Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

R&D Project

AI-assisted short answer grading: comprehensive classification and evaluation of the existing state of the art techniques

Evgeniya Ovchinnikova

Submitted to Hochschule Bonn-Rhein-Sieg,
Department of Computer Science
in partial fullfilment of the requirements for the degree
of Master of Science in Autonomous Systems

Supervised by

Prof. Dr. Paul Plöger
M.Sc. Deebul Nair

Month 20XX

Abstract

Assignment evaluation is one of the most time-consuming parts of teacher's work. Therefore nowadays many assignment types, such as multiple-choice answers or fill-the-gap, are automated. Free text answer grading is a harder task. There are two types of free text assignments: automated essay grading (AES) and short answer grading (ASAG). AES systems are already successfully used by some organizations including Educational Testing Service (ETS). However, ASAG is still not popular, because the existing systems haven't shown sufficient robustness yet. This research concentrates on development of an AI assistant for grading assessments and exams in the fields of computer science, electrical engineering, physics and other technical disciplines. In this case answers are shorter than one paragraph and concrete. Furthermore, stylistics and spelling are not of interest, only the meaning of the answer should be taken into account. This study focuses on comprehensive analysis of existing ASAG methods. Based on this analysis a new ASAG system is developed. It uses similarity between the students' and teachers' answers for grading. Such sentence similarity measures as ROUGE, BLUE and various k-skip-n-gram distances in combination with Kernel Principal Component Analysis (KPCA) are considered. A significant advantage of this vectorization technique is that it allows an efficient clusterization of small amount of data, which is often a case for ASAG, since an offline course can be attended by 10-300 people and not by thousands. KPCA has already been shown to be more powerful than word2vec for word embedding, and this work shows that it is more powerful than doc2vec for paragraph embedding as well.

Acknowledgements

I would like to express my great appreciation to Professor Plöger and Mr. Nair, my research supervisors, for their thoughtful advice and assistance during the my RnD project. I would also like to thank Stifterverband for accepting me as a part Digital Learning Transfer Fellowship, which was a significant support for the project.

Contents

List of Figures

List of Tables

1 Introduction

The process of teaching is very challenging and takes considerable time and effort. In many areas, especially computer science, it requires keeping the curriculum and assignments up to date and relevant. Personal communication with students and giving feedback on tests and homework is also essential as well as time-consuming. These activities are the most visible to students, but there are many more things a teacher or professor must do, such as paperwork, organizational issues, to name just a few. For this reason it is important to optimize the time dedicated to each different aspect of work. Keeping course content relevant as well as personal communication cannot be easily outsourced, but grading tests and assignments is usually straightforward enough and does not require a professor's expertise. Students often make similar mistakes, making the task well-suited for automation using machine learned models. This clearly shows that it is an ideal task to be automated.

Assignments and test answers include different aspects, such as text, pictures and programming code. In this work we focus on the text. There are several main text answer types: "fill the gap", short as well as long answer essays [1]. The first type does not represent a complex problem because there would normally be only one possible answer and the only challenges that one can face here are handwriting recognition, if the test was not on a computer; or spelling correction, if the incorrectly spelled answers can be accepted. Two other types are more challenging from the text analysis point of view. Automated essay scoring (AES) focuses on evaluating texts longer than one paragraph, whereas automated short answer grading (ASAG) concentrates on answers consisting of a small number of sentences [1]. The other important difference is that ASAG only concerns the with the meaning of the answer, while for AES the writing style is also very important [2].

AES is a rather complex task. Some massive open online courses (MOOC), such as those offered by EdX, MIT and Harvard's non-profit organization HarvardX, and organizations like ETS have already started using the automated essay grading systems [3], [4]. This is a logical step, since most of the online courses are either free or very cheap in comparison to the ones in real universities and institutes and it is impossible to find many teachers who are willing to work for free. AES is usually based on comparison of already graded essays and the ones to grade [5]. One of the algorithms they use is an algorithm developed by a contestant [6] of "The Hewlett Foundation: Automated Essay Scoring" competition on Kaggle [7] combined with EASE library [8], which can be used for machine learning text classification. Although AES has been received rather positively [4], [9], [10], there are still many people who have strong doubts about the feasibility of automated essay grading. Many people, including even Noam Chomsky, have signed a petition "Professionals Against Machine Scoring Of Student Essays In High-Stakes Assessment" [11]. Moreover, there were a lot of papers criticizing either state of the art AES or its general appropriateness [12], [13], [14].

AES area is more relevant for the humanities. This research concentrates on developing an AI assistant for grading assessments and exams in the fields of computer science, electrical engineering, physics and other technical disciplines. Here the answers are usually shorter and more concrete. Furthermore, stylistics and spelling are not of interest, but that depends on the individual professor. There is a number of systems developed for this purpose, but ASAG is still not widely used. The reason is that these systems are not robust enough and often work well only on a dataset it was trained on or in a very narrow knowledge area [15]. Therefore, this work focuses on ASAG. This task is viable and still not solved completely, whereas the solution should be helpful for both professors and students. It can allow professors more time for other teaching activities and provide students with a prompt and unbiased feedback on their work.

An important feature of our system is that it is meant for offline courses. An online course or centralized examination can have a very large amount of attendants, whereas an offline university course would usually be restricted by 30 students for a master program and around 300 students for a bachelor program, which means that the algorithm must perform well with a small amount of data.

2

Problem statement

3

State of the Art

3.1 Natural language processing

3.1.1 Natural language processing development

This work focuses on ASAG that is a very narrow subfield of Natural Language Processing (NLP). Therefore it is important to consider what actually is NLP first.

NLP is an very important research area that developed from the simple algorithms those could be performed on punch cards to huge sufisticated and heavy calculations those are performed by compamies like Google and Facebook now [16]. NLP is based on many other subject areas, such as linguistics, mathematics, computer science, artificial intelligence, robotics, and psychology. It can be applied in such areas as machine translation, summary generation, simplification of the texts, multilingual and cross-language information retrieval, diagnosing, chat-bots development, etc. [18]. The common idea among these tasks is that the computer must truely understand the language, the meaning of the phrases, sentences and whole texts. However, we cannot just tell the machine to "understand", its "understanding" must be formalized. This formalization is the subject of NLP. In a broader sense NLP task includes all levels of language processing and understanding: acoustic (speech recognition), analytical (syntactics and semantics), contextectual (anaphora, inference of referents, etc.), inferential (detection of inferences and implicatures), planning (sense, text and speech generation). Otherwise NLP can be narrowed mainly to analytical part, i.e. be considered as a set of techniques and methods for grammatical structure extraction and analysis for text understanding [19]. Current work is not interested in such aspects of NLP as speech recognition and generation or in detailed research on grammars, the most interesting NLP aspect is an information extraction – "subset of knowledge discovery and data mining that aims to extract useful bits of textual information from natural language texts" [18]. However, text generation can be another useful tool for teachers who are interested in automated generation of examination questions, and speech recognition can be interesting for oral exams aytomation in the future. Therefore we will briefly consider all seven stages that may include a task of NLP [18], [20]:

- Phonetics
- Morphology
- Lexis
- Syntax
- Semantics
- Discourse
- Pragmatics

Phonetics doesn't extract any meaning of the words, it is only focused on pronunciation. At this level an NLP system is focused more on a signal-processing than a linguistic part, though linguistic tools can be helpful in disambiguation of badly-parsed text.

Morphology is the most low-level stage of text processing: it considers the structure of the words, words' parts and how these parts influence the meaning of the words. For example, if "s" is added to a noun that would be a plurality morpheme and "s" added to a verb would indicate that it is the verb used in combination with a noun or pronoun that describes a third person.

Lexis considers the level of the words and parts of speech. Unlike in morphology at this level the meaning of the separate words is extracted regardless of their forms, e.g. the words "mouse", "mice" and "mouse's" can be the same lexical entity.

Syntax is the third level that considers the grammar of the language. It breaks the sentence into separate members of a sentence, i.e. subject (nominative), attribute (verb), case put after (object), etc., and reorganizes the data in such a way that can be analyzed by computer. This analysis includes grammar and parsing [20].

Semantic is a higher level analysis that considers extraction of meaning of words and sentences.

Discourse the highest level information extraction that considers the information extraction of text parts larger than a sentence, or the whole text. It deals with the whole structure of the document. The knowledge from this level can be very helpful for text summarization.

Pragmatics is the outer level of information extraction that includes the knowledge about the world, which is not contained in the document. It consists of two stages: finding ambiguities and disambiguation. The are following types of ambiguities: one-word long lexical (homonym) and referential ("it", "he", etc.), several-words long syntactic, sentence-long semantic, local (context requiring, which is usually a great issue for translation). Disambiguation includes the following methods: probabilistic (simple choice of the most common meaning of the word, optionally – most possible in the subject field of this text), conditional probability (choice of the most common meaning of the word in the subject field of this text), context (most probable meaning of this word when it is surrounded by certain words or grammatical structure), world models (disambiguation by usage of world states) [20]. The conditional probability can be especially important for this work, because the tests are subject-specific and would normally use only one meaning of the certain word. For example, the word "operation" in a computer science exam will most likely not mean mean "surgery" .

History and prediction of integration of the methods mentioned above into NLP over is shown in Fig. 3.1. One can clearly see how the computers could grasp gradually more general meaning of the text. From understanding of single words to acquiring the meaning of whole sentences and paragraphs. Such differences are always easy to see in machine translation. One can compare the translation given by Google-translator (`translate.google.com`) that uses the bleeding-edge NLP-methods and PROMT (`http://www.online-translator.com/`) that translates almost word by word. PROMT uses rule-based and statistical machine translation, whereas Google-translator uses eight layers of Long Short-Term Memory-Recurrent Neural Network (LSTM-RNN). The original sentence was taken from the beginning of "Faust" by Johann Wolfgang Von Goethe: "Ihr nähert sich wieder, schwankende Gestalten, Die vor sich selbst ist der Trüben Blick gezeigt.". The translation by Google is "You approach again, swaying figures, who in front of themselves is shown the murky gaze.", by PROMT – "You approach again, fluctuating shapes, early to yourselves once the murky look shown.". PROMT couldn't even detect the German structure of the sentence and just put the English words in the initial order. Google translation is not perfect, but it is clear that it translates the whole sentence and not only the separate words. However, even a relatively good automated translator has problems with textual ambiguity and especially stylistics and it will take some time before it is able to translate the literature like human translators: "Again ye come, ye hovering Forms! I find ye, As early to my clouded sight ye shone!" [22].

Though semantic approach is slowly starting being used more commonly, syntactic approach is still used more often. That is why it is important to mention the main methods used there [16]:

- Keyword Spotting – unambiguous buzzwords detection. It is used in the following systems:
  - Ortony's Affective Lexicon – for words categorization;
  - PageRank – Google's page ranking algorithm;
  - TextRank – graph-based ranking model for text processing.
- Lexical Affinity – allows to detect not only unambiguous words, but also

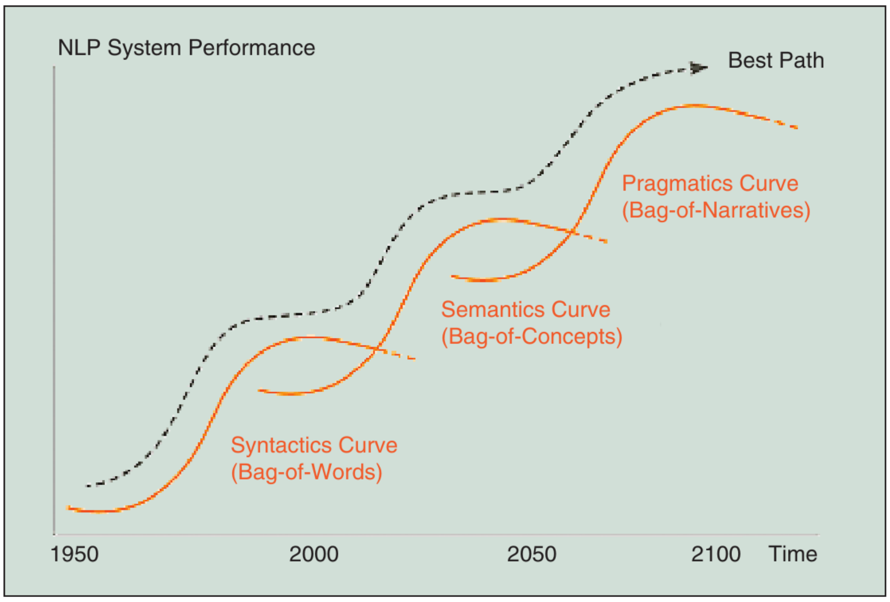


Figure 3.1: NLP developement through three different eras or curves [16]

assign words to a certain group with a given probability based on a certain text corpora, e.g. ”operation” can be medical with 40%, computational with 30% and military – with 30%. The problem of this approach is that it can easily be overfitted and work only for one area on which word probabilities it was trained.

- Statistical NLP includes the following machine learning algorithms:
  - maximum-likelihood
  - expectation maximization
  - conditional random fields
  - support vector machines (SVM)

  The method allows to learn by training on a large text corpora the valence of the buzzwords, while also taking into consideration punctuation and how often certain words occur together. The problem of this method is semantic weakness: it's too focused on the buzzwords and cannot work efficiently enough with other words, therefore it requires a very large amount of text to train.

Semantic or concept-based level is fundamentally different: it is focused on the meaning of sentences as a whole and not on spotting of separate buzzwords, learning the concepts and not the words. One can distinguish the following concept-based NLP approaches [16]:

- internal knowledge-based techniques (endogenous) – uses only information froinly to analytical part, i.e. be considered as a set of techniques and methods for grammatical structure extraction and analysis for text understam a given document. It integrates machine learning in order to conduct lexical-semantic analysis of a large amount of textual information to extract concepts. The following machine learning methods are used:
  - latent semantic analysis – vector documents in a terms space;
  - latent Dirichlet allocation – assigning terms to topics;
  - genetic algorithms – probabilistic pseudo-stochastic search.
- external knowledge-based techniques (endogenous) – used external information about the world. There are two main categories of those:
  - taxonomic or ontology-based approaches are usually concerned about the "isA" relations between objects (e.g. "Albert Einstein") and concepts (e.g. "physicist"). The problems of this approach are bad scalability and that the representation of knowledge is too strictly-defined and typical. It overdefines the object and considers only its "proper" usage, for example an idea of a "key" can be related to a "picklock" or a "wireless keycard", as they are used to open a door, but not to other possible usages of the key, as a sort of weapon or a beer opener.
  - noetic or semantic knowledge-based include all human brain-based techniques. It focuses on semantic parsing nd linking separate linguistic components creating meaningful constructions. The following machine learning methods are used for this:
    - neural networks
    - deep learning
    - sentic computing

3.1.2 From linguistics to NLP

All the concepts mentioned above originate in pure linguistics and it is obvious that NLP area would strongly rely on it. However, it is a hard question, how much NLP should rely on pure linguistics and which modifications should be applied to linguistic methods so they would be efficiently applicable to NLP. Linguistics provides a highly-formalized model of the language. Raskin in [21] names the following manifestations and interpretations of linguistic formality:

- mathematical notation usage
- mechanical-symbol-manipulaticn-device (MSMD) approach:
  - collecting all information without any filtering and representing it in a mathematical form, e.g. formal grammar
  - creating a number of subsets of linguistic items with common features
  - all the subsets together must cover all the words those were considered
  - it is assumed that a native speaker must be able to differentiate all these items intuitively, without any specific knowledge

This approach seems formal enough to directly transfer it to a format acceptable for machine text processing without fundamental concept changes. However, it is not that easy and direct application of the method would usually lead to a bad result. One must know certain laws of a correct usage of pure linguistics in application to NLP. It is important to understand that the problem should originate in a target field (NLP), whereas approach and terms can come from the source field (linguistics). Otherwise it would be impossible to reach the target and use the source efficiently [21]. Plainly linguistic problems should not be formulated for NLP-tasks, however, linguistic approaches remain crucially important for NLP [19]. The NLP task includes the following [18]:

- thought processes
- representation and meaning of the language input

Table 3.1: Linguistics vs NLP [21]

Linguistics	NLP
research on usage of sounds and its meaning for the language	parsing spoken language to a written form for information extraction
structuring the meaning of the text	understanding the meaning of the text
categorizing different linguistic structure levels	making a text source-independent by extracting all relevant information
separating linguistics and encyclopedic information	using encyclopedic information in combination with linguistic approaches
formal (set of rules) representation of the result	making the information practically accessible
language in general	**narrow part of the language**

- world knowledge of the program

Table 3.1 shows the main differences between NLP and pure linguistics those help to understand how to use linguistics for the task of NLP.

The last point is especially important for the current work. It shows that it is common to use a very limited subset of the words existing in a language for NLP in general. It is also the case for our task, because each exam is dedicated to a certain subject with its own lexicon. Moreover, the formal nature of examinations allows to discard all informal and colloquial words. A student is normally not allowed to use sentences with slang and colloquial terminology similar to the following examples: "Yukio Mishima was a totally screwed wacko and a bent, but it is not the only reason why we all like him", "You should never forget to shiled the qubits properly, otherwise your experiment will be absolutely screwed". In the set of words those a formal exam would allow, "screw" as a verb should only mean "an act of turning a screw or other object having a thread" (`https://www.google.de/search?q=Dictionary`) and "bent" would only be a past or an adjective form of "bend". That is why it is not efficient to try using general linguistic methods, though they are supported by the whole power of the language. Instead it makes sense to focus on NLP-methods those would allow to concentrate on specific subject-specific words. Raskin names the following advantages of the sublexicons [21]:

- on average it contains several hundred words instead of 500 000
- ambiguity is drastically reduced due to the reduced number of meaning homonyms (see examples above)
- only a very limited extralinguistic knowledge of the world is needed

However, it is important not to forget that this approach can overlimit the lexicon and make the approach not extrapolatable enough. For example, if one develops an exam in the area of physics, he or she might want to reduce the set in such a way that a word "induction" would mean only "electromagnetic or magnetic induction" and it cannot mean "mathematical induction". It doesn't make much sense though, because adding this sense to such word is not very costly and in general it is possible to use a similar set of words for mathematics and physics, which would allow to have more train exam samples. Moreover, this approach is dangerous because even in one area it is not always possible to foresee all possible vocabulary that will be used.

3.1.3 NLP methods for ASAG

In this section some general NLP methods those are used in various ASAG systems are considered. More specific, e.g. syntactic pattern matching or parse tree matching, are considered in details in ASAG section.

N-grams is a method that breaks a sentence into a set of word sequences of certain length. For example, bigrams and trigrams for a sentence "Penrose's model of quantum consciousness is rather questionable" are the following: bigrms – {"Penrose's model", "model of", "of quantum", "quantum consciousness", "consciousness is", "is rather", "rather questionable"}, trigrams – {"Penrose's model of", "model of quantum", "of quantum consciousness", "quantum consciousness is", "consciousness is rather", "is rather questionable"}. A similar approach can be used on a word level. For example, bigrams for a word "quantum" are the following: {"qu", "ua", "an", "nt", "tu", "um"}.

Skip-grams is a method in which "n-grams are still stored to model language, but they allow for tokens to be skipped" [23]. Guthrie [23] defines k-skip-n-grams for a sentence containing words $w_1, w_2, ..., w_n$ in the following way:

$$\{w_{i1}, w_{i2}, ..., w_{in} | \sum_{j-1}^{n} i_j - i_{j-1} < k\}, \tag{3.1}$$

where n is strictly higher than one. For better understanding of the concept the following example can be made. A sentence "A new art colour for our Irish poets: snotgreen. " [24] broken into bigrams and 2-skip-bigrams takes the following form: bigrams = {"A new", "new art", "colour for", "for our", "our Irish", "Irish poets", "poets snotgreen"}, 2-skip-bigrams = {"A new", "A art", "A colour", "new art", "new colour", "new for", "for our", "for Irish", "for poets", "our Irish", "our poets", "our snotgreen", "Irish poets", "Irish snotgreen", "poets snotgreen"}. One can see that in many cases it creates senseless sequences of the words like "A art", however, k-skip-n-gram "for Irish" would make total sense, because the sentence without word "our" is as valid as the initial one. Moreover, the wrong k-skip-n-grams are too random and often grammatically senseless, that is why they can with very low probability create false positives, which means that it is at least harmless to use them.

Tokenization is breaking a sentence into separate words tokens and removing punctuation, which is basically extraction of unigrams [25]. For example, a sentence "Tokenization is breaking a sentence into separate words tokens and removing punctuation, which is basically extraction of unigrams" will take the following form: {Tokenization"", "is","breaking", "a", "sentence", "into", "separate", "words", "tokens", "and", "", "removing", "punctuation", "which","unigrams","is","basically","extraction","of","unigrams"}.

Stemming and lemmatization are techniques those allow to returned a word in a specific form to its original form, e.g. a verb "is" is lemmatized as "be", genitive case "des Herzens" – as "das Herz", plural "mice" – as "mouse", etc. [26].

Spelling correction allows to change the wrongly written words to the correct form, e.g. "corrrect" – "correct". However, it can be dangerous in cases when there are several possible correction options: "fhy" – "fry" or "fly".

Noun phrase chunking is a method for breaking sentences into subconstituents, i.e. verbs, nouns, and prepositional phrases [27]. For example: "The saddest music in the world." – {("the", "DT"), ("saddest", "JJ"), ("music", "NN"), ("in", "IN"), ("the", "DT"), ("world", "NN") }, where "DT" is a definite article, "JJ" – adjective, "NN" – noun and "IN" – preposition.

Pronoun or anaphora resolution – methods those allow to identify to which word refers a pronoun or anoun phrase. For example, in a sentence "One of the outstanding traits of Nabokov's writing is that he could write, and actually wrote, equally well in Russian and English, which allows speakers of both languages read his books in original." "he" is "Nabokov" [28].

Stop words spotting and removal – finding and removing overly common and thereby semantically non-selective words, such as "a", "an", "the", "its", etc. [29]. For example "A cat sit on a mat." transforms to "cat sit mat".

Part Of Speech (POS) tagging – assigning part of speech to each word in a sentence: "Many ideas of Buddhismus are strongly supported by evolutionary biology." – { ("many", "adjective"), ("ideas", "noun"), ("of", "preposition"), ("Buddhismus", "noun"), ("are", "verb"), ("strongly", "adjective"), ("supported", "verb"), ("by", "preposition"), ("evolutionary", "adjective"), ("biology", "noun")} [30].

Term frequency (TF) is how often a word occurs in a text corpora, e.g. in students' answers:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}, \tag{3.2}$$

where $n_{i,j}$ is a number of occurrences of a ith word in a jth text, e.g. in a student's answer [31].

Term frequency-inverse document frequency (TF-IDF or tfid) is a measure of importance of the word to a text in a corpus. It is proportional to the word's occurrence frequency:

$$tfid_i = log\frac{|T|}{|t : w_i \in T|}, \tag{3.3}$$

where $|T|$ is the numbers of texts in the corpus, w_i is an ith word and t is a number of texts where the word w_i occurs [31].

Entropy-variation (EV) is a measure of uncertainty of word occurrence in a text corpora:

$$normH(w) = -\frac{1}{n}\sum_{i=1}^{n} p(w)log_2 p(w_i), \tag{3.4}$$

where $normH(x)$ is a normalized entropy, $p(w)$ – probability of the word w occurrence in a text and n – number of texts in a corpora [31].

Textual entailment (TE). A hypothesis entails a text iff a human infers a high likelihood of the hypothesis while reading the text [32].

Word alignment is such a feature that measures a proportion of semantically similar words in two pieces of text [33]. The similar words or phrases are found by a word aligner that uses a paraphrase database. An example of processing of two sentences by the aligner is depicted in Figure 3.2.

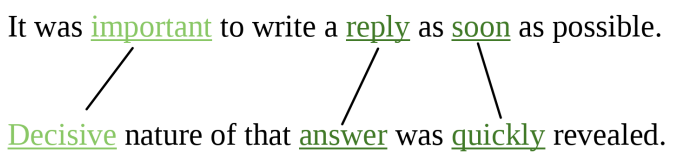

It was important to write a reply as soon as possible.
Decisive nature of that answer was quickly revealed.

Figure 3.2: Word alignment. Words those are strongly similar are denoted by dark-green and weekly similar – by light-green.

3.1.4 NLP software tools used for ASAG

All of the ASAG systems considered below in this work use either Java or Python. The following software tools were used:

- Global Vectors for Word Representation (GloVe) – log-bilinear regression unsupervised algorithm for word vectorization (Python/Octave) (`https://nlp.stanford.edu/projects/glove/`)
- word2vec – log-bilinear prediction-based bag-of-words and skip-n-gram algorithms for vectorizing words developed by Google, can be used with Python, Java, C and Scala (`https://code.google.com/archive/p/word2vec/`)
- Natural Language Toolkit (NLTK) – a Python tool for NLP. Mainly used in scientific projects (`https://www.nltk.org/`).
- Industrial-Strength Natural Language Processing (spaCy) – another Python tool for NLP. The tool is meant for industrial usage, but it is also used in scientific projects (`https://spacy.io/`)
- WordNet – a lexical database for English language (`https://wordnet.princeton.edu/`)
- extracting DIstributionally related words using CO-occurrences (DISCO) – Java application for calculating similarity between words or short texts, finding the most similar words in texts, e.g. "smart" and "clever", and finding the related words, e.g. "beer" $\rightarrow$ "Kölsch", alcohol, foam, barrel. It requires a database of word similarities (word space) precalculated on a large text corpora by "DISCO Builder". It is similar to word2vec and GloVe. Architecture of the system is shown in Fig. 3.3. DISCO Builder is trained on Wikipedia and British National Corpus (BNC) (`http://www.linguatools.de/disco/disco_en.html`).
- InfoMap NLP Software – a package for LSA (`http://infomap-nlp.sourceforge.net/`)

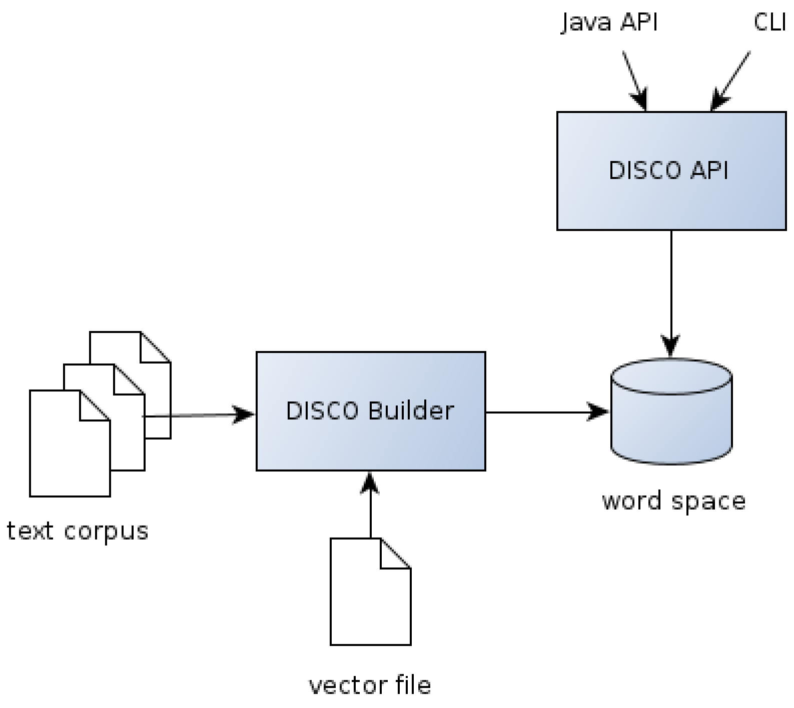


Figure 3.3: Architecture of DISCO [34].

- MontyLingua – Java/Python NLP toolkit (`http://alumni.media.mit.edu/~hugo/montylingua/doc/MontyLingua.html`)
- PC-KIMMO – morphological analysis two-level processor (`http://www.ai.mit.edu/courses/6.863/doc/pckimmo.html`)
- Spell Checker Oriented Word Lists (SCOWL) – spelling correction for English (`https://github.com/en-wl/wordlist`)
- TreeTagger – POS annotation tool for many languages (`http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/`)
- CASS – a tool for chunking (`http://spraakdata.gu.se/svedk/cass_swe.html`)
- Pointwise Mutual Information (PMI-IR) – uses AltaVista's Advanced Search

query syntax to calculate the matching between word pairs by calculating the probability of co-occurrence of the words based on web pages corpus.

- Stanford parser – a statistical tool for structure of sentences parsing (`https://nlp.stanford.edu/software/lex-parser.shtml`)
- Apache OpenNLP – Java NLP toolkit (`https://opennlp.apache.org/`)
- Morpha – Java tool for morphological analysis (`https://github.com/stanfordnlp/CoreNLP`)
- MaltParser – a tool data-driven sentence dependency parsing (`http://www.maltparser.org/`)

3.2 Automated grading

Research in natural language processing (NLP) started in 1950. The following topics were and still are the main focus of NLP researchers: automatic translation, information retrieval, summarizing of texts, question answering, information extraction and topic modeling [16]. Sixteen years later, in 1966, the first work on automated grading was published [17]. That first work was dedicated to essay grading and it laid a foundation to a large field that now includes automated essay and short answer grading. The main differences are listed in Table 3.2 [2], [1], [74]. This section is focused on analysis of current state of the art in these two areas.

Table 3.2: ASAG vs AES

	Short answer	Essay
Length	$\leq$ one paragraph	$>$ one paragraph
Evaluation criterion	Content	Style, content
Openness	Closed	Open
Question format	Formulated according to strict rules	Can be general
Correct answer	Exists	Doesn't exist

3.2.1 AES

Overview of the systems

In recent years massive tests, such as language proficiency exams (TOEFL, IELTS, TestDaF, DELF, etc.) or MOOC assessments, are becoming increasingly popular. All language tests and many MOOC assignments contain parts where the students have to either summarize a given text or write his/her own text or essay. Scoring these long text is a huge task that requires tremendous amount of human power. Language tests are rather expensive, thereby it is possible to find enough people to grade them. The idea of MOOCs, however, is to make the courses from the best universities affordable for people and that is why it is why scoring is a big issue in this case. It is possible to automate multiple-choice tests, but the quality of such check of knowledge is very low. Therefore, it was essential to automate free text grading as well. Moreover, it is still important for the language tests, because the autograding would allow to give a faster and less biased response.

Both, automated and human essay grading, have several important issues those must be faced – see table 3.3 [5]. Therefore, starting the automated grading was a long process, but now it is rapidly developing.

The first multiple-choice grader was developed in 1938, whereas the first idea of the free text grader appeared only in 1960's in a work of Ellis Page [5]. The system was called Project Essay Grade (**PEG**). The approach was based on usage of proxes and trins – the variables recognizable by a machine and a human graders accordingly, e.g. a number of words could be a prox and elaboration – a trin. The method was based on training the multiple-regression algorithm to grade like humans in a train set, not on the analysis of the text itself. The main features of the texts were the number of words and word length, which allowed to reach a correlation of 71% (77% in the later releases) with the human graders. The correlation between the machine grading and human graders was shown to be higher than the correlation between different human garders [5]. This grader, however, was very vulnarable to "gibberish" texts, i.e. the ones containing completely senseless

Table 3.3: Issues of machine and human grading

Automated grading	Human grading
It is hard to define an impersonal criterion for text style quality: many long-text tasks require not only correctness of content, but also a good writing style and/or usage of certain constructions. In tasks like ”Describe a graph”, which a participant has to do on TestDaF and IELTS, one should only use certain constructions and be relevant, however, in essay part the style, which can be estimated differently even by human examiners, plays an important role.	Human graders must be specifically trained, which requires a massive amount of time and money.
Though it is not intentionally malicious, autograder can still be biased. For example, if students with similar writing style usually deliver works of bad quality, the system can overlearn in such a way that it would rate badly even good works of similar style.	Human graders are biased more likely than a machine. Therefore test organizations try to avoid such bias by a complex cross-validation, testing and changing the assignments in such a way that the possible bias is reduced, which requires a lot of time and financial resources.
Usage of AES may cause negative reaction of examenees.	The high costs reduce the number of people who can afford to take an exam.
AES might have problems with gibberish detection.	”Human factor” like sickness or any other problems of the grader may cause a significant increase of grading time or decrease of grading quality.

words like "bzhykhlvyazsk" or "bdvfngjvsbcdhkuhlhdsskmlklewnmx" [9]. The words are long and the text can be as long as the writer please, however, it is not a real essay and by a human teacher it would be graded with zero points. The 77% correlation was obtained in absence of such important test cases. It shows a great importance of proper dataset that would cover most of the possible test cases.

Almost 40 years after the development of Page's approach of using the human graders' scores instead of a real understanding of text by computer as a main training criterion was of current interest. For example, **E-Rater** developed by ETS (the organization that conducts TOEFL and GRE) in 1990's operated in the similar way and also used multiple-regression method combined, however, with NLP, which allowed to include grammatical accuracy. The latest (2006) version had an additional feature that took into account word overlap between test and train essays, which was considered rather advanced, because it allowed to reduce importance of essay length, which essentially should not be an indicator of quality. This system is now used in GRE, TOEFL and several other ETS's tests. Correlation with human scores is very high – 97%, again higher than between human raters [5]. However, there is still a lot of criticism. For example, it was shown that examnee can submit a text – so-called "word salad" – that is long enough, contains all right and long words and constructions, but still doesn't make any sense, and the system will still grade it highly, for it meets all the needed requirements [35]. Another point voiced by critics is that the system can grade a good and creative essay, which doesn't meet some formal requirements, badly. That is another reason why many people including prominent scientists like Noam Chomsky argue that AES brings no good to the education [11]. It is true, however, such requirements is one of the examples when people expect from machine a way better performance than from humans. A human rater receives a massive amount of written works those must be graded in a short time. Human grading in this case is as formal as machine grading and creativity can be punished by a bad grade as possibly as in case of machine grading. Moreover, human graders are shown to have less unity among each other in their judgements than with machine, as it was shown in experiments with both, PEG and E-Rater [5]. Individual approach is important, but it is feasible only in small groups. In case of massive courses it is equally formal, no matter if it is an automated or human grading.

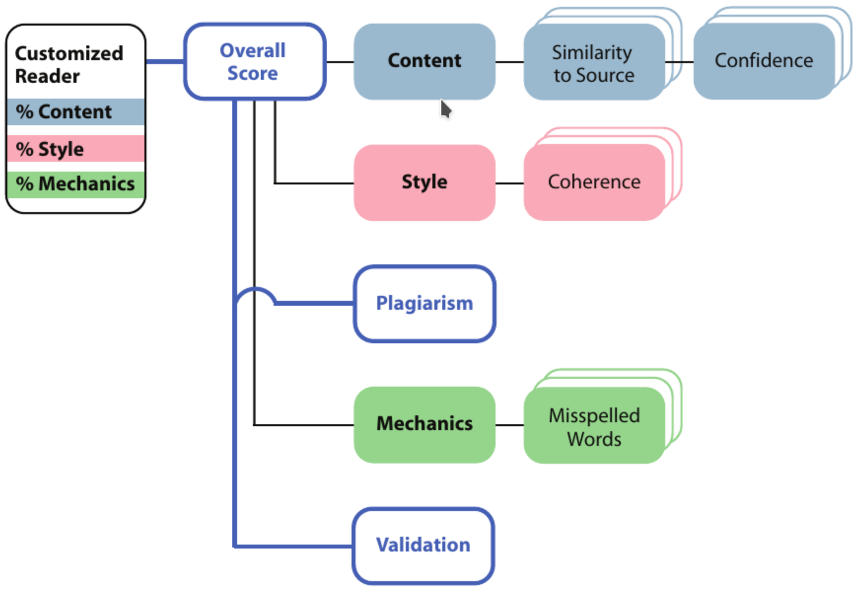


Figure 3.4: Architecture of IEA [37].

PEG and E-Rater had the same base idea. A fundamentally new approach was introduced in 1998 by Landauer in "An Introduction to Latent Semantic Analysis" (LSA) [36]. The main difference was that the method started caring about the meaning of the text and not only on breaking it into linguistically non-meaningful features those will be weighted and learned in relation to the human grades. In this method statistical computations on a large corpus of text are used to extract the contextual meaning of words, which places the method semantic part NLP curve (Figure 3.1) [36]. Moreover, the goal of methods developers was to imitate the way humans obtain and classify knowledge from the piece of text, as well as they resolve ambiguities by context [5]. LSA-based essay grader was called Intelligent Essay Assessor (**IEA**). Its architecture is shown in Fig. 3.4. This system is closer to ASAG than classical AES, because it focuses only on context and not the writing style. Moreover, it doesn't use the human raters' grades as a reference value, but learns based on essay examples – domain-representative text. These essays are divided in quality categories, and a text to be graded is compared to them and rated with the same grade that the most similar essays. The main idea of LSA is an collecting of various contexts of a word and thereby word meaning similarity and possible text patterns where the word may occur. LSA can be used for estimation of word-word, word-phrase and phrase-phrase similarity. It is important that for this LSA doesn't use any external knowledge. The method consists of the following steps:

- a text must be represented as a matrix, in which each row is a unique word and each column – context of the corresponding word. For example, for a set of sentences {("The sun is bright.", "S1"), ("Such a bright, bright student!", "S2"), ("Lying student is lying in the sun.", "S3"), ("That's a lie!", "S4")} it may take the form shown in Table. 3.4 (note: the sentences went through lemmatization and stop words removal, because only the meaningful words are of interest). This example also shows how important it is to have a large text corpora in order to use LSA efficiently: in these four sentences words "bright" and "lie" were used in completely different meanings and thereby it is not possible to derive much sense from them.

Table 3.4: LSA matrix example

	S1	S2	S3	S4
sun	1	0	1	0
bright	1	1	0	0
student	0	1	0	0
lie	0	0	2	1

- Single Value Decomposition (SVD) is applied to the matrix. In SVD a rectangular matrix $\boldsymbol{M}$ of a size $m \times n$ is decomposed in the following way:

$$\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*, \tag{3.5}$$

  where $\boldsymbol{U}$ is a unitary matrix of a size $m \times m$, $\boldsymbol{\Sigma}$ – diagonal matrix that contains non-negative real numbers of a size $m \times n$ and $\boldsymbol{V}^*$ – conjugate transpose of a unitary matrix of a size $m \times n$. SVD-decomposition for the matrix above is shown below:

$$\boldsymbol{U} = \begin{pmatrix} \mathit{sun} & -0.46 & 0.65 & 0.67 & 0.51 \\ \mathit{bright} & -0.12 & 0.81 & 0 & -0.57 \\ \mathit{student} & -0.02 & 0.43 & -0.67 & 0.61 \\ \mathit{lie} & -0.88 & -0.28 & -0.33 & -0.21 \end{pmatrix} \tag{3.6}$$

$$\boldsymbol{\Sigma} = \begin{pmatrix} 2.46 & 0 & 0 & 0 \\ 0 & 1.7 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0.24 \end{pmatrix} \tag{3.7}$$

$$\boldsymbol{V}* = \begin{pmatrix} S1 & S2 & S3 & S4 \\ -0.24 & -0.06 & -0.09 & -0.36 \\ 0.65 & 0.73 & -0.15 & -0.16 \\ 0.67 & -0.67 & 0 & -0.33 \\ -0.28 & 0.15 & 0.4 & -0.86 \end{pmatrix} \tag{3.8}$$

- Next step is dimensionality reduction: one can removes coefficients from $\boldsymbol{\Sigma}$ and keeps only as much coefficients as much dimensions are needed. In the case, for example, we can take two dimensions, so 1 and 0.24 can be nullified. In this case initial matrix takes the form shown in Table 3.5. After nullification the SVD equations will take the following form:

$$\boldsymbol{U} = \begin{pmatrix} \mathit{sun} & -0.46 & 0.65 \\ \mathit{bright} & -0.12 & 0.81 \\ \mathit{student} & -0.02 & 0.43 \\ \mathit{lie} & -0.88 & -0.28 \end{pmatrix} \tag{3.9}$$

$$\boldsymbol{\Sigma} = \begin{pmatrix} 2.46 & 0 & 0 & 0 \\ 0 & 1.7 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{3.10}$$

$$\boldsymbol{V}* = \begin{pmatrix} S1 & S2 & S3 & S4 \\ -0.24 & -0.06 & -0.09 & -0.36 \\ 0.65 & 0.73 & -0.15 & -0.16 \end{pmatrix} \tag{3.11}$$

Table 3.5: LSA reduced dimensionality example

	sun	bright	student	lie
S1	0.59	0.43	0.04	0.33
S2	0.96	1.02	0.95	0.12
S3	0.49	0.53	0.06	-0.1
S3	0.21	-0.22	2.02	0.85

- Based on the decomposition the words are clusterized by sense over the chosen number of dimensions: words coordinates are taken from $\boldsymbol{U}$ and sentences – from $\boldsymbol{V}^*$. In Figure 3.5 one can see the results of LSA clusterized by k-Nearest-Neighbor (KNN) for the example we consider. Because of the size of the "dataset" the results are not satisfactory, but one can already see that the word "lie" right near "S3", where it occurred twice and relatively close to "S4" though it was not put into the same cluster. Word "bright" is in one cluster with both sentences it was in. "Student" is in one cluster with "S2" and not far from "S3", and "sun" is clusterized incorrectly, but it is between "S1" and "S3", so it could be classified better if more dimensions were used. SImilar sentences are also close to each other. Implementation of this example can be found via the link: `https://github.com/elhele/AI-assisted-short-answer-grading/blob/master/src/LSA_tiny_example.py`.

This method can fail in case an essay is too original, but in this case the text can be just marked as "unclassifiable" and be sent to the human teacher. The method was tested with Graduate Management Admission Test® (GMAT®) essays and showed correlation with human raters of 86%, whereas the correlation among human raters was 87% and 86% for different essay types. It is interesting that IEA was very positively met by students – 98% of the ones who tested it very willing to take more tests graded by this system. The system has a user-friendly interface –

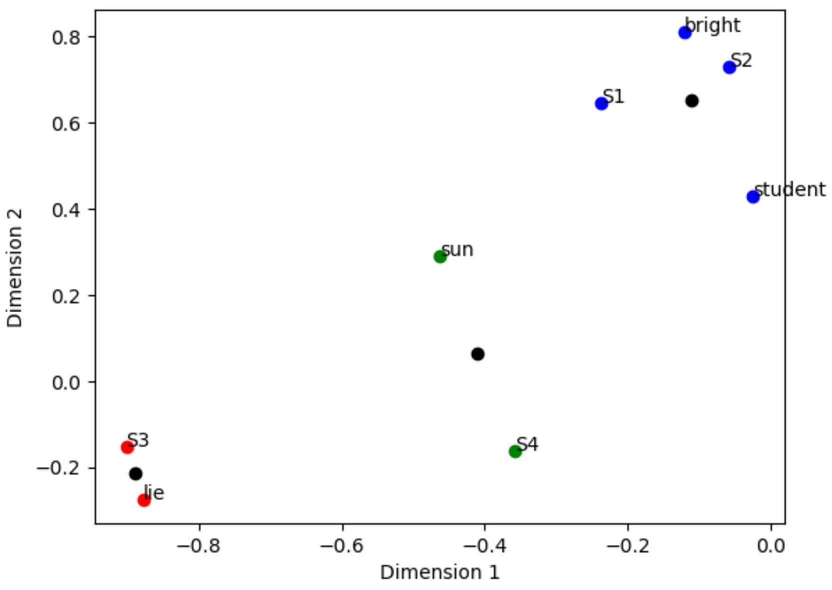
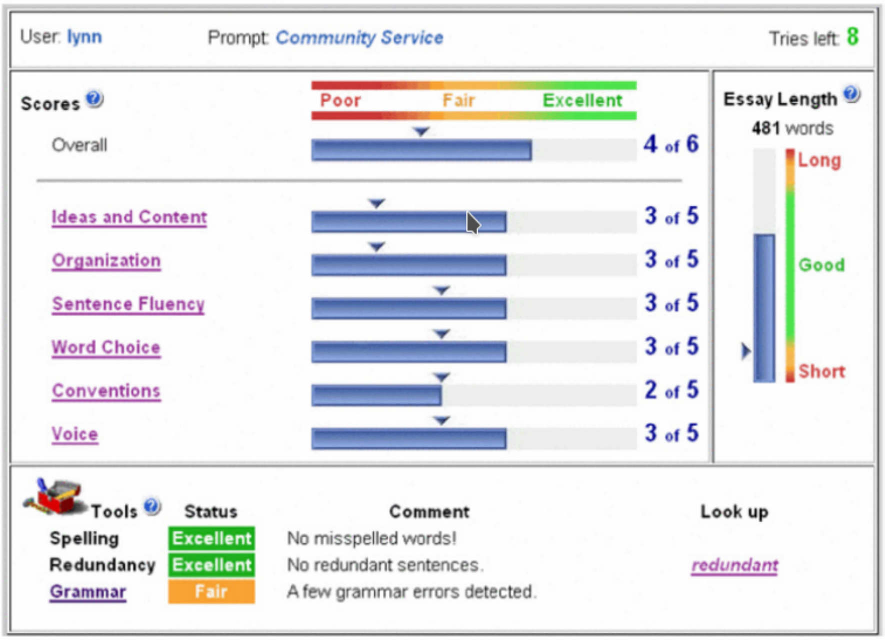


Figure 3.5: LSA clustering for the following set of sentences: set of sentences {(”The sun is bright.”, ”S1”), (”Such a bright, bright student!”, ”S2”), (”Lying student is lying in the sun.”, ”S3”), (”That’s a lie!”, ”S4”)}. Block dots are cluster centers; red, green and blue ones – three different clusters.

see Figure 3.6.

Next important step in AES was an integration of artificial intelligence. **IntelliMetric**^{TM} or its enterprise version MY Access!® didn’t follow coprus-based path of LSA work and, as PEG e-rater, mostly relies on NLP. However, it tries to emulate the work of human brain and uses CogniSearch^{TM} – an NLP AI system, which is able to perform POS tagging, morphological analysis, spelling recognition, and collocation grammar (the whole algorithm is, however, not published openly and protected by patents). It allows the system to go deeper into the language structure and rate essays in accordance to the norms of language – see Figure 3.7. For grading the systems requires a large set of pre-graded essays. The system was tested with GMAT® prompts and showed human-computer correlation of 97% on average. MY Access!® is widely used in USA: in California, Pennsylvania, Massachusetts, Virginia, Oregon, and Texas. Moreover, the following companies use the system



User: lynn Prompt: Community Service Tries left: 8

Scores

Poor Fair Excellent

Overall	4 of 6
Ideas and Content	3 of 5
Organization	3 of 5
Sentence Fluency	3 of 5
Word Choice	3 of 5
Conventions	2 of 5
Voice	3 of 5

Essay Length

481 words

Long

Good

Short

Tools	Status	Comment	Look up
Spelling	Excellent	No misspelled words!	
Redundancy	Excellent	No redundant sentences.	redundant
Grammar	Fair	A few grammar errors detected.	

Figure 3.6: Interface of IEA [37].

as well: The College Board, ACT, Harcourt Assessment, Inc., CTB/McGraw Hill, Thomson Learning. Microsoft, Apple Computer, AOL and Sun Microsystems.

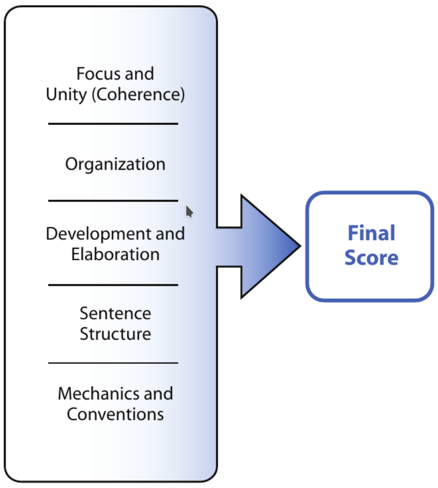

Focus and Unity (Coherence)
Organization
Development and Elaboration
Sentence Structure
Mechanics and Conventions
Final Score

Figure 3.7: IntelliMetric^{TM} grading model [37].

Bayesian Essay Test Scoring sYstem^{TM} (**BETSY**) [39] is a probabilistic AES system that performs text classification based on a large training corpus (1000 texts). To rate the essays BETSY performs the following:

- words and word pairs training
- database statistics evaluation
- uncommon words and word pairs removal
- stop words removal
- lemmatization
- removal of wrongly classified essays

The system uses such NLP features as word frequency in a certain context, specific words and phrases, sentence length, number of verbs, noun-verbs pairs. It combines PEG, early e-rater and LSA methods.

One of the important steps in AES was a kaggle competition **"The Hewlett Foundation: Automated Essay Scoring"** [7]. The winning algorithms are not published, but there is an open information about some principles of their work from a winner of a second place. His team uses the following features: number of sentences, lengths of sentences and words, and presence or absence of certain words. The features were combined with random forest (RF) and regression. An important feature of the approach is that the system can detect a work, which cannot be graded with a good confidence, send it to a human rater and include the grade he/her gave to this work into the train dataset. This method has a name "active learning" [6]. The dataset contains eight subsets with 12978 essays in a test set and 4254 in a train set – see "Essay_Set_Descriptions.zip" that can be downloaded on the following page: `https://www.kaggle.com/c/asap-aes/data`, last visited 20.04.2018, registration is required. Essays length is between 150 and 550 words [40].

The main idea of **Cummins**'s [41] work is a generalization of AES: he claims that the most hindering problem of most of the AES systems is that they are not general enough and require a lot of retraining each time a new essay badge has to be graded. Cummins addresses this problem by using multi-task learning, which grasps the general and specific features of the essays by using a shared-representation vectorized model of the essays, and preference-ranking model, which is based basically on a perceptron classifier. All essays are tokenized. The following features are used:

- POS tagging
- n-grams (1-3)
- unique words number
- words length
- sentences length
- linking words presence (”nevertheless”, ”moreover”, etc.)
- grammatical and spelling mistakes
- grammatical relations between constituents in a clause

A dataset from ”The Hewlett Foundation: Automated Essay Scoring” was used to evaluate the algorithm. Human-computer correlation is between 60% and 80% for different essay subsets.

The most well-known and widely used approaches has already been named, however, it is important to consider at least one of the newest approaches. One of the latest AES systems was developed by **Madala** [40] in 2018. Nine NLP features are used for the grading:

- reachness of the used vocabulary
- relation between a given word number limit and actual length of an essay
- length of sentences
- voice – active is preferred
- tense – must be consistent

- spelling
- grammar
- semantic similarity:
  - between the essay and a given topic
  - between the parts of essay – how coherent it is

The features are preprocessed and combined with the following machine learning techniques: kNN, linear regression and SVM. The system is implemented in Python and uses the following libraries: NLTK, SpaCy. A dataset from "The Hewlett Foundation: Automated Essay Scoring" was used to evaluate the algorithm. The results, depending on the essay type, are in range between 73% to 93%.

Table 3.6 summarized the AES systems' performance over the years.

Gibberish detection

The systems considered in a previous section show impressive results. However, even the most recent ones are vulnerable to various sorts of gibberish, because they mostly use the datasets with real tests written by the students who are were unaware of being graded automatically. Therefore, it is important to consider the topic of gibberish detection separately. Moreover, some of the gibberish detection techniques can be used for ASAG as well, though most of the ASAG systems don't put enough attention to this problem.

There are the following types of gibberish [42], [43] those also obey the NLP development curve from Figure 3.1:

- word level ("keyboard-banging") – words don't make sense and represent random letters sequences, e.g. "bzhyhlvyazsk". PEG was vulnerable even to this gibberish type. This can be spotted by detecting ill-formed words.

Table 3.6: Comparison of AES systems

Year	System	Approach	Accuracy
1968	PEG	comparison to human grading, multiple-regression	77%
1999	IEA	domain-representative text, LSA	86%
2003	IntelliMetric^{TM}	comparison to human grading, NLP + unknown patented techniques	97%
2005	BETSY^{TM}	large essays corpus, NLP-features, Bayesian theorem	80%
2012	The Hewlett Foundation: Automated Essay Scoring	comparison to human grading, NLP, RF and linear regression	81%
2013	E-Rater	comparison to human grading, multiple-regression + NLP	97%
2016	Cummins	comparison to human grading, NLP-features, multi-task learning, preference-ranking model on a perceptron	60%-80%
2018	Madala	comparison to human grading, NLP features, kNN, linear regression, SVM	73% to 93%

- sentence level (”word salad”) – essay contains real words, but in a random order, e.g. ”Went umbrella grump. Handbook keyboard trump. Begone.”. The previous and this gibberish types were solved in e-rater. It can be spotted by a sentence analysis and POS tagging.
- paragraph level (”copied-prompt”) – essay only or mostly consists of an essay topic copied multiple times. This gibberish type can be easily detected by spotting the phrases from the topic in the text.
- text level:

  - ”unexpected-topic” – essay is a sensible well-written text, but on a topic completely different from the required one, e.g. the topic was ”Character of Natasha Rostova from War and Peace” and the essay was about mating habits of baboons.
  - ”bad faith” – each part of essay separately makes sense and related to the main topic, but the parts are incoherent to each other.

For these gibberish types Madala offers a semantic similarity solution, the essay should be similar to the topic and its parts – to each other.

ASAG systems are unlikely to be vulnerable to ”keyword-banging” and ”unexpected topic”, because in one way or another they normally use keyword spotting. ”Copied-prompt” issues are also unlikely, because the question for ASAG must not contain too much information about the topic. However, ASAG can have problems with a word salad in case in contains important words from the example answer. ”Bad faith” answer can also be the case, because the sentences in an answer can be put in a wrong, non-sensible order.

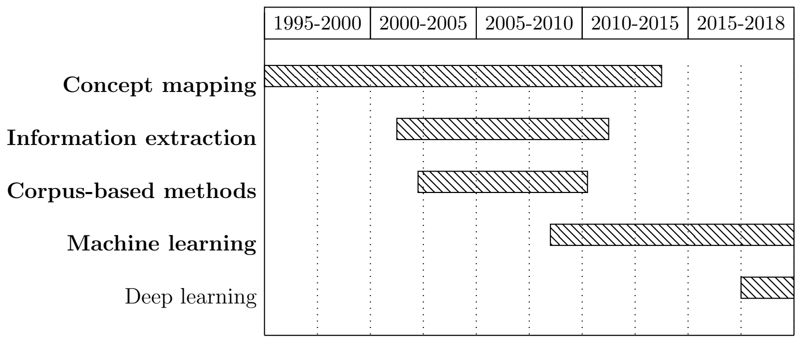
3.2.2 ASAG

ASAG is a form of automated grading of texts that focuses on short answers – from one sentence to one paragraph. Apart from that the answer must contain some external knowledge not mentioned in the question and the grader mostly concerns about the meaning of the answer and not about writing style, though the last point is under discussion, because a professor or teacher might want to include such grading criteria as spelling or grammar. ASAG has the same benefits in comparison to human graders those are mentioned in Table 3.3, but it has clearer criteria of correctness and it is less vulnerable to gibberish, because the ASAG solutions usually use more concrete criteria than AES. For example, if the simplest AES system was based only on text and word lengths, the simplest ASAG system would care about keywords and thereby it is less vulnerable to gibberish. Therefore, a simplified ASAG task is more feasible than simplified AES. ”Simplified” is because in the end both systems from different starting points are heading to the truly-human understanding of meaning and writing style, thereby becoming rather similar.

However, while the ASAG systems are still limited, it is important that answers and also questions should be formalized. The questions must meet certain criteria [1]:

- The question must not contain any information that can let a student to obtain the answer from the question, i.e. the student must be able to formulate the answer based only on his or her external knowledge;
- In general the question must require a strictly natural language answer. Current work focuses on the natural language part. However, in our project we are going to allow a combination of natural language, formulas, pictures and programming code.
- The question must be formulated in such a way that student could give an answer between one sentence and one paragraph length.
- The question must be formulated in the least ambiguous way to increase the level of close-ended responses.

When the correctly-formulated questions are prepared and the student answers obtained, one can start building an ASAG system. A general approach for development of such a system is shown in Fig.3.8. A historical overview of different approaches to ASAG-systems is shown Gantt chart below [1], [2]. The following part of the current work is dedicated to detailed consideration of these approaches.



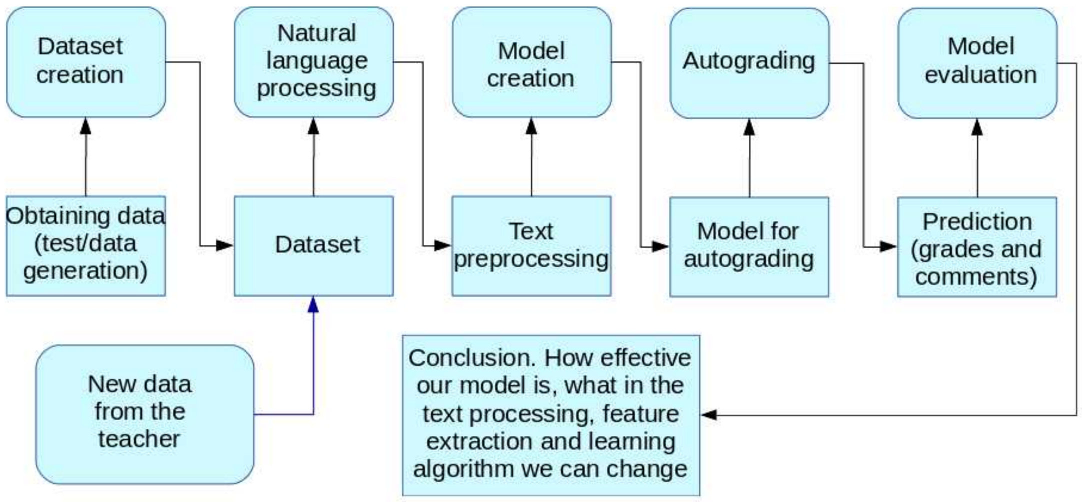

Dataset creation
Natural language processing
Model creation
Autograding
Model evaluation
Obtaining data (test/data generation)
Dataset
Text preprocessing
Model for autograding
Prediction (grades and comments)
New data from the teacher
Conclusion. How effective our model is, what in the text processing, feature extraction and learning algorithm we can change

Figure 3.8: A general workflow for an ASAG system developement [2].

Concept/facet mapping

The oldest direction in ASAG research is, in its pure form, mostly interesting from the historical point of view, for it is not being actively used nowadays. The main idea is searching for the presence/absence of certain concepts those students can mention on their answers. This can be used either on a sentence or facet level. The facet normally contains three or two words [2]. This approach is in a way similar to textual entailment, because entailment also concerns, if one sentence means the same that another, contains the same concept – entails it – or not.

The biggest disadvantage of this approach for ASAG is that it can detect the presence of the concept, but not if it was mentioned in a correct way [32]. For example, a student can write "Q: What is PCA? A: PCA is a method that affects dimensionality. It increases it.", which is incorrect, but the algorithm may still accept or partially accept this answer, because PCA really affects the dimensionality and it might match the concept, whereas the second part can be not accounted into the same concept estimation.

The first concept mapping system was developed by **Burstein** in 1996 [2]. It uses Lexical Conceptual Structure representation, which means that the system detects a presence of certain words and grammatical structures. The approach was rather limited, because it only allowed a certain type of question – the hypothesis-style, which means that the answer to this question must contain several explanations of the hypothesis.

Another mapping system is Automatic Text Marker (**ATM**). It operates with facets: ATM divides example and students' answers into pieces containing a couple of words each, and estimates the number of common concepts in those small parts of example ans students' answers. It is important that a higher weight can be assigned to the piece with a keyword [2].

A concept mapping system developed by ETS is called a Concept Rater or **c-rater**. It focuses on sentence-level comparison of example and students' answers. An example answer is intentionally formulated in such a way that each sentence contains only one concept. Sentence comparison is rule-based and includes a paraphrase recognizer that can distinguish syntactic variations, pronoun references, morphological variations, and synonyms. Moreover, it uses spelling correction [44].

Later concept mapping stated including machine learning. **Wang** in [73] compares pure heuristics-based concept mapping with data-driven classification by SVM and combination of those two methods. In a combination of methods SVM was used with bag-of-words features, whereas pure machine learning used unigrams and bigrams. The best result was reached using the combination of methods.

Table 3.7 shows the comparison of methods above. Accuracy between certain numbers means that the algorithm gives different correlation with human graders on different datasets.

Information extraction

Information extraction (IE) has already been cosidered in NLP section of the work. For ASAG it is also basically a pattern or fact template matching, template merging and resolution of referential ambiguities [2], [46]. It allows to extract a structures

Table 3.7: Concept/facet mapping systems

System	Year	Approach	Accuracy
Burstein	1996	lexical conceptual structure representation	48% to 83%
ATM	2001	weighted comparison of few-words facets	unknown
c-rater	2003	sentence-level paraphrase recognition	87% to 94%
Wang	2008	concept mapping with SVMs	92%

information from unstructured sources. A significant advantage of this approach is its robustness. To certain extend it is immune to grammatically wrong sentences and therefore requires less text preprocessing. The main IE approches are listed below [1], [46]:

- **parse tree matching** uses specific templates for correct and wrong answers. Each answer is being preprocessed and compared to templates with different grades. An example of such approach is **AutoMark** grader.
- **regular expression matching** applies a regular expression patter to all elements of the students' answers. Examples of such approach are **WebLAS** and **PMatch** graders:
  - WebLAS is an older system. It automatically extracts the patterns from example answers and then offers the grader to approve it and its synonyms and give a certain weight to each patter. Then the algorithm goes through the students' replies and matches the patterns. Thereby it is possible to give a wide range of grades – based on how much patterns were matched and their weight [2].
  - PMatch is suitable only for very short answers – not longer than one sentence. The system can match short and simple word sequences. It can inform students that words in their replys are either misspelled or unknown to the system and offer spelling correction options [1], [2].
- the main idea of **boolean phrase matching** is matching of key-phrases between an example and answers to grade. To be able to deal with synonyms and word forms the systems with boolean matching use thesaurus and lemmatization. Phrases with the same meaning are added to the needed sentence

representation as "or" and the required ones – as "and" [1]. An example of this is shown in Fig.3.9. It is taken from the work by **Thomas** [47], which describes an example of such an autograding system. Due to "or" the system has a certain flexibility, however, as a correct answer it could only accept a perfect match with one of the phrases from the tree.

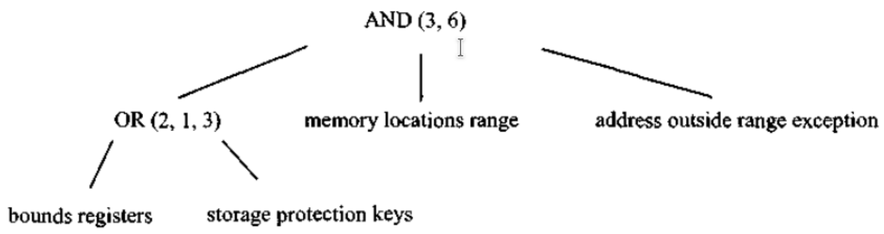


Figure 3.9: An example of boolean and/or tree for the answer that must contain the following: "bounds registers", "range memory locations,", "address outside range exception" [47]

- **syntactic pattern matching** is a variety of methods using different syntactic structures:
  - **auto-marking** requires specifically developed grammars those contain the hand-crafted patterns to match. This system is hard to use for people unfamiliar with the rules those were used for grammar construction [1].
  - In **Oxford-UCLES** system Pulman and Sukkarieh uses Hidden Markov part-of-speech model in combination with Noun Phrase and Verb Group finite state machine. Their approach excludes templates merging and ambiguity resolution, because they assume that the student replies in their dataset don't contain unnecessary information and ambiguous phrases. IE is combined with such machine learning methods as Naive Bayesian learning, inductive logic programming and decision trees. The best result was achieved with Naive Bayesian learning [46].
  - **IndusMarker** as well as auto-marking uses specifically constructed grammars for word- and phrase-level matching, but also includes spell checking and correction. The system developers introduced a "Question

Answer Markup Language" where the users can input the correct answers [2].

- **syntactic-semantic pattern matching**:
  - **eMax** uses both syntactic and semantic analysis. User of the system must provide one or multiple correct answers and highlight important elements of them. It is important that the system provides a confidence level, so a human grader could check the low-confidence answers him/herself [2], [1].
  - **FreeText Author** operates on a sentence-level. Every sentence is being semantically analyzed and the system returns a certain feedback that is compared to the feedback from the syntactic-semantic templates those are automatically created based on example answers. User of the system can specify keywords and accepted synonyms. The system has a user-friendly interface and it is easy to use [2], [1].
- **semantic word matching** is based on individual terms matching. An example of such a system is **Auto-Assessor**. To rate answers it uses such NLP tools as bag-of-words, WordNet.NET and SharpNLP. The system can only accept one example answer, but of this example it automatically generates a range of acceptable solutions. It grades based on can only grade one-sentence answers. Auto-Assessor has a user-friendly interface [2], [1].
- **Lexical Resource Semantics (LRS) representation matching** system – **CoSeC-DE** – was developed by Hahn in 2012. The system uses Corpus of Reading Comprehension Exercises in German (CREG) [48]. The answers are preprocessed using TreeTagger and dependency parser MaltParser and afterwards the system creates syntax-semantics-interface representations and then builds lexical resource semantics representations. Comparison of example and the answer to grade is performed by alignment afterwards. It is more accurate than bag-of-words model, because it allows to avoid consider a sentence of a higher level and avoid such problems: consider the sentences below: "CoSeC-DE is more advanced than AutoMark", "CoSeC-DE outperforms AutoMark", "AutoMark is more advanced than CoSeC-DE".

Bag-of -words would assume that the first and third sentences are the same and the second one is different, whereas the first two are the same and the third one contradicts them. CoSeC-DE evaluates these sentences correctly [49].

Table 3.8: Information extraction systems

System	Year	Approach	Accuracy
AutoMark	2002	parse tree matching	92.5%
WebLAS	2002	regular expression matching	unknown
PMatch	2012	regular expression matching	90%
Thomas	2003	boolean phrase matching	86%
auto-marking	2003	syntactic pattern matching	88%
Oxford-UCLES	2005	syntactic pattern matching	87%
IndusMarker	2008	syntactic pattern matching	96%
eMax	2009	syntactic-semantic pattern matching	unknown
FreeText Author	2009	syntactic-semantic pattern matching	unknown
Auto-Assessor	2011	semantic word matching	unknown
CoSeC-DE	2012	LRS representation matching	86.3%

From the Gantt chart above and Burrows's review paper one can see that this research direction tends to stay in the past as well as the concept mapping, but Hasanah [1] in 2016 year's paper argues that it is a valid method that can be used in combination with corpus-based methods, which can help to deal with synonyms, in the future.

Similarity measures

Next two sections are dedicated to corpus-based and machine learning ASAG systems. Both of them extensively use various text similarity measures. Therefore, this section focuses on summarization of the similarity measures those are essential for understanding of further sections.

BiLingual Evaluation Understudy (**BLEU**) was initially used in machine translation, but later it become used in all fields of machine text-processing including ASAG systems. The key concept of this metric is a modified unified n-gram precision (MUP) measure: all n-grams and number of their occurances are taken into account, n-grams of certain type are summed up and divided by the total number of n-grams. It is calculated in the following way [50]:

$$MUP_n = \frac{\sum\limits_{C \in \{AnswerSentences\}} \sum\limits_{n-gram \in C} count_{matches}(n-gram)}{\sum\limits_{C' \in \{AnswerSentences\}} \sum\limits_{n-gram \in C'} count_{matches}(n-gram')}. \tag{3.12}$$

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{1-r/c} & \text{if } c \leq r \end{cases}, \tag{3.13}$$

where BP is a brevity penalty, c – length of the student's answer and r – effective reference corpus length.

$$SIM_{BLUE} = BP \times e^{\sum_{n=1}^{N} w_n \times log(MUP_n)}, \tag{3.14}$$

where N is a maximum size of an n-gram and w_n – weight of the corresponding n-gram size. For estimation of BLUE similarity usually $log(SIM_{BLUE})$ is used.

An extended version of BLEU – Evaluating Responses with BLEU (**ERB**) adds recall accommodating to precision that uses modified brevity penalty (MBP) factor. The ERB similarity is calculated in the following way [51]:

$$SIM_{ERB}(answer) = MBP(answer) \times e^{\sum_{n=0}^{N} \frac{log(MUP(n))}{N}}, \tag{3.15}$$

where n – length of n-grams-sequence and N – highest possible value of n.

Recall-Oriented Understudy for Gisting Evaluation (**ROUGE**) was initially developed for evaluation of quality of the summaries. There are the following implementation of ROUGE [52]:

- ROUGE-N – estimates n-gram with $n > 1$ co-occurrences, similar to BLEU.
- ROUGE-L – finds and calculates the length of longest common subsequence of two answers.

- ROUGE-W – finds similar subsequences and takes them into account according to weights.
- ROUGE-S – uses skip-bigram.
- ROUGE-SU – uses the same idea that ROUGE-S, but also takes into account unigrams.

N-gram co-occurrence – number of similar n-grams [52]

$$SIM_{n-gram} = \frac{N_{sim}}{N_{total}}, \tag{3.16}$$

where N_{sim} – number of matching n-grams between the student's and example answers and N_{total} – total number of n-grams in student's answer.

Jaccard distance – percent of intersection of two sets (words of answers) [53]:

$$SIM_{Jaccard} = \frac{answer_1 \cap answer_2}{answer_1 \cup answer_2}. \tag{3.17}$$

Levenshtein distance is a "minimum number of operations needed to transform one string into the other" [54]):

$$SIM_{Levenshtein} = \frac{max(length(currentWord, keyword)) - minLev}{length(keyword)}, \tag{3.18}$$

where *minLev* is the distance between the current keyword the most similar word in a student's answer. If it is larger or equal to the minimal length of the two words, $SIM_{Levenshtein}$ nullifies.

Sørensen-Dice coefficient – a number of overlap words in two answers:

$$SIM_{Dice} = \frac{2 \times (answer_1 \cap answer_2)}{|answer_1| + |answer_2|}. \tag{3.19}$$

Jaro-Winkler distance calculates the number of similar characters in two answers taking into account their order. Similarity is calculated in the following way [55]:

$$SIM_{Jaro} = \begin{cases} 0 & 0 \text{ if } m = 0 \\ \frac{1}{3}(\frac{m}{answer_1} + \frac{m}{answer_2} + \frac{m-t}{m}) & \text{otherwise,} \end{cases} \tag{3.20}$$

where m is the number of matching characters and t – half of the total transpositions number.

Keywords coverage rate – basically it is a count of present and absent keywords or their synonyms [53]. It can be used, for example, with Levenshtein distance:

$$keywordCoverage = \sum_i SIM_{Levenshtein}/numberOfKeywords, \tag{3.21}$$

Similarity measures bases on the distance (*length*) between the concepts, e.g. from WordNet taxonomy, calculated based on node-counting with the following similarity metric:

$$maxsim(W, C) = maxSIM_x(W, w_i), \tag{3.22}$$

where C is a word class, W – a word of the class from the example answer, w_i – a word of the class from the student answer, SIM_x – one of the following similarity measures [15]:

- **shortest path**:

$$SIM_{shortestPath} = \frac{1}{length}, \tag{3.23}$$

  where *length* is the shortest path between the concepts calculated based on node-counting.

- **C-Rater** (has already been considered in "Concept/facet mapping" section):

$$SIM_{c-rater} = -log\frac{length}{2 \times D}, \tag{3.24}$$

  where *length* is calculated similar to the shortest path approach and D is the maximum depth of the taxonomy.

- **Lesk**:
$$SIM_{Lesk} = f(two\_concepts\_overlap). \tag{3.25}$$

- **Wu and Palmer**:
$$SIM_{Wu} = \frac{2 \times depth(least\_common\_subsumer)}{depth(concept_1) + depth(concept_2)}. \tag{3.26}$$

- **Resnik**:
$$SIM_{Resnik} = -logP(least\_common\_subsumer), \tag{3.27}$$
where $P(concept)$ is a probability to find an instance of the concept in a given corpus.

- **Lin**:
$$SIM_{Lin} = \frac{2 \times logP(least\_common\_subsumer))}{logP(concept_1) + logP(concept_2)}. \tag{3.28}$$

- **Jiang and Conrath**:
$$SIM_{Jiang} = -\frac{1}{logP(concept_1) + logP(concept_2) - 2 \times logP(least\_common\_subsumer)}. \tag{3.29}$$

- **Hirst and St-Onge**:
$$SIM_{Hirst} = f(strength\_of\_pair\_of\_synonym\_sets), \tag{3.30}$$
which is based on lexical chains detection in WordNet taxonomy.

Vector-based similarities require a prior vectorization of the words, e.g. with word2vec. The following vector similarities can be calculated [56]:

- **cosine** distance:
$$SIM_{cos} = \frac{\boldsymbol{answer_1} \times \boldsymbol{answer_2}}{||\boldsymbol{answer_1}|| \times ||\boldsymbol{answer_2}||}, \tag{3.31}$$

- **Euclidean** distance:

$$SIM_{Euclidean} = 1 - \sqrt{\sum_{i=1}^{k}(answer_{1_i} - answer_{2_i})^2}, \tag{3.32}$$

- **Manhattan** distance [53]. Each answer is represented by a vector, e.g. $\boldsymbol{answer_1} = (answer_{1_1}, answer_{1_2}, ..., answer_{1_n})$, where n is a length of the answer in words. The similarity is calculated between the student's answer and multiple example answers in the following way:

$$SIM_{Manhattan} = 1 - \sum_{k=1}^{n} |answer_{1_k} - answer_{2_k}|/m, \tag{3.33}$$

  where m is the total number of words in two answers. As a reference answer the most similar example answer is chosen, i.e. similarity between this and student answer would be taken into account for grading.

Corpus-based methods

This group of methods is widely used for analyzing of large texts. It has already been considered in AES section, but corpus-based methods can also be useful for ASAG, because they allows a better paraphrasing detection by efficient interpreting of synonyms [2]. These are statistical methods those use information from the large amount of text, so short answers should be preprocessed in order for corpus-based methods to be applied. Below the most significant corpus-based methods are considered.

Willow [57] is a system developed on the base of **Atenea** [51] – the first system that introduced ERB similarity measure. The system extensively uses various shallow NLP techniques, such as stemming and lemmatization and disambiguation. Basically, Willow is based on LSA and ERB with NLP preprocessing. The architecture of the system is shown in Fig. 3.10. It doesn't include LSA part explicitly, but it is normally being used in combination with LSA: LSA + Willow (NLP (stemming + removal of closed-class words + no disambiguation) + ERB).

The combination is calculated in the following way:

$$SIM_{Willow} = \alpha \times SIM_{PureWillow} + (1 - \alpha \times SIM_{LSA}), \tag{3.34}$$

where α is a parameter that defines the weights of LSA and Willow parts, the best result was reached with $\alpha = 0.489$. WordNet and BNC for English and EuroWordNet for Spanish were used for LSA.

The system had a user-friendly interface (see Fig. 3.11) and it worked with English and Spanish.

In **Short Answer Measurement of Text (SAMText)** author uses modified LSA. He finds the regular LSA unsuitable for ASAG due to the following reasons [58]:

- LSA is ineffective when the answer is shorter than around 200 words, which is often the case in ASAG.
- Relevant corpus doesn't always exist. LSA trained on general texts can be rather useless for specific areas and it is not always easy to find a large enough corpus for this specific area.
- Most of the existing LSA-based systems require large pregraded datasets. Therefore, a solution might only work well on some specific dataset and an attempt to extend it to a new area of knowledge would fail. That is one of the reasons why it doesn't make much sense to compare the accuracy of various ASAG systems given in the published papers – everybody uses their own dataset, which can be overly convenient for their method, and often there is no published code to test the solutions on the similar datasets.
- To use LSA one must find an optimal dimensionality.

Whereas a regular LSA uses a large corpus to model semantic relatedness, the given LSA modification is based on inverted index data structure, which requires general-corpus and topic-specific keywords to create the corpora. The advantage of the approach is that it allows to use a subject-specific sublanguage instead of the

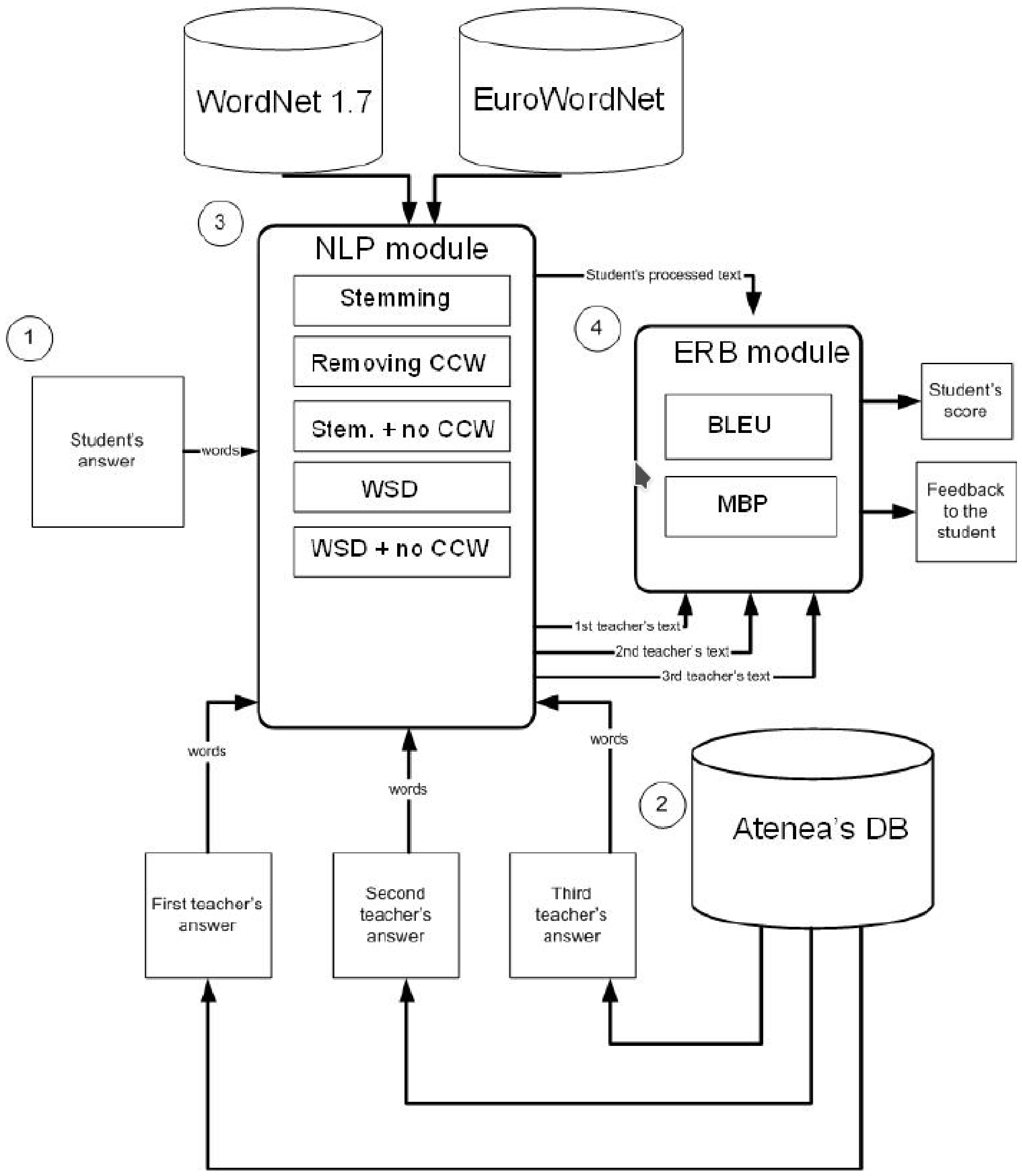
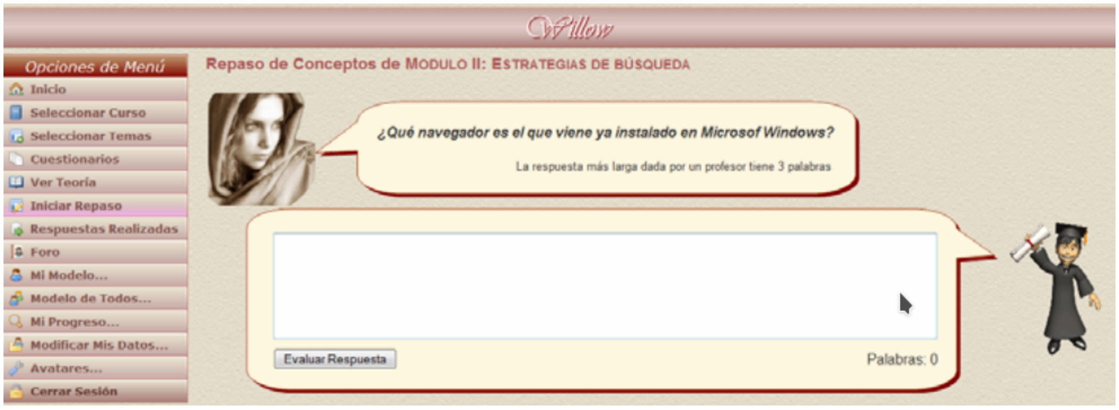


Figure 3.10: Willow's architecture [57], where "CCW" is "closed-class words" and "WSD" – word sense disambiguation.

whole language. Thereby, modified LSA can work properly with short answers and only requires a few example answers [2].



Repaso de Conceptos de MODULO II: ESTRATEGIAS DE BÚSQUEDA

¿Qué navegador es el que viene ya instalado en Microsof Windows?

La respuesta más larga dada por un profesor tiene 3 palabras

Figure 3.11: Interface of Willow [57].

Mohler [15] focuses on unsupervised ASAG systems and evaluates ten of them on the same dataset obtained from Data Structures course. The dataset consists of 80 questions, 2273 students' answers and two sets of grades given by two different people. It is important, because, as it was mentioned above, it is usually hard to compare the systems tested on completely different datasets. Moreover, the dataset was published and thereby the other authors later also have the opportunity to use this dataset to evaluate their algorithms [33], which allows to obtain comparable results. The following methods were evaluated:

- knowledge-based methods (see Equations 3.23-3.30):
  - shortest path, accuracy: 44.14%.
  - C-Rater, accuracy: 22.31%.
  - Lesk, accuracy: 36.3%.
  - Wu and Palmer, accuracy: 33.66%.
  - Resnik, accuracy: 25.20%.
  - Lin, accuracy: 39.16%.
  - Jiang and Conrath, accuracy: 44.99%.
  - Hirst and St-Onge, accuracy: 19.61%.
- corpus-based methods :

  - LSA (considered in details in ”AES” section) trained on BNC and English Wikipedia (September 2017 version). For implementation of LSA InfoMap package is used. Accuracy: 67.35% – on reduced computer science Wikipedia corpus with adding new good students’ answers to the examples list.
  - Explicit Semantic Analysis (ESA or ESA Wikipedia) – concept-vector features approach trained on Wikipedia (September 2017 version) corpus, each article is a concept vector. This algorithm is implemented by the authors themselves based on work by Gabrilovich and Markovitch [59]. Accuracy: 64.98% – on full Wikipedia corpus with adding new good students’ answers to the examples list.

It was shown that the accuracy of LSA strongly depends on the size of training corpus and on how close the corpus topic is to the topic of exam, i.e. the better results for the Data Structures test was obtained using domain-specific corpus that contained Wikipedia articles on computer science. However, for ESA the size is more important and it performs worse on a smaller domain-specific corpus. Also, in addition to the example correct answers the system added the best student answers to the list of examples, which helped to improve the quality of grading significantly.

It is interesting that the result of LSA, which was said to be unsuitable for ASAG in Bukai’s work [58] shows the best results in Mohler’s work [15]. However, the best result is 67.35%, therefore it is still questionable, if LSA an be used for ASAG.

As it was shown by Mohler [15], LSA performs better when trained on domain-specific corpus. Therefore, **Klein** [56] trains LSA on even more specific corpus that consists only of partially manually-graded student’s answers. To determine the distance between the answers, a linear combination of cosine and Euclidean similarities (see Equations 3.31-3.32) was chosen:

$$\begin{aligned} SIM_{combined} = \alpha \times SIM_{Euclidean}(\boldsymbol{answer_1}, \boldsymbol{answer_2}) + \\ +(1-\alpha) \times SIM_{cos}(\boldsymbol{answer_1}, \boldsymbol{answer_2}), \end{aligned} \tag{3.35}$$

where α is a parameter between 0 and 1 that allows to assign a weight to the distances. However, it was shown that the best result is reached when $\alpha = 1$, i.e. when only cosine similarity is used.

The system would find the graded answer, which is the most similar to the student's answer, and gives it the corresponding grade. It is important that in case if there is no graded assignment that is similar enough, the student's answer is marked as "unassigned". LSA here is used in combination with clustering. Three clustering methods to choose the assignments to grade manually were used:

- random choice of assignments to be graded manually
- k-means clustering to divide the students' answers into semantically-similar clusters: one or two answers from each cluster should be graded manually
- modified similarity-based Min-Max algorithm: first answer to grade manually is chosen randomly and the new ones are chosen in such a way that the answer should be the least similar to the closest answer in already built manually-graded set:
  $index = argmin_{index}(max_j(SIM_{combined}(\boldsymbol{answerToAdd_{index}}, \boldsymbol{gradedAnswer_j})))$

The Min-Max algorithm was shown to be the most efficient and k-means was on the second place: the total amount of answers was 49, Min-Max algorithm allowed to reduce the number of manually graded answers to 35, k-means – to 40 and random choice – between 41 and 45. The accuracy of the grading is fairly high – between 80% and 95%. However, to really reduce the teacher's workload, it would require a lot of pregraded works from previous exams those must be similar to the current one, otherwise it only allows to reduce the number of works to grade by around 20%-30%, which doesn't worth the effort.

The work of **Noorbehbahani** [52] deserves a special attention, for it contains a good description of various metrics:

- n-gram co-occurrence – see Equation 3.16. Accuracy: 48%.

- BLUE 3.14 – see Equation 3.14. Noorbehbahani mentions the following drawbacks of the system [52]:
  - BLUE only recognizes exactly-matching words, which means that it discards even misspelled words, and will certainly ignore synonyms;
  - all words have the same weights, which means that the essential for the meaning of the answer special terms would have as much meaning as articles or transition words;
  - the system compares the student answer with all reference answers and counts matches from different references;
  - brevity penalty punishes all answers those are longer than the reference ones, which doesn't always makes sense.

  Accuracy: 36%.

- M-BLUE – the system proposed by Noorbehbahani. It is based on the same idea, but includes the following modifications:
  - spelling correction;
  - acceptance of synonym-n-grams, understanding of paraphrasing: the system generates synonym example answers based on the given reference answers in such a way that it is as close as it is possible to the student answers, but still keeps the meaning of the reference answers;
  - weighting of the n-grams by their importance, i.e. special terms should have higher weights, whereas an n-gram like "and the" can have a nearly-zero weight;
  - the system compares student's answers with all the example answers, but the final similarity measure is calculated only with the most similar one.
  - brevity penalty is calculated based on weights, because the length of the answer doesn't help to estimate the correctness;
  - the final M-BLUE similarity measure is calculated in the following way:

$$SIM_{M-BLUE} = \lambda \times BP_r \times SIM_{M-BLUE} + (1 - \lambda) \times S_0, \tag{3.36}$$

where BP_r is a modified weighted brevity penalty, $SIM_{BLUE_r a}$ – weighted similarity measure between a student answer and one reference answer, S_0 – measure for calculating the common-words order similarity and λ is a similarity components weight parameter, which is taken in range between 0.5 and 1. In Noorbehbahani's work $\lambda = 0.85$ was chosen experimentally. The whole derivation of the equation 3.36 is intentionally omitted, because the meaning of the equation members is clear from the list items above and the formulas for them can be found in [52].

Accuracy: 85%.

- ERB – see equation 3.15. Accuracy: 41%.
- ROUGE. Accuracy: 22%.
- LSA. Accuracy: 51%.
- Combination of LSA and n-grams – answers are broken into sentences, each of them is a vector for LSA and closeness of the answers is evaluated as a cosin distance. Separately n-gram n-gram co-occurrence is calculated. After this, two similarities are combined:

$$SIM_{LSA+n-grams} = \frac{SIM_{LSA} + SIM_{n-gram}}{2}. \tag{3.37}$$

  The method can be modified by a more complex – weighted – calculating of the similarity. Accuracy: 59%.

The dataset used in this paper was obtained from a computer science e-learning course. It contained 45 questions, 300 example answers and 237 student's answers.

There were several important papers on corpus-based methods published 2012. The first one by **Li** [53] concentrates on various similarity metrics, which allows us to extend the list of the metrics given by Mohler and Noorbehbahani. The following similarity metrics are considered:

- cosine distance – see Equation 3.31. Accuracy: 75.3%.

- Jaccard distance – see Equation 3.17. Accuracy: 73.9%.
- Sørensen-Dice coefficient see Equation 3.19. Accuracy: 76.6%.
- Manhattan distance – introduced by Li for ASAG. See Equation 3.33. Accuracy: 76.2%.
- Keywords coverage rate – see Equation 3.21. Accuracy: 64.6%.

The similarity measures were tested separately and then combined with different weights those were determined by linear regression.

A distinctive feature of this paper is that they evaluate non written, but spoken answers, which makes preprocessing harder, and requires speech-recognition tools instead of, for example, spelling correction. However, the rest of the work is similar to the regular written ASAG. The author claims that the accuracy of the method is 77.8% and it is very impressive, eepesially taking into account that the correlation between the human raters was 80.5%. However, the used dataset was rather simple. Here are some examples of the answers: "Hello, Xiao Li! This is Bill speaking. What's up?", "Good idea! How much is a ticket?", "All rightSee you then! Bye!" [53]. It would be a valid assumption that the results on more complex dataset, e.g. Mohler's one, would be much lower, because even the keyword spotting alone has given 64.6% correlation with a human grader. Though the results of this work might be not very impressive, it is a good article to move from corpus-based methods to machine learning, because the approach of using various distances as different features will be helpful for machine learning.

Another paper published in 2012 by **Gomaa** [54] also uses a combination of several types of methods – on string-based and corpus-based. String-based methods are represented by the following:

- Character-based distance measures
  - Levenshtein distance – see Equation 3.18.
  - Jaro-Winkler distance – see 3.20.

  - Needleman-Wunsch algorithm – dynamical-programming algorythm initially used for protein or nucleotide sequences similarity calculations. It searches for the best alignment over the whole two answers.
  - Smith-Waterman algorithm – also from dynamic programming, looks for similarities over conserved domain of two answers.
  - n-gram co-occurrence see Equation 3.16.

- Term-based distance measures
  - Manhattan distance – see Equation 3.33.
  - cosine similarity – see Equation 3.31.
  - Sørensen-Dice distance – see Equation 3.19.
  - Euclidean distance – see Equation 3.32.
  - Jaccard distance – see Equation 3.17.
  - Matching Coefficient – number of non-zero values over the term axes those are similar for two answer-vectors.
  - Overlap coefficient – modified Sørensen-Dice distance: if only a subset of one answer is similar to the whole other answer, these answers are 100% match.

Corpus-based similarity methods are represented by the following:

- LSA
- ESA
- PMI-IR
- DISCO

ASAG is performed in three stages:

- answer similarities are calculated by all string-based methods above.
- DISCO similarities are calculated, at the same step lammatization and stop-words removal are made.

- combination of first two steps and deriving the grade based on maximum and average overall similarities

To evaluate the efficiency of the introduced methods, Mohler's dataset is used. The best human-computer correlation among string-based methods was achieved by n-gram co-occurrence – 39.8%. The best DISCO result was shown after training on Wikipedia corpus – 47.5%. The three methods mentioned above were combined and the total result reached 50.4%. This work shows not very high results in comparison to many other papers, however, there are several important points:

- it uses Mohler's dataset, which makes the result comparable. Moreover, the obtained result is not that low if we compare it to the other algorithms trained on the same data.
- The algorithm was trained on whole Wikipedia corpus. It is very likely that the result would be better if only computer science Wikipedia would have been used, as it helped to improve LSA's performance in Mohler's work.
- The work also shows the importance of combination of various metrics and methods and thereby leads to using them as features in machine learning.

Pribadi [72] uses Mohler's dataset with word overlap features based on Dice, Jaccard and cosine distances. It doesn't create much of a scientific contribution to what has already been done, however, an interesting analysis of the dataset structure was performed. It shows that a simple word overlap doesn't always help and its usefulness depends on the answer type. Some questions require a short and very concrete answer that cannot be paraphrased, e.g. "What are the access-specifiers that can be used in a C++ class definition?"-"Private and public.". In this case a very high accuracy can be achieved by using a word overlap approach. Yet, when a question implies a long answer that contains words those can be replaces by valid synonyms, word overlap may lead to false-negative results. For example, in a question-answer pair "What is the role of a prototype program in problem solving?"-"To simulate the behaviour of portions of the desired software product" "software product" can be replaced be "program" and simulate – by "mimic" without a significant sense alteration. An answer can even be formulated completely differently, e.g. "Abstraction and reusability", and still be correct. The examples are taken from Mohler's dataset. It shows an importance of usage of various features, not only word-overlap.

Table 3.9 shows the comparison of various corpus-based methods those are in part of the cases combined with various similarity metrics. A dataset column was not added intentionally, because most of the works use their own datasets. The works using Mohler's dataset marked as "(MD)".

Table 3.9: Corpus-based systems and metrics

System	Year	Approach	Accuracy
Atenea	2004	ERB + LSA	50%
Willow	2006	ERB + LSA + NLP	56%
SAMText	2006	modified LSA	69%-72%
Mohler	2009	LSA	67.35% (MD)
Klein	2011	LSA + clustering	80%-95%
Noorbehbahani	2011	M-BLEU	85%
Gomaa	2012	bag of words	50.4% (MD)
Li	2012	Jaccard, Dice and Manhattan distances, keywords coverage	77.8%
Pribadi	2017	Jaccard, Dice and cosine distances	unknown (MD)

Machine learning

Machine learning (ML) requires a preprocessing that would allow to retrieve features for further reduction of the ASAG task to regression or classification problem. Previous section of the work focused on corpus-based methods and similarity metrics. First works were mostly using one or two metrics combined with certain weights. However, the latest works started using a combination of several techniques with automatically chosen wights and it was shown that this approach gives better results. This led to an important idea of combination of several techniques as features for further machine learning. And moreover, to an idea of combination of several machine learning techniques in the later machine learning works.

The oldest ML system – **e-Examiner** [60] focuses various ROUGE metrics (considered in details in "Corpus-based methods and metrics" section) in combination with cosine distance and such NLP techniques as tocenization, morphological analysis and stopword finding and removal. The architecture of the system is shown in Fig. 3.12. It is a Web-based stand-alone service written in Java that uses MySQL to store the answers.

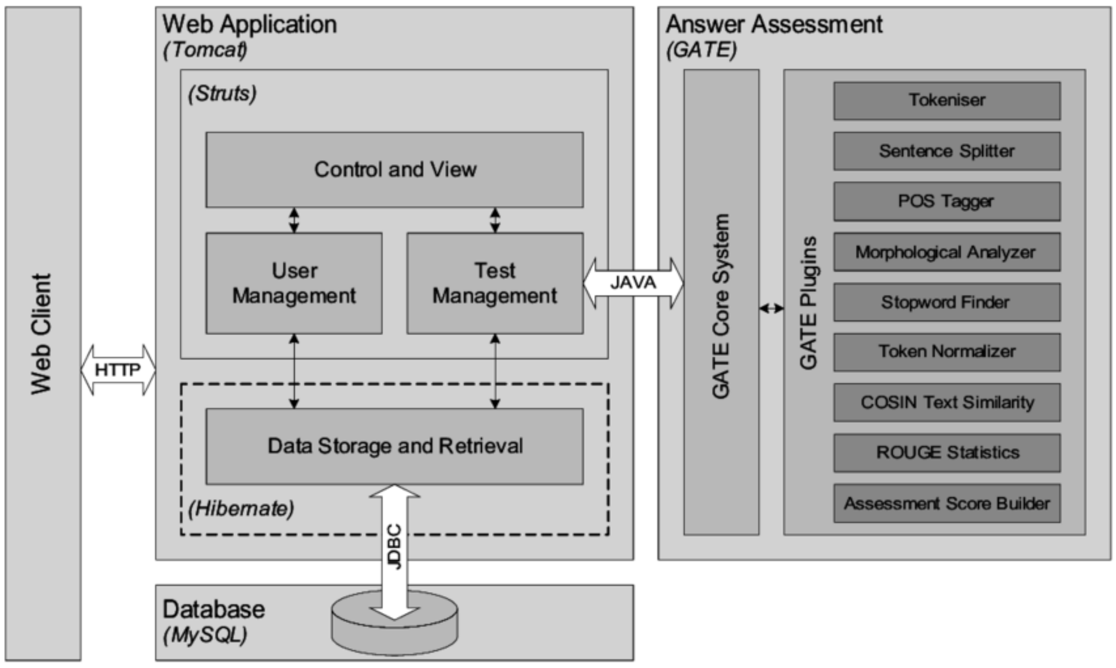
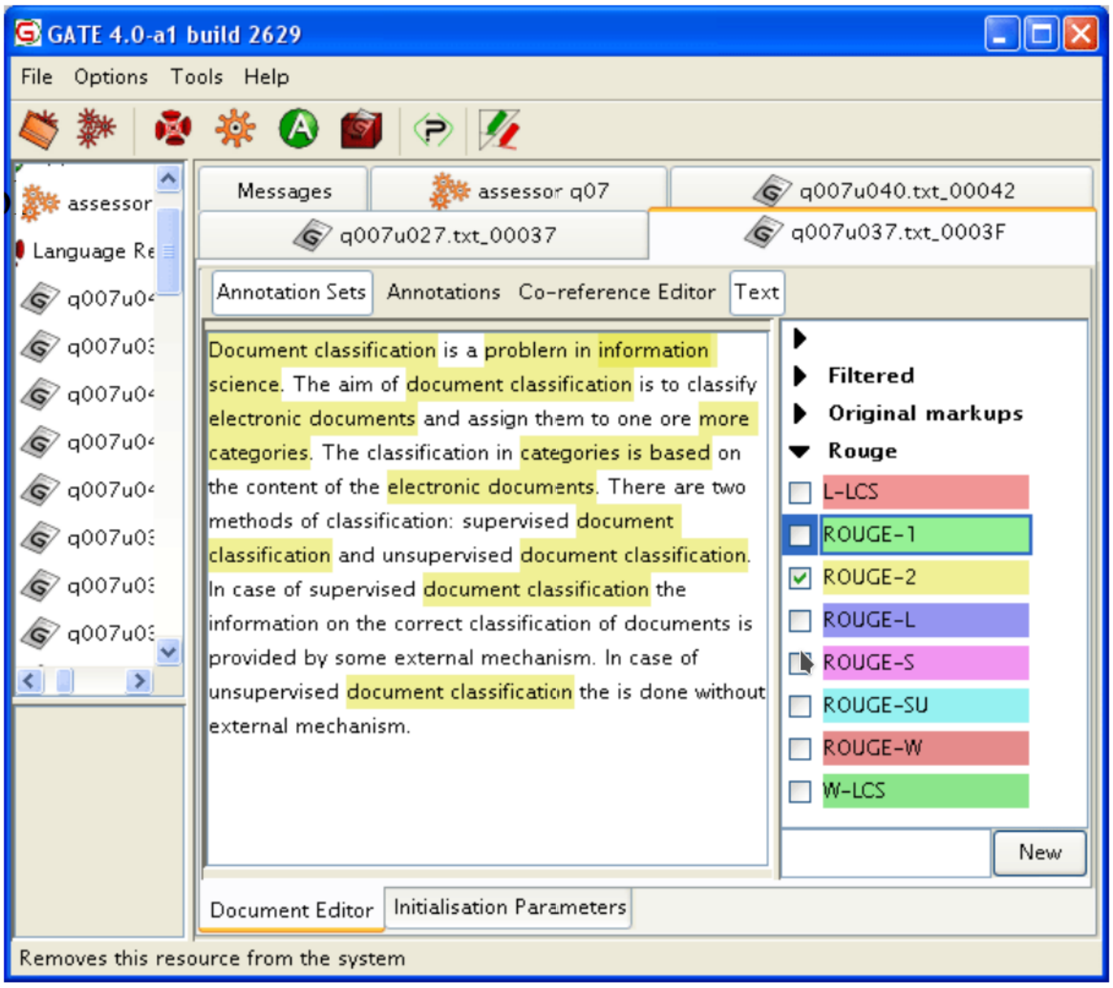


Figure 3.12: Architecture of e-Examiner [60].

The system has a user-friendly interface, where user can choose which of the ROUGE techniques should be applied – see Fig. 3.13.

To evaluate the performance of the system datasets from The Institute for Information Systems and New Media (IICM) and Faculty of Engineering, Al-Quds University, Jerusalem were chosen. IICM consists of 5 computer science questions, 5 example answers and 23 sets of students' answers. From the second dataset were taken 3 questions, 3 example answer and also 23 sets of student's answers. The



Document classification is a problem in information science. The aim of document classification is to classify electronic documents and assign them to one ore more categories. The classification in categories is based on the content of the electronic documents. There are two methods of classification: supervised document classification and unsupervised document classification. In case of supervised document classification the information on the correct classification of documents is provided by some external mechanism. In case of unsupervised document classification the is done without external mechanism.

Figure 3.13: Interface of e-Examiner [60].

best result – 81% – was achieved using a combination of all ROUGE techniques and cosine similarity.

Content Assessment Module (**CAM**) [61] is a system for grading of assignments on English as a Second Language (ESL) test. A dataset for evaluation of the system is taken from this test. It includes 75 questions 566 students' answers, having an example answer is not required. In this method the answers are divided into a graded and test sets and the answers from the test set are compared to the ones from the graded set using a number of NLP techniques: tokenization, lemmatization, spelling correction and noun phrase chunking – for preprocessing, and similarity metrics – for preprocessed text similarity calculation. For these tasks the following tools were used:

- MontyLingua
- PC-KIMMO
- Spell Checker Oriented Word Lists (SCOWL)
- TreeTagger
- CASS
- WordNet
- PMI-IR
- Stanford parser

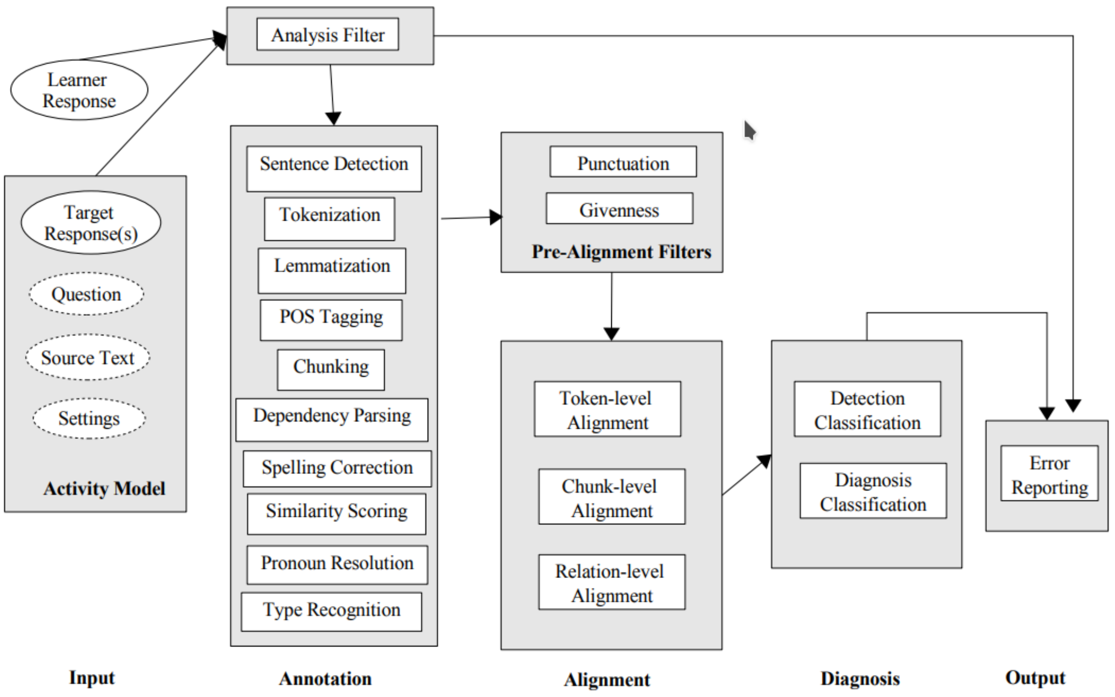


Figure 3.14: Architecture of CAM [61].

The following features were used [61]:

- Keyword overlap – percent of graded-test aligned keywords
- Graded and test overlaps – percents of aligned graded and test tokens
- Graded and test chunk – percents of aligned graded and test chunks
- Graded and test trigrams – percent of aligned graded and test trigrams
- Unigram match – percent of identical graded and test unigrams
- Similarity match – percent of similar tokens
- Type match – percent of type-resolved token alignments
- Lemma match – percent of lemma-resolved token alignments
- Synonym match – percent of synonym-resolved token alignments
- Variety of match – number of token-level alignments types

These features are used with Tilburg Memory-Based Learner (TiMBL) classifier and thereby reach 88% of human-computer correlation. TiMBL is based on KNN classifier modified in such a way that the nearest neighbors are searched in a decision-tree structure [62]. It is open-source and can be downloaded on the following website: `https://languagemachines.github.io/timbl/`. The last commit in TiMBL's github repository (`https://github.com/LanguageMachines/timbl`) was on 7th of February 2018, which shows that it is still relevant.

Comparing Meaning in Context (**CoMiC-EN**) [63] and **CoMiC-DE** [64] are systems for grading reading comprehension assignments in English and German correspondingly. The work is strongly based to CAM approach. It uses the same dataset (for CoMiC-EN) and ML method – TiMBL. However, there are the following differences:

- whereas there is only strict unigram-unigram and chunk-chunk comparison in CAM, CoMiC can also compare unigrams with chunks. It is important, because a several words often have a similar meaning to the one word, e.g. "green-eyed monster"-"jealousy".

- whereas CAM just removes punctuation and words already mentioned in a question from the answers, CoMiC keeps it to save the coherence of the sentence.

- an updated architecture. As a basis for its architecture CoMiC employs Unstructured Information Management Architecture (UIMA), which is an efficient architecture for working with large amounts of unstructured information – "whose intended meaning is only loosely implied by its form" [65], which is natural language texts, pictures, videos, etc. Structured information would be the one that has a direct relation between the information and form, e.g. a database. UIMA-based architecture of CoMiC is shown in Fig. 3.15.

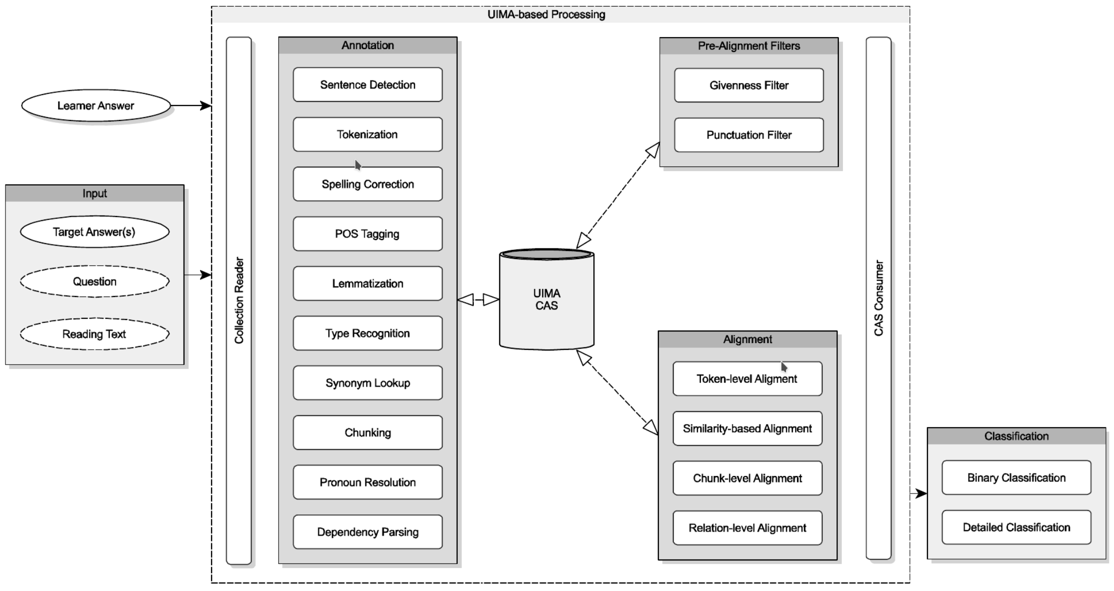


Figure 3.15: Architecture of CoMiC [63].

- CoMiC uses different NLP tools for their extraction:
  - Apache OpenNLP toolkit instead of MontyLingua and CASS
  - Morpha instead of PC-KIMMO
  - MaltParser instead of Stanford parser

  The rest of the tools (SCOWL, TreeTagger, WordNet and PMI-IR) remained the same.

- CoMiC uses the following features: "cosine distance, dot product, weighted overlap, Levenshtein distance, Euclidean distance, modified value difference, Jeffrey divergence and numeric overlap" [63]. All of the have already been described above.

- CoMiC-EN is not only capable to estimate if the answer was correct or not, bit also provide a detailed information, i.e. which concepts were missing and which were added.

CoMiC shows an insignificantly better result than CAM – 88.4% vs 88%. Nevertheless, the main idea of the paper was to introduce an UIMA architecture and new NLP tools for ASAG. It is also important that CoMiC-EN corpus is available for other researchers, which allows them to compare their results with the ones in the paper. It is not suitable for the current work though, because we focus on computer and natural science.

CoMiC-DE is conceptually identical to CoMiC-EN, but focuses on German languag and uses Corpus of Reading comprehension Exercises in German (CREG). It is the first work that considers automated grading of reading comprehension in German. One can suggest that English and German languages might require slightly different NLP-prepsocessing, however, in this paper the same tools are used and results of English and German ASAG were identical.

Hou's [31] is interesting because he introduces POS tagging those were not widely used in ASAG before. Hou preprocesses the students' answers by removing punctuation and digits, and lemmatization. The following features were used for SVM-classification:

- POS

- TF

- TF-IDF

- EV

For evaluation of the algorithm was taken a dataset that contains 9 questions with 38 student answers for each. Algorithm reached 71.9% of human-computer correlation.

In **Zbontar**'s [66] paper a stacking method is used. The main idea of the stacking is combining not only of several metrics, but also of several machine learning methods. The following preprocessing was used: lowercasing, removal of all punctuation, spelling correction, breaking text into 4- and 6-character-grams and creating bag-of-words model based on the n-grams. It is important that character n-grams were used, for usually sentences are broken into word-n-grams. Word-n-grams were already considered before. The following machine learning methods were used: Ridge Regression (RR), SVM – with lineal and RBF kernels, Gradient Boosting Machine (GBM), RF and KNN. Different combination of methods were used with different preprocessing:

- RR, GBM and RF – with 4-grams
- RR, SVM-RBF, GMB – with spelling correction + 4-grams
- RR, SVM-RBF – with 6-grams
- RR, SVM-RBF, KNN – with spelling correction + 6-grams
- SVM-RBF and -linear, KNN – with bag-of-word model based on 4-grams + latent semantic indexing with 200 components
- SVM-RBF, GBR, KNN – with bag-of-word model based on 4-grams + latent semantic indexing with 500 components

For evaluation of the method a Kaggle dataset [67] was used. As Mohler's dataset, this one also contains grades of two human graders. To avoid ambiguity, author just discards the answers graded differently. For the rest of the answers only the grade given by the first human was considered for the learning.

Another work from ETS organization by Madnani [68] describes a Reading for Understanding (**RfU**)framework for evaluation of summaries. The following features were used:

- BLEU. Contributes negatively to the score.
- ROUGE. Contributes positively to the score.
- CopiedSumm – number of occurrences of n-grams (n>2) common with the topic formulation divided by the length of a student's answer, basically copied-prompt gibberish detection. Contributes negatively to the score.
- CopiedPassage – number of occurrences of n-grams (n>2) common with the topic formulation divided by the length of topic formulation. Contributes negatively to the score.
- MaxCopy – maximum n in n-grams copied from the topic formulation. Contributes negatively to the score.
- FirstSent – number of task formulation sentences those have common n-grams (n>1) with the first sentence of the answer. Contributes positively to the score.
- Length in sentences. Not mentioned how it contributes to the score.
- Coherence – semantic similarity between the answer and an the topic – related to "bad faith" and unexpected-topic gibberish detection. Contributes positively to the score.

FirstSent and Coherence were shown to be least useful features, whereas BLUE, ROUGE and CopiedSumm – the most useful. It is different that the contribution of the features into the score are different for different summary topics. The features are combined with a logistic regression classifier. To evaluate the algorithm a dataset containing 2695 summaries of two topics – 1016 for the first one and 1679 for the second one – was used. The most important of the metrics mentioned here for ASAG related to technical subjects are BLUE, ROUGE and coherence.

SemEval [32] paper is a result the Joint Student Response Analysis and 8th Recognizing Textual Entailment Challenge. The approach is based on usage of TE. One of the important take-away messages from this paper is that the algorithm considers not just a student answer-example answer entailment, but a joint student answer plus question-example answer entailment. The reason is that the students tend to omit the information already given in an answer to save time, whereas an example question may contain a longer version of solution. Participants of the competition used such approaches as LSA and such metrics as BLEU, ROUGE and other similar ones. To evaluate the algorithm the Student Response Analysis corpus was used. The corpus consists of two parts: BEETLE contains 56 questions, each of which has one or several example answers, and around 3000 students' answers; and SCIENTSBANK that contains 197 questions, each of which has one or several example answers, and around 10,000 students' answers. The best result was achieved by ETS: 70% for BEETLE and 62% for SCIENTSBANK. ETS team used the following features: lexical similarity, n-grams similarity, BLUE and PERP. PERP is a similarity measure based on finding sequences of edit operations [69].

Sultan [33] uses the following methods:

- word alignment:
  - proportion of words semantically similar to an example answer in a student's answer
  - proportion of words semantically similar to a student's answer in an example answer

  It is important to use both features, because both, student's or an example answers, may have different number words those are not changing the meaning but change the percentage of aligned words, e.g. introductory words.

- Semantic vector cosine similarity based on word2vec [70]. Each sentence is represented as sum of lemmatized words' vectors.
- TF-IDF
- answer length

It is interesting that whereas SemEval combines a question and student's answer, Sultan subtracts the words used in a question from the students' answers, which would make the students' answers equally far from an example answer in case it contains a partial reformulation of the question. It can help to avoid copied-prompt gobberish, but may too much reduce an answer. A ridge regression model from Python's library sklear was used to classify the students' answers. The algorithm was tested on Mohler's dataset and human-computer correlation reached 63%.

Deep learning methods were already widely applied to AES, but so far there is only one paper by **Zhang** [71] that applies a deep learning approach to ASAG. The author proposes usage of Deep Belief Networks (DBN) that uses restricted Boltzmann machine and compares it with such classic classifiers as Naive Bayes, logistic regression, decision tree, artificial neutral network and SVM. An important idea proposed in the paper is usage not only of an answer model, as it was done in the papers considered above, but also taking into account a question and a student models. Question model is based on spotting various knowledge components in the questions. In a way in is similar to LSA approach, because a matrix with knowledge components-rows and question-columns is built for the question classification. Student model is based on Bayesian Knowledge Tracing. It means that the system keeps track of student performance during examination and considers it as a Markov Chain. Basically, this model shows the learning progress of a students in various knowledge areas. This model allows to predict, if student will answer a next question from a certain knowledge area correctly. Therefore, the following features were used:

- answer model:
  - TF-IDF match
  - TF-IDF-based cosine similarity
  - answer length
  - weighted text similarity
  - LSA
- question model:
  - question type based on question-knowledge components matrix

    - question difficulty – derived from a training corpus and based on a percent of good and bad grades given to answers to the question

- student model:
    - probabilistic prediction of answer correctness

To evaluate the algorithm a dataset collected from a college physics exam was used. It consists of 482 questions answered by 158 students. Each student answered around 280 questions. It was shown that DBN outperforms all classifiers except for SVM. Including question and student models helps to improve the results greatly. Moreover, it is interesting that answer model contributes to accuracy and precision (how many of returned values are true positives) of grading, whereas question and student models contribute more to recall (how many true positives are found). Using all three feature models combined with SVM Zhang achieved 85% of grading accuracy.

The most important ASAG works have already been considered, however, it is essential to have a glimpse to the latest 2018 works. **Wang** [73] represents the sentences as a weighted GloVe vector sum and trains the weights in such a way that the distance between similarly-graded answers is minimized and between differently-graded – maximized. After the training vectors are clusterized by KNN based on cosine distance between them. To evaluate the algorithm two datasets are used: X-CSD and SciEntsBank. X-CSD has 87 questions with 2442 students' answers and SciEntsBank has 135 questions with 4969 students' answers. The answers are graded from 0 to 5. Human-computer correlation is not provided, but root-mean-square error (RMSE) for X-CSD is 1.133, while state of the art RMSE is 0.887; RMSE for SciEntsBank is 0.606, while state of the art RMSE is 0.643.

3.3 State of the art summary

"Natural language processing development" section provides a brief overview of a language structure and the general trends of NLP development (Fig. 3.1). Understanding of linguistic base of NLP helps to see how it is possible to preprocess text in order to improve the process of grading, e.g. by spelling correction,

Table 3.10: Machine learning systems

System	Year	Approach (features + ML algorithms)	Accuracy
e-Examiner	2007	ROUGE (all), cosine similarity + linear regression	81%
CAM	2008	keyword, token, chunk and triple overlaps + + TiMBL-based classifier	88%
CoMiC-EN CoMiC-DE	2011	keyword, token, chunk and triple overlaps + + TiMBL-based classifier	88.4%
Hou	2011	POS tags, TF, TF-IDF, EV + SVM	71.9%
Zbontar	2012	bag-of-words of character-n-grams+ + RR, SVM-RBF, GMB, KNN, RF	73.892%
RfU	2013	BLUE, ROUGE, answer length, coherence, common with topic formulation n-grams + +logistic regression	52%-65%
SemEval	2013	Dzikovska team [2]: word overlap, Lesk (see Mohler's paper description), cosine distance, F1(WordNet relation)+ + C4.5 decision tree; ETS team [69]: lexical similarity, n-grams, BLUE, PERP + logistic regression	62%-70%
Sultan	2016	word alignment, cosine semantic similarity, answer length, TF-IDF + ridge regression	63%
Zhang	2016	cosine semantic similarity, answer length, TF-IDF, weighted text similarity, question difficulty, student-based answer correctness prediction + DBN, SVM	85%
Wang	2018	cosine distance + KNN	unknown, RMSE= =0.606

lemmatization and disambiguation. Moreover, it shows the importance of usage of sublanguage that can reduce the amount of words from thousands to several hundreds. Considering various fields of NLP is also important, because ASAG is not a separate scientific discipline – most or all of the ASAG techniques has already been used in other NLP-fields, for example, BLUE – for translation and ROUGE – for evaluation of summaries. It is always important to look for the new methods in various adjacent fields.

AES is the most adjacent to ASAG area. The very first AES and ASAG methods were on the opposite sides of automated grading methods: for ASAG it was keyword-spotting, a very specific approach, and for AES it was evaluation of the most general features – the length of words and sentences. However, in later works one can see how these two areas are moving towards each other and towards understanding of real meaning and style of the text. ASAG and AES share some common methods, e.g. LSA, but not all of the methods from AES are interesting for ASAG. For example, ASAG is much less than AES vulnerable to gibberish on the word level.

AES research started earlier than ASAG and that is why papers on this topic contain some useful general information. The first important idea is the comparison of human graders and computers. The correlation between the human rates was lower – around 76% – than even the very first AES system. ASAG questions are more straightforward, which usually rises the correlation between humans to 80%-85%, but it is still far from perfect. That is why 80%-90% result for an ASAG system is already satisfying. Moreover, it is not possible to reach 100% correlation with a human rater, because the human raters simply give different grades.

Another important moment is the reason why even the first AES system achieved good results. It was trained on rather convenient dataset. The system was vulnerable to any sort of gibberish, but was only trained and tested on a meaningful dataset. It shows that it is always important to test the method on different datasets and, to compare to other works, try to use the same datasets those were used in those works. Moreover, it is essential to keep in mind that certain types of gibberish, namely "word salad" and "bad faith", can become a problem not only for an AES system, but also for ASAG. The importance of datasets one can see also in further ASAG papers, when in the paper that presents the system it achieves 87% to 94% [44] and, when tested on another dataset by other author, it reaches only 22% [15]. That is why all the accuracies provided in the tables above or below don't provide enough clarity on how good the methods are. However, without testing teach system ourselves on different kinds of data, it is the only way to compare them at least approximately. The performance of the most significant ASAG systems is shown in Tables 3.7, 3.8, 3.9 and 3.10.

Current work focuses on machine learning methods for ASAG and that is why concept or facet mapping and information extraction are mentioned mostly to complete the picture. However, the "Corpus-based methods" section was considered in details, because many machine learning approaches use similarity metrics those were mentioned there. In table 3.11 the similarity measures are summarized. The accuracy column is taken from Noorbehbahani's (marked as "N") [52], Mohler's ("M") [15], Li's ("L") [53], Gomaa's ("G") [54] and e-Examiner ("E") [60] works. It is important not to forget that the accuracy was obtained on different datasets, and therefore the comparison is not entirely valid. Also, Noorbehbahani doesn't mention, which of the ROUGE methods was used, and it is confusing, because in other papers [60], [68] it performed significantly better.

All works except for one concentrate on various parameters of students' answers and their similarity to the example or well-graded answer. Moreover, answers to each question are usually considered separately. Zhang [71], however, offers to take into account difficulty of the questions and students' dynamic over the questions into account. These features can improve results significantly, but they also require a large amount of data from one examination.

Most of the authors use their own datasets. However, there is one similarity that occurs relatively often – a dataset contains grades from two human raters. As mentioned above, human raters' grades are often significantly different. Two main solutions to this problem was named:

- taking an averaging grade. This method is used in several papers [15], [33], however, human-computer correlation, which is calculated based on the average value, is questionable. Each human rater has a certain set of rules for grading. Averaging of grades derived from different sets of rules gives a senseless number, for grades here cannot be considered as linear functions.
- discarding questions graded differently [66]. It is reasonable, because these questions must be ambiguous. However, in certain datasets, e.g. Mohler's, it would cause discarding of almost half of the questions.

Table 3.11: Similarity metrics tested separately

System	Accuracy
BLEU	36% (N)
ERB	41% (N)
M-BLEU	85% (N)
ROUGE	22% (N), 81% (E)
n-gram co-occurrence	48% (N) 43.5% (G)
Jaccard distance	73.9% (L) 34.9% (G)
Levenshtein distance	64.6% (L), 31.7% (G)
Sørensen-Dice distance	76.6% (L) 37.9% (G)
Jaro-Winkler distance	24.5% (G)
cosine distance	75.3% (L), 37.7% (G)
Euclidean distance	33.8% (G)
Manhattan distance	76.2% (L)
shortest path	44.14% (M)
C-Rater	22.31% (M)
Lesk	36.3% (M)
Wu and Palmer	33.66% (M)
Resnik	25.20% (M)
Lin	39.16% (M)
Jiang and Conrath	44.99% (M)
Hirst and St-Onge	19.61% (M)
LSA	51% (N), 67.35% (M)
ESA	64.98% (M)

Both approaches have significant disadvantages, therefore it is one of the problem that should be addressed.

As for the machine learning, most of the papers use ready methods, such as SVM, RF, KNN and regression, those can be simply imported from standard libraries. The only paper that used a more complex method is the one by Zhang [71], however, proposed DBN didn't perform better than SVM. That shows another possible direction of ASAG improvement.

4

Solution

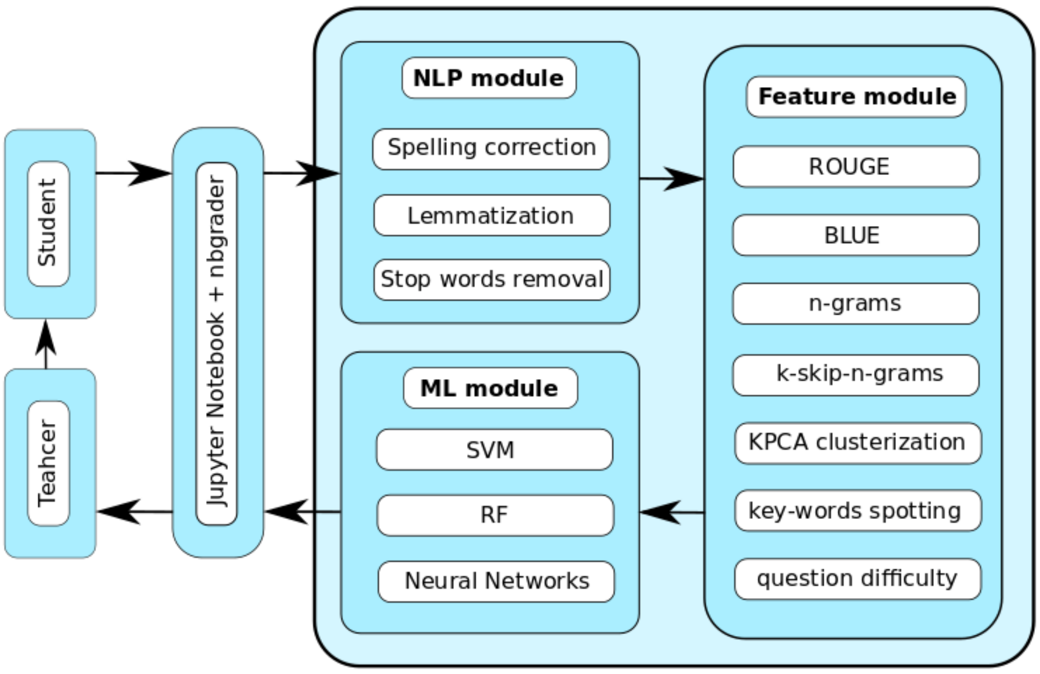


Figure 4.1: Architectureo of our system.

4.1 Proposed algorithm

4.2 Implementation details

5 Evaluation and results

Implementation and measurements.

6 Conclusions

6.1 Contributions

6.2 Lessons learned

6.3 Future work

English/German – the same NLP tools used in CoMiC-EN and -DE.

Bibliography

[1] U. Hasanah, A. E. Permanasari, S. S. Kusumawardani and F. S. Pribadi, "A review of an information extraction technique approach for automatic short answer grading," 2016 1st International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE), Yogyakarta, 2016, pp. 192-196.

[2] Burrows S., Gurevych I., Stein B. The eras and trends of automatic short answer grading //International Journal of Artificial Intelligence in Education. – 2015. – T. 25. – No. 1. – C. 60-117.

[3] Balfour, S. P. (2013). Assessing writing in MOOCS: Automated essay scoring and Calibrated Peer Review. Research & Practice in Assessment, 8(1), 40–48.

[4] S. Dikli An overview of automated scoring of essays //The Journal of Technology, Learning and Assessment. – 2006. – T. 5.– No. 1.

[5] Blood, Ian. "Automated essay scoring: a literature review." Teachers College, Columbia University Working Papers in TESOL & Applied Linguistics 11.2 (2012): 40-64.

[6] `https://github.com/VikParuchuri/vikparuchuri-affirm/blob/master/on-the-automated-scoring-of-essays.md`, last visited 19.04.2018

[7] `https://www.kaggle.com/c/asap-aes`, last visited 19.04.2018

[8] `https://github.com/edx/ease`, last visited 19.04.2018

[9] M. Shermis and B. Hammer. 2012. Contrasting state-of-the-art automated scoring of essays: analysis. Technical report, The University of Akron and Kaggle

[10] Alikaniotis D., Yannakoudakis H., Rei M. Automatic text scoring using neural networks //arXiv preprint arXiv:1606.04289. 2016.

[11] `http://humanreaders.org/petition/index.php`, last visited 19.04.2018

[12] Perelman, Les. "Critique of Mark D. Shermis & Ben Hamner,"Contrasting state-of-the-art automated scoring of essays: analysis"." Journal of Writing Assessment 6.1 (2013).

[13] Byrne, Roxanne, Tang, Michael, Truduc, John & Tang, Matthew. (2010). eGrader, a software application that automatically scores student essays: with a postscript on the ethical complexities. Journal of Systemics, Cybernetics & Informatics, 8(6), 30-35.

[14] Ramineni, Chaitanya, and David M. Williamson. "Automated essay scoring: Psychometric guidelines and practices." Assessing Writing 18.1 (2013): 25-39.

[15] Mohler, Michael, and Rada Mihalcea. "Text-to-text semantic similarity for automatic short answer grading." Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics. Association for Computational Linguistics, 2009.

[16] E. Cambria and B. White, "Jumping NLP Curves: A Review of Natural Language Processing Research [Review Article]," in IEEE Computational Intelligence Magazine, vol. 9, no. 2, pp. 48-57, May 2014.

[17] Page, E. B. (1966). "The Imminence of Grading Essays by Computer" Phi Delta Kappan, 47(5), 238-243.

[18] Chowdhury, Gobinda G. "Natural language processing." Annual review of information science and technology 37.1 (2003): 51-89

[19] Steedman, Mark. "Natural language processing." Handbook of perception and cognition (2nd ed.). Artificial intelligence (1996): 229-266.

[20] Ahmad, Saad. "Tutorial on Natural Language Processing." Artificial Intelligence 810.161 (2007).

[21] Raskin, Victor. "Linguistics and natural language processing." Machine Translation: Theoretical and Methodological Issues. Cambridge University Press, Cambridge (1987): 42-58.

[22] Von Goethe, Johann Wolfgang, "Faust", January 4, 2005 [EBook #14591], `http://www.gutenberg.org/files/14591/14591-h/14591-h.htm`

[23] Guthrie, David, et al. "A closer look at skip-gram modelling." Proceedings of the 5th international Conference on Language Resources and Evaluation (LREC-2006). sn, 2006.

[24] Joyce, J. (1969). Ulysses. London: Bodley Head.

[25] `https://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html`, last visited 19.04.2018

[26] `https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html`, last visited 19.04.2018

[27] `http://www.nltk.org/book/ch07.html`, last visited 19.04.2018

[28] Mitkov, Ruslan, Branimir Boguraev, and Shalom Lappin. "Introduction to the special issue on computational anaphora resolution." Computational Linguistics 27.4 (2001): 473-477.

[29] `https://nlp.stanford.edu/IR-book/html/htmledition/dropping-common-terms-stop-words-1.html`, last visited 19.04.2018

[30] `https://nlp.stanford.edu/software/tagger.shtml`, last visited 19.04.2018

[31] Hou, Wen-Juan, and Jia-Hao Tsao. "AUTOMATIC ASSESSMENT OF STUDENTS'FREE-TEXT ANSWERS WITH DIFFERENT LEVELS." International Journal on Artificial Intelligence Tools 20.02 (2011): 327-347.

[32] Dzikovska, M. O., Nielsen, R. D., Brew, C., Leacock, C., Giampiccolo, D., Bentivogli, L., Clark, P., Dagan, I., and Dang, H. T. (2013). SemEval-2013

Task 7: The Joint Student Response Analysis and Eighth Recognizing Textual Entailment Challenge. In M. Diab, T. Baldwin, and M. Baroni, editors, Proceedings of the Second Joint Conference on Lexical and Computational Semantics, pages 1-12, Atlanta, Georgia.

[33] Sultan, Md Arafat, Cristobal Salazar, and Tamara Sumner. ”Fast and easy short answer grading with high accuracy.” Proceedings of NAACL-HLT. 2016

[34] `http://www.linguatools.de/disco/disco_en.html`, last visited 19.04.2018

[35] Marti A. Hearst, ”The debate on automated essay grading” (http://www.knowledge-technologies.com/presskit/KAT_IEEEdebate.pdf)

[36] Landauer, Thomas K., Peter W. Foltz, and Darrell Laham. ”An introduction to latent semantic analysis.” Discourse processes 25.2-3 (1998): 259-284.

[37] Dikli, Semire. ”An overview of automated scoring of essays.” The Journal of Technology, Learning and Assessment 5.1 (2006).

[38] Rudner, Lawrence M., Veronica Garcia, and Catherine Welch. ”An evaluation of IntelliMetric^{TM} essay scoring system.” The Journal of Technology, Learning and Assessment 4.4 (2006).

[39] Rudner, Lawrence M., and Tahung Liang. ”Automated essay scoring using Bayes’ theorem.” The Journal of Technology, Learning and Assessment 1.2 (2002).

[40] Madala, Deva Surya Vivek, et al. ”An empirical analysis of machine learning models for automated essay grading.” PeerJ PrePrints (2018).

[41] Cummins, Ronan, Meng Zhang, and Ted Briscoe. ”Constrained multi-task learning for automated essay scoring.” Association for Computational Linguistics, 2016.

[42] Shermis, Mark D., and Jill Burstein, eds. Handbook of automated essay evaluation: Current applications and new directions. Routledge, 2013.

[43] `http://www.d.umn.edu/~tpederse/Courses/CS8761-FALL04/Project/Readme-SanLorenzo.html`, last visited 20.04.2018

[44] Leacock, Claudia, and Martin Chodorow. "C-rater: Automated scoring of short-answer questions." Computers and the Humanities 37.4 (2003): 389-405.

[45] Wang, Hao-Chuan, Chun-Yen Chang, and Tsai-Yen Li. "Assessing creative problem-solving with automated text grading." Computers & Education 51.4 (2008): 1450-1466.

[46] Stephen G. Pulman and Jana Z. Sukkarieh. 2005. Automatic short answer marking. In Proceedings of the second workshop on Building Educational Applications Using NLP (EdAppsNLP 05). Association for Computational Linguistics, Stroudsburg, PA, USA, 9-16.

[47] Pete Thomas. 2003. The evaluation of electronic marking of examinations. In Proceedings of the 8th annual conference on Innovation and technology in computer science education (ITiCSE '03), David Finkel (Ed.). ACM, New York, NY, USA, 50-54. DOI=http://dx.doi.org/10.1145/961511.961528

[48] `https://github.com/carolscarton/CREG-MT-eval`, last visited 19.04.2018

[49] Hahn, Michael, and Detmar Meurers. "Evaluating the meaning of answers to reading comprehension questions a semantics-based approach." Proceedings of the Seventh Workshop on Building Educational Applications Using NLP. Association for Computational Linguistics, 2012.

[50] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL '02). Association for Computational Linguistics, Stroudsburg, PA, USA, 311-318. DOI: https://doi.org/10.3115/1073083.1073135

[51] Prez, Diana, Alfio Gliozzo, and Carlo Strapparava. "Automatic Assessment of Students free-text Answers underpinned by the Combination of a BLEU-inspired algorithm and Latent Semantic Analysis." (2005).

[52] Noorbehbahani, Fakhroddin, and Ahmad A. Kardan. ”The automatic assessment of free text answers using a modified BLEU algorithm.” Computers & Education 56.2 (2011): 337-345.

[53] Yanling Li and Yonghong Yan, ”New similarity measures for automatic short answer scoring in spontaneous non-native speech,” International Conference on Automatic Control and Artificial Intelligence (ACAI 2012), Xiamen, 2012, pp. 1398-1402.

[54] Gomaa, Wael H., and Aly A. Fahmy. ”Short answer grading using string similarity and corpus-based similarity.” International Journal of Advanced Computer Science and Applications (IJACSA) 3.11 (2012).

[55] `https://rosettacode.org/wiki/Jaro_distance`, last visited 19.04.2018

[56] R. Klein, A. Kyrilov, and M. Tokman. 2011. Automated assessment of short free-text responses in computer science using latent semantic analysis. In Proceedings of the 16th annual joint conference on Innovation and technology in computer science education (ITiCSE ’11). ACM, New York, NY, USA, 158-162.

[57] Prez-Marn, Diana, and Ismael Pascual-Nieto. ”Willow: a system to automatically assess students free-text answers by using a combination of shallow NLP techniques.” International Journal of Continuing Engineering Education and Life Long Learning 21.2-3 (2011): 155-169.

[58] Bukai, Ohad Lisral, Robert Pokorny, and Jacqueline A. Haynes. ”Automated short free-text scoring method and system.” U.S. Patent Application No. 11/895,267.

[59] E. Gabrilovich and S. Markovitch. 2006. Overcoming the brittleness bottleneck using Wikipedia: Enhancing text categorization with encyclopedic knowledge. In Proceedings of the National Conference on Artificial Intelligence (AAAI), Boston.

[60] Guetl, Christian. (2008). Moving towards a Fully Automatic Knowledge Assessment Tool. International Journal of Emerging Technologies in Learning. 3.

[61] Bailey, S. & D. Meurers (2008). Diagnosing meaning errors in short answers to reading comprehension questions. In J. Tetreault, J. Burstein & R. D. Felice (eds.), Proceedings of the 3rd Workshop on Innovative Use of NLP for Building Educational Applications (BEA-3) at ACL'08. Columbus, Ohio, pp. 107-115.

[62] Daelemans, Walter, et al. "Timbl: Tilburg memory-based learner." version 6 (2007): 07-03.

[63] Meurers, Detmar, et al. "Integrating parallel analysis modules to evaluate the meaning of answers to reading comprehension questions." International Journal of Continuing Engineering Education and Life Long Learning 21.4 (2011): 355-369.

[64] Meurers, Detmar, et al. "Evaluating answers to reading comprehension questions in context: Results for German and the role of information structure." Proceedings of the TextInfer 2011 Workshop on Textual Entailment. Association for Computational Linguistics, 2011.

[65] Ferrucci, David, and Adam Lally. "UIMA: an architectural approach to unstructured information processing in the corporate research environment." Natural Language Engineering 10.3-4 (2004): 327-348.

[66] Zbontar, J. "Short Answer Scoring by Stacking" ASAP'12 SAS methodology paper (2012).

[67] `https://www.kaggle.com/c/asap-sas`, last visited 19.04.2018

[68] Madnani, Nitin, et al. "Automated scoring of a summary-writing task designed to measure reading comprehension." Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications. 2013.

[69] Heilman, Michael, and Nitin Madnani. "ETS: Domain adaptation and stacking for short answer scoring." Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013). Vol. 2. 2013.

[70] Baroni, Marco, Georgiana Dinu, and Germn Kruszewski. ”Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors.” Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Vol. 1. 2014.

[71] Zhang, Yuan, Rajat Shah, and Min Chi. ”Deep Learning+ Student Modeling+ Clustering: a Recipe for Effective Automatic Short Answer Grading.” EDM. 2016.

[72] Pribadi, Feddy Setio, et al. ”Automatic short answer scoring using words overlapping methods.” AIP Conference Proceedings. Vol. 1818. No. 1. AIP Publishing, 2017.

[73] Wang, Tianqi, et al. ”Identifying Current Issues in Short Answer Grading.” (2018).

[74] Ziai, Ramon, Niels Ott, and Detmar Meurers. ”Short answer assessment: Establishing links between research strands.” Proceedings of the Seventh Workshop on Building Educational Applications Using NLP. Association for Computational Linguistics, 2012.

[75] Roy, Shourya, Himanshu S. Bhatt, and Y. Narahari. ”An Iterative Transfer Learning Based Ensemble Technique for Automatic Short Answer Grading.” arXiv preprint arXiv:1609.04909 (2016).

[76] Yin, Wenpeng, et al. ”Comparative Study of CNN and RNN for Natural Language Processing.” arXiv preprint arXiv:1702.01923 (2017).

[77] Zanzotto, Fabio Massimo, Marco Pennacchiotti, and Alessandro Moschitti. ”A machine learning approach to textual entailment recognition.” Natural Language Engineering 15.4 (2009): 551-582

[78] `https://ipython.org/notebook.html`, last visited 19.04.2018

[79] `https://github.com/jupyter/nbgrader`, last visited 19.04.2018

[80] Brito, Eduardo & Sifa, Rafet & Bauckhage, Christian. (2017). KPCA Embeddings: an Unsupervised Approach to Learn Vector Representations of Finite Domain Sequences.

[81] Cover, T.M., Geometrical and Statistical properties of systems of linear inequalities with applications in pattern recognition, 1965 (hghlighted words are exchanged)

[82] `https://www.researchgate.net/project/reading-group-machine-learning-AI`, last visited 19.04.2018