GPT-2, and T5 as examples that illustrate the most important differences between transformer architectures in the automated feedback setting – the pre-training procedure, the training data, unidirectional or bidirectional (or both), and the autoregressive property. The following sections about the revision models consider this transformer background information to determine the most suitable transformer architecture for the task.

5.1.2 Rephrase Revision

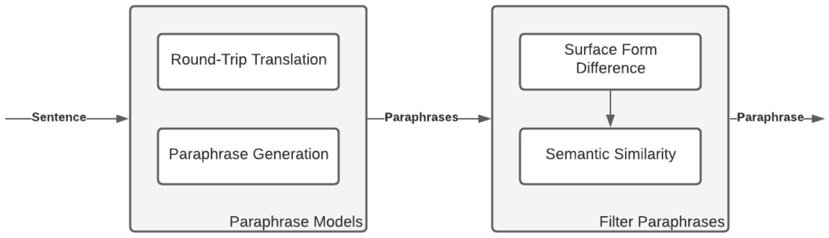


FIGURE 5.1: The flow of constructing paraphrase revisions.

The rephrase revision model consists of two paraphrase techniques: round-trip translation (RTT) and paraphrase generation (Figure 5.1). RTT is a technique that translates a text into a different language and back to the original language, which results in a paraphrase of the original text. Aroyehun and Gelbukh, 2018 show that RTT is effective and robust for data augmentation in French, Spanish, Hindi, and German. For the translation task, we use pre-trained transformer models (T5 for German to English and Facebook's transformer model for English to German[3]).

The second paraphrase model is a pre-trained T5 model[4] fine-tuned on Google's PAWS paraphrase dataset[5]. T5 supports bidirectional language modeling that allows T5 to reason about words based on the word's context in both directions and T5 supports the autoregressive unidirectional property to generate natural language. In addition, T5 achieves state-of-the-art performance in summarization, text classification, question answering, and more (Raffel et al., 2019). Since a paraphrasing model depends on natural language comprehension and language generation, we select T5 for the paraphrase revision task. Pre-training transformer networks is expensive and takes a long time, and to address that, we use the pre-trained T5 model available on HuggingFace[6].

The RTT and paraphrase generation model are used to construct sentence-level paraphrases, and the next step is to determine the best paraphrase. A good paraphrase is similar in meaning but different in surface form, where a similar surface form indicates a similar sentence structure with similar words. To measure this, we assess semantic similarity with BERTscore (a bidirectional transformer-based semantic similarity model, Zhang et al., 2020) and we assess surface form similarity with BLEU (Papineni et al., 2001), the most common - n-gram based - machine translation metric (Zhang et al., 2020). In the end, the model selects the paraphrase with the

[3]`https://huggingface.co/facebook/wmt19-en-de`
[4]`https://huggingface.co/Vamsi/T5_Paraphrase_Paws`
[5]`https://github.com/google-research-datasets/paws`
[6]`https://huggingface.co/Vamsi/T5_Paraphrase_Paws`

highest semantic similarity and with the lowest surface form difference out of a pool of RTT and generated paraphrases.

5.1.3 Rewrite Revision

In contrast to the rephrase revision's paraphrase generation model, this revision model's goal is to generate an essay sentence unrelated to the original essay sentence, while the generated sentence makes sense in the original sentence's context. In practice, essay sentence generation depends on the context of the original sentence to allow the text comprehension model to generate context-aware sentences. In the following section we provide an overview of *RevisionCoach*'s text generation architecture and discuss the fine-tuning process to elaborate upon free parameter decisions.

Text Generation Architecture

Similar to paraphrase generation, we use a T5 model for the bidirectional understanding and for the autoregressive property. The bidirectional understanding supports the model's goal to generate a sentence based on a context before and after the input sentence, and the autoregressive property enables the model to generate natural text. On the contrary, we fine-tune the pre-trained T5 transformer on the essay dataset containing similar essays that contain relevant content with respect to the assignment's task. To start, we transform the essays in a (sentence before, original sentence, sentence after) format to fine-tune the T5 transformer. T5 uses the masked spans of tokens as a pre-training objective that is similar to our fine-tuning objective (where the original sentence is the masked span of tokens). Fine-tuning a large transformer model is resource-demanding; and therefore, we use a Google Colab premium cloud GPU.

For each essay's input sentence we select two sentences as context: one before and one after the input sentence. As a result, the first and last sentence of the essay can learn only from a single sentence as context, but that is a trade-off we are willing to make.

An important part of text generation is the decoding setup to generate natural, not-repeatable and reasonable text. There are several options to generate text from the transformer model's output. In itself, the output consists of probabilities for each token in the vocabulary for each token in the generated output, meaning that an output sentence of 10 tokens with a vocabulary of 10.000 has a shape of [10, 10.000]. There are several options to decode the output probabilities, including Greedy Search, Beam Search, Top-K sampling, and Top-p sampling (supported by the transformer library called HuggingFace[7]). Each option tries to generate the most natural-looking text and tries to prevent text repetition. Although some options can generate multiple text variations, the essay text generation models only require one output text. We use beam search with 4 beams and greedy decoding, 1 return sequence, and a length penalty of 0.6, to produce a single feedback text (similar to the decoding strategy for all NLP tasks of Raffel et al., 2019). By considering more beams - or paths in a search tree - we encourage the model to search beyond a single next best token. In the following section, we discuss the fine-tuning process of the T5 text generation model.

[7] `https://huggingface.co/blog/how-to-generate`

Fine-Tuning T5

Parameter	Value
Learning Rate	0.001
Steps	2^{18}
Optimizer	AdaFactor
Decoding Strategy	Greedy decoding
Loss	Cross-entropy

TABLE 5.2: T5 fine-tuning parameters (Raffel et al., 2019)

Transformer architectures are time-consuming and resource-demanding to train; and therefore, we use the pre-trained T5-base model[8] for the fine-tuning process. Although fine-tuning is less time-consuming than pre-training, it is infeasible to do a complete grid search to find the perfect parameters for all parameters (for instance, epochs, learning rate, warmup steps, optimizers, all optimizer parameters). To address this, we re-use the fine-tuning parameters of T5's original work (Table 5.2).

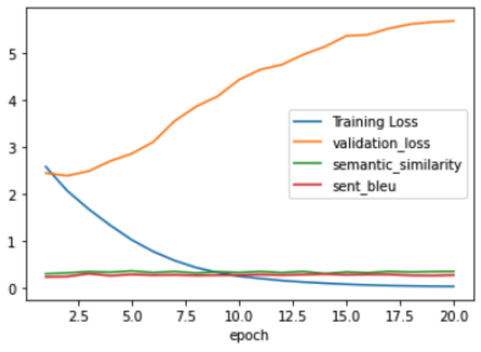
Although most parameters in Table 5.2 are determined to work best with the pre-trained T5 architecture, Raffel et al., 2019 mentions that the 'steps' parameter is chosen as a trade-off between high-resource (large datasets) and low-resource tasks (overfits quickly). *RevisionCoach* uses small datasets compared to T5's datasets; and therefore, we expect the fine-tuning to not require $2^{18} = 262$k steps. To monitor the fine-tuning process we include the training loss, validation loss (for 100 randomly sampled essays that we exclude from the training dataset), semantic similarity (between the model prediction and validation dataset labels), and sentence BLEU (n-gram overlap). In the open-ended sentence generation revision case there is not a single correct answer, and as a consequence, the training and validation loss do not represent the actual loss between the predictions and correct answer. Therefore, we include the sentence BLEU and semantic similarity to generalize the single correct sentence from the validation dataset to multiple related answers because the loss includes only the difference between the generated sentence and the expected sentence.

Figure 5.2 shows the fine-tuning results using two essay datasets for 20 epochs, where Figure 5.2a uses the essays from the ASAP Challenge's second task (available here[9]) and Figure 5.2b uses essays about genetically modified organisms (GMO). The following listing shows the training input structure for three subsequent sentences (sentence 0 to sentence 2).

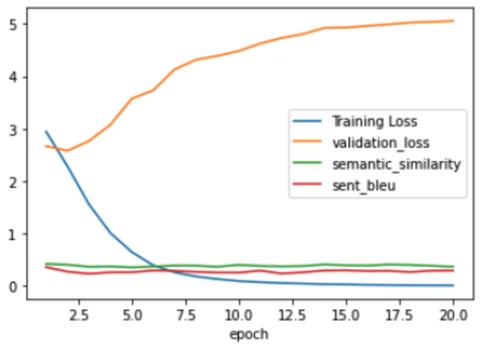
```
input: [sentence 0] <extra_id_0> [sentence 2]
label: <extra_id_0> [sentence 1] <extra_id_1></s>
```

The extra id's represent unique mask tokens that the pre-trained T5 model uses during the masked language modeling pre-training stage[10] and </s> indicates the end of the sentence. Although the original fine-tuning process of Raffel et al., 2019 takes $262k$ steps, the models are trained for 20 epochs resulting in $20 * 6597 = 132k$ and $20 * 745 = 15k$ steps for the ASAP and GMO essays respectively. In Figure 5.2 we observe that the models slightly overfit in both cases, because the validation loss increases and the training loss decreases. On the contrary, we observe that the semantic similarity and sentence BLEU do not decrease significantly (Figure 5.2c and

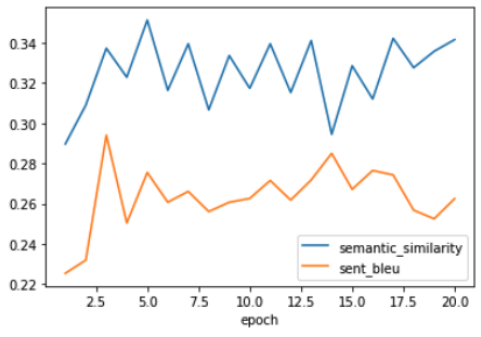
[8] https://huggingface.co/t5-base
[9] https://www.kaggle.com/c/asap-aes
[10] https://huggingface.co/transformers/model_doc/t5.html



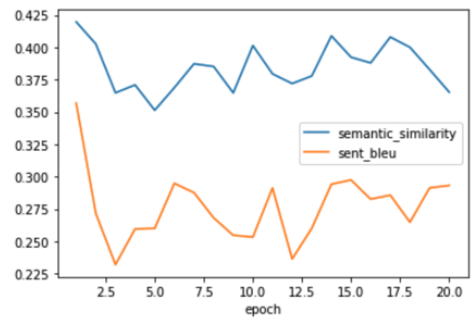
(A) Fine-tuning analysis of a pre-trained T5-base model for 20 epochs using 6597 training essays and 100 validation essays.



(B) Fine-tuning analysis of a pre-trained T5-base model for 20 epochs using 745 training essays and 83 validation essays.



(C) Fine-tuning analysis of a pre-trained T5-base model for 20 epochs using 6597 training essays and 100 validation essays.



(D) Fine-tuning analysis of a pre-trained T5-base model for 20 epochs using 745 training essays and 83 validation essays.

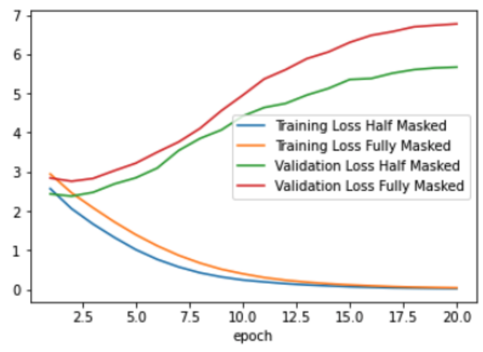
FIGURE 5.2: Fine-tuning analysis using two essay datasets.

Figure 5.2d). Based on both observations, the models are slightly overfitting - although it is challenging to assess whether the overfitting is an issue because the predicted sentences are possibly not similar to the expected answer, while at the same time, the predicted sentence can be valid in the essay context. Manual experimentation shows that the predicted sentences make sense in the essay's context (Appendix B).

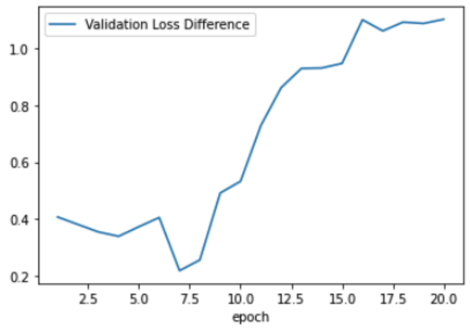
Originally, T5 is pre-trained with masked random spans modeling, that masks part of sentences contrary to the previous fine-tuning input where entire sentences are masked. To investigate the effects on the transformer model when parts of the sentences are masked we run the experiment with the following setup:

```
input: [sentence 0] [first half of sentence 1] <extra_id_0> [sentence 2]
label: <extra_id_0> [second half of sentence 1] <extra_id_1></s>
```

Figure 5.3 shows that the validation loss difference between fully masked and partly masked sentences becomes larger during the fine-tuning process, and as a result, the partly masked sentences causes less overfitting. The partly masked model's initial validation loss is smaller because the target labels contain fewer words; however, that does not explain the increasing difference between the validation losses (Figure 5.3b). At the same time, both models converge to a similar training loss (Figure 5.3a). The difference is minimal, and the validation loss does not directly relate to overfitting because multiple predictions can be correct while there is one expected prediction. Second, the partly masked input results in faster convergence of the training loss (Figure 5.3a). Moreover, we analyse the predictions of the partly



(A) Fine-tuning difference of a pre-trained T5 model between fully masked and partly masked sentences.



(B) Validation loss difference between fully masked and partly masked input sentences. The loss is calculated by subtracting the partly masked validation loss from the fully masked validation loss.

FIGURE 5.3: Difference in losses between two models. Partly masked refers to training labels that mask the second half of the sentence, while fully masked refers to training labels that mask the complete sentence.

masked and fully masked fine-tuned models (due to space considerations, we included the predictions in Appendix B). As expected, the fully masked sentence fine-tuning results in completely different sentence predictions because the model has no knowledge of the original sentence. On the other hand, the partly masked sentence fine-tuning results are more similar to the original sentence while it still results in valid sentence predictions that are different from the original sentence. Furthermore, two interesting observations are that the T5 base predictions (without additional fine-tuning) do not make sense and that the T5 fully masked fine-tuned model frequently predicts a sentence that repeats the sentence before.

To conclude, based on the fine-tuning analysis we observe that partly masked sentences result in small advantages, and we observe that the original 2^{18} steps are not necessary for *RevisionCoach*'s small essay datasets because the training losses converge relatively fast (Figure 5.3a). *RevisionCoach* uses partly masked sentences as input as a result of the small advantages. Contrary to text generation that uses a transformer model, the synonym revision model depends on part-of-speech tags and WordNet to analyse the text structure.

5.1.4 Reword Revision

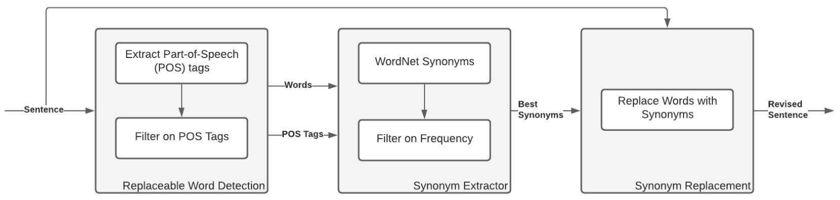


FIGURE 5.4: The flow of constructing synonym-based revisions.

Replacing content words in sentences with synonyms requires the model to understand the text structure. Not all words can directly be replaced with their synonyms. For instance, replacing a verb with a synonym is tricky - it can lead to a different semantic meaning of the sentence, and the correct tense and form (verb conjugation) of the verb are required for a correct sentence. The synonym revision process follows the synonym data augmentation technique of Zhang and LeCun, 2016 and consists of three steps (visualized in Figure 5.4): (1) select replaceable words, (2) select the best synonym, and (3) replace the words by their best synonym.

In first step *RevisionCoach* considers a word replaceable when it is not a stopword or a verb. To detect the words' part-of-speech (POS), we use the NLTK POS tagger[11]. After manual experimentation with verb conjugation, we noticed that the accuracy to construct natural sentences was tricky, and since not all words have to be replaced to construct a revision, we decided to exclude verbs. Other than verbs, *RevisionCoach* considers only alphabetical characters and excludes stopwords.

Second, we select the best synonym based on the synonym's frequency in the NLTK brown corpus, a million-word English corpus with a variety of text categories such as news. We use WordNet[12] to get the potential synonyms for each word.

The last step replaces the replaceable words with the most frequent synonym to construct a revision. When no synonym is found the original word is not replaced or removed.

5.2 Scoring Models

In this section, we discuss the implementation details of the scoring models. The scoring models are based on existing research. To not complicate the score comparison process, we transform the scores such that a higher score correlates with a better text quality.

5.2.1 Readability

$$206.835 - 1.015\left(\frac{\text{total words}}{\text{total sentences}}\right) - 84.6\left(\frac{\text{total syllables}}{\text{total words}}\right) \tag{5.1}$$

Readability tests are designed to identify the text's reading difficulty. For that purpose, the Flesch-Kincaid Readability Reading Ease is one of the widely used readability metrics to estimate the required grade level to understand a text (Si and Callan, 2001). Flesch-Kincaid Reading Ease disregards the difficulty of understanding a concept. In addition, the best Flesch-Kincaid readability score depends on the assignment task because different texts require different readability levels (newspapers are more formal than opinion pieces, for example). In general, a higher Flesch-Kincaid readability score indicates that the text is more difficult to read. An advantage of the Flesch-Kincaid readability is that the score calculation is fast because it requires only information about the total number of words, sentences, and syllables.

RevisionCoach uses the NLP python library spaCy[13] to calculate the Flesch Kincaid Reading Ease score with Equation 5.1. Finally, we transform the readability score ($100 - \textit{readability score}$), such that a higher readability score indicates an easier to read text.

[11] `https://www.nltk.org/book/ch05.html`
[12] `https://wordnet.princeton.edu/`
[13] `https://spacy.io/universe/project/spacy_readability`

5.2.2 Coherence

In contrast to the readability formula, coherence is harder to calculate, and therefore, we designed a custom coherence score for *RevisionCoach* to illustrate that simple scores are effective to produce localized feedback using *RevisionCoach*'s architecuture. In this section, we explore related methods and additional background information to construct an effective coherence scoring model. Automatic assessment of text coherence is an important task in NLP, for example in text generation, in human-produced text evaluation, and in finding the best sentence ordering in summarization (Elsner, Austerweil, and Charniak, 2007). Li et al., 2018 propose a solution using the relationships between words as coherence score, that achieves state-of-the art results on the common ASAP automated essay scoring challenge[14]. However, the model requires coherence scores (or overall essay scores) for each essay that requires additional manual annotation. Therefore, we select an older technique that focuses on semantic similarity between subsequent sentences to approximate coherence scores (Lapata and Barzilay, 2005) that requires no additional manual annotations. The intuition is that subsequent sentences are assumed to be similar in meaning for coherent texts. One of the main drawbacks of this approach is that it has trouble to detect sudden shifts in topics, for example between paragraphs (Elsner, Austerweil, and Charniak, 2007). For the purpose of *RevisionCoach*, that specifically targets small essays, we hypothesize that this approach can approximate text coherence properly without adding additional manual annotation requirements.

A key component of all text similarity methods is to transform text into numerical representations, which are typically called word or text embeddings. Without numerical representations, models are unable to reason about text and find relations between words. The methods to construct text embeddings shifted over the years, and to provide an overview of the methods, we discuss LSA, Word2Vec, ELMo, and language models.

LSA and Word2Vec

Latent semantic analysis (LSA, Landauer, Foltz, and Laham, 1998) is one of the earlier methods to construct word embeddings. A complete explanation of LSA is beyond the scope of this thesis project and we refer to Landauer, Foltz, and Laham, 1998 for additional reading. To summarize, LSA starts with a matrix of occurrence values for unique words (also called a bag of words), where the columns represent documents or sentence contexts and the rows represent unique content words. Next, LSA applies singular value decomposition to the matrix to construct three matrices. The product of the three matrices equals the original matrix. By reducing the dimensions of the three matrices, usually starting with the least significant and lowest values, a least-square fit can be constructed of the original matrix. Unique words in the least-square fit that do not occur in one of the documents, but do frequently occur alongside unique words of that document, receive a higher occurrence value for the document it does not appear in. Therefore, given the assumption that the unique content words in the rows properly represent the document's (or sentence's) content, the columns are considered contextualized text embeddings. On the other hand, the rows represent contextualized word embeddings. Besides constructing contextualized text and word embeddings, LSA can be used to find discussed topics in text, also called topic modeling.

[14]`https://www.kaggle.com/c/asap-aes`

A similar approach is used in Word2Vec to capture semantic similarity (Mikolov et al., 2013). Word2Vec calculates the word embeddings using a deep neural network. On large datasets, LSA becomes computationally expensive while Word2Vec's neural network approach performs better (Mikolov et al., 2013). As a consequence, neural nets are more capable of considering a large corpus of words. A wide range of neural network architectures is proposed to solve the task of computing word embeddings.

ELMo and BERT

ELMo is a more recent embedding technique that captures contextual relations using a language model (Peters et al., 2018). A language model task is a probability distribution over a sequence of words, an important task for NLP and text comprehension (Jozefowicz et al., 2016). In other words, a language model tries to understand which words are most probable based on the preceding word probabilities, usually by pre-training on a large text dataset. ELMo is bidirectional language model (the probability of each word depends on words before and after) using a bi-LTSM (bi-directional long short-term memory) recurrent neural net architecture.

In addition to the ELMo architecture, transformer-based architectures show more potential in predicting contextualized word embeddings (Ethayarajh, 2019). Ethayarajh, 2019 shows that BERT suggests a more nuanced contextualization than ELMo, while both include context in the word embeddings. The nuance in contextualization is based on that two words in the same sentence do not necessarily share the same meaning when they share the same context (Ethayarajh, 2019), which is more the case in the BERT architecture than in ELMo. Another interesting point is that the self-similarity of BERT's words embeddings is less than ELMo's, where the self-similarity is the average cosine similarity of embeddings in different contexts, making BERT's embeddings more context-specific than ELMo's embeddings (Ethayarajh, 2019).

Based on the research, we conclude that transformer-based embeddings are most capable of capturing the text's semantics. To take advantage of the newer transformer models to capture semantic similarity, we calculate the semantic similarity with the bidirectional transformer architecture T5, where the input is:

```
stsb sentence 1: [sentence 1]   sentence 2: [sentence 2]
```

T5 is pre-trained on the semantic textual similarity benchmark (STS-B) where 'stsb' is used as prefix for all training data. In the global SuperGLUE benchmark, T5 is one of the highest-scoring models for the STS-B[15]. Larger T5 models have better performance (Raffel et al., 2019), but unfortunately, it also requires a significant amount of RAM and it takes longer to generate results. Because *RevisionCoach* needs to calculate hundreds of semantic similarities, and because less RAM requirements are more convenient, we select T5-base - the baseline model - as pre-trained T5 model[16]. According to the evaluation of Raffel et al., 2019, the baseline T5 model performs relatively well with a STS-B Pearson Spearman Correlation of 88.02 compared to the largest T5 architecture in the SuperGLUE leaderboard with 93.1.

[15]SuperGLUE (visited on 31 August): https://gluebenchmark.com/leaderboard
[16]https://huggingface.co/t5-base

$$\frac{1}{2}\left(\frac{\sum_{i=0}^{|S|-1}\mathrm{sim}(S[i],S[i+1])}{|S|-1}+\frac{\sum_{i=0}^{|S|-2}\mathrm{sim}(S[i:i+1],S[i+1:i+2])}{|S|-2}\right) \tag{5.2}$$

where:

$\|S\|$	= Number of sentences in the essay
$S[x], S[\mathrm{x}_1{:}x_2]$	= Essay sentence at index x, and essay sentences at indices x_1 to x_2
$sim(x,y)$	= Semantic similarity between texts x and y

The coherence model in *RevisionCoach* measures the similarity between pairs of sentences and averages the similarities to construct a final coherence score. To allow for both local and (more) global coherence, we select pairs of one sentence and pairs of two subsequent sentences. Equation 5.2 shows the essay's coherence score calculation using the sentence similarity pairs' average coherence scores.

5.2.3 Cohesion

Third, the cohesion scoring module focuses on connective words and represents the number of connective words divided by the number of unique words in an essay. Similar to the other scores, the cohesion score is not necessarily better when it is higher or lower because it depends on the essay's task. To gather a corpus of connective words we combine connective words from GrammarBank[17] with connective words from the University of Pennsylvania[18]. *RevisionCoach* uses the constructed connective word corpus to count the occurrences of connective words after tokenization.

5.2.4 Formality

Finally, the formality scoring method uses the F-Score formula from Heylighen and Dewaele, 1999. To construct all necessary inputs for the F-Score, *RevisionCoach* first tokenizes the sentences and gathers the POS tags with the NLTK POS Tagger. Finally, the following formula calculates the F-Score (Heylighen and Dewaele, 1999):
F = (noun frequency + adjective freq. + preposition freq. + article freq. – pronoun freq. – verb freq. – adverb freq. – interjection freq. + 100)/2

5.3 Feedback Controller

Type	Name	Core Technique(s)	Data
Revision	Paraphrase	RTT and T5	Google PAWS
Revision	Text generation	T5	Essays
Revision	Synonym	WordNet synonyms, NLTK POS tagger	NLTK's brown corpus
Score	Readability	Flesch-Kincaid Reading Ease (SpaCy)	-
Score	Coherence	T5 Semantic Textual Similarity	-
Score	Cohesion	NLTK tokenization	-
Score	Formality	F-Score	-

TABLE 5.3: Summary of the scoring models and revision models.

[17] https://www.grammarbank.com/connectives-list.html
[18] https://www.cis.upenn.edu/~nlp/software/discourse.html

The feedback controller uses the scoring models and revision models (Table 5.3) to construct a feedback collection for an input essay. To summarize, the feedback controller filters on significant score improvements (where an improvement of more than 5% is significant) and removes duplicate feedback. In addition, the feedback controller handles requests to calculate the impact of the student's improvement suggestions.

5.4 User Interface Design

RevisionCoach's UI is implemented in the Google Colab Notebook using iPython widgets[19]. Three low-level UI controllers control the header, the essay view, and the improvement view, all of which use the feedback controller to retrieve and process relevant data. On top of the three UI controllers, a general UI controller allows for communication between the low-level UI controllers.

[19]https://ipywidgets.readthedocs.io/en/latest/

Chapter 6

Evaluation

In this evaluation we investigate RevisionCoach's feedback accuracy with an expert-based experiment. Writing experts formulate feedback on sentence-level, which we compare to RevisionCoach's feedback. Since it is challenging to accurately compare feedback and to translate feedback similarity into an evaluation number, we simplify the evaluation by considering a pre-defined feedback structure consisting of a sentence-level highlight, an assessment category, and a mistake importance. *RevisionCoach* can construct feedback in a similar structure, and we ask experts to do the same. This way, we can compare expert feedback to *RevisionCoach* feedback by comparing the sentence-level highlights, assessment categories, and mistake importances. And finally, the feedback is compared in terms of inter-rater agreement (IRR) and mean square error (MSE) to evaluate the system's accuracy.

6.1 Method

There are several options to evaluate essay feedback systems. Burstein, 2004 evaluates e-rater 2.0's performance by comparing the predicted scores to human judged scores. Woods et al., 2017 approaches it differently and shows experts the results of different feedback algorithms. For each comparison, the experts choose whether they strongly prefer or moderately prefer one of the algorithms. Both evaluation strategies evaluate the feedback system's capability to propose accurate feedback.

Andersen et al., 2013 uses a different setup where students use the system and answer a questionnaire afterward. The questionnaire focuses on system usability and the student's perception of the system's usefulness, usability, clarity, and accuracy. Although a student's perception is important, it does not accurately represent the feedback's accuracy because students are not experts. On the contrary, experts have a better understanding of appropriate and accurate feedback - enabling them to evaluate a feedback system more accurately.

In this evaluation, we focus on the feedback system's accuracy. Feedback accuracy is measurable and relevant since an educational system that makes fewer mistakes is intuitively more educational. We recognize the underlying implicit assumption: accurate feedback systems built on top of educational research translate into positive long-term learning effects. Although this assumption is not validated, there are arguments favoring an accuracy-centered evaluation for feedback systems. First, accuracy centered validation is commonly used to evaluate textual feedback systems (Woods et al., 2017; Zhang and Litman, 2020; Bernius, Krusche, and Bruegge, 2021), and in numerical feedback systems that evaluate with the ASAP dataset[1] (Burstein, 2004; Yang et al., 2020; Uto, Xie, and Ueno, 2020; Mayfield and Black, 2020; Ormerod,

[1]The ASAP dataset is part of a Kaggle competition sponsored by The Hewlett Foundation: `https://www.kaggle.com/c/asap-aes`

Malhotra, and Jafari, 2021). Second, feedback accuracy relates directly to the feedback system and disregards course design, repetitive usage of the system, and other external factors that all influence long-term learning effects. Combining all factors in a single experiment complicates the experiment design and complicates the evaluation of individual factors, and therefore, we narrow it down to feedback accuracy. Most of the work of this thesis is focused on a single factor; the functionality of feedback systems. The educational aspect is important for any functional feedback system, but in this thesis, we use existing educational practices and research to define strict requirements, assumptions, and design goals. Therefore, we argue that this thesis's main contribution concerns the feedback system itself. Consequently, taking the evaluating system's accuracy is reasonable because it concretely evaluates the system's capability to find mistakes in text and propose (pre-defined and changeable) related tips and suggestions. In the next section, we discuss the experiment procedure that evaluates *RevisionCoach*'s feedback accuracy.

6.2 Study

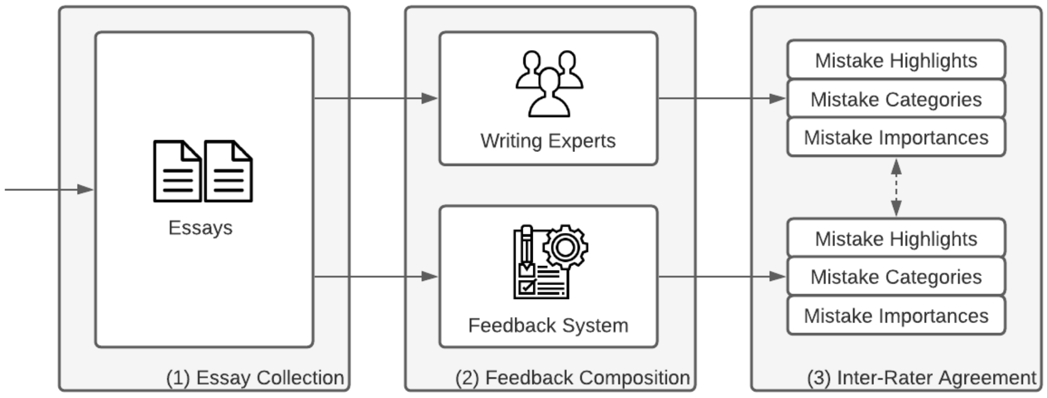


FIGURE 6.1: Experiment setup flow

Since we refrain from classroom experiments and focus on feedback accuracy evaluation, the experiment setup is compact. In Figure 6.1 the experiment flow is visualized. Starting with (1), we select four essays that contain writing mistakes. Subsequently in (2), two writing experts and *RevisionCoach* compose feedback items consisting a sentence-level highlight (to indicate the mistake's location), a mistake category (cohesion, coherence, formality, or readability), and a mistake importance (1-3, where 3 indicates the most important mistake). In the following section we discuss the study biases and the study setup.

6.2.1 Study Biases

For the evaluation it is important to consider biases, and therefore, we start with explaining the three considered experiment biases. Unfortunately, feedback is biased because it depends on the expert's personal opinions and experience.

The first bias concerns the expert's importance scoring procedure because experts can have individual opinions about a mistake's importance. *RevisionCoach* supports feedback differentiation and assigns the mistake importances relative to each other for a single essay, and as a result a major mistake in a bad essay is not the same as the same mistake in a relatively well-written essay. Although mistake importances relative to the entire essay dataset is fairer in terms of official grading, *RevisionCoach*'s goal is not to grade essays but to assist in the learning process, and we expect experts to rate mistake importances relative to the mistakes in the same essay that we encourage by letting the experts evaluate one essay at a time. Although other studies show experts several essay snippets or a more extensive collection of essays, both options introduce scoring biases in this evaluation because it becomes less clear how to score the essay's mistake importances. In addition, the evaluation participants cannot provide scores relative to a large number of essays, because identifying sentence-level mistakes and feedback is time-consuming (the current evaluation took experts between 20 and 60 minutes). In the end, we expect to minimize the mistake scoring bias - that occurs because the mistakes depend upon the mistake's context - by embracing context-dependent scores and by showing evaluation participants a single essay text at a time.

Secondly, by considering more experts, the individual opinions of experts are less impactful for the final evaluation decreasing the individual scoring bias. Using more experts or annotators to improve the experiment's validity is commonly used in practice, for instance in the well-known ASAP AES challenge[2] to construct essay scores.

Last, evaluation participants receive the exact same evaluation instructions and general information of the assessment category. To not introduce a new scoring bias, we do not include information about *RevisionCoach*'s scoring approach. In the following paragraphs, we provide an outline of the study setup.

6.2.2 Study Setup

The experiment uses Google Docs that contains the instructions, relevant information (information about the considered writing assessment categories), allows for the expert's input, and conveniently stores the results in Google Drive. In the evaluation document, the evaluation participant needs to copy-and-paste individual sentences, assign an assessment category, and assign a mistake importance. Because providing sentence-level feedback for student essays is time-consuming, we ask evaluation participants to select at most the top 2 most important mistakes per assessment category. This way, it is possible to include four essays in the evaluation session. For the evaluation we use the publicly available ASAP essays with indices 2085 and 3096 (randomly selected from the essay data, available here[3]) and two essays from a TU Delft experiment that resulted in a dataset of genetically modified organism (GMO) essays. Obviously, we excluded the evaluation essays from the training dataset to train the revision models. And finally, the last step is to process the evaluation data. To this end, we use an automatic feedback parser that automatically loads the google document evaluation results into Python, including the sentence selection, the related assessment categories, and the related severities.

The last challenging step in the evaluation is to compare *RevisionCoach*'s feedback accuracy to other systems. First, we construct a random baseline that iterates through all essay sentences and annotates sentences as 'contains a mistake' with a 50% chance and attaches a random mistake importance. To illustrate the result for 4 assessment categories and 10 essay sentences, the random baseline is expected to select 5 mistakes per assessment category and 20 mistakes in total. Although evaluation participants are tasked to find two sentences for each assessment category,

[2]For more details: `https://www.kaggle.com/c/asap-aes/data`

[3]`https://www.kaggle.com/c/asap-aes/data`

RevisionCoach can find more mistakes. Ideally, the baseline is not restricted in the mistake selection because the top-2 most important feedback can still be identified by consdering the assigned mistake importances.

Furthermore, we asked two students (MSc Computer Science) to do the expert's task to collect a second baseline to compare *RevisionCoach* against. Last, we considered various feedback systems, but unfortunately, it is challenging to find feedback systems that output sentence-level feedback for coherence, cohesion, readability, or formality. In the end, we use Grammarly Premium (30$ per month) and Microsoft Editor (6,99$ per month) in the evaluation - both systems predict formality and readability on sentence-level. Because Grammarly and Microsoft Editor do not supply sentence-level mistake importances, we assign an average fixed importance of 2 to the discovered mistakes. In addition, Microsoft Editor's clarity and conciseness categories are compared to *RevisionCoach*'s readability category and Microsoft Editor's formality to *RevisionCoach*'s formality. For Grammarly, we compare the clarity, tone adjustment, and fluency to readability, and the formality to formality.

6.3 Limitations

To close this chapter, we discuss the limitation of this evaluation. The ideal experiment evaluates the student's learning progress over a longer period to measure the feedback system's long-term impact on a student's learning progress. For instance, Burstein, 2004 evaluates the impact of *Criterion* on student writing performance once a week during the fall 2002 term. A similar experiment setup can evaluate the system's effect on the student's long-term learning effects.

One of the original goals of *RevisionCoach* is to encourage long-term learning effects, and although the ideal experiment evaluates long-term learning effects, it is not an option for this thesis as a result of time and cost restrictions. Long-term effects take time to measure, and it is challenging to continuously measure the effects of the system on a large group of students (Burstein, 2004 is an example of a long-term learning effect evaluation). An obvious solution that addresses the time and cost issue is to consider a single session that targets short-term learning effects of the feedback system. Although a single evaluation session is an option, this setup disregards the original goal to focus on long-term learning effects. On top of that, systems that correct essays directly can have a positive short-term effect on essay quality, but on the contrary, it does not stimulate the student to process mistakes, and it does not encourage self-regulated learning. Therefore, the benefit of a single evaluation session concerning long-term learning effects is minimal. Considering the disadvantages of short-term learning effect evaluation, we focus on the feedback system's accuracy instead. In the following chapter we present *RevisionCoach*'s evaluation results.

Chapter 7

Results

To start, Table 7.1 shows the evaluation essays' statistics of the GMO and ASAP essay datasets. The readability is a number from 0 (extremely difficult to read) to 100+ (very easy to read), and formality varies between 0 and 100% where a higher percentage indicates a more formal text. Furthermore, coherence is a number between 1 and 5 where 5 indicates the most coherent text, and cohesion is a percentage that increases when the text is more cohesive.

	Readability		Coherence		Cohesion		Formality	
Metric	ASAP	GMO	ASAP	GMO	ASAP	GMO	ASAP	GMO
Mean	24.74	50.44	1.24	1.29	0.18	0.20	56.89	66.28
Standard Deviation	10.27	20.90	0.40	0.38	0.03	0.04	6.24	6.60
Minimum	4.30	-19.63	0.34	0.57	0.06	0.03	42.86	50.99
Maximum	51.75	184.45	2.95	2.96	0.26	0.35	97.01	96.52

TABLE 7.1: *RevisionCoach*'s score statistics of all GMO and ASAP task 2 essays. For an equal comparison, we compare 114 GMO essays (all) to 114 ASAP task 2 essays (randomly sampled from 1800 in total).

Apart from the dataset statistics, in this chapter we report the evaluation results in terms of Inter-Rater Reliability (IRR) and mean square error (MSE). The following formulas show the IRR calculation for essays and assessment categories, that represents the agreement of two predictors about the sentence's mistake categories.

$$IRR_{category}(essay, category) = \frac{\sum_{i=0}^{|S_{essay}|} \begin{cases} 1 & \text{if } X_{i,c} = Y_{i,c} \\ 0 & \text{if } X_{i,c} \neq Y_{i,c} \end{cases}}{|S_{essay}|} \tag{7.1}$$

$$IRR_{essay}(essay) = \frac{\sum_{c}^{C} IRR_{category}(essay, c)}{|C|} \tag{7.2}$$

$$IRR = \frac{\sum_{e}^{E} IRR_{essay}(e)}{|E|} \tag{7.3}$$

where:

$|S_{essay}|$ = Number of sentences in the essay

X, Y = Importances for category c, where $X_{i,c} = \begin{cases} 1 & \text{if importance} \in \{1,2,3\} \\ 0 & \text{if importance} = 0 \end{cases}$

C = List of categories: coherence, cohesion, readability and formality

E = List of essays: GMO and ASAP

In contrast to the IRR calculation, the MSE takes the mistake importance into account, and therefore, the following formulas use a different definition for X's and Y's importance predictions:

$$MSE_{category}(essay, category) = \frac{\sum_{i=0}^{|S_{essay}|} (X_{i,c} - Y_{i,c})^2}{|S_{essay}|} \tag{7.4}$$

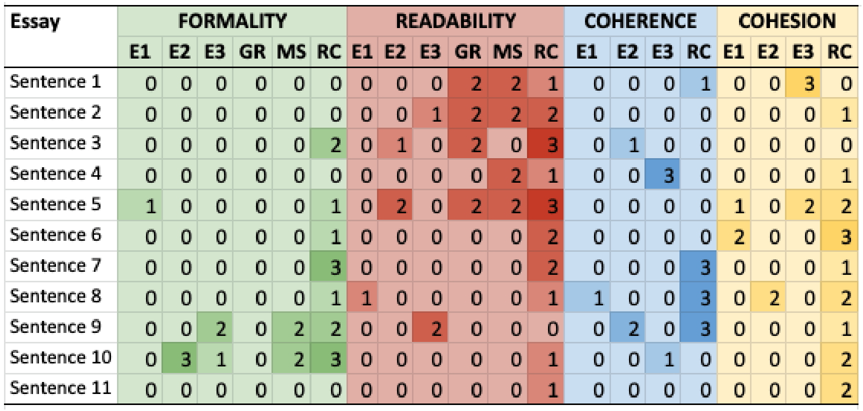
$$MSE_{essay}(essay) = \frac{\sum_{c}^{C} MSE_{category}(essay, c)}{|C|} \tag{7.5}$$

$$MSE = \frac{\sum_{e}^{E} MSE_{essay}(e)}{|E|} \tag{7.6}$$

where:

X, Y = Importances for the category, where X_i =importance

To aggregate the IRR and MSE metrics, we compute the IRR and MSE for each assessment category (Equation 7.1 and Equation 7.4) and take the average for each essay (Equation 7.2 and Equation 7.5). Finally, the average over all essays reflects the aggregated IRR and MSE (Equation 7.3 and Equation 7.6). Although the aggregates provide a summary of *RevisionCoach*'s evaluation results, a single essay's evaluation results clarify the meaning of the importance scores.



Essay	FORMALITY						READABILITY						COHERENCE				COHESION			
	E1	E2	E3	GR	MS	RC	E1	E2	E3	GR	MS	RC	E1	E2	E3	RC	E1	E2	E3	RC
Sentence 1	0	0	0	0	0	0	0	0	0	2	2	1	0	0	0	1	0	0	3	0
Sentence 2	0	0	0	0	0	0	0	0	1	2	2	2	0	0	0	0	0	0	0	1
Sentence 3	0	0	0	0	0	2	0	1	0	2	0	3	0	1	0	0	0	0	0	0
Sentence 4	0	0	0	0	0	0	0	0	0	0	2	1	0	0	3	0	0	0	0	1
Sentence 5	1	0	0	0	0	1	0	2	0	2	2	3	0	0	0	0	1	0	2	2
Sentence 6	0	0	0	0	0	1	0	0	0	0	0	2	0	0	0	0	2	0	0	3
Sentence 7	0	0	0	0	0	3	0	0	0	0	0	2	0	0	0	3	0	0	0	1
Sentence 8	0	0	0	0	0	1	1	0	0	0	0	1	1	0	0	3	0	2	0	2
Sentence 9	0	0	2	0	2	2	0	0	2	0	0	0	0	2	0	3	0	0	0	1
Sentence 10	0	3	1	0	2	3	0	0	0	0	0	1	0	0	1	0	0	0	0	2
Sentence 11	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	2

FIGURE 7.1: Overview of a single essay's (GMO 1) evaluation results. E stands for expert, GR for Grammarly, MS for Microsoft, and RC for *RevisionCoach*. The numbers indicate no mistake (0), minor mistake (1), average mistake (2), and major mistake (3). And finally, the color intensity reflects the number.

In Figure 7.1 we provide an overview of a single essay's evaluation results. The columns in Figure 7.1 represent importance predictions - (referred to by X and Y in the MSE and IRR formulas) - for a certain assessment category (formality, readability, coherence, and cohesion) and predictor (experts, Grammarly, Microsoft Editor, and *RevisionCoach*). For the evaluation, we compare the automated feedback systems to the random and student baseline using the expert evaluation data, where the student baseline's data is from the students' evaluation results. Given mistake predictions A and B, we compare the predictions using the IRR and MSE, where A and B can be a baseline, a system, or an expert. In the comparison, a higher IRR indicates a higher agreement about the writing mistakes, and therefore, a higher IRR is better. On the other hand, a lower MSE indicates a lower difference between the expected and predicted feedback importances, and therefore, a lower MSE is better.

The expert columns in Figure 7.1 contain less importance scores above 0 compared to the system columns because we ask experts to find the top 2 most severe mistakes while the systems select all possible mistakes. To allow for a fair comparison, we consider the top 2 most important mistakes of the systems to compare against the expert importances. As a consequence, the mistake annotations are sparse and the minimum IRR is high - which is illustrated by Equation 7.1 and Figure 7.1 - because the expert and systems select at most 2 out of 11 sentences per assessment category, and as a result, the minimum $IRR_{category}$ is $(11-4)/11 = 63\%$ (when different mistakes are selected $X_{i,c} \neq Y_{i,c}$ for 4 out of 11 sentences). A large portion of the IRR reflects the agreement of sentences that contain no mistake (the rows containing 0's in Figure 7.1), although the evaluation task is to find mistakes and not to find sentences without mistakes. Regardless of a high minimum IRR of 63%, the IRR is comparable between systems to gain insight in which predictions result in the highest agreement about the selected mistakes.

On the other hand, the MSE reflects the error between the expected and predicted mistake importances. For the IRR the difference between 0 (no mistake) and 1 to 3 (any mistake) is the most important, while for MSE the difference between 1 and 3 is relatively more important than between 0 and 1. As a result, MSE focuses on the accuracy of feedback importance predictions.

	Experts (Prof. or PhD)			MSc CS Students	
Essay	Baseline	*RevisionCoach*	Students	Baseline	*RevisionCoach*
all sentences	70%	73%	74%	69%	71%
expert selection	53%	83%	46%	48%	81%

TABLE 7.2: IRR between the random baseline and evaluation participants and between *RevisionCoach* and the evaluation participants. The IRR is calculated for all sentences in the essay and for the evaluation participant's selected highlights.

In addition to considering all sentences in the metric calculation, we calculate MSE and IRR based on the expert sentence selection (Table 7.2). The expert sentence selection includes all sentences that contain a mistake according to the expert. In Figure 7.1 the zeros do not necessarily indicate that the sentence contains no mistake according to the evaluation participants because the participant selects at most two sentences per assessment category. Therefore, we consider the expert sentence selection that directly reflects the participant's assessment of the writing mistakes disregarding the unrated sentences. For the expert sentence selection, we evaluate the system's capability to predict the expert importances, and consequently, we consider all system's predicted importances and not only the top 2 most important mistakes.

		Experts (Prof. or PhD)					MSc CS Students			
Essay	Category	BL	GR	MS	RC	S	BL	GR	MS	RC
ASAP	Coherence	3.59	-	-	2.82	3.28	2.67	-	-	2.62
	Cohesion	2.99	-	-	2.33	3.83	4.01	-	-	0.50
	Formality	1.69	1.75	1.75	0.92	2.17	4.60	6.38	6.38	3.00
	Readability	3.20	3.33	3.00	1.58	2.75	4.75	4.00	5.00	2.38
GMO	Coherence	2.81	-	-	3.17	0.75	3.34	-	-	1.88
	Cohesion	2.57	-	-	1.58	2.83	3.24	-	-	1.50
	Formality	3.08	3.33	1.67	1.50	2.50	3.21	4.75	4.25	2.00
	Readability	1.71	1.67	1.67	1.25	1.92	2.27	2.62	2.62	1.38
	Average	2.70	2.52	2.02	1.89	2.50	3.51	4.44	4.56	1.91

TABLE 7.3: Experiment results of the system's ability to predict expert and student mistake importance assessments. All values are the mean square error (MSE) compared to the expert's or student's importance score for expert or student selected sentences. The MSE is reported for the random baseline (BL), Grammarly (GR), Microsoft Editor (MS), *RevisionCoach* (RC), and student baseline (S).

To provide more insights in the final aggregations of the evaluation results, we include Table 7.3 that contains the MSE scores per category for the expert selection and the final average MSE scores. Grammarly's (GR) and Microsoft Editor's (MS) final scores are listed for readability and formality aggregates because GR and MS predict readability and formality only.

	Expert 1	Expert 2	Expert 3	*RevisionCoach*
Expert 1	100.0%	74.1%	71.1%	80.0%
Expert 2	74.1%	100.0%	68.3%	66.9%
Expert 3	71.1%	68.3%	100.0%	71.8%
RevisionCoach	80.0%	66.9%	71.8%	100.0%

TABLE 7.4: Inter-Rater Reliability (IRR) between experts measured for all essay sentences.

Furthermore, Table 7.4 shows the agreement between experts for all sentences to illustrate the disagreement between experts. Although *RevisionCoach* does not perfectly agree with experts, we do not expect a perfect agreement because experts disagree about sentence-level mistakes.

And finally, we provide results in Table 7.5 related to the revision categories that are used by *RevisionCoach* to construct feedback and to find potential improvements. The rewrite, rephrase, remove, and reword models are the same for the GMO and ASAP dataset, while the generation revision models are trained separately on the GMO and ASAP dataset.

	Rewrite	Rephrase	Remove	Reword
Coherence	74%	19%	4%	4%
Cohesion	26%	18%	44%	12%
Readability	22%	24%	32%	22%
Formality	36%	18%	39%	7%
Average (GMO Essays)	30%	26%	26%	17%
Average (ASAP Essays)	42%	16%	34%	9%
Average	37%	20%	31%	12%

TABLE 7.5: *RevisionCoach*'s revision model usage to construct feedback for the GMO and ASAP evaluation essays.

Chapter 8

Discussion

In this chapter, we provide an analysis of the evaluation results and answers to the research questions. First, a reflection on the explored automated feedback systems.

During this thesis, we learned that direct feedback generation is promising but complicates evaluation and requires a significant amount of data and manual annotations. As a result, direct feedback generation systems are challenging to generalize and use because educational systems require proper evaluation. On the other hand, innovative ways to explain the decision process behind feedback generation increase the usability of feedback generation in an educational setting, because that allows students to reason for themselves about the proposed feedback. Second, we learned that feedback linking (for instance, using peer review data) is a promising direction to take. Apart from the challenge to provide the model with information about the feedback's content, the more challenging issue is that this method heavily relies on manually annotated feedback data. Considering the educational purpose of automated feedback systems, the suggested feedback needs to be educational, and student-level feedback – who are learning the material – is not comparable to expert-level feedback. *RevisionCoach* addresses these issues by generating categorized feedback and removing the need for manual data annotations. The following sections discuss *RevisionCoach*'s evaluation results.

8.1 Interpretation of *RevisionCoach*'s Evaluation Results

In this section, we analyse the results that are presented in chapter 7 that relate to the GMO and ASAP essay datasets. In Table 7.1, we observe that the GMO essays have a better readability, coherence, cohesion, and formality compared to the ASAP essays. Therefore, we expect that experts find it easier to indicate mistakes in the ASAP essays because they contain more mistakes on average. Although the coherence, cohesion, and formality statistics are relatively similar for ASAP and GMO, the readability differs in terms of standard deviation (10.27 against 20.90 for ASAP and GMO) and minimum and maximum values. For this reason, we expect that *RevisionCoach* finds more potential improvements (and feedback) for the readability assessment category. To start the discussion, we consider the evaluation overview of a single essay (Figure 7.1).

Grammarly and Microsoft Editor find fewer formality and readability mistakes than *RevisionCoach* (Figure 7.1), and at the same time, all mistakes that Grammarly and Microsoft Editor find are identified by *RevisionCoach*. Moreover, the expert predictions are similar to the system predictions when we exclude unrated sentences (the zeros in Figure 7.1), while the expert predictions are less similar when you consider *RevisionCoach*'s top-2 most important mistakes and all essay sentences. Because *RevisionCoach*'s top-2 mistakes are different from the expert's annotations, a possible conclusion is that *RevisionCoach* is unable to identify the most important mistakes -

but on the contrary, *RevisionCoach* importance predictions for the expert's mistakes are similar. In this section we investigate this contradiction further using the aggregate evaluation results.

Furthermore, Figure 7.1 is merely an example of a single essay, but it provides the opportunity to visualize the evaluation results and to extract sentences that *RevisionCoach* rates as important mistake and the experts do not. In the following listing we use formality as example (from Figure 7.1):

```
Sentence 7: People do not know the extent of what they are
    eating and we may not know for many years to come. (E1-E3:
     formality importance 0, RevisionCoach: formality
    importance 3)
```

Objectively, sentence 7's formality can be improved because it contains several redundant words. Although experts 2 and 3 can consider the mistake less important than other mistakes because they found the maximum of 2 mistakes, expert 1 annotated only one mistake for the formality category, indicating that the expert did not find a second mistake. A single example does not contribute to a final conclusion; however, combined with the fact that all participants' evaluations took between 20 and 60 minutes, the example illustrates the challenging task of providing sentence-level feedback and importance scores for four assessment categories and four essays.

In contrast to the overview of a single essay, Table 7.2 shows aggregated results to show the difference in agreement for all sentence selection and expert sentence selection. As mentioned in chapter 7, the minimum IRR for all sentences is 63%, and therefore, even the random baseline scores relatively well for all sentences (70%). *RevisionCoach* and the student baseline (73% and 74% respectively) have a higher IRR than the random baseline by a small margin for all sentences, indicating that both *RevisionCoach* (the top-2 most important feedback) and the student baseline have learned to perform better than randomly guessing sentences that require feedback. For all sentences the differences in IRR are small, while for the expert sentence selection the differences are considerable (53%, 83%, and 46% for the random baseline, *RevisionCoach*, and the student baseline), indicating that *RevisionCoach* has a high agreement with the experts about sentences that contain certain mistakes. Moreover, *RevisionCoach* has a higher agreement with the experts than with the students for both sentence selections. The expert selection and all sentence selection IRR results correlate with the previously mentioned contradiction that *RevisionCoach* accurately identifies mistakes (83%) and accurately predicts mistake importances (1.89 MSE) but relatively inaccurately finds the top-2 most important mistakes (73%). To address the contradiction we consider two possibilities: (1) *RevisionCoach* predicts too many mistakes for high-quality sentences and (2) it is difficult for experts to find the top-2 most important mistakes. Although we analyse the second possibility more by comparing the agreement between experts about writing mistakes, we leave the contradiction for future work because possibility (1) requires a secondary time-consuming evaluation session with experts and more data to address. In the following paragraph we discuss Table 7.3 that includes the importance scores in the evaluation results.

In Table 7.3 *RevisionCoach* outperforms the other systems and baselines in the ability to predict a mistake's importance. All systems have a lower MSE than the random baseline indicating that all systems learned to provide more accurate than random feedback. When we compare *RevisionCoach* to deployed writing feedback systems, *RevisionCoach*'s formality and readability MSE is lower than Grammarly's and Microsoft Editor's for all categories and all essays, which indicating that *RevisionCoach* is better capable of predicting expert importance scores than popular education systems for the evaluated essays. Moreover, *RevisionCoach* and Microsoft Editor (1.89 and 2.02 MSE respectively) predict the expert's mistake importances more accurately than the master students' baseline (2.50 MSE). In fact, *RevisionCoach* has a lower MSE than the students' baseline in all categories except for the GMO essays' coherence assessment. The following essay snippet of GMO 2 illustrates why *RevisionCoach*'s coherence predictions of GMO result in a high MSE:

```
5     When you insert a gene into the plant's DNA, new
   proteins will be produced by that plant and the effects
   are still unknown as there are a lack of long-term studies
    into it.
6     The risks can be big.
7     People do not know the extent of what they are eating
   and we may not know for many years to come.
8     For economies it may not be good news either.
9     If we can guarantee perfect crops every time, what is
   it to say that a few corporations won't take control of
   the agriculture market - forcing all other farmers out of
   business?
10    Whether of not we should continue to progress GMO
   research and its commercial use is a pointless question -
   it's already happening whether we want it to or not.
11    The question is, are we ready to deal with the
   consequences?
```

From the above sentences, *RevisionCoach* indicates that sentences 7 to 9 contain major coherence mistakes. Although experts 1 and 2 agree that sentences 8 and 9 contain a (minor) coherence mistake, there is a large error in terms of importance prediction, and as a result, *RevisionCoach*'s coherence MSE for the GMO essays is high. To elaborate upon this difference, we hypothesize that humans intuitively include time in their decision-making process such that when three subsequent sentences are wrong, fixing the sentence in the middle solves the problem automatically. On the other hand, *RevisionCoach* does not consider the impact of potential sentence improvements, and therefore, fixing one of the three sentences is independent of the other sentences' mistakes. In a way, this illustrates the iterative learning process that *RevisionCoach* aims to achieve by allowing students to make gradual improvements to individual sentences, and after the improvement, *RevisionCoach* will consider the improvement in the feedback prediction process. Unfortunately, *RevisionCoach*'s iterative feedback is not included in this evaluation session because it considers a single time step. To solve *RevisionCoach*'s time limitation, the feedback model needs to consider potential improvements of the most significant writing mistakes and to include that information in the mistake importance scores. Predicting potential improvements of sentences can be approximated by *RevisionCoach*'s revision models, but the actual student's writing improvements can be entirely different. At the same time, experts face a similar challenge because experts implicitly assume that improving a single sentence addresses the sentence's context simultaneously, while that depends upon the student's sentence correction. After the correction, the expert's writing assessment (similar to *RevisionCoach*) can change as a result of the student's writing improvements of the previous timestep. Again, this illustrates the time-consuming feedback process to support learning by deliberate practice.

Other than the comparisons of the baselines and systems, we look into the agreement between experts in Table 7.4. Hypothetically, the sentence-level feedback annotation task is based on personal opinions and experience, and therefore, we expect the agreement to be low. However, the experts' agreement is lower than expected (between 68.3% and 74.1%) and *RevisionCoach* has the highest agreement with expert 1 and expert 3 (80.0% and 71.8% respectively) compared to the other experts. Surprisingly, the average agreement between pairs of experts ($(0.742 + 0.711 + 0.683)/3 = 71.2\%$) is lower than the average agreement of *RevisionCoach* with each expert ($(0.800 + 0.669 + 0.718)/3 = 72.9\%$), that supports the argument that *RevisionCoach*'s feedback predictions approximate expert-level feedback predictions. Furthermore, the low IRR between experts indicates that the time-consuming sentence-level annotation task is difficult and inconsistent for experts.

Finally, we look into *RevisionCoach*'s suggested revisions to compare against the related work of Woods et al., 2017 that transforms essay-level scores into sentence-level scores using the remove revision strategy. In contrast to Woods et al., 2017 that only uses remove revisions, *RevisionCoach* uses the rewrite (text generation) revision the most, followed by remove, rephrase, and reword revisions (Table 7.5). Interestingly, coherence mostly uses the rewrite revision (74%), while the other assessment categories rely more on the remove revision (32% to 44%). Moreover, Table 7.5 shows a difference in revision model usage for the GMO and ASAP essays, that shows that the essay dataset's quality influences the revision model usage.

Moreover, we notice that localized (sentence-level) feedback for readability, coherence, formality, and cohesion is a challenging and time-consuming task for experts, while at the same time, it is easier to assign the mistake's severity. In fact, we believe that this analysis benefits *RevisionCoach* and other automated feedback systems, since it illustrates the feedback consistency (*RevisionCoach* is deterministic and the assessment categories follow pre-defined scoring algorithms) and accuracy (low MSE) of automated feedback. Automated feedback systems can assist teachers to localize feedback - apart from an essay-level rubric grade - to individual sentences and to provide more structured feedback for students. In addition, automated feedback systems can point experts to potential writing mistakes that are otherwise undetected to increase the feedback's effectiveness to assist in the student's learning process.

8.2 Answers to the Research Questions

To conclude the evaluation discussion, we refer back to the original research questions:

1. How to support learners with feedback on essays?
2. How can feedback be presented to improve the student's learning process?

For the first question, we assist students to learn by deliberate practice and to encourage self-regulation. Although the evaluation does not address long-term learning effects, *RevisionCoach* considers these educational practices. To start, deliberate practice requires clear learning tasks and goals, to motivate improvement, to provide feedback, and to provide iterative and repeated opportunities for gradual refinement and improvement. Because *RevisionCoach* targets writing skills, the tasks and goals refer to the writing assessment categories: cohesion, coherence, readability and formality. Students can pick any category to focus on, and additionally, *RevisionCoach* motivates students to directly improve individual sentences because it immediately shows the score improvement for an assessment category (in *RevisionCoach*'s prototype UI). Apart from the score improvement, *RevisionCoach* shows feedback in the form of a mistake highlight, an assessment category, a correction category (reword, rephrase or rewrite), and a correction suggestion. In addition, *RevisionCoach* can show pre-defined formative feedback that is linked to the assessment and correction category. And finally, *RevisionCoach* allows students to improve the essay iteratively and to incorporate gradual refinements and improvements on sentence-level. Second, self-regulated learning is encouraged because students set their own learning goals that relate to the acceptable writing assessment category scores. *RevisionCoach*'s feedback assists students to learn and to motivate themselves to write high-quality text.

Furthermore, the second research question relates to differentiated learning, to support novice to advanced learners with appropriate feedback. *RevisionCoach* can present four layers of feedback and can provide additional pre-defined tips and hints that relate to the feedback. In addition, *RevisionCoach* has a notion of mistake importance, and therefore, *RevisionCoach* can direct students to the essay's most concerning writing mistakes. Although *RevisionCoach*'s prototype UI illustrates how *RevisionCoach*'s feedback can be presented, the user study is left for future work.

Chapter 9

Conclusion

In this thesis, we propose the automated writing feedback system *RevisionCoach* that follows the writing process intuition to iteratively assist students to improve the essay's readability, coherence, cohesion, and formality. *RevisionCoach* follows educational theory by assisting students with self-regulated learning by deliberate practice (iterative learning, repeated opportunities for gradual improvement, clear tasks, and localized feedback) and to provide differentiated learning (different levels of feedback to support novice to advanced learners). At the same time, *RevisionCoach* is designed to follow the strict requirement to reduce the reliability of manual data annotation, and as a result, *RevisionCoach* requires only relevant essays (or texts) as a dataset. Furthermore, we present a *RevisionCoach* user interface prototype that encourages students to think about writing mistakes. And finally, *RevisionCoach* is designed to not construct open-ended feedback in order to prevent common natural text generation evaluation difficulties and to facilitate effective feedback evaluation. In the evaluation, experts and students are tasked to construct feedback similar to *RevisionCoach*'s feedback by indicating a highlight, a writing assessment category, and a mistake importance. In the end, the experts' and students' feedback is compared to *RevisionCoach*'s predicted feedback to evaluate *RevisionCoach*'s ability to construct accurate feedback.

To summarize *RevisionCoach*'s evaluation results, we observe that *RevisionCoach*'s overall ability to identify writing mistake importances is more accurate (1.89 MSE) than the random baseline (2.70 MSE), student baseline (2.50 MSE), Grammarly (2.52 MSE), and Microsoft Editor (2.02 MSE). Although the results are promising, the student baseline has a lower MSE for the coherence assessment category for the GMO essays and Grammarly and Microsoft Editor do not supply importance scores – a fixed importance of 2 is used for all identified mistakes – and Grammarly and Microsoft Editor do not support coherence and cohesion feedback. Furthermore, we discovered the contradicting result that *RevisionCoach* accurately identifies mistakes (83% IRR) and accurately predicts mistake importances (1.89 MSE) but relatively inaccurately finds the top-2 most serious mistakes (73% IRR against the random baseline's 63% IRR). The first explanation is that *RevisionCoach* mistakenly identifies mistakes in high-quality sentences, but unfortunately, this requires a secondary time-consuming evaluation session with experts and more data to address, and therefore, we leave it for future work. An alternative explanation addresses the difficult challenge for experts to identify the top-2 most important mistakes for four assessment categories and four essays, that is supported by the relatively low agreement between experts (71.2% IRR on average) compared to the agreement of *RevisionCoach* and the experts (72.9% on average). To conclude, *RevisionCoach*'s feedback importance predictions are more accurate than the random baseline, student baseline, Grammarly, and Microsoft Editor, and *RevisionCoach*'s agreement with experts about writing mistakes is higher than the agreement between the experts, indicating that *RevisionCoach* is capable of accurate feedback importance predictions and that *RevisionCoach* approximates expert-level mistake identification.

To address the research questions (how to support learners with feedback on essays and how can feedback be presented to improve the student's learning process?), we use learning by deliberate practice, self-regulated learning, and differentiated learning to encourage effective learning, to prepare students for learning outside the university, and to assist novice to advanced learners to learn how to write. *RevisionCoach* follows the differentiated learning by offering several layers of feedback (a mistake highlight, an assessment category, a correction category, and a correction suggestion). Furthermore, *RevisionCoach* follows learning by deliberate practice and self-regulated learning because *RevisionCoach* provides feedback for pre-defined writing assessment categories – coherence, cohesion, readability, and formality – to allow students to determine clear learning tasks and goals, to motivate improvement, and to allow students to incorporate iterative and gradual improvements.

Throughout this thesis project, we learned that open-ended feedback evaluation is challenging to evaluate and that feedback assessment categories are important. Although open-ended feedback is promising in terms of content-related feedback, it lacks feedback structure, and therefore, it is challenging to evaluate. Furthermore, we learned that formative and localized feedback is difficult to construct for experts and that automated systems show promising results to support the expert by providing consistent and high-quality feedback. Although educational systems' feedback accuracy is relatively achievable to evaluate, educational systems' long-term learning effects are challenging to evaluate. Moreover, manually annotated educational datasets are challenging to find or to construct, which drives the need for innovative methods and models to construct localized feedback with as little as possible manual data annotation. Throughout the thesis project, it was challenging to combine educational theory with computer science in order to construct an effective educational system. Apart from AES and AWE models, we recognize that any automated educational system needs to adhere to at least one educational principle to encourage long-term learning effects combined with a focus on high feedback accuracy. Ideally, any deployed automated feedback system needs a study about the long-term learning effects to ensure the system's effectiveness to stimulate positive learning effects.

Chapter 10

Future Work

RevisionCoach is based upon the requirement that strictly limits the usage of manual annotations, which provides opportunities for data augmentation of large datasets. In principle, we suggest a hybrid approach of *RevisionCoach* and direct feedback generation, where *RevisionCoach* is used to augment the dataset and to structure the feedback. Given a large feedback dataset with sentence-level highlights and attached feedback (peer reviews, expert feedback, or both), *RevisionCoach* can categorize all sentence-level highlights. As a result, *RevisionCoach* categorizes the feedback in readability, formality, coherence, cohesion, and no category. A potential next step is to train a transformer model with the following input sequence for training:

```
fb before: [sentences before]  highlight: [mistake highlight
    ]  after:  [sentences after]  category: [category]
    feedback: <extra_id_0>
```

After fine-tuning the model, the following input sequence to generate feedback and a category:

```
fb before: [sentences before]  highlight: [highlight]
    after:  [sentences after] feedback:
```

As a result, the model generates a feedback text and a corresponding category and has access to more information about the highlight's mistake. *RevisionCoach* can be used to double-check the assessment category for additional verification. Of course, other model structures are possible, where separate models are used to predict feedback text and mistake categories. Furthermore, more scoring models can be included to have more assessment categories.

In contrast to generating open-ended feedback that is challenging to evaluate, categorized feedback can be evaluated against *RevisionCoach*, and expert evaluation becomes easier because experts know what assessment category the feedback addresses. Second, for students it is easier to determine the feedback usefulness because the feedback text should relate to the assessment category.

Moreover, *RevisionCoach*'s scoring and revision models are extensible and left for future work. More advanced scoring models require additional data annotation but potentially increase *RevisionCoach*'s ability to assess the essay's quality in certain assessment categories. On the other hand, the text generation revision model can be improved in terms of training data and preprocessing to generate better texts. In the current essay sentence generation setup, the transformer is not informed about the essay's quality and related information directly, and it is interesting to evaluate the potential revision texts with different preprocessing methods. In addition, an intermediate training step to teach the transformer about high-quality texts - for example, scientific papers and other articles - can help the generated revision's quality. The ideal revision model works for all essay types and topics, and transformers are capable to train and learn from a large variety of texts.

Last, *RevisionCoach* needs additional expert evaluation sessions to evaluate the feedback accuracy with a higher certainty because more expert evaluations decrease the impact of the expert's individual opinions. As a result, the comparison between the average agreement between experts and the average agreement between experts and *RevisionCoach* is more reliable. In addition, evaluating more essays and more sentences provides better insights in the feedback importance prediction's recall and precision. Furthermore, *RevisionCoach* has an educational focus and requires multiple evaluation sessions with students to evaluate the long-term learning effects on the students' writing skills.

Appendix A

Data Exploration

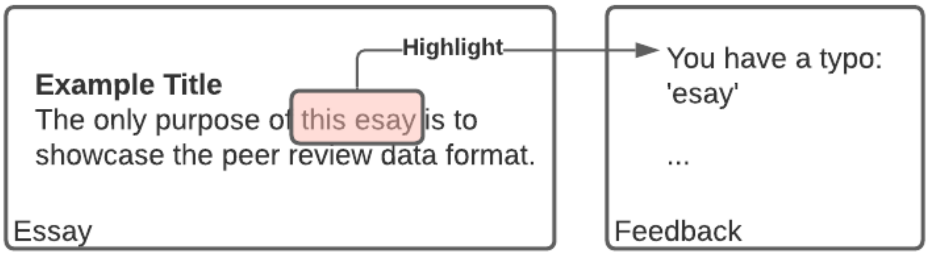

Highlight
Example Title
The only purpose of this esay is to showcase the peer review data format.
Essay
You have a typo: 'esay'
...
Feedback

FIGURE A.1: Highlight and feedback definitions

During the data exploration, we focus on a single appropriate large peer review dataset to narrow down the data exploration. Most of FeedbackFruit's assignments are similarly structured but vary in size, where each peer review item contains a highlight, a linked feedback, and an entire essay (as visualized in Figure A.1).

To narrow down the data exploration further, we select a single assignment from the peer review dataset. As a result, it becomes intuitively easier to find useful relationships and common writing mistakes when the essays are similar in content. In order to find the dataset's assignment, we follow the following steps:

1. Filter on assignments outside of Europe to avoid GDPR restrictions.
2. Filter on the number of submissions and on the number of feedback annotations. Each submission is a single essay submitted by a student.
3. Filter on essay assignment descriptions. Some assignments result in widely varying essays that make it difficult to find meaningful relationships and common mistakes. We assume that the essays cover the same main topic.

Based on the above steps, we selected an assignment that tasks students to write an opinion piece about a news article. We refer to this assignment as $ASSIGNMENT_{COVID}$ since the news article is about COVID cases in the UK. An overview of the data is in Table A.1.

The next step is to preprocess the data to clean, filter and remove noise. Afterward, we explore the processed data. Preprocessing is discussed in detail in the following section.

Submitted essays	419
Number of students	419
Feedback items	2366
Feedback criterion	Writing, General Comments, Engaging, Overall Assessment, Persuasive
Annotation types	Critique: 1030, compliment: 845, not specified: 491

TABLE A.1: Data statistics of $ASSIGNMENT_{COVID}$.

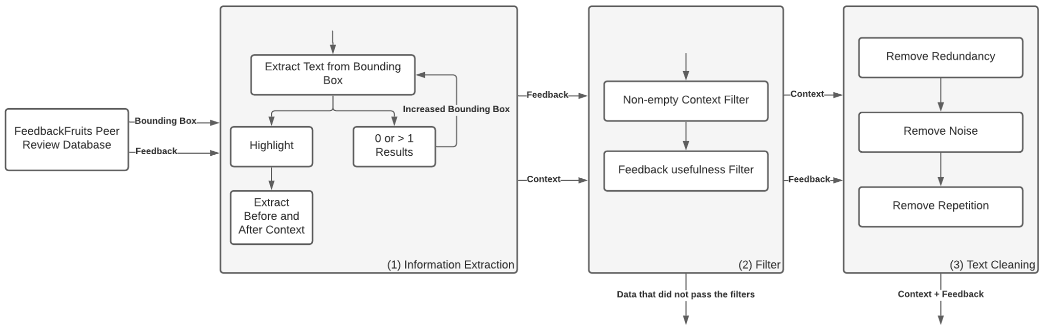


FIGURE A.2: Preprocessing steps for feedback data. *Context* consists of a highlight, before context, and after context. The bounding boxes in the dataset are the coordinates of the highlight in the essay PDF.

A.1 Dataset Preprocessing

The goal of data preprocessing is to improve the quality of the dataset and to remove noise. The noise is caused primarily by students that select incomplete highlights or that write nonsense as feedback. To address the quality and noise, the data preprocessing stage entails three steps (as visualized in Figure A.2): (1) extract missing highlights, (2) remove non-informative feedback, and (3), clean feedback items. We discuss each step individually.

Figure A.2 step (1) addresses the highlight text extraction from the $ASSIGNMENT_{COVID}$ dataset. Since highlights are frequently incomplete and depend on the sentences before and after, we include the highlight's context up to a fixed number of words. $ASSIGNMENT_{COVID}$ uses the FeedbackFruit's feedback structure, where highlights are represented by coordinate bounding boxes that indicate the highlight's position in the essay PDF. We use an iterative algorithm to extract the context - a highlight, words before, and words after - as described in algorithm 1.

Algorithm 1: Context extraction based on a coordinates bounding box

```
Input: $bounding\_box$, $essay$;
Output: $context$ or $None$;
while $bounding\_box.width < essay.width$ do
    $text \leftarrow$ extract_pdf_text($bounding\_box$);
    $matches \leftarrow$ search_matches($essay$, $text$);
    if $matches = 1$ then
        return extract_context($match$, $constants.MAX\_CONTEXT\_WORDS$);
    else
        $bounding\_box.width * 5\%$;
    end
end
```

The second step (2 in Figure A.2) removes uninformative feedback which requires some rules to separate ’useful’ from ’not useful’ feedback. The following feedback is considered to be not useful and is removed using a rule-based filtering module:

1. Grammatical or spelling related feedback
2. Feedback containing quotes because quotes contain original essay text which is not real ‘feedback’ text
3. Feedback targeting specific parts of the text, for instance "the introduction needs some work". This would require advanced document parsing to predict the function of pieces of text.
4. Feedback containing less than 3 words

The last preprocessing step (3 in Figure A.2) cleans text to reduce the vocabulary size in the essays and feedback. This process reduces the search space for any model. We use a limited number of simple cleaning rules to avoid unnecessary cleaning that results in information loss. The rules remove symbol repetitions (for instance ‘!!!!!!’ becomes ‘!’), remove redundant whitespaces, remove unicode characters, map percentage styles (‘23 %’ and variations to ‘23%’), map prices (‘$ 100‘ and variations to ‘100$’), map numbers (‘1000.0’ and variations ‘1,000.0’), and map similar symbols (all quote variations to ‘"’).

A.2 Exploration

After the assignment selection and preprocessing, we explore the data with text similarity methods. Figure A.3 shows a 2D visualization of the semantic similarity (a high semantic similarity means that the texts’ meaning is similar) between feedback texts, using the text embedding model “RoBERTa". A text embedding is a numerical representation of text. We cluster the embeddings in two steps: (1) cluster the embedding vectors and (2) reduce the dimensions to visualize the embeddings in 2D. Embeddings that are close have a higher semantic similarity.

For the first step, the challenge is to find a metric that evaluates the clusters because the cluster label for each item is unknown (this is called unsupervised clustering). When the cluster labels are known, it is possible to calculate accuracy, precision, and related metrics for each clustering method. On the other hand, we need

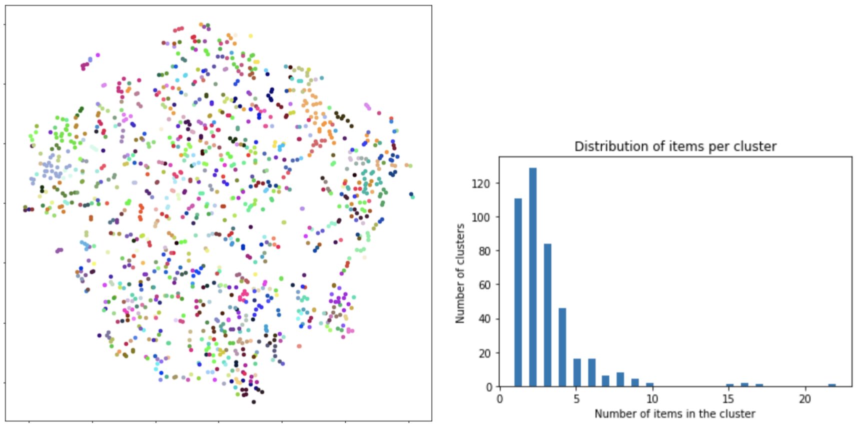


FIGURE A.3: Peer feedback text clustered on semantic similarity and the number of items per cluster. The colors in the left graph represent 427 clusters.

other ways to measure the clustering methods' performances for unsupervised clustering. Liu et al., 2010 indicates various metrics to measure the clusters' quality, for example, by measuring the compactness of clusters and separation between clusters. Liu et al., 2010 finds that the S_Dbw index performs best when considering the impact of well-separated clusters, the impact of noise, the impact of various cluster densities, the impact of nearby clusters, and the impact of unequal cluster sizes. On the other hand, Yanchi Liu et al., 2013 finds that S_Dbw (and all other metrics evaluated by Liu et al., 2010) has difficulty addressing the impact of various cluster shapes, while CVNN does not in certain circumstances (Yanchi Liu et al., 2013). Although CVNN is promising, it is based on nearest neighbours and it is required to specify *k* (the number of nearest neighbours to consider), which is challenging to determine. On the other hand, S_Dbw is promising, but it is computationally expensive to compute. Therefore, we choose the Silhouette (similar in performance compared to S_Dbw excluding the impact of nearby clusters (Liu et al., 2010)) and Calinski-Harabasz indices (similar in performance compared to S_Dbw including nearby clusters), to evaluate cluster quality. We assume that the number of clusters is between 256 and 1024 for 1229 items, to force the grid search to find more variation in text similarity but not assign each item to an individual cluster. Table A.2 shows the top 5 results of the result of the grid search, that tries to maximize the Silhouette and the Calinski-Harabasz indices. The first row performs best overall.

The challenge of the second step is to represent the computed embeddings and clusters in a single 2D graph. Since the resulting embeddings have 512 dimensions it is hard to visualize. To solve that we use PCA to reduce the dimensions to 30 and t-SNE to reduce the dimensions to two, in order to visualize the clusters in 2D (visualization in Figure A.3). This approach is proposed by Maaten and Hinton, 2008 to visualize high-dimensional data effectively.

Figure A.3 clearly shows that clusters are recognized and illustrates that the feedback texts share similar content. Manual experimentation shows that the feedback texts in clusters share similarities, for example, the feedback texts target persuasiveness, use of evidence, or engagement. It requires significant manual effort to investigate the clusters in detail, which we leave for future work.

Clusters	S_cos	S_eucl	C	Parameters	Model
427	0.0880	**0.0396**	**3.0709**	{'affinity': 'cosine', 'linkage': 'complete', 'distance_threshold': 0.6}	**agg**
287	0.0529	0.0171	3.0382	{'affinity': 'cosine', 'linkage': 'average', 'n_clusters': None, 'distance_threshold': 0.6}	agg
938	0.0721	0.0385	2.8769	{'affinity': 'cosine', 'linkage': 'complete', 'n_clusters': None, 'distance_threshold': 0.4}	agg
676	**0.0950**	0.0464	2.8149	{'affinity': 'cosine', 'linkage': 'complete', 'n_clusters': None, 'distance_threshold': 0.5}	agg
912	0.0686	0.0358	2.7941	{'affinity': 'cosine', 'linkage': 'average', 'n_clusters': None, 'distance_threshold': 0.4}	agg

TABLE A.2: Top 5 gridsearch results of clustering algorithms AgglomerativeClustering, DBSCAN, MeanShift, KMeans for text embeddings clustering. Agg is AgglomerativeClustering, S_cos is the Silhouette Cosine index, S_eucl is the Silhouette Euclidean index, and C is the Calinski-Harabasz index. Values in bold show the most promising results per column.

Appendix B

T5 Text Generation Fine-Tuning Predictions

This appendix contains T5's sentence predictions for the text generation revision model. The inputs are the sentences from the ASAP essay that is used in the evaluation and the fine-tuning essay dataset does not contain this essay.

```
Input:
 > Today I am going to talk about <extra_id_0> Should they be
     removed from the shelf if they are found offensive?
Expected:
 > certain materials , such as books , music , movies , and
    magazines .
T5 Base Prediction
 > offensive books. offensive books. offensive books.
    offensive books. offensive books? offensive books. Should
     they be removed from the shelf if they are found
    offensive?
T5 Fine-Tuned Half Masked Prediction
 > the books that should be removed from the shelf if they
    are found offensive.
T5 Fine-Tuned Fully Masked Prediction
 > I believe that certain materials should not be removed
    from the shelves if they are found offensive.
_____________________________________
Input:
 > Today I am going to talk about certain materials, such as
    books, music, movies, and magazines. Should they be
    removed <extra_id_0> In todays world it is hard to please
     every living soul.
Expected:
 > from the shelf if they are found offensive ?
T5 Base Prediction
 > ??? Should they be removed??? Should they be removed??.
T5 Fine-Tuned Half Masked Prediction
 > or should they be removed from shelves?
T5 Fine-Tuned Fully Masked Prediction
 > I am going to talk about how people in this world would
    feel if they were able to take these materials off the
    shelf.

_____________________________________
```

```
Input:
 > Should they be removed from the shelf if they are found
    offensive? In todays world it <extra_id_0> Every one is
    going to disagree about something in there life time.
Expected:
 > is hard to please every living soul .
T5 Base Prediction
 > is a constant debate. is inevitable. is inevitable. is
    inevitable. is inevitable. is inevitable. It is
    inevitable.
T5 Fine-Tuned Half Masked Prediction
 > isn't just any one thing that doesn't agree with.
T5 Fine-Tuned Fully Masked Prediction
 > Yes i think they should be removed from the shelf if they
    are found offensive.
__________________________________________
Input:
 > In todays world it is hard to please every living soul.
    Every one is going <extra_id_0> Life as we know it,
    everybody lives it a different way, and in a different
    place, but everybody has read a book, listened to music,
    and even read a magazine a time or to.
Expected:
 > to disagree about something in there life time .
T5 Base Prediction
 > through through through through through through a
    different way. Everyone has read a book, listened to
    music, and even listened to a magazine. Everybody is
    going through Life. Everybody is going through Life.
    Everybody is going through Life. Everybody is going
    through Life. Everybody is going through Life. Everybody
    is going through
T5 Fine-Tuned Half Masked Prediction
 > to feel the same way we do.
T5 Fine-Tuned Fully Masked Prediction
 > I cannot fully grasp the concept in my head of hiding
    materials from the public.
```

Bibliography

Anaby-Tavor, Ateret et al. (Nov. 2019). "Not Enough Data? Deep Learning to the Rescue!" In: *arXiv:1911.03118 [cs]*. arXiv: 1911.03118. URL: http://arxiv.org/abs/1911.03118 (visited on 02/02/2021).

Anders Ericsson, K. (Nov. 2008). "Deliberate Practice and Acquisition of Expert Performance: A General Overview". In: *Academic Emergency Medicine* 15.11, pp. 988–994. ISSN: 10696563, 15532712. DOI: 10.1111/j.1553-2712.2008.00227.x. URL: https://onlinelibrary.wiley.com/doi/10.1111/j.1553-2712.2008.00227.x (visited on 08/09/2021).

Andersen, Øistein E et al. (2013). "Developing and testing a self-assessment and tutoring system". In: *Proceedings of the eighth workshop on innovative use of NLP for building educational applications*, pp. 32–41.

Aroyehun, Segun Taofeek and Alexander Gelbukh (2018). "Aggression Detection in Social Media: Using Deep Neural Networks, Data Augmentation, and Pseudo Labeling". In: *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pp. 90–97.

Bernius, Jan Philip and Bernd Bruegge (2019). "Toward the Automatic Assessment of Text Exercises". In: *Software Engineering (Workshops)*. Software Engineering (Workshops), pp. 19–22.

Bernius, Jan Philip, Stephan Krusche, and Bernd Bruegge (June 2021). "A Machine Learning Approach for Suggesting Feedback in Textual Exercises in Large Courses". In: *Proceedings of the Eighth ACM Conference on Learning @ Scale*. Virtual Event Germany: ACM, pp. 173–182. ISBN: 978-1-4503-8215-1. DOI: 10.1145/3430895.3460135. URL: https://dl.acm.org/doi/10.1145/3430895.3460135 (visited on 06/24/2021).

Boud, David and Nancy Falchikov (2007). *Rethinking assessment in higher education: learning for the longer term*. OCLC: 252861992. London: Routledge. ISBN: 9780203964309 9786610738823 9781134152100 9781134152148 9781134152155 9781280738821. URL: http://www.dawsonera.com/depp/reader/protected/external/AbstractView/S9780203964309 (visited on 03/09/2021).

Burstein, Jill (2004). "Automated Essay Evaluation: The Criterion Online Writing Service". In: *Ai magazine* 25.3, pp. 27–27.

Campos, Ricardo et al. (Jan. 2020). "YAKE! Keyword extraction from single documents using multiple local features". In: *Information Sciences* 509, pp. 257–289. ISSN: 00200255. DOI: 10.1016/j.ins.2019.09.013. URL: https://linkinghub.elsevier.com/retrieve/pii/S0020025519308588 (visited on 03/16/2021).

Celikyilmaz, Asli, Elizabeth Clark, and Jianfeng Gao (June 2020). "Evaluation of Text Generation: A Survey". In: *arXiv:2006.14799 [cs]*. arXiv: 2006.14799. URL: http://arxiv.org/abs/2006.14799 (visited on 01/26/2021).

Chen, Hongbo and Ben He (2013). "Automated Essay Scoring by Maximizing Human-Machine Agreement". In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1741–1752.

Chen, Yen-Yu et al. (2010). "An Unsupervised Automated Essay- Scoring System". In: *Natural Language Processing*, p. 7.

Cummins, Ronan, Meng Zhang, and Ted Briscoe (2016). “Constrained Multi-Task Learning for Automated Essay Scoring”. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, pp. 789–799. DOI: 10.18653/v1/P16-1075. URL: http://aclweb.org/anthology/P16-1075 (visited on 12/16/2020).

Devlin, Jacob et al. (May 2019). “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *arXiv:1810.04805 [cs]*. arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805 (visited on 01/08/2021).

Dikli, Semire (2016). “An Overview of Automated Scoring of Essays”. In: *The Journal of Technology, Learning and Assessment* 5.1.

Dong, Fei and Yue Zhang (2016). “Automatic Features for Essay Scoring – An Empirical Study”. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, pp. 1072–1077. DOI: 10.18653/v1/D16-1115. URL: http://aclweb.org/anthology/D16-1115 (visited on 12/16/2020).

Dong, Fei, Yue Zhang, and Jie Yang (2017). “Attention-based Recurrent Convolutional Neural Network for Automatic Essay Scoring”. In: *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*. Vancouver, Canada: Association for Computational Linguistics, pp. 153–162. DOI: 10.18653/v1/K17-1017. URL: http://aclweb.org/anthology/K17-1017 (visited on 12/14/2020).

Elsner, Micha, Joseph Austerweil, and Eugene Charniak (2007). “A unified local and global model for discourse coherence”. In: *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pp. 436–443.

Ethayarajh, Kawin (Sept. 2019). “How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings”. In: *arXiv:1909.00512 [cs]*. arXiv: 1909.00512. URL: http://arxiv.org/abs/1909.00512 (visited on 07/01/2021).

Farra, Noura, Swapna Somasundaran, and Jill Burstein (2015). “Scoring Persuasive Essays Using Opinions and their Targets”. In: *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*. Denver, Colorado: Association for Computational Linguistics, pp. 64–74. DOI: 10.3115/v1/W15-0608. URL: http://aclweb.org/anthology/W15-0608 (visited on 12/17/2020).

Foltz, Peter W, Darrell Laham, and Thomas K Landauer (1999). “The Intelligent Essay Assessor: Applications to Educational Technology”. In: *Interactive Multimedia Electronic Journal of Computer-Enhanced Learning*, pp. 939–944.

Gao, Silin et al. (2020). “Paraphrase Augmented Task-Oriented Dialog Generation”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, pp. 639–649. DOI: 10.18653/v1/2020.acl-main.60. URL: https://www.aclweb.org/anthology/2020.acl-main.60 (visited on 02/03/2021).

Ghufron, Muhammad Ali and Fathia Rosyida (Dec. 2018). “The Role of Grammarly in Assessing English as a Foreign Language (EFL) Writing”. In: *Lingua Cultura* 12.4, p. 395. ISSN: 2460-710X, 1978-8118. DOI: 10.21512/lc.v12i4.4582. URL: https://journal.binus.ac.id/index.php/Lingua/article/view/4582 (visited on 07/01/2021).

Gibbs, Graham and Claire Simpson (2005). “Conditions Under Which Assessment Supports Students’ Learning”. In: *Learning and teaching in higher education*, pp. 3–31.

Graesser, Arthur C et al. (2004). “Coh-Metrix: Analysis of text on cohesion and language”. In: *Behavior research methods, instruments, \& computers* 32.2, pp. 193–202.

Graham, Steve and Dolores Perin (2007). “A meta-analysis of writing instruction for adolescent students.” In: *Journal of Educational Psychology* 99.3, pp. 445–476. ISSN: 0022-0663. DOI: 10.1037/0022-0663.99.3.445. URL: http://doi.apa.org/getdoi.cfm?doi=10.1037/0022-0663.99.3.445 (visited on 07/01/2021).

Graves, Alex (Nov. 2012). “Sequence Transduction with Recurrent Neural Networks”. In: *arXiv:1211.3711 [cs, stat]*. arXiv: 1211.3711. URL: http://arxiv.org/abs/1211.3711 (visited on 03/10/2021).

Guénette, Danielle (Mar. 2007). “Is feedback pedagogically correct?” In: *Journal of Second Language Writing* 16.1, pp. 40–53. ISSN: 10603743. DOI: 10.1016/j.jslw.2007.01.001. URL: https://linkinghub.elsevier.com/retrieve/pii/S1060374307000021 (visited on 05/07/2021).

Hattie, John and Helen Timperley (Mar. 2007). “The Power of Feedback”. In: *Review of Educational Research* 77.1, pp. 81–112. ISSN: 0034-6543, 1935-1046. DOI: 10.3102/003465430298487. URL: http://journals.sagepub.com/doi/10.3102/003465430298487 (visited on 06/16/2021).

Hellman, Scott et al. (2020). “Multiple Instance Learning for Content Feedback Localization without Annotation”. In: *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*. Seattle, WA, USA → Online: Association for Computational Linguistics, pp. 30–40. DOI: 10.18653/v1/2020.bea-1.3. URL: https://www.aclweb.org/anthology/2020.bea-1.3 (visited on 12/04/2020).

Heylighen, Francis and Jean-Marc Dewaele (1999). “Formality of Language: definition, measurement and behavioral determinants”. In: *Interner Bericht, Center “Leo Apostel”, Vrije Universiteit Brussel*, p. 38.

Higgins, Derrick et al. (2004). “Evaluating Multiple Aspects of Coherence in Student Essays”. In: *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pp. 185–192.

Ibarra-Sáiz, María Soledad, Gregorio Rodríguez-Gómez, and David Boud (July 2020). “Developing student competence through peer assessment: the role of feedback, self-regulation and evaluative judgement”. In: *Higher Education* 80.1, pp. 137–156. ISSN: 0018-1560, 1573-174X. DOI: 10.1007/s10734-019-00469-2. URL: http://link.springer.com/10.1007/s10734-019-00469-2 (visited on 03/09/2021).

Jin, Cancan et al. (2018). “TDNN: A Two-stage Deep Neural Network for Prompt-independent Automated Essay Scoring”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, pp. 1088–1097. DOI: 10.18653/v1/P18-1100. URL: http://aclweb.org/anthology/P18-1100 (visited on 12/17/2020).

Jozefowicz, Rafal et al. (Feb. 2016). “Exploring the Limits of Language Modeling”. In: *arXiv:1602.02410 [cs]*. arXiv: 1602.02410. URL: http://arxiv.org/abs/1602.02410 (visited on 07/01/2021).

Ke, Zixuan and Vincent Ng (2019). “Automated Essay Scoring: A Survey of the State of the Art”. In: *IJCAI* 19, pp. 1–9.

Kennedy, Declan, Áine Hyland, and Norma Ryan (2006). “Writing and Using Learning Outcomes: a Practical Guide”. In: *University College Cork*, p. 30.

Kirschner, Paul A. and Carl Hendrick (Feb. 2020). *How Learning Happens: Seminal Works in Educational Psychology and What They Mean in Practice*. 1st ed. Abingdon, Oxon ; New York : Routledge, 2020.: Routledge. ISBN: 978-0-429-06152-3.

DOI: 10.4324/9780429061523. URL: https://www.taylorfrancis.com/books/9780429591891 (visited on 06/17/2021).

Knight, S. et al. (June 2020). "AcaWriter: A Learning Analytics Tool for Formative Feedback on Academic Writing". In: *Journal of Writing Research* 12.vol. 12 issue 1, pp. 141–186. ISSN: 2030-1006, 2294-3307. DOI: 10.17239/jowr-2020.12.01.06. URL: https://www.jowr.org/abstracts/vol12_1/Knight_et_al_2020_12_1_abstract.html (visited on 11/10/2020).

Krishna, Kalpesh, John Wieting, and Mohit Iyyer (2020). "Reformulating Unsupervised Style Transfer as Paraphrase Generation". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, pp. 737–762. DOI: 10.18653/v1/2020.emnlp-main.55. URL: https://www.aclweb.org/anthology/2020.emnlp-main.55 (visited on 02/03/2021).

Landauer, Thomas K, Peter W. Foltz, and Darrell Laham (Jan. 1998). "An introduction to latent semantic analysis". In: *Discourse Processes* 25.2-3, pp. 259–284. ISSN: 0163-853X, 1532-6950. DOI: 10.1080/01638539809545028. URL: http://www.tandfonline.com/doi/abs/10.1080/01638539809545028 (visited on 07/01/2021).

Lapata, Mirella and Regina Barzilay (2005). "Automatic Evaluation of Text Coherence: Models and Representations". In: *IJCAI*, p. 6.

Li, Xia et al. (2018). "Coherence-Based Automated Essay Scoring Using Self-attention". In: *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*. Vol. 11221. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 386–397. ISBN: 978-3-030-01715-6 978-3-030-01716-3. DOI: 10.1007/978-3-030-01716-3_32. URL: http://link.springer.com/10.1007/978-3-030-01716-3_32 (visited on 05/05/2021).

Liu, Ming et al. (2016). "Automated Essay Feedback Generation and Its Impact in the Revision". In: *IEEE Transactions on Learning Technologies*, p. 13.

Liu, Yanchi et al. (Dec. 2010). "Understanding of Internal Clustering Validation Measures". In: *2010 IEEE International Conference on Data Mining*. Sydney, Australia: IEEE, pp. 911–916. ISBN: 978-1-4244-9131-5. DOI: 10.1109/ICDM.2010.35. URL: http://ieeexplore.ieee.org/document/5694060/ (visited on 12/15/2020).

Lonsdale, Deryle and Diane Strong-Krause (2003). "Automated rating of ESL essays". In: *Proceedings of the HLT-NAACL 03 workshop on Building educational applications using natural language processing -*. Vol. 2. Not Known: Association for Computational Linguistics, pp. 61–67. DOI: 10.3115/1118894.1118903. URL: http://portal.acm.org/citation.cfm?doid=1118894.1118903 (visited on 11/24/2020).

Louis, Annie and Derrick Higgins (2010). "Off-topic essay detection using short prompt texts". In: *proceedings of the NAACL HLT 2010 fifth workshop on innovative use of NLP for building educational applications*, pp. 92–95.

Maaten, Laurens Van der and Geoffrey Hinton (2008). "Visualizing data using t-SNE." In: *Journal of machine learning research* 9.11.

Mayfield, Elijah and Alan W Black (2020). "Should You Fine-Tune BERT for Automated Essay Scoring?" In: *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*. Seattle, WA, USA → Online: Association for Computational Linguistics, pp. 151–162. DOI: 10.18653/v1/2020.bea-1.15. URL: https://www.aclweb.org/anthology/2020.bea-1.15 (visited on 03/30/2021).

McInnes, Leland, John Healy, and Steve Astels (Mar. 2017). "hdbscan: Hierarchical density based clustering". In: *The Journal of Open Source Software* 2.11, p. 205. ISSN:

2475-9066. DOI: 10.21105/joss.00205. URL: http://joss.theoj.org/papers/10.21105/joss.00205 (visited on 07/01/2021).

McNamara, Danielle S. and Walter Kintsch (Oct. 1996). "Learning from texts: Effects of prior knowledge and text coherence". In: *Discourse Processes* 22.3, pp. 247–288. ISSN: 0163-853X, 1532-6950. DOI: 10.1080/01638539609544975. URL: http://www.tandfonline.com/doi/abs/10.1080/01638539609544975 (visited on 07/27/2021).

McNamara, Danielle S. et al. (Jan. 2015). "A hierarchical classification approach to automated essay scoring". In: *Assessing Writing* 23, pp. 35–59. ISSN: 10752935. DOI: 10.1016/j.asw.2014.09.002. URL: https://linkinghub.elsevier.com/retrieve/pii/S1075293514000427 (visited on 11/12/2020).

Mikolov, Tomas et al. (Sept. 2013). "Efficient Estimation of Word Representations in Vector Space". In: *arXiv:1301.3781 [cs]*. arXiv: 1301.3781. URL: http://arxiv.org/abs/1301.3781 (visited on 07/01/2021).

Nicol, David, Avril Thomson, and Caroline Breslin (Jan. 2014). "Rethinking feedback practices in higher education: a peer review perspective". In: *Assessment & Evaluation in Higher Education* 39.1, pp. 102–122. ISSN: 0260-2938, 1469-297X. DOI: 10.1080/02602938.2013.795518. URL: http://www.tandfonline.com/doi/abs/10.1080/02602938.2013.795518 (visited on 03/09/2021).

Ormerod, Christopher M., Akanksha Malhotra, and Amir Jafari (Feb. 2021). "Automated essay scoring using efficient transformer-based language models". In: *arXiv:2102.13136 [cs]*. arXiv: 2102.13136. URL: http://arxiv.org/abs/2102.13136 (visited on 04/15/2021).

Papineni, Kishore et al. (2001). "BLEU: a method for automatic evaluation of machine translation". In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*. Philadelphia, Pennsylvania: Association for Computational Linguistics, p. 311. DOI: 10.3115/1073083.1073135. URL: http://portal.acm.org/citation.cfm?doid=1073083.1073135 (visited on 08/02/2021).

Peters, Matthew E. et al. (Mar. 2018). "Deep contextualized word representations". In: *arXiv:1802.05365 [cs]*. arXiv: 1802.05365. URL: http://arxiv.org/abs/1802.05365 (visited on 07/01/2021).

Phandi, Peter, Kian Ming A Chai, and Hwee Tou Ng (2015). "Flexible Domain Adaptation for Automated Essay Scoring Using Correlated Linear Regression". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 431–439.

Pintrich, Paul R. (1995). "Understanding self-regulated learning". In: *New Directions for Teaching and Learning* 1995.63, pp. 3–12. ISSN: 0271-0633, 1536-0768. DOI: 10.1002/tl.37219956304. URL: https://onlinelibrary.wiley.com/doi/10.1002/tl.37219956304 (visited on 09/16/2021).

Pruksachatkun, Yada et al. (2020). "Intermediate-Task Transfer Learning with Pretrained Language Models: When and Why Does It Work?" In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, pp. 5231–5247. DOI: 10.18653/v1/2020.acl-main.467. URL: https://www.aclweb.org/anthology/2020.acl-main.467 (visited on 02/11/2021).

Radford, Alec et al. (2018). "Improving Language Understanding by Generative Pre-Training". In: p. 12.

Radford, Alec et al. (2020). "Language Models are Unsupervised Multitask Learners". In: *OpenAI blog* 1.8, p. 9.

Raffel, Colin et al. (2019). "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *arXiv preprint arXiv:1910.10683*, p. 67.

Rogers, Anna, Olga Kovaleva, and Anna Rumshisky (Nov. 2020). "A Primer in BERTology: What we know about how BERT works". In: *arXiv:2002.12327 [cs]*. arXiv: 2002.12327. URL: http://arxiv.org/abs/2002.12327 (visited on 01/09/2021).

Roscoe, Rod D et al. (2012). "Developing Pedagogically-Guided Threshold Algorithms for Intelligent Automated Essay Feedback". In: p. 6.

Rudner, Lawrence M and Tahung Liang (2002). "Automated Essay Scoring Using Bayes' Theorem". In: *The Journal of Technology, Learning and Assessment* 1.2, p. 22.

Si, Luo and Jamie Callan (2001). "A Statistical Model for Scientific Readability". In: *Proceedings of the tenth international conference on Information and knowledge management*, p. 3.

Taghipour, Kaveh and Hwee Tou Ng (2016). "A Neural Approach to Automated Essay Scoring". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, pp. 1882–1891. DOI: 10.18653/v1/D16-1193. URL: http://aclweb.org/anthology/D16-1193 (visited on 11/23/2020).

Thakur, Nandan et al. (Oct. 2020). "Augmented SBERT: Data Augmentation Method for Improving Bi-Encoders for Pairwise Sentence Scoring Tasks". In: *arXiv:2010.08240 [cs]*. arXiv: 2010.08240. URL: http://arxiv.org/abs/2010.08240 (visited on 01/19/2021).

Thomas, Glyn, Dona Martin, and Kathleen Pleasants (2011). "Using self- and peer-assessment to enhance students' future-learning in higher education". In: *Journal of University Teaching \& Learning Practice* 8, p. 17.

Uto, Masaki, Yikuan Xie, and Maomi Ueno (2020). "Neural Automated Essay Scoring Incorporating Handcrafted Features". In: *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics, pp. 6077–6088. DOI: 10.18653/v1/2020.coling-main.535. URL: https://www.aclweb.org/anthology/2020.coling-main.535 (visited on 04/15/2021).

Vaswani, Ashish et al. (Dec. 2017). "Attention Is All You Need". In: *arXiv:1706.03762 [cs]*. arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762 (visited on 01/08/2021).

Verma, Pradeepika and Hari Om (May 2019). "A novel approach for text summarization using optimal combination of sentence scoring methods". In: *Sādhanā* 44.5, p. 110. ISSN: 0256-2499, 0973-7677. DOI: 10.1007/s12046-019-1082-4. URL: http://link.springer.com/10.1007/s12046-019-1082-4 (visited on 05/12/2021).

Villalón, Jorge et al. (2008). "Glosser: Enhanced Feedback for Student Writing Tasks". In: *2008 Eighth IEEE International Conference on Advanced Learning Technologies*. Santander, Cantabria, Spain: IEEE, pp. 454–458. ISBN: 978-0-7695-3167-0. DOI: 10.1109/ICALT.2008.78. URL: http://ieeexplore.ieee.org/document/4561736/ (visited on 11/18/2020).

Witte, Stephen P. and Lester Faigley (May 1981). "Coherence, Cohesion, and Writing Quality". In: *College Composition and Communication* 32.2, p. 189. ISSN: 0010096X. DOI: 10.2307/356693. URL: https://www.jstor.org/stable/10.2307/356693?origin=crossref (visited on 01/05/2021).

Woods, Bronwyn et al. (Aug. 2017). "Formative Essay Feedback Using Predictive Scoring Models". In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Halifax NS Canada: ACM, pp. 2071–2080. ISBN: 978-1-4503-4887-4. DOI: 10.1145/3097983.3098160. URL: https://dl.acm.org/doi/10.1145/3097983.3098160 (visited on 11/13/2020).

Xu, Peng et al. (2020). “MEGATRON-CNTRL: Controllable Story Generation with External Knowledge Using Large-Scale Language Models”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, pp. 2831–2845. DOI: 10.18653/v1/2020.emnlp-main.226. URL: https://www.aclweb.org/anthology/2020.emnlp-main.226 (visited on 02/03/2021).

Yanchi Liu et al. (June 2013). “Understanding and Enhancement of Internal Clustering Validation Measures”. In: *IEEE Transactions on Cybernetics* 43.3, pp. 982–994. ISSN: 2168-2267, 2168-2275. DOI: 10.1109/TSMCB.2012.2220543. URL: http://ieeexplore.ieee.org/document/6341117/ (visited on 12/15/2020).

Yang, Ruosong et al. (2020). “Enhancing Automated Essay Scoring Performance via Fine-tuning Pre-trained Language Models with Combination of Regression and Ranking”. In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, pp. 1560–1569. DOI: 10.18653/v1/2020.findings-emnlp.141. URL: https://www.aclweb.org/anthology/2020.findings-emnlp.141 (visited on 04/08/2021).

Yannakoudakis, Helen, Ted Briscoe, and Ben Medlock (2011). “A New Dataset and Method for Automatically Grading ESOL Texts”. In: *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, pp. 180–189.

Yu, Qian et al. (2020). “Review-based Question Generation with Adaptive Instance Transfer and Augmentation”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, pp. 280–290. DOI: 10.18653/v1/2020.acl-main.26. URL: https://www.aclweb.org/anthology/2020.acl-main.26 (visited on 02/02/2021).

Zesch, Torsten, Michael Wojatzki, and Dirk Scholten-Akoun (2015). “Task-Independent Features for Automated Essay Grading”. In: *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*. Denver, Colorado: Association for Computational Linguistics, pp. 224–232. DOI: 10.3115/v1/W15-0626. URL: http://aclweb.org/anthology/W15-0626 (visited on 11/23/2020).

Zhang, Haoran and Diane Litman (2020). “Automated Topical Component Extraction Using Neural Network Attention Scores from Source-based Essay Scoring”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, pp. 8569–8584. DOI: 10.18653/v1/2020.acl-main.759. URL: https://www.aclweb.org/anthology/2020.acl-main.759 (visited on 04/06/2021).

Zhang, Tianyi et al. (Feb. 2020). “BERTScore: Evaluating Text Generation with BERT”. In: *arXiv:1904.09675 [cs]*. arXiv: 1904.09675. URL: http://arxiv.org/abs/1904.09675 (visited on 01/27/2021).

Zhang, Xiang and Yann LeCun (Apr. 2016). “Text Understanding from Scratch”. In: *arXiv:1502.01710 [cs]*. arXiv: 1502.01710. URL: http://arxiv.org/abs/1502.01710 (visited on 07/29/2021).