Towards the Automatic Classification of Student Answers to Open-ended Questions

by

Jesus Gerardo Alvarado Mantecon

Thesis submitted to the University of Ottawa
in partial fulfilment of the requirements for the
Master of Computer Science degree.

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

One of the main research challenges nowadays in the context of Massive Open Online Courses (MOOCs) is the automation of the evaluation process of text-based assessments effectively. Text-based assessments, such as essay writing, have been proved to be better indicators of higher level of understanding than machine-scored assessments (E.g. Multiple Choice Questions). Nonetheless, due to the rapid growth of MOOCs, text-based evaluation has become a difficult task for human markers, creating the need of automated systems for grading.

In this thesis, we focus on the automated short answer grading task (ASAG), which automatically assesses natural language answers to open-ended questions into correct and incorrect classes. We propose an ensemble supervised machine learning approach that relies on two types of classifiers: a response-based classifier, which centers around feature extraction from available responses, and a reference-based classifier which considers the relationships between responses, model answers and questions.

For each classifier, we explored a set of features based on words and entities. For the response-based classifier, we tested and compared 5 features: traditional n-gram models, entity URIs (Uniform Resource Identifier) and entity mentions both extracted using a semantic annotation API, entity mention embeddings based on GloVe and entity URI embeddings extracted from Wikipedia. For the reference-based classifier, we explored fourteen features: cosine similarity between sentence embeddings from student answers and model answers, number of overlapping elements (words, entity URI, entity mention) between student answers and model answers or question text, Jaccard similarity coefficient between student answers and model answers or question text (based on words, entity URI or entity mentions) and a sentence embedding representation.

We evaluated our classifiers on three datasets, two of which belong to the SemEval ASAG competition (Dzikovska et al., 2013). Our results show that, in general, reference-based features perform much better than response-based features in terms of accuracy and macro average f1-score. Within the reference-based approach, we observe that the use of S6 embedding representation, which considers question text, student and model answer, generated the best performing models. Nonetheless, their combination with other similarity features helped build more accurate classifiers. As for response-based classifiers, models based on traditional n-gram features remained the best models.

Finally, we combined our best reference-based and response-based classifiers using an ensemble learning model. Our ensemble classifiers combining both approaches achieved the best results for one of the evaluation datasets, but underperformed on the remaining two. We also compared the best two classifiers with some of the main state-of-the-art results on the SemEval competition. Our final embedded meta-classifier outperformed the top-ranking result on the SemEval Beetle dataset and our top classifier on SemEval SciEntBank, trained on reference-based features, obtained the 2nd position.

In conclusion, the reference-based approach, powered mainly by sentence level embeddings and other similarity features, proved to generate the most efficient models in two out of three datasets and the ensemble model was the best on the SemEval Beetle dataset.

Acknowledgement

First and foremost, I would like to thank God for providing me with the strength and courage to keep moving forward during my time in the Master's program.

I would like to express my gratitude and appreciation towards my supervisor Dr. Amal Zouaq for her relentless support and guidance. Thank you for believing in me even when I didn't believe in myself. Also, I would like to extend my gratitude to Dr. Jelena Jovanovic and Dr. Jenny McDonald for their feedback and advice during my assistantship.

Thanks to my colleagues at the Tamale lab for their insights, willingness to help and suggestions.

Last but not least, to my rock through the storm, my mother. For your caring and kindness, for the many phone calls and endless words of encouragement, thank you.

TABLE OF CONTENTS

Abstract ii

Acknowledgement iv

TABLE OF CONTENTS v

LIST OF FIGURES viii

LIST OF TABLES ix

LIST OF ABBREVIATIONS xi

Chapter 1. Introduction 1

1.1 Goal 2

1.2 Contributions 2

1.3 Outline 3

Chapter 2. Background and Related Work 4

2.1 Massive Open Online Courses 4

2.1.1 Formative assessments in MOOCs 5

2.1.2 Summative assessments in MOOCs 6

2.2 The Automatic Short Answer Grading task 7

2.2.1 ASAG Approaches 7

2.2.2 Reference-based Approaches 10

2.2.3 Response-based Approaches 12

2.2.4 Hybrid Approaches 14

2.2.5 Artificial Intelligence Techniques for ASAG 15

2.2.5.1 Classification 15

2.2.5.2 Natural Language Processing 16

2.2.5.3 Semantic Web 17

2.2.5.4 Vector space models 19

2.2.5.4.1 N-grams 19

2.2.5.4.2 Word Embeddings 19

2.2.5.4.3 Sentence Embeddings 21
2.2.6. Summary 22

Chapter 3. Research Methodology 24
3.1 General Architecture 24
3.2 Data pre-processing 25
3.3 Feature extraction 27
3.3.1 Response-based features 27
3.3.1.1 N-gram Features 28
3.3.1.2 Entity URI Features 28
3.3.1.3 Entity Mention Features 33
3.3.1.4 Entity Embedding Features 34
3.3.1.5 Mention Embedding Features 35
3.3.2 Reference-based features 36
3.3.2.1 Word Similarity Features 36
3.3.2.2 Semantic Similarity Features 37
3.3.2.3 Answer Embeddings Features 38
3.4 Classifiers, combination and stacking 40

Chapter 4. Experiments and Results 42
4.1 Datasets Description 42
4.1.1 INSIGHT dataset 42
4.1.2 SemEval datasets 45
4.1.3 Datasets features statistics 48
4.2 Evaluation Metrics 52
4.3 INSIGHT Dataset Results 53
4.3.1 Response-based models 53
4.3.2 Reference-based models 57
4.4 SemEval Beetle Dataset Results 59
4.4.1 Response-based models 59
4.4.2 Reference-based models 63
4.5 SemEval SciEntBank Dataset Results 65
4.5.1 Response-based models 65

4.5.2 Reference-based models 68
4.6 Feature Combination and Stacking 70
4.7 SemEval State-of-the-art Comparison 75
Chapter 5. Discussion 78
5.1 Classification Algorithms 78
5.2 Response-based and Reference-based Models 78
5.2.1 Semantic Similarity vs Word Similarity 79
5.2.1 Embedding Representations 80
5.3 Stacking Efficiency 81
5.4 Limitations 82
5.5 Future Work 84
Chapter 6. Conclusion 85
References 87
Annex I 93
Annex II 109

LIST OF FIGURES

Figure 1. Word2Vec architecture models extracted from (Mikolov, Le, & Sutskever, 2013) 21
Figure 2. InferSent's sentence encoding-based model extracted from (Conneau, Kiela, Schwenk, Barrault, & Bordes, 2017) 22
Figure 3. General pipeline 25
Figure 4. Responses pre-processing. 26
Figure 5. Feature set extraction overview. 27
Figure 6. TF-IDF vectorization process for unigrams. 28
Figure 7. Semantic web annotations extraction process. 30
Figure 8. TAGME Semantic Web resource association. 30
Figure 9. DBpedia Spotlight Semantic Web resource association. 31
Figure 10. TF-IDF vectorization process for entity URI features. 32
Figure 11. Example of entity mention and entity URIs using DBpedia Spotlight. 33
Figure 12. Word2Vec, Wiki2Vec & Zhou's model comparison. 34
Figure 13. Extraction and aggregation of Entity Embeddings 35
Figure 14. Word overlap between model answer & student answer. 37
Figure 15. Semantic Similarity feature extraction process for a student answer. 38

LIST OF TABLES

Table 1. Datasets overview 42
Table 2. Survey questions on INSIGHT dataset 44
Table 3. Answers distribution and statistics on INSIGHT dataset 44
Table 4. Sample of student answers on the INSIGHT dataset 45
Table 5. Labeling scheme for SemEval datasets per task. 46
Table 6. Sample questions, student answers and model answers for the SemEval datasets 47
Table 7. SemEval datasets description per label 48
Table 8. Number of N-grams found on each dataset (train set and test set combined) 49
Table 9. Five most frequent n-grams per dataset 50
Table 10. Number of Entity URIs found on each dataset (train set and test set combined when applicable) 50
Table 11. Number of Entity Mentions found on each dataset (train set and test set combined when applicable) 51
Table 12. Percentage of Entity URIs found in Wiki2Vec model 51
Table 13. Percentage of Entity Mentions found in GloVe model 51
Table 14. Accuracy & macro average f1-score on INSIGHT dataset (10FCV) with response-based features 54
Table 15. SVM results per label on INSIGHT dataset (10FCV) with response-based features 56
Table 16. RFC results per label on INSIGHT dataset (10FCV) with response-based features 57
Table 17. Accuracy and Macro-F1 results on INSIGHT dataset (10FCV) with reference-based features 59
Table 18. Accuracy & macro average f1-score on SemEval Beetle test set with response-based features 60
Table 19. SVM results per label on SemEval Beetle test set with response-based features 62
Table 20. RFC results per label on SemEval Beetle test set with response-based features 63
Table 21. Accuracy and Macro-F1 results on SemEval Beetle test set with reference-based features 64
Table 22. Accuracy & macro average f1-score on SemEval SciEntBank test set with response-based features 66

Table 23. SVM results per label on SemEval SciEntBank test set with response-based features 67
Table 24. RFC results per label on SemEval SciEntBank test set with response-based features. 68
Table 25. Accuracy and Macro-F1 results on SemEval SciEntBank Test Set with reference-based features 70
Table 26. Top performing models on the reference-based approach..... 71
Table 27. Stacking/combination of response-based models on INSIGHT dataset (10FCV) 72
Table 28. Stacking/combination of response-based models on SemEval Beetle test set 72
Table 29. Stacking/combination of response-based models on SemEval SciEntBank test set 72
Table 30. Stacking of response-based and reference-based models on INSIGHT dataset (10FCV) 73
Table 31. Stacking of response-based and reference-based models on SemEval Beetle Test Set 74
Table 32. Stacking of response-based approach and reference-based approach on SemEval SciEntBank Test Set 74
Table 33. Ranking of top performing results on SemEval2013 datasets in terms of macro average f1-score 76
Table 34. Number of top performing models trained on each classifier per dataset 78

LIST OF ABBREVIATIONS

ACC	Accuracy
AES	Automatic Essay Scoring
ANN	Artificial Neural Network
API	Application Programming Interface
ASAG	Automatic Short Answer Grading
ASAP	Automated Student Assessment Prize
BLEU	Bilingual Evaluation Understudy Score
BOW	Bag of Words
BTK	Bayesian Knowledge Tracing
cMOOC	Connectivist Massive Open Online Course
CS	Cosine Similarity
DBN	Deep Belief Network
DT	Decision Tree
DUM	Dummy Classifier
EE	Entity Embedding
EM	Entity Mention
ETS	Educational Testing Service
EU	Entity URI
GBM	Gradient Boost Machine
GLOVE	Global Embeddings
HTTP	HyperText Transfer Protocol
IDF	Inverse Document Frequency
KNN	K-nearest Neighbors
LOD	Linked Open Data
LR	Logistic Regression
LSA	Latent Semantic Analysis
M. A	Model Answer
MCQ	Multiple Choice Question
ME	Mention Embedding

MJ	Mention Jaccard Index
ML	Machine Learning
MO	Mention Overlap
MOOC	Massive Open Online Course
NB	Naïve Bayes
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
OWL	Web Ontology Language
PEG	Project Essay Grading
PERP	Paraphrase Edit Rate with the Perceptron
POS	Part-of-Speech
Q	Question Text
RDF	Resource Description Framework
RF	Random Forest
RFC	Random Forest Classifier
RNN	Recurrent Neural Network
RR	Ridge Regression
S. A	Student Answer
SciEntBank	Science Entailments Corpus
SemEval	Semantic Evaluation Workshop
SNLI	Stanford Natural Language Inference
SOLO	Structured of Observed Learning Outcomes
SPARQL	SPARQL Protocol and RDF Query Language
SVD	Singular Vector Decomposition
SVM	Support Vector Machine
SVR	Support Vector Regressor
TF-IDF	Term Frequency – Inverse Document Frequency
TTR	Type Token Ratio
UA	Unseen Answers
UD	Unseen Domains
UJ	URI Jaccard Index

UO	URI Overlap
UQ	Unseen Questions
URI	Uniform Resource Identifier
WJ	Word Jaccard Index
WO	Word Overlap
xMOOC	Extended Massive Open Online Course

Chapter 1. Introduction

In the last decade, higher education has shown an increase in demand (Statistics Canada, 2017). Due to the inability of traditional higher education settings to suffice the need of accessible quality education at large scale, more and more people are recurring to Massive Open Online Courses (MOOCs) (Yuan & Powell, 2013). MOOCs offer quality online higher education courses at large scale in multiple domains. Most of these MOOCs are freely accessible but include a pay option for certificates issued by the course's offering institution. Additionally, universities have started to offer distance learning programs for the sake of increasing size and availability.

Within these platforms, the assessment of student knowledge still relies on machine-scored assessment such as multiple-choice questions (Yuan & Powell, 2013). Even though open-ended assessments have proven to be better at evaluating student's knowledge (Reilly, Stafford, Williams, & Corliss, 2014), they are not feasible at large scale due to the number of human resources needed to manually evaluate them. Thus, automatic methods are necessary to improve the diversity of questions and assessments in online and large class settings.

The automatic evaluation of open-ended assessments poses several difficulties ranging from the use of different question design taxonomies, unstandardized grading schemes and generalization of pipelines to multiple domains, to more complex challenges as providing systems with the ability to understand natural language. For this reason, the automatic evaluation of student (short) answers to open-ended questions, known as ASAG (automatic short answer grading), presents an important, yet to be solved, challenge.

1.1 Goal

Our goal is to examine the interest of machine learning algorithms with natural language processing (NLP) techniques for the assessment of student knowledge through the ASAG task. Following Sakaguchi's work (2015), we consider the ASAG task from two perspectives: a) Similarities among all student answers, called the *response-based approach*, and b) similarities between student answers and the expected answer (also known as model answer), called the *reference-based approach*.

In these two approaches, we extract statistical, textual and semantic features based on language modeling techniques and analyze their performance for the discrimination between correct and incorrect students' answers.

Our proposed pipeline encompasses a dual approach that selects models based on their individual performance and takes the top models in an ensemble learning method to answer the question: Which of the two approaches is more effective? And, can the combination of reference-based and response-based features produce a more accurate model for the ASAG task?

Within each approach, we address secondary questions regarding the predictive power of state-of-the-art features. We question the performance of sentence embeddings over word embeddings as high-dimensional vector representations of text, and the contribution of token level features based on semantic Web technologies to the ASAG task.

1.2 Contributions

Given the several different combinations of classification algorithms, feature sets and third party technologies used for the training of our models, the contributions of our research are as follows:

- We propose a domain-independent pipeline combining the predictive power of response-based and reference-based features for the ASAG task.
- We compare the performance of response-based and reference-based models.
- We analyze the impact of features based on semantic Web annotations in the context of ASAG in contrast to traditional NLP features (Alvarado et al., 2018).

1.3 Outline

In chapter 1, we give our motivation and goals behind our research. Chapter 2 describes the ASAG task, current state of the art pipelines for ASAG using machine learning techniques, and provides background on the useful features and the techniques used for their extraction. In chapter 3, we propose our methodology and detail each component. Then chapter 4 presents the results obtained by applying our pipeline on three different standard datasets, and compare them to state-of-art results. In chapter 5, we discuss these results and state the limitations and possible future work. Finally, chapter 6 concludes our research.

Chapter 2. Background and Related Work

In this section, we establish the context in which this research is conducted. First, we give a brief introduction to MOOCs. Then, we present the ASAG task, its multiple approaches and focus on machine learning and NLP techniques and features in ASAG systems. Finally, we offer background on the specific techniques and features used in our research.

2.1 Massive Open Online Courses

Massive open online courses offer an alternative for quality higher education at large scale providing tools for the teaching and formative or summative evaluation of student knowledge worldwide.

The first MOOC was developed by Stephen Downes and George Siemens in the year 2008. It was a course named '*Connectivism and Connected Knowledge'* offered by the University of Manitoba (Daniel, 2012) which had 25 on site students and reached 2,300 online students.

It was based on the connectivism theory of learning (Siemens, 2004), which emphasizes on learning through connections of decentralized knowledge nodes (concepts, fields, ideas, opinions, communities), making MOOCs (cMOOCs in this case), facilitators for learning and sharing of information (through RSS, forum discussions, blogs) with instructors guiding students through the knowledge offered.

Similarly, a different kind of MOOCs, called xMOOCs (extended massive open online courses), raised in popularity triggered by Stanford's *'Artificial Intelligence'* online course in the year 2011 (Rodriguez, 2012). It was attended by 160,000 students worldwide from which 20,000 successfully completed the course. This course was different than its precursors by offering lectures through

videos (hosted in *YouTube*[1]) and assessing participants' progress through assignments with deadlines, tests and exams, providing feedback and a final grade. Teachers in xMOOCs have the same role as in face-to-face class settings.

On the following year (2012), capitalizing on the success of Stanford's MOOC, many MOOC platforms were launched. *Udacity*[2] was launched in February 2012 founded by Sebastian Thrun, David Stavens and Mike Sokolsky. Then, in April, *Coursera*[3] went online founded by Stanford professors Andrew Ng and Daphne Koller. That same year, in May, the Massachusetts Institute of Technology (MIT) in partnership with Harvard University created *edX*[4] (Daniel, 2012). All these providers are still active, and keep on offering online courses from several institutions. In all these platforms, the assessment of student learning is one of their main challenges.

2.1.1 Formative assessments in MOOCs

Known as *"assessment for learning"*, formative assessment supports the student's learning by providing feedback and guidance during ongoing evaluations. It is a combined effort of classroom discussions, homework, teacher guidance and tests, to adapt the teaching process to each student as the course advances with prompt feedback as one of its principal components (Boston, 2002). While it has been found that these kinds of assessments have pedagogical benefits promoting critical thinking (Gikandi, Morrow, & Davis, 2011), their complexity in implementation and the unbalanced student/instructor ratio have prevented their proliferation (Xiong & Suen, 2018). Among the most popular assessments formats for implementation at large scale we can find

[1] https://www.youtube.com/
[2] https://www.udacity.com/
[3] https://www.coursera.org/
[4] https://www.edx.org/

question answering systems, inline multiple-choice questions (MCQ) and short-answer questions, discussion platforms and peer assessments (Xiong & Suen, 2018; Hollands & Devayani, 2014).

2.1.2 Summative assessments in MOOCs

Summative assessments are examinations evaluating the resulting knowledge obtained by students at the end of a course. Nowadays, most MOOCs (based on xMOOCs) rely on summative assessments for the evaluation of student knowledge using multiple choice questions (MCQ) (Xiong & Suen, 2018).

Since summative assessments have a higher impact on a student's academic life, their accountability (Xiong & Suen, 2018) is still a controversial topic. In this context, the evaluation of open-ended summative assessments presents a challenging task.

The interest in open-ended assessments is based on their suggested ability to capture higher levels of knowledge in learning outcomes (Reilly, Stafford, Williams, & Corliss, 2014). This is in part due to students' perception of open-ended assignments requiring higher levels of cognitive processing for which they prepare through deep-level approaches to learning (Scouller, 1998).

The idea of using computational methods for the evaluation of open-ended assessments dates to 1966 (Page, 1966), where Page acknowledged the need of automated systems due to the expensive and time-consuming effort needed for the marking of open examinations at large-scale.

Within automated open-ended assessments, we find short answer grading (ASAG) and essay scoring (AES). Burrows et al. (2014) differentiates their inputs based on their a) sizes: essays are larger than short answers, b) focus: AES evaluation is based on writing skills while ASAG evaluate answers' content (Burstein, Leacock, & Swartz, 2001), and c) openness, meaning that short answers' questions are not as broad as essay questions. Further categorization of systems for ASAG and AES divide their implementations into fully-automatic and semi-automatic (Alvarado

Mantecon, Abdi Ghavidel, Zouaq, Jovanovic, & McDonald, 2018). Semi-automatic approaches assist teachers in the correct and fast evaluation of student assessments, such as the work of Basu et al. (2013) which groups similar responses (response-based clustering), while fully-automatic systems do not rely at all on teachers' participation.

In this work, we propose a fully automatic ASAG system.

2.2 The Automatic Short Answer Grading task

The ASAG is a well-known task. Burrows et al. (2014), defines it as the automatic assessment of student knowledge through short answers to objective questions using computer programs. In their work, they define short answers as the output of questions that meet the following criteria: 1) The answer requires recalling knowledge not mentioned in the question. 2) The answer should be in natural language. 3) An answer's size between one sentence and one paragraph is required. 4) The answer must focus on domain content. 5) The question should be open but constricted to the question's objective. The required length for a free-form text produced by a student to be considered a short-answer varies. In some cases, answers' lengths from one word to hundreds (Shermis, 2015), or even few hundred words (Madnani, Loukina, & Cahill, 2017) are accepted.

2.2.1 ASAG Approaches

One of the main literature reviews for ASAG is the one of Burrows et al. (2014) who identified five categories to analyze 35 state-of-the- art ASAG systems:

- *Concept mapping*: Evaluation based on the identification of concepts at the sentence level within a short answer. Leacock and Chodorow (2003) exemplify this category. They first transform students' answers and expected answers by replacing morphological and syntactic variations of words with their base forms, replacing misspelled words and resolving pronouns.

Then they run a ruled-based algorithm for the alignment of similar concepts between the response and the reference answer also called the model answer.

- *Information extraction*: This category is characterized by matching patterns extracted from available material (course material, expected answers to questions on teacher designed assessments, etc.) to students' answers. For instance, Dessus et al. (2000), compare the semantic space obtained from a teacher designed text (called notion) with a Latent Semantic Analysis (LSA) to the semantic space produced by the student text (with LSA as well).
- *Corpus-based methods*: This includes methods that rely on the statistical analysis of large corpora for the identification of similar or related terms to enhance paraphrase recognition. An example is provided by Klein et al. (2011). They employed LSA and Singular Vector Decomposition (SVD) on their entire corpus of student answers to generate a semantic space on which they calculate distance measures between graded and non-graded assessments. Close assessments are then graded in a similar manner.
- *Machine learning*: These methods generate predictive or clustering models based on statistical features extracted from written text (natural language features). We delve into this category in the following sub-section.
- *Evaluation*: In this category, the authors describe competitions for the comparison of ASAG systems on shared datasets. They provide guidelines for multiple ASAG challenges, datasets and baselines. For example, in the year 2012, the William and Flora Hewlett Foundation launched the *Automated Student Assessment Prize* (*ASAP*), through *Kaggle*[5], with a competition for the automatic scoring of essays (*ASAP-AES*[6]) and another for the automatic

[5] https://www.kaggle.com/
[6] https://www.kaggle.com/c/asap-aes

scoring of short answers (*ASAP-SAS*[7]). The latter provided train and test set of over 17,000 and 5,000 answers respectively to 10 different questions, with an average length of 50 words per answer. In total, the contest registered 152 submissions making available the methodology papers and code of the top 5 performing systems. Another example is the annual *International Workshop on Semantic Evaluation*, specifically in the year 2013, known as *SemEval-2013*[8] (Dzikovska et al.,2013). It presented *The Joint Student Response Analysis and 8th Recognizing Textual Entail Challenge* which, under the assumption that a correct student answer would entail the expected answer to a given question, required participants to generate a system capable of classifying student answers into 5-way, 3-way and 2-way judgements (labels). The challenge provided two datasets, with their corresponding test sets of unseen answers and questions, with over 3,000 answers each.

More generally, ASAG systems can also be categorized as follows (Sakaguchi, Heilman, & Madnani, 2015) :

- *Reference-based approaches*: Use features based on the similarity or dissimilarity between students' answers and expected answers and questions.
- *Response-based approaches*: Extract features strictly from the corpus of students' answers and work with the assumption that answers with the same label have characteristics in common.
- *Hybrid approaches*: which combine the two previous ones.

[7] https://www.kaggle.com/c/asap-sas
[8] https://www.cs.york.ac.uk/semeval-2013/

2.2.2 Reference-based Approaches

In this section, we present ASAG systems that fall into Burrow's machine learning category and follows the reference-based approach.

Reference-based features, within ASAG, are produced by comparing a student answer and a model answer at different levels. They are commonly embodied by similarity measures and cover various aspects of similarity between texts: content, structure and style (Bär, Zesch, & Iryna, 2011).

A popular text similarity feature that convey content similarity based on counting matching sequences of words is BLEU (Papineni, Roukos, Ward, & Zhu, 2002). Originally intended to evaluate machine generated text translations, it calculates the ratio of unigrams in a candidate translation (student answers in the context of ASAG) that appear on the reference translation (model answer) limited by the unigram frequency on the reference translation.

PERP (Heilman & Madnani, 2012) is another well known similarity feature. It calculates similarity based on the sequences of edit operations neccessary to transform a student answer into the model answer. While the Levenshtein distance calculates the number of insertions, substitutions and deletions needed to convert one string to another, PERP uses a perceptron-based discrminative learning algorithm to learn the cost of a bigger set of edit operations (e.g stemming match, synonym alignment, number of word shifts, insertion, deletion, phrasal paraphrasing, substitution) to calculate a similarity measure between two sentences.

Other more complex similarity features based on semantics were also proposed (Bär et al. (2012): Pairwise word similarity computes the averaged inverse document frequency (IDF) from words in two matching WordNet graphs, one extracted from a student response and the other from the reference answer. Another reference-based feature, Explicit Semantic Similarity, calculates Pearson correlation between two weighted vectors of concepts generated by an encoder that uses

Wikipedia to find, extract and weight concepts according to their use (Gabrilovich & Markovitch, 2007).

Other works focused on features that compare stylistic aspects of text. Based on McCarthy and Jarvis (McCarthy & Jarvis, 2010), the sequential TTR (type-token ratio) correlation compares the texts' vocabulary richness, calculating the ratio of unique tokens present in a text sequence. Function word frequency is another stylistic feature used to calculate the Pearson correlation between two function word frequency vectors, one for each text to be compared.

Finally, some syntactic features were also explored. For example, Bär et al. (2012) used sequences of *n* part-of-speech (POS) to calculate a text structure similarity measure. These sequences are used to generate frequency vector representations of texts and compute the Jaccard coefficient (also known as Jaccard index) between them.

In the literature, we can find many examples of ASAG systems that rely on reference-based features. One of the earliest systems of ASAG using NLP reference-based features was PEG (Project Essay Grading), a regression model for the assessments of essays based on correlations of text-level features, syntactic and lexical features (e.g. average word length), between human graded essays and students' essays. Although its focus was AES, its influence on ASAG was important.

Zhang et al., (2016) compared six different machine learning algorithms for the ASAG task to a popular deep learning model at the time: Deep Belief Network (DBN). Their approach combined three models: 1) An answer model capturing similarities between student and model answers (length difference, matching IDF, TF-IDF based cosine similarity, LSA). 2) A question model representing the difficulty level of each answer based on knowledge components (commonly used terms within the domain). 3) A student model generated with Bayesian Knowledge Tracing (BTK)

which outputs the probability that a student learned a new concept based on past performance. They combined their features to train the following models: Naïve Bayes (NB), Logistic Regression (LR), Decision Tree (DT), Artificial Neural Network (ANN), Support Vector Machine (SVM) and a DBN. Their results showed DBN as their best resulting model followed by SVM.

In one of the most recent related work, (Saha et al., 2018), we observe a combination of token level (word) and sentence level features in the reference-based approach. They combine their own sentence embedding representation of an answer (based on InferSent[9]) with word overlap and histograms of partial similarity based on words and POS tags, to train a Multinominal Logistic Regression classifier.

2.2.3 Response-based Approaches

The ASAG systems presented in this section use ML techniques with NLP features following the response-based approach. The response-based approach's main idea is to extract features from the entire corpus of natural language student responses to open-ended questions. Some of the features described in the response-based models can also be used in the reference-based approaches. For example, the bag-of-words (BOW) model can be used to calculate Jaccard coefficient between model answer and student answer, but also to generate vector space representations from all student answers to train a classification model.

While reference-based features focus on similarity measures, response-based features thrive from the use of sparse and dense vector space representations of student answers to identify syntactic, semantic and lexical characteristics. However, the use of similarity measures is not limited to

[9] https://github.com/facebookresearch/InferSent

reference-based models. For instance, Klein et al. (2011) use LSA, trained on student answers alone, to calculate distances among labeled and non-labeled student answers.

Statistical language modeling, such as word and character n-grams, are also commonly used in response-based models. Sakaguchi et al. (2015) generated a sparse binary vector representation of each answer indicating the appearance of word and char n-grams from the entire corpus of student responses in the answer (among other features). In Alvarado Mantecon et al. (2018), we trained multiple classifiers using sparse vector space models holding TF-IDF values of all n-grams, from unigrams to trigrams and their combinations, extracted from all student answers (the full paper can be found in Annex I).

Lexical and syntactic features also play a part in the response-based approach. Based on (Sil, Shelton, Ketelhut, & Yates, 2012), the length of responses, as well as length of a paragraph (number of sentences), were used to detect disengagment of students usually associated to the generation of poorly composed answers (e.i short or repetitive). Part-of-speech tagging was used by (Hou, Tsao, Li, & Chen, 2011) to generate two features: POS tag of preceding word and POS tag of next word combined with other NLP features (TF-IDF n-grams, etc).

In the following, we present some ASAG systems that implement the response-based approach. McDonald et al., (2011) trained a Naïve Bayes (NB) classifier on several features with their best features being the combination of n-grams, word length, first word, and a hand-crafted regular expression.

The winner of the *ASAP-SAS* 2012, Luis Tandalla[10], used a Random Forest (RF) model trained on frequencies of unigrams, bigrams and trigrams to predict labels. Then, he used the model

[10] https://storage.googleapis.com/kaggle-competitions/kaggle/2959/media/TechnicalMethodsPaper.pdf

predictions, along with relevant unigrams, bigrams and a vector of binary elements, where each element represents a concept found in an answer, detected by specific regular expressions designed for each question, to train RF classifiers and Gradient Boost Machine (GBM) models to generate an average score.

The second top performing ASAG system, developed by Jure Zbontar[11], opted for a more complex approach using machine learning classifiers and regressors, but with simple NLP features. First, he trained multiple models on character four-grams, and six-grams models and trained variations using latent semantic indexing. The models used were: RF, GBM, Ridge Regression (RR), Support Vector Regression (SVR) and K-nearest neighbors (KNN). Then, he stacked them into a final Ridge Regression model.

In recent years, Madnani et al. (2017), used character and word n-grams, combined with answer length and triples capturing syntactic relationships between answers. They trained multiple RF and SVM classifiers and regressors trying to identify if a particular learner performed the best for ASAG following the response-based approach.

2.2.4 Hybrid Approaches

Hybrid approaches combine reference-based and response-based features. In the *SemEval-2013* competition, *ETS* (Heilman & Madnani, 2013) consistenly provided the best results. It consists of a Logistic Regression (LR) model with response-based features: N-grams (unigrams to trigrams), char n-grams (5 to 8 characters), and two sets of similarity features: 1) BLEU score between student answers and model answers, and among student answers. 2) PERP score between student answers and model answers, and among student answers.

[11] https://storage.googleapis.com/kaggle-competitions/kaggle/2959/media/jzbontar.pdf

Likewise, Sakaguchi et al., (2015) proposed the stacking of two Support Vector Regressors for the scoring of student answers, one for response-based features (response length, unigrams and bigrams, character n-grams, syntactic and semantic features), and one for reference-based features (BLEU similarity metric, cosine similarity between word2vec embeddings for answer representations, etc.), thus combining both approaches. Their stacking consisted in first training the response-based model and then using its predicted scores along with the reference-based features for the stacked SVR.

Another interesting hybrid proposal, (Roy, Bhatt, & Narahari, 2016), relies on a domain independent pipeline. They train an initial classifier on response-based features (TF-IDF BOW), a second classifier on a common shared feature space between domains generated by reference-based features (BLEU, LSA and word2vec similarities) and classical canonical correlation analysis (CCA), and ensemble them.

2.2.5 Artificial Intelligence Techniques for ASAG

In this section, we present a brief description of the multiple techniques used in our proposed pipeline. We start with the Machine Learning classification task and present the algorithms used. Then, we give a short background for our features including an introduction to the Semantic Web and its technologies, which were used for some of our features.

2.2.5.1 Classification

In Machine Learning, classification is the task of generating predictive statistical models, through supervised learning, capturing patterns mapping labels (classes) to data points (instances) (Osisanwo et al., 2017).

The typical classification problem consists on, given a labeled set of instances (train set), training a model (classifier) using a classification algorithm capable of assigning labels to new unseen instances (test set).

In this thesis, we rely on three classification algorithms:

- *Random Forest Classifier*: Tree-based ensemble meta-classifier. It consists of growing multiple independent decision trees with equally distributed random subsets of features. Due to its random nature, this classifier handles better the noise than other types of classifiers. It is also less prone to over-fitting (Breiman, 2001). We use this classifier as implemented by the *Scikit-Learn*[12] library.
- *Support Vector Machine*: This classifier projects training data into a high dimensional space and constructs a linear hyperplane that separates data points into two classes. This optimal hyperplane is defined as the hyperplane with the largest separation, known as maximal margin, between vectors of two classes. (Cortes, 1995).
- *Stacked Logistic Regression*: Ensemble meta-classifier as implemented by *Mlxtend*[13]. It trains multiple first-level classifiers on the complete provided dataset, and stack them by training a Logistic Regression meta-classifier on their predictions. Logistic regression is based on the sigmoid function, to handle outliers, and is suitable for categorical predictions.

2.2.5.2 Natural Language Processing

Natural Language Processing is an application of Artificial Intelligence and computational linguistics with the purpose of providing computers with the ability of understanding natural

[12] https://github.com/scikit-learn/scikit-learn
[13] http://rasbt.github.io/mlxtend/

language through features and techniques covering linguistic aspects such as morphology, syntax, semantics and discourse (Khurana, Koli, Khatter, & Singh, 2017).

Among its popular application, we can mention Information Retrieval, Question Answering, Text Summarization and Text Generation (Ledeneva & Sidorov, 2010).

Burrows et al. 2014 provide an overview of the most common NLP techniques used in ASAG systems. Spelling correction appears as the most frequently used lexical technique. Then, stemming (reduction of words to their root form by removing prefixes and suffixes) for morphological analysis and POS for syntactic techniques. Lastly, LSA was fairly used for the extraction of semantic features and the removal of punctuation marks was the most commonly used surface-level technique.

2.2.5.3 Semantic Web

The Semantic Web, as described by Tim Berners-Lee, is an extension of the Word Wide Web (created by him in 1989) intended to provide machine-readable meaning from the Web's content so that software agents could perform more sophisticated tasks using structured interlinked data (Berners-Lee, Hendler, & Lassila, 2001) known as Linked Data. In the Semantic Web, meaning is encoded as subject-predicate-object triples using RDF (Resource Description Framework) where subject and object are represented by a Universal Resource Identifier (URIs) reachable through the HTTP protocol. The Semantic Web relies on knowledge bases that can be queried using SPARQL (SPARQL Protocol and RDF Query Language). These knowledge bases can also be built using the Web Ontology Language (OWL), which in turn relies on RDF and extends RDFS capabilities, allowing better expressiveness (Horrocks, Patel-Schneider, F., & Harmelen, 2003).

One of the most popular and largest knowledge bases nowadays is DBpedia (Lehmann, et al., 2015), making its content available for everyone under the Linked Open Data (LOD) principle. It

extracts knowledge from Wikipedia's[14] (free online encyclopedia's) infoboxes and maps them to an ontology of 320 classes and 1,650 properties at the time of its original release (2012). The most recent version, DBpedia 3.2, covers 685 classes and 2,795 properties.

The development of Semantic Web knowledge bases triggered the emergence of tools known as semantic Web annotators, which, usually link concepts found in text to a resource on the Web (Gagnon, Zouaq, & Jean-Louis, 2013). In our work, we particularly focus on two of the main semantic annotators APIs: DBpedia Spotlight[15] and TAGME[16].

DBpedia Spotlight is a semantic Web annotator that enables the automatic linking of text documents to the Linked Open Data cloud using DBpedia URIs (Daiber, Jakob, Hokamp, & Mendes, 2013). Its pipeline is composed of four stages: spotting, candidate selection, disambiguation and configuration. In the spotting stage, based on the lexicon of DBpedia resources that point to Wikipedia pages, a string matching algorithm is used, LingPipe Exact Dictionary-Based Chunker[17], to identify mentions disregarding unnecessary common words (i.e. verbs, adjectives, adverbs). Then, the candidate selection stage maps the spotted words to the DBpedia resources on the Web, and retrieves possible candidate URIs. After that, the disambiguation stage tries to select the correct DBpedia resource by using the context around the spotted words. DBpedia spotlight relies on a vector space model holding TF-ICF (term-frequency inverse candidate frequency) values of each DBpedia resource. Using these values, the annotator computes a context vector for each candidate DBpedia resource and ranks them based on their similarity (cosine

[14] https://en.wikipedia.org/wiki/Main_Page
[15] https://github.com/dbpedia-spotlight/dbpedia-spotlight
[16] https://github.com/gammaliu/tagme
[17] http://alias-i.com/lingpipe/

similarity) to the context vector of the spotted word. Finally, it is possible to configure some parameters such as the confidence to help make the decision on the right candidate.

TAGME is another semantic Web annotator specialized in short text (Ferragina & Scaiella, 2010). It uses Wikipedia's anchors as its main source of annotations and has a dictionary of 3M anchors linked to 2.7M Wikipedia pages. It first extracts the words in text by scanning its anchor dictionary. Then, in the disambiguation phase, it calculates a collective agreement measure which represents the average relatedness of each candidate link of a spotted word to the candidate links of all the other spotted words. Finally, an anchor pruning process takes place removing non-meaningful annotations based on the probability of the spotted word to be an anchor and a coherence measure between the anchor and its associated link. In our work, we used the two afore-mentioned annotators to identify concepts in students' answers.

2.2.5.4 Vector space models

2.2.5.4.1 N-grams

N-grams, or n-gram analysis, is one of the most popular probabilistic language model technique used in NLP. It captures the usage of *n* word sequences on a given text where *n* stands for the sequence length (Schonlau, Guenther, & Sucholutsky, 2017). Sparse vector space models can be created based on n-grams. One of the most popular text-weighing scheme for the generation of sparse bag-of-words (sBoW) is TF-IDF which represents the importance of a given term in a corpus of documents (Xu, Chen, Weinberger, & Sha, 2013).

2.2.5.4.2 Word Embeddings

Another language modeling technique used in our research is word embeddings. Specifically, word embeddings map words into dense continuous spaces. By doing so, they attempt to capture the semantic and syntactic aspects of a word. Their implementation attempts to solve the issue of high

dimensionality, known as the "curse of dimensionality", present in the use of n-grams in large scale text. They group words with similar context close to each other under the assumption that close words hold the same meaning. (Li & Yang, 2017).

We make use of three pre-trained embeddings models trained using the following techniques: Stanford's GloVe[18], a Google's Word2Vec[19] variation called Wiki2vec[20] and FastText from Facebook.

GloVe (Pennington, Socher, & Manning, 2014), is a count-based unsupervised learning method to generate vector space representations of words. Initially, it relies on a co-occurrence word-to-word matrix on which elements represent the frequency of a word appearing in the context of another word. In this methodology, the model learns meaning through the ratio of co-occurrence probabilities.

Word2Vec (Mikolov, Le, & Sutskever, 2013), is a predictive-based unsupervised method for the generation of embeddings. It presents two model architectures: continuous bag-of-words (CBOW) and skip-grams. In CBOW, a feedforward Neural Net Language Model (NNLM) with a shared projection layer is trained as traditional bag-of-words. This makes it capable of predicting a word based on a set of words (context). In contrast, the skip-gram architecture produces a model that predicts context to a given word. Figure 1 shows both architectures.

Similarly, Wiki2Vec generates vector space representations of DBpedia entities from Wikipedia Dumps. It maps similar DBpedia resources, which are equivalent to Wikipedia hyperlinks, together based on the same two architectures.

[18] https://github.com/stanfordnlp/GloVe
[19] https://code.google.com/archive/p/word2vec/
[20] https://github.com/idio/wiki2ve

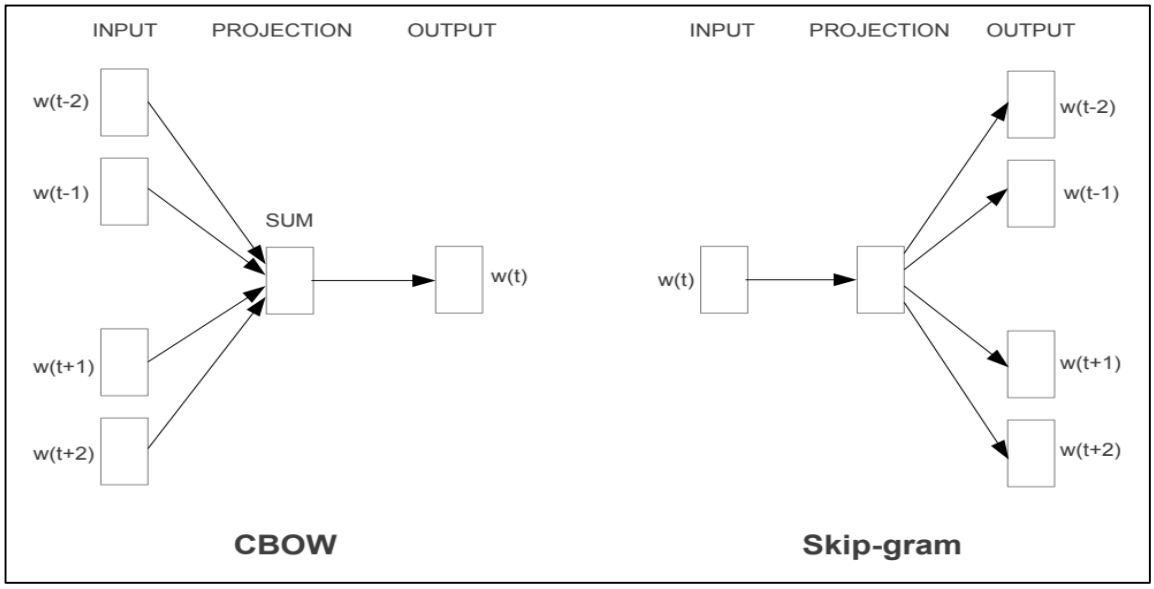


Figure 1. Word2Vec architecture models extracted from (Mikolov, Le, & Sutskever, 2013)

FastText (Bojanowski, Edouard, Joulin, & Mikolov, 2016), is an embedding methodology that takes into account the morphology of a word when training word vectors. In fact, the skip-gram models (Mikolov et al., 2013) ignore these variations. To solve this, the input words are represented by a set of character n-grams and the word itself, so that FastText learns vector representations of character n-grams, and outputs the total sum of those character n-gram embeddings.

2.2.5.4.3 Sentence Embeddings

Just like word embeddings, but for larger text, sentence embeddings project sentence representations into continuous spaces. These techniques produce similar vector representations for sentences that are semantically and syntactically similar.

InferSent is a methodology for unsupervised learning of dense sentence representations created at Facebook (Conneau, Kiela, Schwenk, Barrault, & Bordes, 2017). The idea behind this methodology is to use a data set of natural language inference from which sentence encoders can learn useful text representations. Their training architecture presents various levels (Figure 2).

First, they use a bi-directional long short-term memory (Bi-LSTM) recurrent neural network (RNN) to encode sentences from Stanford Natural Language Inference (SNLI) dataset. In this step, pre-trained GloVe models are used for the extraction of fixed word embeddings. Then, they compute a series of matching methods to capture information from the encoded inputs (an encoded premise sentence and an encoded hypothesis sentence). Finally, the resulting vector is fed into a multi-layer perceptron 3-way classifier followed by a softmax hidden layer. Once the learning rate passes the threshold of 10^{-5}, training stops.

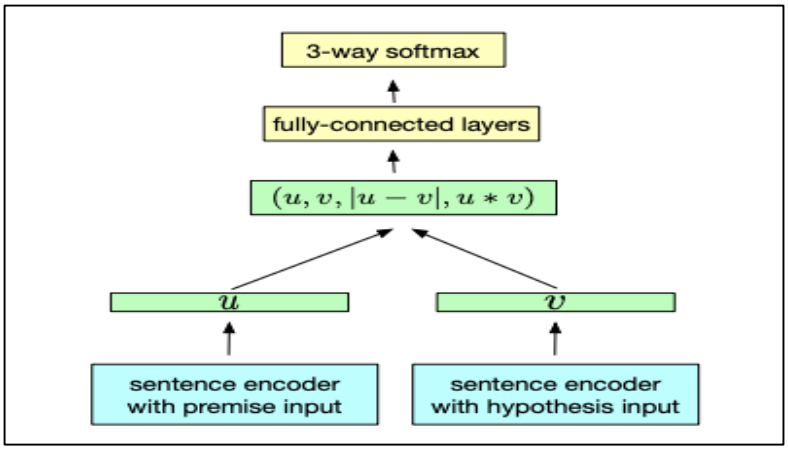


Figure 2. InferSent's sentence encoding-based model extracted from (Conneau, Kiela, Schwenk, Barrault, & Bordes, 2017)

2.2.6. Summary

Our approach incorporates many of the above-mentioned techniques. In our pipeline, a large range of features, with various configurations, in the response-based and reference-based approaches, are used to train multiple classifiers on different standard ASAG datasets. In the response-based part of our pipeline, we use traditional n-grams but we also test vector space models based on DBpedia URIs and mentions. We also explore the use of entity embeddings, i.e. vector space representations of DBpedia URIs and mentions, for the representation of students' answers.

As for reference-based features, we propose the use of semantic Web annotations for the calculation of similarity measures (word overlap and Jaccard index) and experiment sentence embeddings for the calculation of similarity (cosine similarity) between students' responses and model answers, instead of aggregated word embeddings. Finally, we propose a hybrid approach that use stacking to aggregate all our features. The following chapter offers a detailed description of our approach.

Chapter 3. Research Methodology

This chapter presents the general research methodology for the automatic classification of students' short answers. First, it describes our general architecture. Then, it details the dual approach followed for feature extraction:

a. *Response-based features*: Set of features extracted directly from model answers and student responses.

b. *Reference-based features*: This set of features calculate the similarity among questions, students' answers and model answers.

Finally, this chapter explains the classification methods used to generate automatic models that classify responses into correct and incorrect answers.

3.1 General Architecture

Our general pipeline (Figure 3) can be described as follows:

1 ***Data pre-processing***: This includes several pre-processing steps such as lemmatization and the removal of stop words (NLTK stop words list) and punctuation marks.

2 ***Feature extraction***: Feature extraction and generation of vector space models for the response-based approach and reference-based approach separately.

3 ***Classification task***: Training and evaluation of classification algorithms using available vector space models for each approach.

4 ***Feature combination & stacking***: This is based on a two-step process: 1) in each approach, we train and evaluate classifiers combining features and meta-classifiers stacking previously generated models based on their performance. 2) We generate and evaluate a final meta-

classifier with the top performing models from the response-based and reference-based approaches.

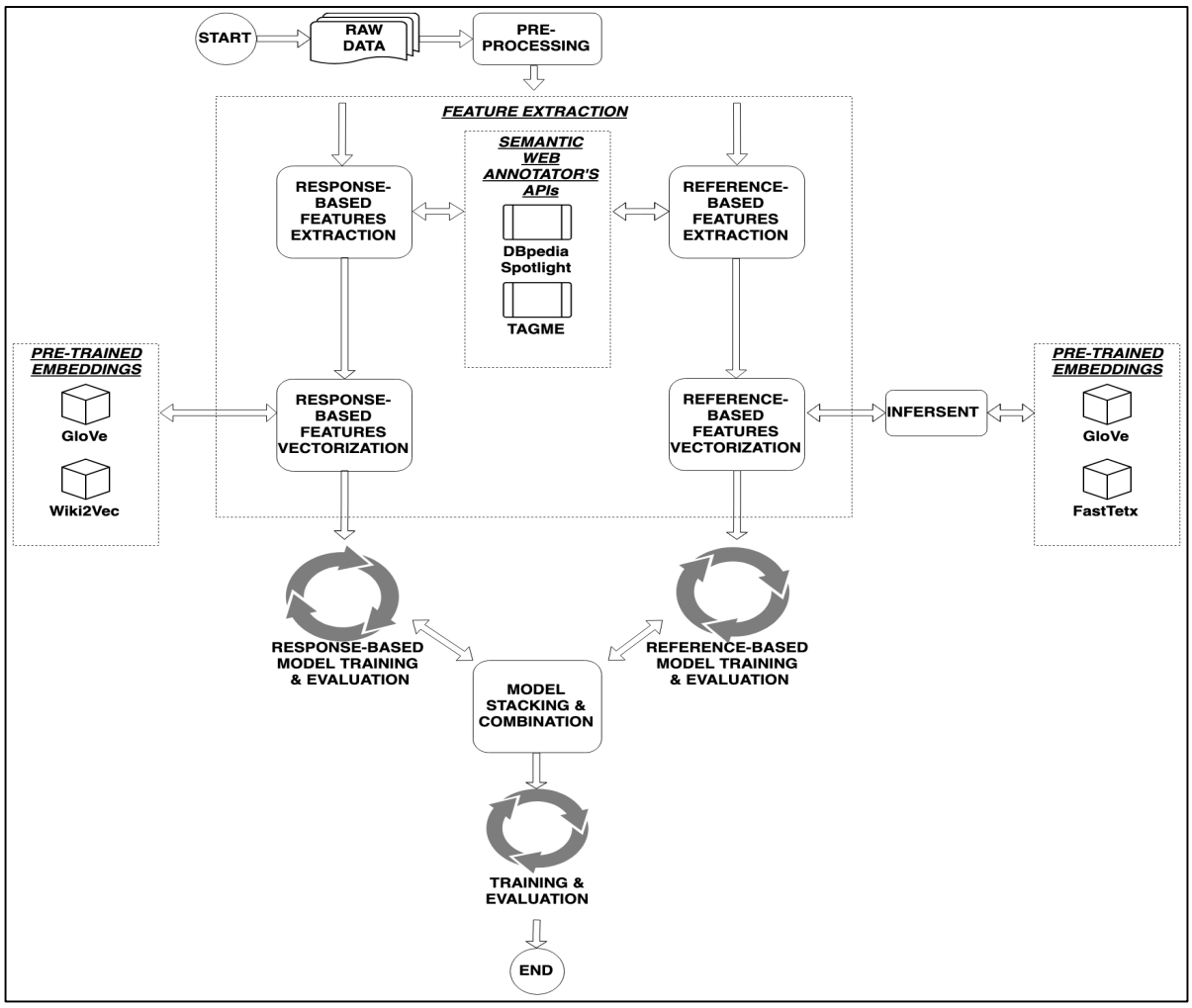


Figure 3. General pipeline.

In the following sections, we detail the various steps followed in our pipeline.

3.2 Data pre-processing

First, we to transform all the answers to lower case and remove punctuation marks. Then, we make use of the *NLTK*[21] platform, which provides multiple tools for NLP in Python and multiple lexical

[21] https://github.com/nltk/nltk

databases to perform two tasks: stop word removal and lemmatization of both verbs and nouns (Figure 4). Stop words are disregarded as non-informative elements. We remove them by simply filtering all word tokens in each answer using *NLTK*'s stopwords lexicon. Then, we lemmatize each word in students' answers and expected answers (e.g. "studying" / "study") using the *WordNet*[22] lexical database for the English Language through *NLTK*. While the pre-processing of data is a critical step within most machine learning architectures, especially for real world data, some of our features require an extraction from unaltered student answers. In fact, we applied pre-processing only for the extraction of lexical features (i.e. N-grams and word similarity features).

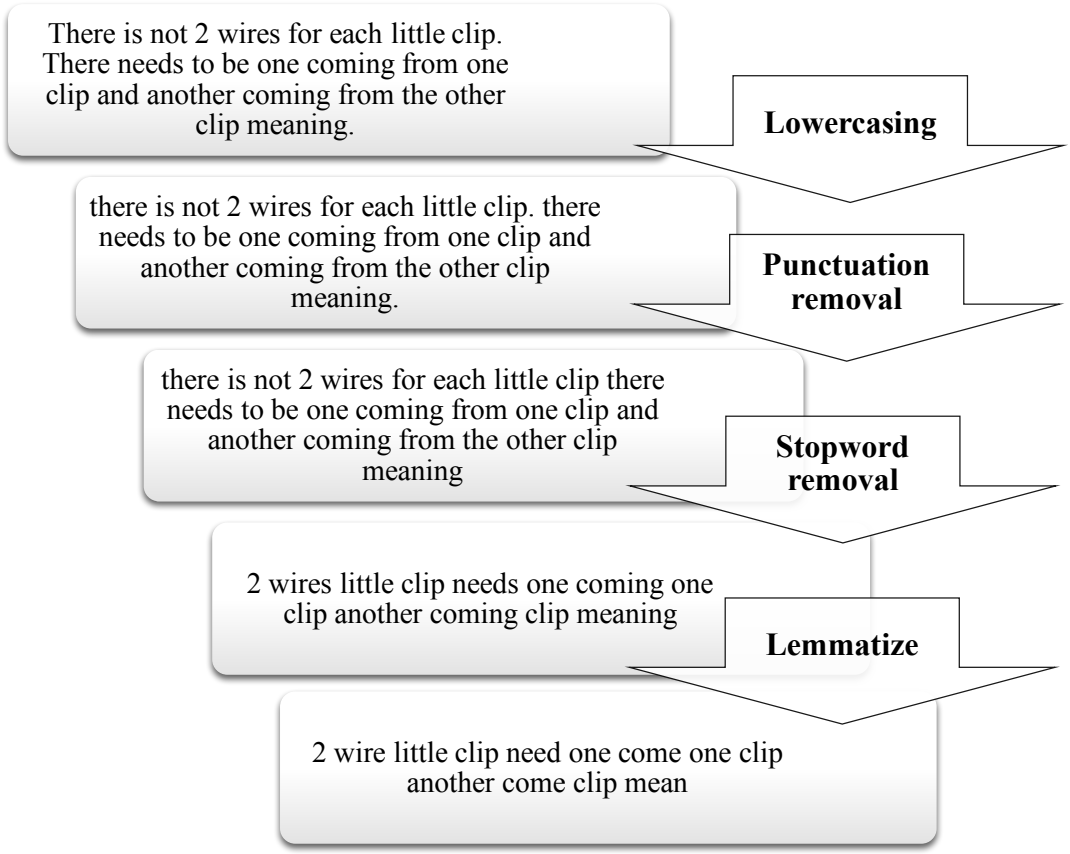


Figure 4. Responses pre-processing.

[22] https://wordnet.princeton.edu/

3.3 Feature extraction

We divide our features into response-based features and reference-based features (Figure 5). We define reference-based features as those that compute some similarity between a students' answers and model answers, and that use the question text in some cases. As for response-based features, we refer to semantic and lexical features extracted from the entire corpus of available students' answers and model answers.

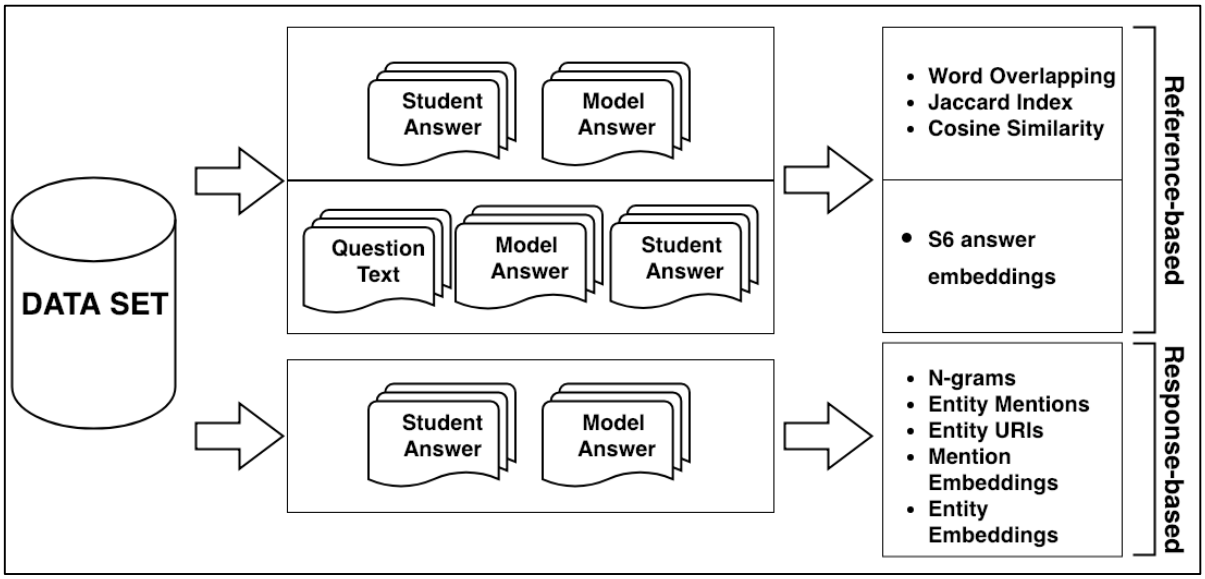


Figure 5. Feature set extraction overview.

3.3.1 Response-based features

We explored five different types of features: N-grams, Entity Mentions (EM), Entity URI (EU), Mention Embeddings (ME) and Entity Embeddings (EE).

For most of these features (n-grams, entity mentions, entity URIs), we build a vector space model using TF-IDF values to represent the importance of a feature. This metric is often used as a term-weighting scheme which, given multiple text documents, represents the frequency of a word (term frequency) within a text document, balanced out by the rarity of that same word (inverse document

frequency) across all text documents. By doing so, TF-IDF minimizes the effect of words that are in general frequent but not important (e.g. stopwords such as 'the', 'a',). TF-IDF is calculated with the formula:

$$tf - idf_{i,j} = \mathrm{tf}_{\mathrm{i,j}} \times (\log \frac{1+\mathrm{n_d}}{1+\mathrm{df_i}} + 1) \quad (1)$$

Where $\mathrm{tf}_{\mathrm{i,j}}$ is the number of occurrences of a term i in the document j, $\mathrm{n_d}$ is the total number of documents and $\mathrm{df_i}$ the number of documents in which the term i appears. In our case, each document represents either a student answer or a model answer.

As for mention embeddings and entity embeddings, we generate a single embedding representation of a document (student answer or model answer) by taking the average of the embedding representation of each entity mention or entity URI present in that document.

3.3.1.1 N-gram Features

We create a vector representation for each answer using either unigrams, bigrams, trigrams, and their combination. In total, we generate four vector space representations, one per n-gram type, where each vector holds the TF-IDF value of each item found in the answers. Figure 6 shows an example of the extraction and vectorization process for unigrams.

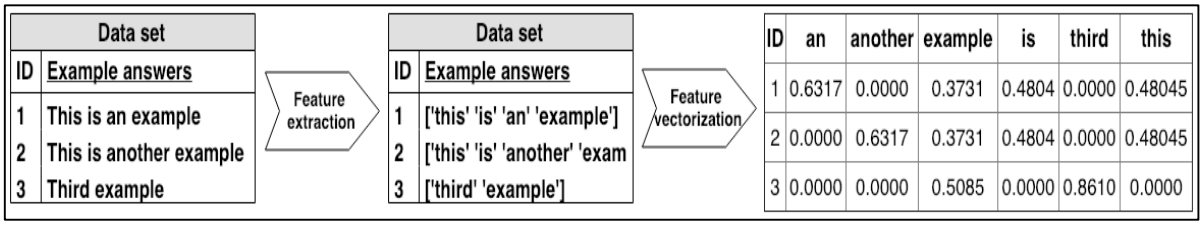


Figure 6. TF-IDF vectorization process for unigrams.

3.3.1.2 Entity URI Features

These features are based on URIs extracted from answers using two Semantic Web annotators: DBpedia Spotlight and TAGME.

Annotations returned by DBpedia Spotlight contain the spotted expressions (Entity Mentions) found in text with their associated DBpedia resource identifier (Entity URI).

TAGME is developed specifically for short text annotation. Its ability to handle short text is due to a configurable threshold which enables disambiguation based on the most-common sense when the threshold is high, which is recommended for short text. We set the threshold at the recommended default value, favoring most-common sense disambiguation (Ferragina & Scaiella, 2010).

TAGME retrieves Semantic Web annotations that rely on Wikipedia's anchor texts and their corresponding URIs. TAGME also offers an online endpoint accessible through an HTTP request. Figure 7 illustrates the annotation extraction process using both annotators.

For both responses, the most important part of the annotations is, for DBpedia Spotlight the '*@URI'* and '*@surface'* fields, and for TAGME '*title'* and '*spot'* which we link to entity URI and entity mention respectively. Based on TAGME's *'title'* field and DBpedia Spotlight's '*@URI*' we are able to find the corresponding Wikipedia and DBpedia resources. Since TAGME's *'title'* is a Wikipedia's anchor text, it can be resolved by adding it at the end of the Wikipedia URI: *http://en.wikipedia.org/wiki/* (Figure 8).

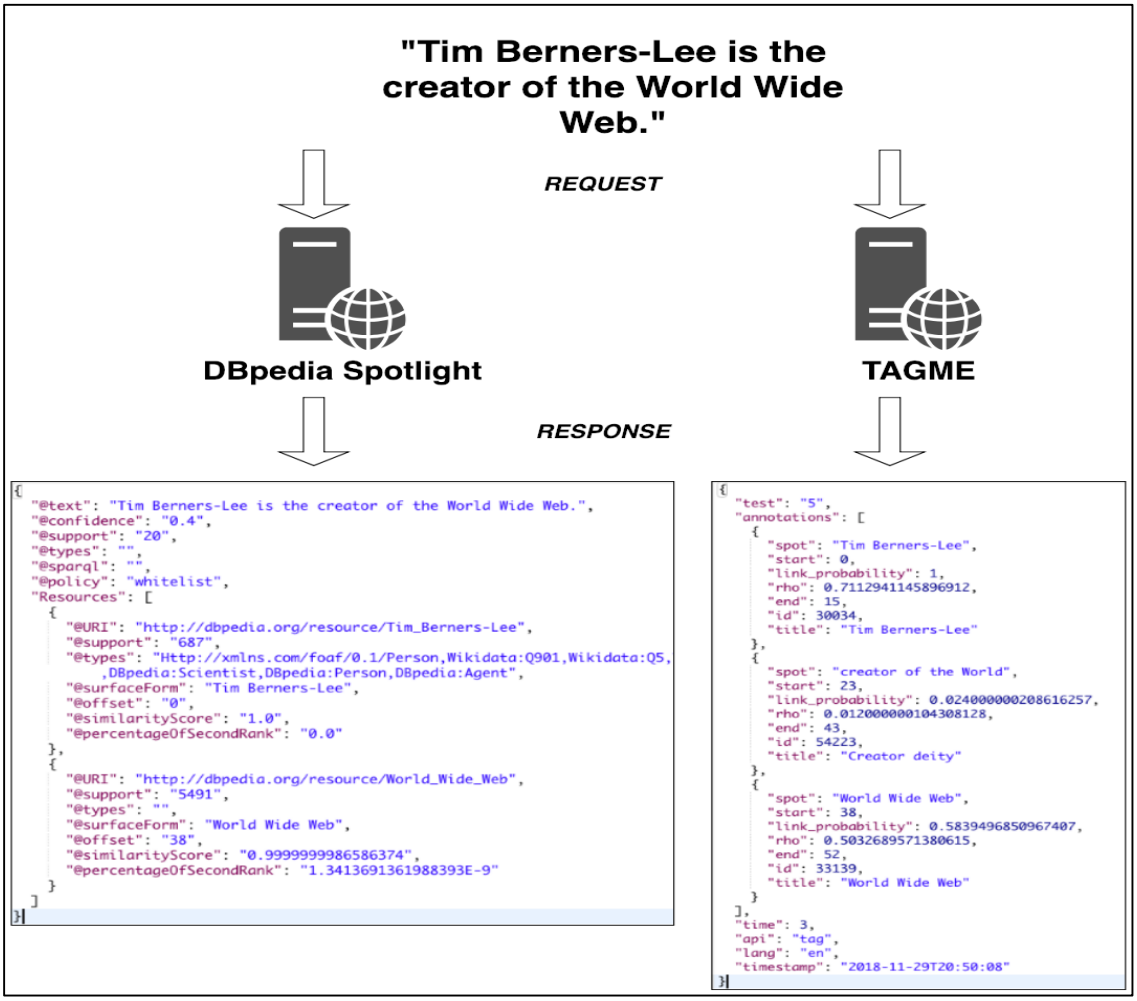


Figure 7. Semantic web annotations extraction process.

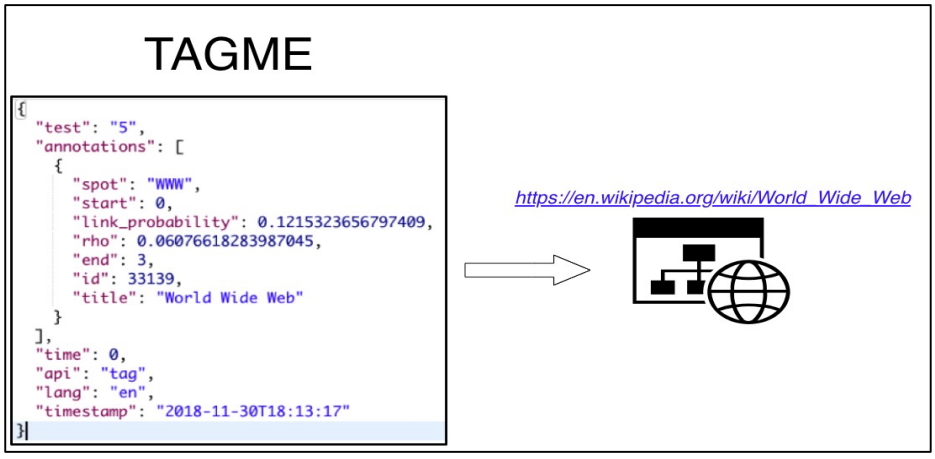


Figure 8. TAGME Semantic Web resource association.

As for DBpedia Spotlight's *'@URI'*, it designates a DBpedia resource which can be directly mapped to its corresponding Wikipedia page (i.e. http://dbpedia.org/page/Tim_Berners-Lee refers to https://en.wikipedia.org/wiki/Tim_Berners-Lee) as shown in Figure 9.

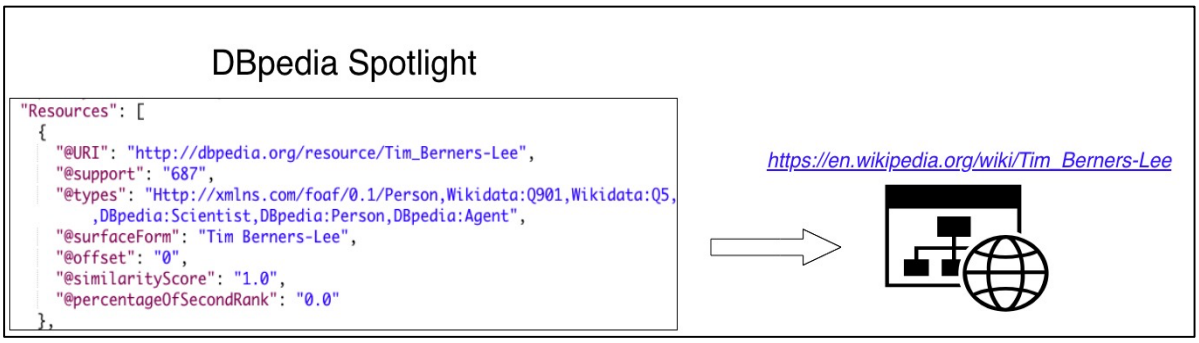


Figure 9. DBpedia Spotlight Semantic Web resource association.

For entity URIs features, the basic unit in the built vector space model is the URI of a DBpedia resource (e.g. http://dbpedia.org/resource/Tim_Berners_Lee).

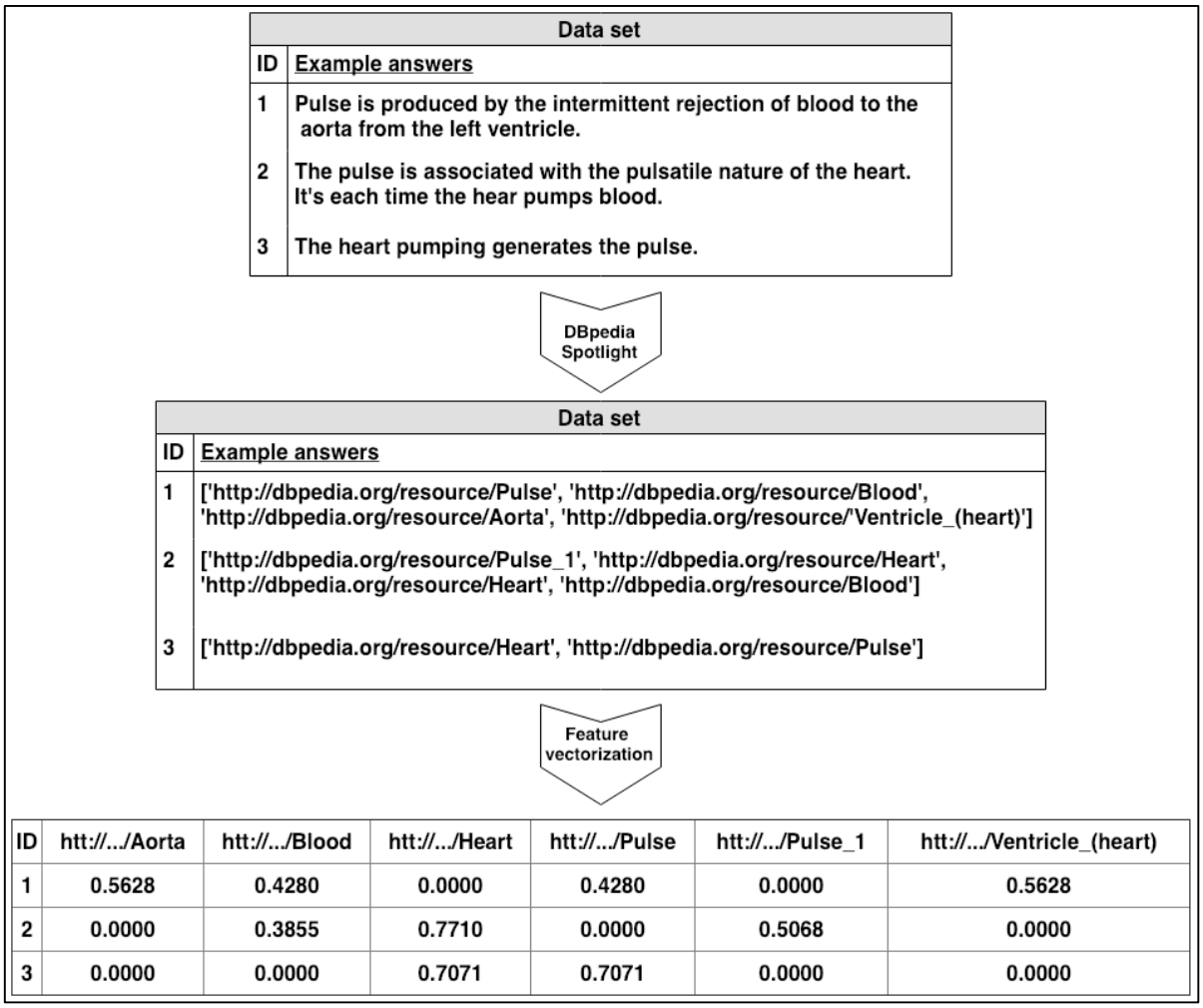
For each answer, we send a *get* request to both annotators, and receive a list of entity URIs and their associated entity mentions. Then, as with n-gram features, we compute a vector space representation relying on TF-IDF where each document is the list of entity URIs in an answer. The resulting vector representation holds the TF-IDF (see formula 1) value of the entity URIs present in an answer. The length of the vector its equal to the total number of entity URIs extracted from our entire dataset (vocabulary). Figure 10 illustrates the vectorization process.

In total, we generated four vector space models:

- Spotlight_URI: Vector space created from entity URIs extracted using DBpedia Spotlight.
- TAGME_URI: Vector space created from entity URIs extracted using TAGME.
- Intersection: Vector space created from the set intersection of entity URIs extracted using both DBpedia Spotlight and TAGME.

- Union: Vector space created from the set union of entity URIs extracted using both DBpedia Spotlight and TAGME. We perform the union to increase the number of available features per answer due to the difficulty of extracting significant statistics on short text.

One interesting aspect of these features is that vector-based representations using semantic annotations are not only smaller than n-grams representations but may also capture relevant terms for a given subject domain.



ID	htt://.../Aorta	htt://.../Blood	htt://.../Heart	htt://.../Pulse	htt://.../Pulse_1	htt://.../Ventricle_(heart)
1	0.5628	0.4280	0.0000	0.4280	0.0000	0.5628
2	0.0000	0.3855	0.7710	0.0000	0.5068	0.0000
3	0.0000	0.0000	0.7071	0.7071	0.0000	0.0000

Figure 10. TF-IDF vectorization process for entity URI features.

3.3.1.3 Entity Mention Features

In this feature, the basic unit for building vector space models is the spotted sequence of words, (mentions) found in answers. This means that an Entity Mention can be a unigram, but also a bigram or trigram.

Here, one entity mention might refer to multiple entity URIs or multiple entity mentions might refer to the same entity URI depending on the context of an answer (Figure 11).

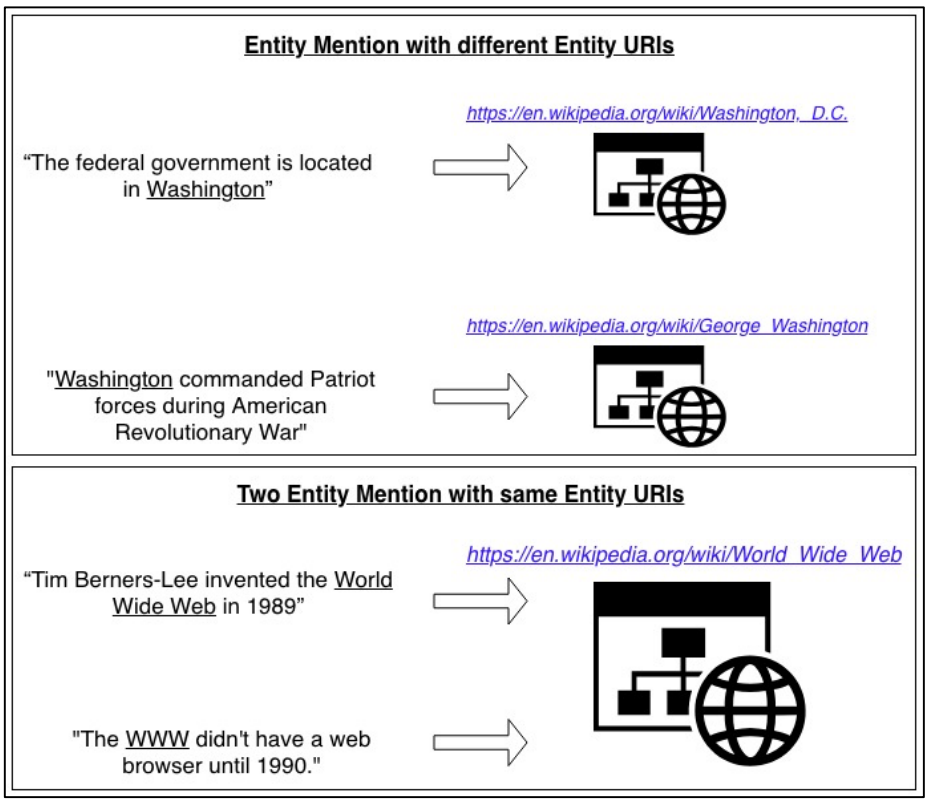


Figure 11. Example of entity mention and entity URIs using DBpedia Spotlight.

In conjunction with TF-IDF, and compared to N-gram models, our hypothesis is that these features allow us to focus our classification efforts on relevant elements and omit non-meaningful words that might bias our model.

Once again, we represent each answer with its extracted entity mentions and compute the TF-IDF value to build its vector representation. As in entity URI features, we have one vocabulary per

configuration with four vector space models as the final output. Our configurations, analogous to those described for entity URIs are Spotlight_Mention, TAGME_Mention, Intersection and Union.

3.3.1.4 Entity Embedding Features

Starting from the entity URIs already obtained, we now rely on a Wiki2Vec pre-trained model presented by Zhou et al. (2017) to compute a single embedding representation per answer. Here we tried to test entity embeddings as the main representation for the answer.

Zhou's model is s a Wiki2Vec model trained on Wikipedia dumps from the year 2014. It replaces Wikipedia hyperlinks with DBpedia entities (entity URI) as shown in Figure 12. Because of this, we consider it a suitable match for the extraction of embedding representations of entity URIs.

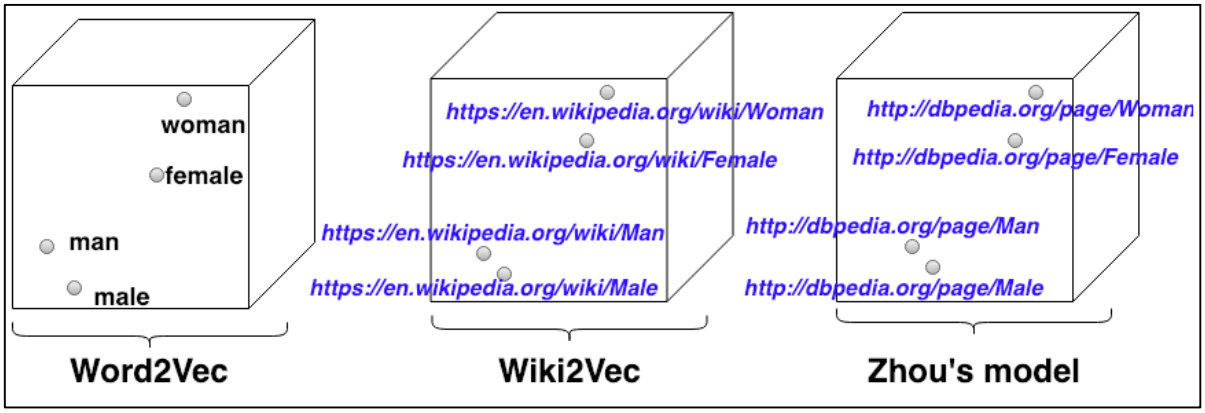


Figure 12. Word2Vec, Wiki2Vec & Zhou's model comparison.

The feature extraction and vectorization process for these features consists in first retrieving entity URIs, using the already introduced semantic web annotators DBpedia Spotlight and TAGME, from each answer present in our dataset. Then, per answer, we query the pre-trained Wiki2Vec model with each entity URI to obtain its corresponding embedding. Finally, we compute a single embedding representation of the answer by calculating the average of the entity embedding vectors (Figure 13).

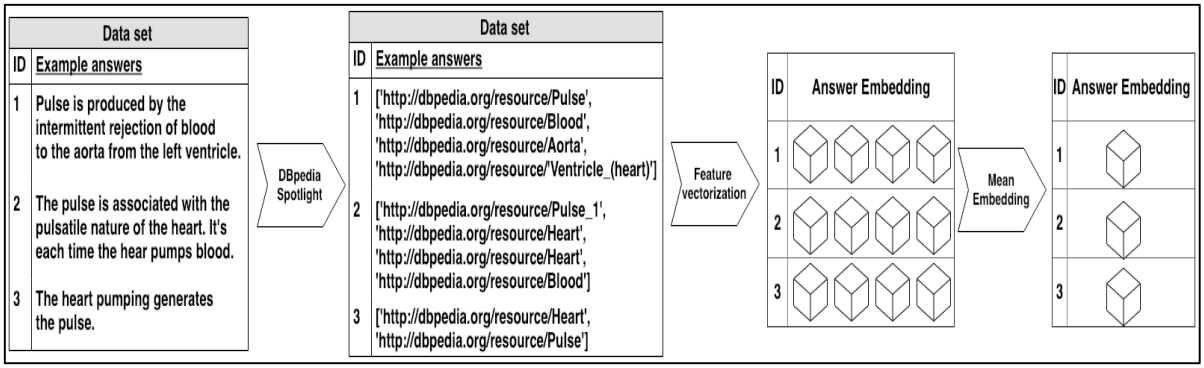


Figure 13. Extraction and aggregation of Entity Embeddings

In total, we generate 4 models. Each model is created with a specific configuration depicting the source from which its base entity URIs were obtained: DBpedia Spotlight (Spotlight_URI), TAGME (TAGME_URI), Intersection and Union.

3.3.1.5 Mention Embedding Features

This set of features describes single embedding representations of answers computed by querying Stanford's GloVe model, pre-trained on Wikipedia dumps from 2014, with the already existing entity Mentions.

In contrast to Word2Vec, GloVe relies on a co-occurrence matrix from which a count-based model is derived. Out of 1.6 million tokens retrieved from the 2014 Wikipedia dump, only the 400,000 most frequent terms make up the model's vocabulary.

We extract word embeddings for each entity Mention present in each answer and aggregate them for a single representation by calculating the mean. Once again, we create 4 different vector spaces models, each with its own base configuration. We average word embeddings obtained from querying GloVe with entity mentions.

3.3.2 Reference-based features

In this approach, we consider the similarities between the three main elements of our dataset: model answers, students' answers and questions.

We make use of various similarity measures such as Overlapping of elements, Jaccard similarity coefficient, cosine similarity in combination with Semantic Web Annotators (DBpedia Spotlight and TAGME) and sentence embeddings and more precisely the InferSent model (Conneau, Kiela, Schwenk, Barrault, & Bordes, 2017) to generate twelve different features.

We define Overlapping as the number of elements, either word tokens, entity mentions or entity URIs present in sets. E.g. let A be a set of entity URIs found on $S.A_1$ and B another set of entity URIs found on $S.A_2$, we calculate the Overlapping as:

$$Overlap(A, B) = |A \cap B| \quad (2)$$

We also calculate the Jaccard similarity coefficient as follows:

$$Jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|} \quad (3)$$

Again, the already mentioned Semantic Web annotators are used in the same way as in the response-based approach: to extract entity URI and entity mentions features from each student answer, model answer and questions.

3.3.2.1 Word Similarity Features

Word similarity features are a lexical set of features that analyses similarity between model answers and students' answers. We compute two features, the Jaccard similarity coefficient between model answer and student answer as shown by formula (3) and the number of word tokens that they have in common (word overlap) calculated with formula (2) (Figure 14).

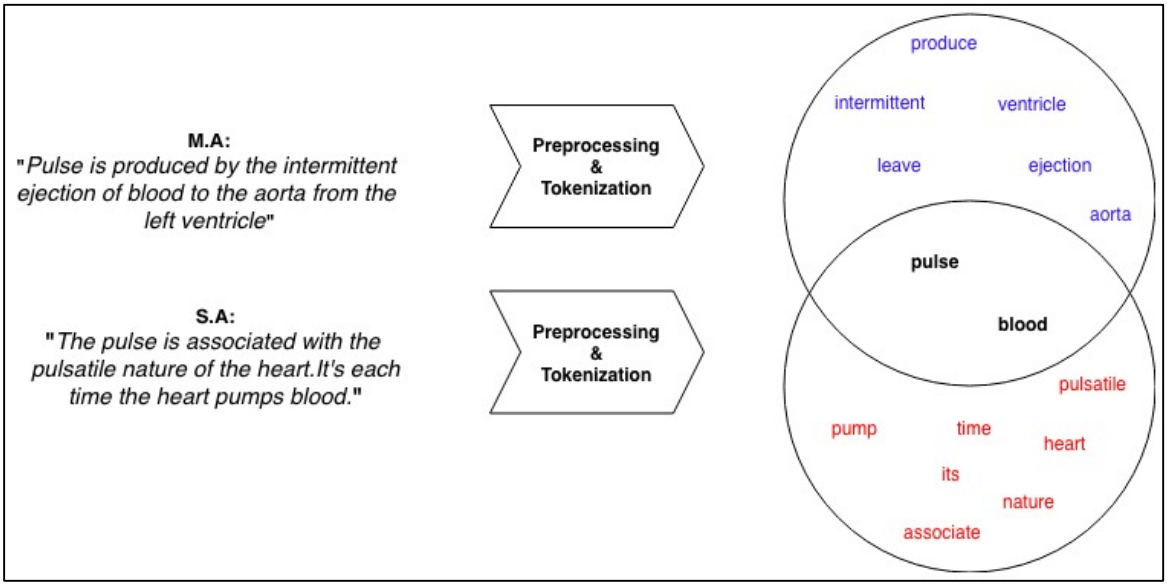


Figure 14. Word overlap between model answer & student answer.

Both features are normalized to a value between 0 and 1 using min-max normalization as follows:

$$x_n = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (4)$$

3.3.2.2 Semantic Similarity Features

Four sets of features are generated combining semantic Web annotations (entity URIs and entity mentions) with Jaccard index (3) and overlap (2).

- URI Overlap: Number of overlapping entity URIs between model answer and student answer.
- Mention Overlap: Number of overlapping entity mentions between model answer and student answer.
- URI Jaccard index: Similarity measure between sets of entity URIs from model answer and student answer.
- Mention Jaccard index: Similarity measure between sets of entity URIs from model answer and student answer.

Each feature set can be computed on two different configurations considering the semantic Web annotator used for the extraction. On Figure 15 we detail how we produce these features before normalization:

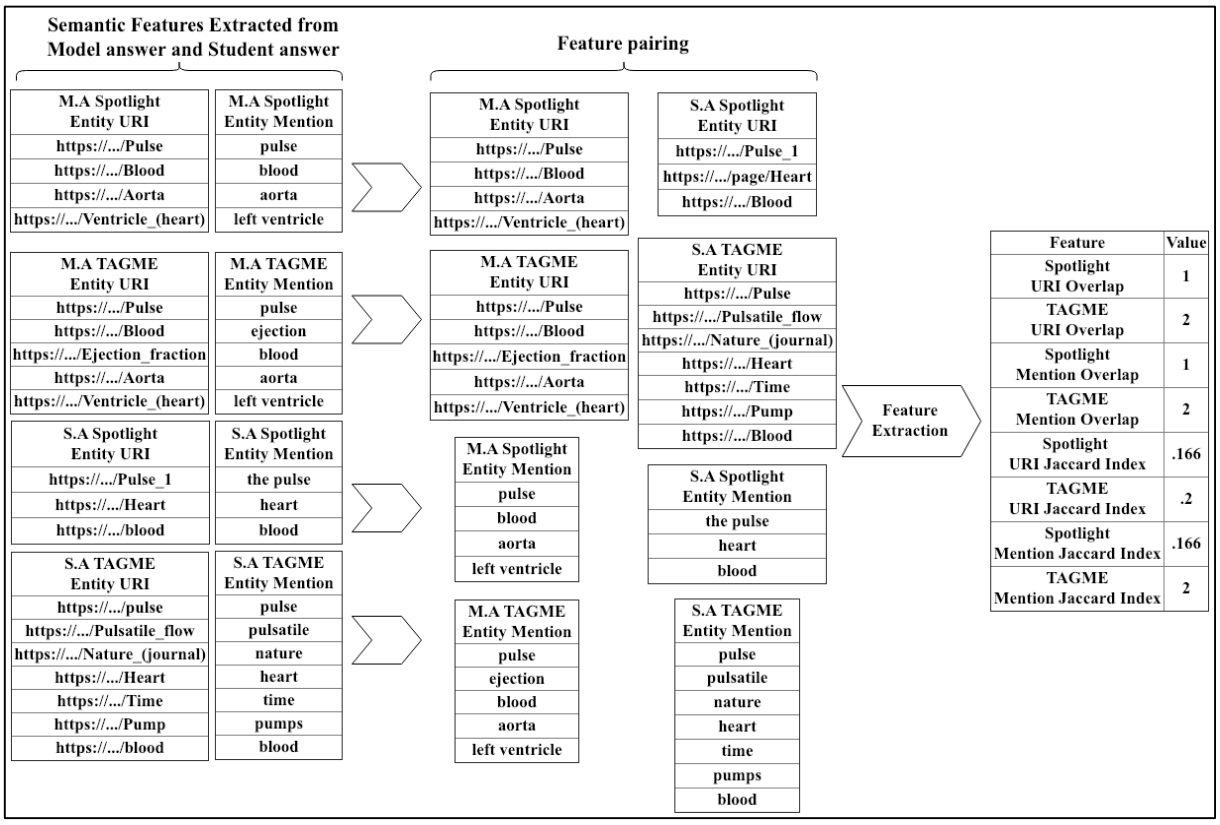


Figure 15. Semantic Similarity feature extraction process for a student answer.

3.3.2.3 Answer Embeddings Features

We compute two separate features here: The Cosine Similarity and S6 embedding. Both make use of the InferSent method, with either pre-trained GloVe or fastText models, for the extraction of answer embeddings.

InferSent (Conneau, Kiela, Schwenk, Barrault, & Bordes, 2017) is a sentence embedding method that provides semantic representations for English sentences. It relies on pre-trained word embeddings models, GloVe or FastText, from which InferSent takes its initial word vectors. These sentence embeddings have two different purposes: to calculate the cosine similarity between the

elements (student answer and model answer) and to extract an embedding representation (Saha, Dhamecha, Marvaniya, Sindhgatta, & Sengupta, 2018) of six components (S6) computed as:

$$S_{feat}(q, r, a) = (MA * SA, |MA - SA|, MA * Q, |MA - Q|, SA * Q, |SA - Q|) \quad (5)$$

where (Q, SA, MA) represents a triple formed by the sentence embeddings provided by InferSent for a question (Q), model answer (MA) and student answer (SA). Each element is either an element-wise multiplication of vectors or an absolute element-wise difference. Then, to combine these six elements into one representation, we simply concatenate them altogether into one vector, which we call S6 embeddings.

The cosine similarity feature is computed by extracting answer embeddings, from a student answer and its corresponding model answer, with InferSent and calculating their cosine similarity as expressed by the formula:

$$\cos\theta = \frac{\vec{\mu}\cdot\vec{v}}{||\vec{\mu}||\,||\vec{v}||} \quad (5)$$

Results varies between -1 and 1, where the higher the value the more similar student answer and model answer are, while near zero values represent dissimilarity between the two answers and negative values points represent opposite meanings.

In a similar fashion, for the S6 embedding features, we generated sentence embeddings by querying InferSent with students' answers and model answers, but this time we also extract the embeddings corresponding to the question. Then, we applied the already introduced formula (5) proposed by Saha et al. (2018) to generate a six-component representation which captures the missing information between model answer and student answer, $(MA * SA, |MA - SA|)$, as well as the novel information provided by the student answer and model answer when compared to the question, $(MA * Q, |MA - Q|)$ and $(SA * Q, |SA - Q|)$ correspondingly. Finally, for a single embedding representation we concatenated all six elements.

3.4 Classifiers, combination and stacking

We selected three machine learning algorithms for the generation of models for the assessment of student's answers given an open-ended question: A Majority-class classifier, Random Forest (RFC) and Support Vector Machine (SVM).

Our first classifier's purpose is to establish baseline scores. A majority-class classifier, implemented by *Scikit-Learn* as DummyClassifier (DUM), outputs the most frequent class from the train-set whenever tested with a new instance.

RFC is an ensemble meta-classifier based on bootstrap aggregation. It generates multiple decision trees sampling the given dataset into various sub-sets (with repetition). It outputs the most frequently predicted class (mode) by each individual tree.

We also use the SVM classification algorithm. It tries to find a hyperplane, in an n-dimensional space, that correctly separates data points. Given that our number of features exceeds the number of instances (students' answers per dataset) we use a linear kernel and follow a one-vs-one strategy which we deemed appropriate for our binary classification problem.

For the response-based approach, we trained three models for each (feature set, configuration). This amounts to 60 models in total. Then, we either combine or stack the top performing models based on the following evaluation measures:

- *Accuracy boost*: We pick the two best performing models in terms of overall accuracy.
- *Precision boost*: for each label, we select the best performing models in terms of precision to either combine them or stack them.
- *F1-score boost*: We either combine or stack the best performing model in terms of f1-score for each label.

If the two selected models were trained using the same classification algorithm we combine their features and proceed to use the same classifier. Otherwise (different classification algorithms), we stack both models into a Logistic regression meta-classifier as implemented by Mlxtend.

As for the reference-based approach, we generated 15 models per classification algorithm, one per each feature set (word similarity, semantic similarity, cosine similarity and S6) and their combinations, for a total of 45 models.

Finally, to improve our results even further, we stacked the best performing response-based and referenced-based models based on macro average f1-score. We generated two Logistic Regression meta-classifiers per dataset trained either on the labels or stacked class-probabilities of the first level classifiers.

Chapter 4. Experiments and Results

In this chapter, first we introduce the datasets used to evaluate our models. Then, we provide some descriptive statistics of the extracted features for each dataset. We also detail our evaluation metrics and analyze the obtained results. Finally, we compare our results obtained on the SemEval Beetle and SciEntBank datasets to some state-of-art results.

4.1 Datasets Description

Our experiments comprise three datasets: McDonald's dataset introduced in (McDonald, et al., 2017), we refer to it as the INSIGHT dataset in this thesis, SemEval Beetle and SemEval SciEntBank (Dzikovska et al., 2013). All of these datasets contain a corpus of labeled short answers provided by students, questions related to one or multiple topics and their corresponding expected answers. The SemEval dataset contains train and test sets of unseen answers. Table 1 gives us a general overview of these datasets.

Dataset	Domain	# Train Answers	# Test Answers	# Tot. Answers	# Questions
INSIGHT	Human biology	1876	0	1876	6
Beetle	Electricity and electronics	3941	439	4380	47
SciEntBank	15 different science domains	4969	540	5509	135

Table 1. Datasets overview

4.1.1 INSIGHT dataset

The INSIGHT dataset consists of a corpus of short answers to open-ended questions from a first-year human biology course (McDonald, Bird, Zouaq, & Moskal, 2017). Answers were collected through a dialog system. All answers have a label, among other metadata, assigned by human

annotators that describe the quality of an answer or its correctness. These labels were negotiated through discussion between two domain experts. Based on the output of the author of the dataset, Dr. Jenny McDonald, we selected a sub-set of answers as follows:

- Only answers from the year 2012. This year registered the highest participation rate with 7,548 answers collected from a total of 15,758 answers received during four years.
- Answers from six unique questions (out of 42 available) which provide a good number of lengthy (deep) responses. Questions that didn't require a deep understanding of the topic to be answered were discarded.

The final selected questions for our sub-set from INSIGHT were designed to encourage deep response using the Structure of Observed Learning Outcomes (SOLO) taxonomy at multi-structural and relational levels. This means that the expected answers should display that the student has several relevant, independent or linked, ideas about the question asked. Table 2 present the surveys questions and model answers used.

ID	Question	Model Answer
Q.1	HR or heart rate is the number of times the heart beats each minute. A normal adult HR is around 72 beats/min. How would you check someone's HR?	You could measure their pulse.
Q.2	What is the pulse?	The pulse is a pressure wave or a pulsatile wave generated by the difference between systolic and diastolic pressures in the aorta.
Q.3	Inotropic state is a term that is sometimes used to describe the	Contractility is the force or pressure generated by the heart muscle during contraction.

	contractility of the heart. Can you describe what is meant by contractility?	
Q.4	If you were 'building' a human being and you wanted to position receptors in the body to monitor blood pressure, where would you put them?	You'd probably want to put them near vital organs and at the main outflow from the heart. It turns out that the main human baroreceptors are located in the carotid sinuses and aortic arch.
Q.5	What feature of artery walls allows us to feel the pulse?	Artery walls are thick and strong and not very compliant
Q.6	Can you explain why you cannot feel a pulse in someone's vein?	You cannot feel a pulse in veins because the blood flow in veins is not pulsatile

Table 2. Survey questions on INSIGHT dataset

After this selection, our new INSIGHT dataset is composed of 1,876 answers from 218 students to 6 questions and their corresponding model answers for a total of 1,882 answers. Model answers on this data set were added as positive instances of 'correct' answers on the train set due to its small size. Table 3 shows the distribution of answers across the selected questions and some descriptive statistics.

Question	**Avg. number of words**	**Min. words**	**Max. words**	**Number of Answers**	**Percentage of Correct Answers**
Q.1	6	1	36	243	65.43%
Q.2	9	1	82	422	17.54%
Q.3	6	1	31	316	33.86%
Q.4	4	1	34	151	54.97%
Q.5	3	1	27	171	25.15%
Q.6	9	1	34	361	31.86%

Table 3. Answers distribution and statistics on INSIGHT dataset

Eighteen labels are associated to the students' answers (McDonald et al. (2017) that range from answers showing disinterest ('*don't-know*' label) and rephrasing of a question ('*self-ref*' label) to responses that fully align with the expected answers ('*correct*' label) or make naïve use of technical expressions ('*naïve*' label). Table 4 display some of those student answers with their corresponding labels.

Question	**Student Answer**	**Label**
Q.5	Elasticity	correct
Q.6	idk lol	dont-know
Q.4	In major arteries of the body, such as the common carotid or the aortic arch	ok
Q.3	ability to change volume	incomplete
Q.6	Ventricle contracts blood ejected into aorta, expanding vessel and increase pressure in vessel, wave of pressure cane felt is pulse	correct

Table 4. Sample of student answers on the INSIGHT dataset

For our purposes, we re-labeled all answers into *'correct'* and *'incorrect'* to ensure a binominal classification problem. For '*correct*' we refer to answers labeled as '*correct*' or '*ok*', and all other answers are considered as '*incorrect*'. This led us to an unbalanced dataset with 65% 'in*correct*' answers and 35% '*correct'*.

4.1.2 SemEval datasets

SemEval (Dzikovska et al., 2013) provides two training datasets Beetle and SciEntBank, that can be used in 5-way, 3-way and 2-way student answer labeling tasks along with three distinct test sets comprising either unseen answers (UA), unseen questions (UQ) or unseen domains (UD). The SciEntBank dataset was collected by Nielsen et al. (2008a) from a corpus of answers to assessment questions, and SemEval Beetle dataset was collected through the BEETLE II dialogue system

(Dzikovska et al., 2010). SciEntBank contains the biggest train set with 4,969 answers for 135 questions on multiple domains, while Beetle provides 3,941 answers for 47 questions within the same domain (Table 1).

Given a student response to a question and a given known correct answer, Dzikovska uses the judgments in Table 5 to label it depending on the task at hand.

Description	**5-way**	**3-way**	**2-way**
Complete paraphrase of the reference answer.	Correct	Correct	Correct
Explicit contradiction of reference answer.	Contradictory	Contradictory	Incorrect
Contains some but not all information from the reference answer.	Partially_correct_incomplete	Incorrect	
Display domain content but does not provide the necessary information.	Irrelevant		
Utterance with no domain content.	Non_domain		

Table 5. Labeling scheme for SemEval datasets per task.

We attempt to solve the 2-way task, which requires to discriminate between *'correct'* and *'incorrect'* answers, with the test set of unseen answers for evaluation. Table 6 shows some questions and answers from both SemEval datasets with their corresponding label according to the 2-way task.

Dataset	Question	M. A	S. A	Label
SciEntBank	You used several methods to separate and identify the substances in mock rocks. How did you separate the salt from the water?	The water was evaporated, leaving the salt.	By letting it sit in a dish for a day.	Incorrect
SciEntBank	What happens to earth materials during deposition?	Earth materials settle out during deposition.	Earth material get deposited and forms a delta.	Correct
Beetle	Why does measuring voltage help you locate a burned-out bulb? Try to answer in terms of electrical states, connections and/ or a gap.	Measuring voltage indicates the place where the electrical state changes due to a gap.	Because if there is a difference in electrical states then there is a gap. That will located the burned-out bulb	Correct
Beetle	What is voltage?	Voltage is the difference in electrical states between two terminals	Is the difference between two terminals	Incorrect

Table 6. Sample questions, student answers and model answers for the SemEval datasets

As shown in Table 7, even though both train sets are unbalanced with a bigger percentage of incorrect answers, we observe that Beetle is the least unbalanced. Also, we notice that SemEval SciEntBank contains the longest answers with an average of 12 words per answer. Another

noticeable difference between SciEntBank and Beetle is that all questions of the SciEntBank dataset have only one model answer each while Beetle questions have from 1 up to 14 possible model answers. Each student answer has an identifier linking it to the model answer to which it was compared.

	SciEntBank		**Beetle**	
	Train	**Test**	**Train**	**Test**
Number of Correct Answers	2008	233	1665	176
Number of Incorrect Answers	2961	307	2276	263
Percentage of Correct Answers	40.41%	43.15%	42.25%	40.09%
Percentage of Incorrect Answers	59.59%	56.85%	57.83%	59.91%
Avg. Number of Words per Answer	12.1	11.89	9.14	9.59
Total number of words	4969	540	3941	439

Table 7. SemEval datasets description per label

4.1.3 Datasets features statistics

In this section, we present some descriptive statistics of all the features extracted from students' answers. In Table 8, we can see the number of n-grams in our three datasets. Statistics regarding the SemEval datasets include their corresponding test sets.

The INSIGHT dataset is the most diverse in terms of unigrams and bigrams by having a bigger rate of unique elements.

	N-gram	**Unique**	**Total**	**Unique rate**
INSIGHT	Unigram	700	6,364	11.00%
	Bigram	2,114	2,589	81.56%
	Trigram	2,590	3,383	76.56%

Beetle	Unigram	678	19,008	3.57%
	Bigram	2,463	14,666	16.79%
	Trigram	3,762	10,666	35.27%
SciEntBank	Unigram	1536	26,581	5.78%
	Bigram	11,222	21,109	53.16%
	Trigram	13,312	16,142	82.47%

Table 8. Number of N-grams found on each dataset (train set and test set combined)

Analyzing the top extracted n-grams (Table 9), we can clearly notice the domain homogeneity in INSIGHT and Beetle. While INSIGHT's and Beetle's top unigrams can be identified in longer n-grams, SciEntBank top unigrams disappear as the sequence of words get longer.

Dataset	**Unigram**	**Freq.**	**Bigram**	**Freq.**	**Trigram**	**Freq.**
INSIGHT	pressure	490	pressure wave	199	aortic arch carotid	34
	blood	400	blood flow	103	pressure wave travel	27
	heart	315	aortic arch	71	wave travel artery	20
	artery	292	low pressure	68	arch carotid sinus	16
	wave	243	heart contract	53	blood flow vein	16
Beetle	bulb	1943	close path	935	close path battery	354
	path	1762	path battery	485	contain close path	263
	terminal	1666	bulb b	466	positive battery terminal	207
	battery	1515	battery terminal	389	bulb b c	186
	close	1226	bulb c	340	bulb close path	114
SciEntBank	water	499	pure sugar	74	fruity cream cooky	36
	make	447	vitamin c	62	cooky pure sugar	23
	different	280	look like	60	cream cooky pure	18

	heat	276	make sound	58	number tree ring	15
	use	262	absorb heat	56	plastic metal wood	14

Table 9. Five most frequent n-grams per dataset

In Table 10, we focus on the number of retrieved entity URIs retrieved from each dataset by DBpedia Spotlight and TAGME. For this feature set, SciEntBank displays the higher rate of unique items with both semantic Web annotators, followed by INSIGHT and finally Beetle. Among annotators, TAGME found more annotations than DBpedia Spotlight in all datasets.

	Entity URI					
	DBpedia Spotlight			**TAGME**		
Dataset	**Unique**	**Total**	**Unique Rate**	**Unique**	**Total**	**Unique Rate**
INSIGHT	143	1,620	8.83%	876	5,054	17.33%
SemEval Beetle	83	3,309	2.51%	960	14,652	6.55%
SemEval SciEntBank	470	4,567	10.30%	4,347	23,580	18.43%

Table 10. Number of Entity URIs found on each dataset (train set and test set combined when applicable)

In most datasets, the unique rate of entity mentions was higher than the unique rate of entity URIs, which confirms that more than one entity mention referred to the same entity URI (Figure 11). For the opposite case, the INSIGHT dataset presented more entity URIs than entity mentions, where an entity mention had different associated URIs depending on the context.

	Entity Mention					
	DBpedia Spotlight			**TAGME**		
Dataset	**Unique**	**Total**	**Unique Rate**	**Unique**	**Total**	**Unique Rate**
INSIGHT	188	1,620	11.60%	806	5,054	15.95%
SemEval Beetle	80	3,309	2.42%	652	14,652	4.45%

SemEval SciEntBank	465	4,567	10.18%	2,843	23,580	12.06%

Table 11. Number of Entity Mentions found on each dataset (train set and test set combined when applicable)

As for entity embeddings, Table 12 presents the coverage of Zhou's model (Zhou, Zouaq, & Inkpen, 2017). For most available entity URIs on each dataset, this pre-trained Wiki2Vec model provided an embedding representation learnt from Wikipedia dumps.

	Found entities on		
	INSIGHT	**Beetle**	**SciEntBank**
Spotlight_URI	97.50%	99.97%	99.47%
TAGME_URI	93.94%	94.88%	97.03%
Intersection	97.11%	99.87%	99.65%
Union	94.63%	95.80%	97.43%

Table 12. Percentage of Entity URIs found in Wiki2Vec model

In contrast, the coverage of GloVe for all entity mentions in each dataset (Table 13), while still on the higher end for Beetle and SciEntBank, dropped in all datasets and affected the INSIGHT dataset in particular.

	Found mentions on		
	INSIGHT	**Beetle**	**SciEntBank**
Spotlight_Mention	50.46%	89.38%	83.58%
TAGME_Mention	62.16%	87.64%	85.83%
Intersection	49%	90.49%	86.55%
Union	65.10%	87.96%	85.46%

Table 13. Percentage of Entity Mentions found in GloVe model

4.2 Evaluation Metrics

After generating our models, we had to choose evaluation strategies and measures based on the datasets characteristics. While test sets for SciEntBank and Beetle were provided, the INSIGHT dataset lacked a test set. Additionally, given its small size, dataset splitting was not a suitable strategy. Thus, we evaluated all models trained using the INSIGHT dataset by running a 10-Fold cross validation, and a train and test evaluation for both SemEval datasets.

The metrics used to assess classifiers' performance are the following:

- Precision: Percentage of predicted answers as correct that are correct. This metric is important as false positives have a high cost. A high precision means that it is less likely to have incorrect answers classified as correct.

$$Precision = \frac{TP}{TP + FP}$$

- Recall: Also known as true positive rate. It tells us the percentage of predicted correct answers captured by our model out of all positive (false negative and true positive) predictions.

$$Recall = \frac{TP}{TP + FN}$$

- F1-score: Harmonic mean between precision and recall.

$$f_{1i} = 2\,\frac{Precision \times Recall}{Precision + Recall}$$

- Accuracy: Measures the total percentage of correctly predicted student answers.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

- Macro Average F1-score: Most important metric, due to comparison purposes with the state of the art. It allows us to evaluate our system by considering the performance of each individual class. It is defined as:

$$Macro\ f_1 = \frac{\sum_{i=1}^{N} f_{1i}}{N}$$

Where *N* is the number of classes (two for the two-way task) and f_{1i} is the f1-score for each class *i.*

4.3 INSIGHT Dataset Results

This section presents the results obtained by applying our proposed pipeline, using both response-based and reference-based features, to the INSIGHT dataset.

4.3.1 Response-based models

First, we present the results obtained through a 10-fold cross validation performed on all models trained using the INSIGHT dataset (Table 14). Within each feature set, top accuracy and macro average f1scores across classifiers are highlighted with bold letters.

The use of unigrams from the n-grams feature set obtained the highest accuracy of 89.12% and highest macro average f1-score of 87.53%. Both results were achieved using RFC.

		DUM		**SVM**		**RFC**	
Feature Set	**Conf.**	**ACC**	**Macro Avg F1**	**ACC**	**Macro Avg F1**	**ACC**	**Macro Avg F1**
Ngrams	Unigrams	0.649	0.3935	0.8552	0.8408	**0.8912**	**0.8753**
	Bigrams	0.649	0.3935	0.7777	0.7282	0.7975	0.7392
	Trigrams	0.649	0.3935	0.729	0.6191	0.729	0.6067

	Ngrams	0.649	0.3935	0.8606	0.8414	0.8744	0.8511
EU	Spotlight	0.649	0.3935	0.738	0.686	0.7585	0.6991
	TAGME	0.649	0.3935	0.836	0.8231	0.8654	0.8438
	Intersection	0.649	0.3935	0.7073	0.6298	0.723	0.6323
	Union	0.649	0.3935	0.8378	0.824	**0.8708**	**0.851**
EM	Spotlight	0.649	0.3935	0.7512	0.7029	0.7596	0.6998
	TAGME	0.649	0.3935	0.857	0.8424	0.8847	0.8668
	Intersection	0.649	0.3935	0.744	0.6942	0.7537	0.6898
	Union	0.649	0.3935	0.8594	0.8449	**0.8865**	**0.8696**
EE	Spotlight	0.649	0.3935	0.7194	0.6469	0.7579	0.6909
	TAGME	0.649	0.3935	0.7548	0.7323	0.8168	0.7761
	Intersection	0.649	0.3935	0.6622	0.5838	0.7218	0.6161
	Union	0.649	0.3935	0.762	0.7393	**0.8324**	**0.797**
ME	Spotlight	0.649	0.3935	0.7097	0.6213	0.7163	0.6205
	TAGME	0.649	0.3935	0.8035	0.7819	0.8209	0.7802
	Intersection	0.649	0.3935	0.6875	0.5938	0.7043	0.5994
	Union	0.649	0.3935	0.8245	**0.8071**	**0.8378**	0.8053

Table 14. Accuracy & macro average f1-score on INSIGHT dataset (10FCV) with response-based features

More details on the evaluation of our models trained on INSIGHT can be found on Tables 15 and 16. For each classifier, we present precision, recall and f1-score per label. Results highlighted in bold are the highest results across configurations per feature set. The baseline dummy classifier obtained a precision of 0.65. a recall of 1 and an F-score of 0.79 across all features for the majority class '*incorrect*'.

The best results for the correct label are the following: a precision of 94% achieved using all the n-grams and an f1-score of 83% using only unigrams. Both results were obtained using RFC (Table 16).

We observe a two-way tie of 89% precision regarding the incorrect label using SVM (Table 15): unigrams in n-grams and TAGME in the entity URI feature set. As for top f1-score for the incorrect label we have unigrams in n-grams (92%), but this time using RFC (Table 16).

		Correct			**Incorrect**		
Feature Set	**Conf.**	**Precision**	**Recall**	**F-score**	**Precision**	**Recall**	**F-score**
Ngrams	Unigrams	0.8	**0.79**	**0.79**	**0.89**	0.89	0.89
	Bigrams	0.78	0.51	0.62	0.78	0.92	0.84
	Trigrams	0.84	0.28	0.42	0.71	**0.97**	0.82
	Ngrams	**0.85**	0.74	**0.79**	0.87	0.93	**0.9**
EM	Spotlight	0.68	0.33	0.44	0.72	**0.92**	0.8
	TAGME	0.73	0.7	0.71	0.84	0.86	0.85
	Intersection	0.61	0.3	0.4	0.7	0.9	0.79
	Union	**0.75**	**0.75**	**0.75**	**0.86**	0.87	**0.87**
EE	Spotlight	**0.67**	0.39	0.49	0.73	**0.9**	0.81
	TAGME	0.65	**0.67**	**0.66**	**0.82**	0.8	0.81
	Intersection	0.53	0.36	0.43	0.71	0.82	0.76
	Union	0.66	**0.67**	**0.66**	**0.82**	0.81	**0.82**
EU	Spotlight	0.68	0.48	0.56	0.76	0.88	0.81
	TAGME	0.75	**0.8**	**0.77**	**0.89**	0.85	**0.87**
	Intersection	0.65	0.36	0.46	0.72	**0.9**	0.8
	Union	**0.76**	0.79	**0.77**	0.88	0.86	**0.87**

ME	Spotlight	0.68	0.33	0.44	0.72	**0.92**	0.8
	TAGME	0.73	0.7	0.71	0.84	0.86	0.85
	Intersection	0.61	0.3	0.4	0.7	0.9	0.79
	Union	**0.75**	**0.75**	**0.75**	**0.86**	0.87	**0.87**

Table 15. SVM results per label on INSIGHT dataset (10FCV) with response-based features

		Correct			**Incorrect**		
Feature Set	**Conf.**	**Precision**	**Recall**	**F-score**	**Precision**	**Recall**	**F-score**
Ngrams	Unigrams	0.91	**0.77**	**0.83**	**0.88**	0.96	**0.92**
	Bigrams	0.9	0.47	0.62	0.77	0.97	0.86
	Trigrams	0.92	0.25	0.39	0.71	**0.99**	0.83
	Ngrams	**0.94**	0.69	0.79	0.85	0.97	0.91
EM	Spotlight	0.73	0.31	0.43	0.71	0.94	0.81
	TAGME	**0.89**	0.56	0.69	0.8	**0.96**	0.87
	Intersection	0.7	0.28	0.4	0.71	0.94	0.8
	Union	**0.89**	**0.62**	**0.73**	**0.82**	**0.96**	**0.88**
EE	Spotlight	0.79	0.42	0.55	0.75	0.94	0.83
	TAGME	0.87	0.56	0.68	0.8	0.95	0.87
	Intersection	0.78	0.29	0.42	0.71	**0.96**	0.82
	Union	**0.89**	**0.6**	**0.71**	**0.81**	**0.96**	**0.88**
EU	Spotlight	0.76	0.46	0.57	0.76	0.92	0.83
	TAGME	0.88	0.71	0.79	**0.86**	**0.95**	0.9
	Intersection	0.74	0.33	0.45	0.72	0.94	0.81
	Union	**0.89**	**0.72**	**0.8**	**0.86**	**0.95**	**0.91**
ME	Spotlight	0.73	0.31	0.43	0.71	0.94	0.81

	TAGME	**0.89**	0.56	0.69	0.8	**0.96**	0.87
	Intersection	0.7	0.28	0.4	0.71	0.94	0.8
	Union	**0.89**	**0.62**	**0.73**	**0.82**	**0.96**	**0.88**

Table 16. RFC results per label on INSIGHT dataset (10FCV) with response-based features

4.3.2 Reference-based models

Table 17 present results in terms of macro average f1-score and accuracy using a 10-fold cross validation on models trained using reference-based features extracted from the INSIGHT dataset. We present here all possible combinations of the word similarity (WS), semantic similarity (SS), sentence cosine similarity (CS) and S6 embedding (S6). For each classifier, we highlighted the highest results, across all feature combinations, with bold lettering.

While for models trained on RFC the use of semantic similarity features didn't improve the performance, our top performing model was generated by combining semantic Web based features with word similarity and the S6 embedding (InferSent with fastText) on SVM. We obtained an accuracy of 93.48% and macro average f1-score of 92.83%.

		RFC		**SVM**	
Feature	**Model**	**ACC**	**Macro Avg F1**	**ACC**	**Macro Avg F1**
S6	GloVe	0.8865	0.8696	0.9016	0.8924
	fastText	0.8974	0.8808	0.9306	0.9237
CS	GloVe	0.6646	0.6305	0.4952	0.4931
	fastText	0.6773	0.6439	0.5465	0.5361
WS	GloVe	0.7038	0.6378	0.6279	0.5900
	fastText	0.7038	0.6378	0.6279	0.5900

SS	GloVe	0.7834	0.7302	0.7014	0.6602
	fastText	0.7834	0.7302	0.7014	0.6602
WS + CS	GloVe	0.7249	0.6928	0.6279	0.5900
	fastText	0.7901	0.7678	0.5989	0.5635
WS + SS	GloVe	0.7635	0.7192	0.7032	0.6611
	fastText	0.7635	0.7192	0.7032	0.6611
SS + S6	GloVe	0.8835	0.8660	0.9143	0.9059
	fastText	0.9052	0.8908	0.9336	0.9268
CS + S6	GloVe	0.8781	0.8601	0.9010	0.8918
	fastText	0.9022	0.8871	0.9330	0.9263
SS + CS	GloVe	0.7822	0.7601	0.7056	0.6638
	fastText	0.7901	0.7961	0.7062	0.6645
WS + S6	GloVe	0.8884	0.8705	0.9016	0.8927
	fastText	**0.9064**	**0.8919**	0.9330	0.9264
WS + SS + CS	GloVe	0.7961	0.7723	0.7038	0.6622
	fastText	0.8275	0.8075	0.7044	0.6625
WS + SS + S6	GloVe	0.8841	0.8669	0.9143	0.9059
	fastText	0.8992	0.8831	**0.9348**	**0.9283**
WS + CS + S6	GloVe	0.8847	0.8666	0.9022	0.8933
	fastText	0.8992	0.8036	0.9326	0.9270
SS + CS + S6	GloVe	0.8848	0.8675	0.9149	0.9065
	fastText	0.9058	0.8915	0.9342	0.9274
SS + WS + CS + S6	GloVe	0.8853	0.867	0.9137	0.9052
	fastText	0.9022	0.8869	0.9342	0.9275

Table 17. Accuracy and Macro-F1 results on INSIGHT dataset (10FCV) with reference-based features

Tables describing in details the results in terms of precision, recall and f1-score were added in Annex II due to space issues. In general, S6 embeddings is the best feature in terms of precision (95%) and f1-score (95%) for the correct label. In contrast, the combination of S6 embeddings with either word or semantic similarities generated the best model for precision on the incorrect label (94%). As for f1-score for the incorrect label, the combinations of three or four features, as long as it contains S6 embeddings, provided the best result (91%). All these models were trained on SVM.

4.4 SemEval Beetle Dataset Results

4.4.1 Response-based models

We present the results in terms of accuracy and macro average f1-score for the Beetle dataset (Table 18) following the response-based approach. Our three classifiers were trained on the same sets of features (n-grams, entity URIs, entity mentions, entity embeddings, mention embeddings). Results in bold depict the highest accuracy and macro average f1-score across classifiers within each feature set. We observe that the highest accuracy 78.82% and macro average f1-score 78.01% were obtained using unigrams and RFC.

		DUM		**SVM**		**RFC**	
Feature Set	**Conf.**	**ACC**	**Macro Avg F1**	**ACC**	**Macro Avg F1**	**ACC**	**Macro Avg F1**
Ngrams	Unigrams	0.5991	0.3746	0.6811	0.6799	**0.7882**	**0.7801**
	Bigrams	0.5991	0.3746	0.6970	0.6942	0.7517	0.7436

	Trigrams	0.5991	0.3746	0.7130	0.6963	0.7244	0.7013
	Ngrams	0.5991	0.3746	0.7016	0.7008	0.7699	0.7612
EU	Spotlight	0.5991	0.3746	0.6105	0.5984	0.6241	0.5698
	TAGME	0.5991	0.3746	0.7084	0.7084	0.7107	0.7059
	Intersection	0.5991	0.3746	0.4396	0.3968	0.6059	0.4123
	Union	0.5991	0.3746	0.7084	0.7084	**0.7289**	**0.7223**
EM	Spotlight	0.5991	0.3746	0.6583	0.6190	0.6424	0.5992
	TAGME	0.5991	0.3746	0.6811	0.6797	0.7312	0.7275
	Intersection	0.5991	0.3746	0.6583	0.6202	0.6538	0.6100
	Union	0.5991	0.3746	0.6902	0.6902	**0.7449**	**0.7384**
EE	Spotlight	0.5991	0.3746	0.6583	0.5944	0.6310	0.5784
	TAGME	0.5991	0.3746	0.6811	0.6918	**0.7472**	**0.7402**
	Intersection	0.5991	0.3746	0.6583	0.3952	0.6059	0.4123
	Union	0.5991	0.3746	0.6902	0.6558	0.7380	0.7304
ME	Spotlight	0.5991	0.3746	0.6014	0.5870	0.6333	0.5803
	TAGME	0.5991	0.3746	0.6355	0.6350	0.7335	0.7277
	Intersection	0.5991	0.3746	0.6036	0.5890	0.6333	0.5818
	Union	0.5991	0.3746	0.6287	0.6276	**0.7494**	**0.7442**

Table 18. Accuracy & macro average f1-score on SemEval Beetle test set with response-based features

Tables 19 and 20 presents the results of each classifier per label (correct, incorrect) using precision, recall and f1-score. Results in bold represent once again the highest value, either precision, recall or f1-score, across configurations in each feature set. The baseline (DUM) for this dataset has a precision of 60% and f1-score of 75%, both for the majority class label '*incorrect'*.

For the correct label, the highest precision value is 73% and the highest f1-score is 74% using unigrams and RFC (Table 20). As for the incorrect label, the highest precision 89% was obtained using entity URIs with TAGME and SVM (Table 19) while the top f1-score 82% is obtained with unigrams and RFC (Table 20).

		Correct			**Incorrect**		
Feature Set	**Conf.**	**Precision**	**Recall**	**F-score**	**Precision**	**Recall**	**F-score**
Ngrams	Unigrams	0.58	0.77	0.66	0.8	0.62	0.7
	Bigrams	0.6	0.75	0.66	0.8	0.66	0.72
	Trigrams	**0.66**	0.6	0.62	0.75	**0.79**	**0.77**
	Ngrams	0.59	**0.81**	**0.69**	**0.83**	0.63	0.72
EM	Spotlight	**0.61**	0.42	0.5	0.68	**0.82**	**0.74**
	TAGME	0.58	0.77	0.66	0.8	0.62	0.7
	Intersection	0.6	0.43	0.5	0.68	0.81	**0.74**
	Union	0.58	**0.85**	**0.69**	**0.85**	0.59	0.69
EE	Spotlight	0.51	0.53	0.52	0.68	**0.66**	0.67
	TAGME	**0.58**	0.81	**0.68**	**0.83**	0.62	**0.71**
	Intersection	0.41	**0.89**	0.56	0.65	0.14	0.23
	Union	0.55	0.78	0.65	0.8	0.57	0.67
EU	Spotlight	0.51	0.55	0.53	0.68	**0.65**	0.67
	TAGME	**0.59**	**0.89**	**0.71**	**0.89**	0.59	**0.71**
	Intersection	0.41	0.88	0.56	0.64	0.14	0.24
	Union	**0.59**	0.87	**0.71**	0.87	0.6	**0.71**
ME	Spotlight	0.5	0.52	0.51	0.67	**0.66**	0.66
	TAGME	**0.53**	0.84	**0.65**	0.82	0.5	0.62

	Intersection	0.51	0.52	0.51	0.67	**0.66**	**0.67**
	Union	0.52	**0.85**	**0.65**	**0.83**	0.48	0.61

Table 19. SVM results per label on SemEval Beetle test set with response-based features

		Correct			**Incorrect**		
Feature Set	**Conf.**	**Precision**	**Recall**	**F-score**	**Precision**	**Recall**	**F-score**
Ngrams	Unigrams	**0.73**	**0.74**	**0.74**	**0.83**	0.82	**0.82**
	Bigrams	0.68	0.72	0.7	0.8	0.78	0.79
	Trigrams	0.7	0.56	0.62	0.74	**0.84**	0.78
	Ngrams	0.71	0.72	0.72	0.81	0.8	0.81
EM	Spotlight	0.58	0.39	0.47	0.67	0.81	0.73
	TAGME	0.64	**0.77**	**0.7**	**0.82**	0.71	0.76
	Intersection	0.6	0.4	0.48	0.67	**0.83**	0.74
	Union	**0.66**	0.73	**0.7**	0.81	0.75	**0.78**
EE	Spotlight	0.56	0.35	0.43	0.65	0.82	0.73
	TAGME	**0.67**	**0.73**	**0.7**	**0.81**	0.76	**0.78**
	Intersection	0.64	0.04	0.07	0.61	**0.98**	0.75
	Union	0.66	0.71	0.68	0.8	0.76	**0.78**
EU	Spotlight	0.55	0.34	0.42	0.65	0.82	0.72
	TAGME	0.62	**0.73**	0.67	0.79	0.7	0.74
	Intersection	0.64	0.04	0.07	0.61	**0.98**	0.75
	Union	**0.65**	0.72	**0.68**	**0.8**	0.74	**0.77**
ME	Spotlight	0.57	0.35	0.43	0.65	**0.83**	0.73
	TAGME	0.65	0.73	0.69	0.8	0.73	0.77

	Intersection	0.57	0.35	0.44	0.65	0.82	0.73
	Union	**0.67**	**0.76**	**0.71**	**0.82**	0.75	**0.78**

Table 20. RFC results per label on SemEval Beetle test set with response-based features

4.4.2 Reference-based models

Now we present the results in terms of accuracy and macro f1-score for the Beetle dataset following the reference-based approach on Table 21. It contains the results obtained by different combinations of our 15 features. Results in bold highlight the highest accuracy and top macro average f1-score per classification algorithm.

		RFC		**SVM**	
Feature	**Model**	**ACC**	**Macro Avg F1**	**ACC**	**Macro Avg F1**
S6	GloVe	0.8769	0.8709	0.8405	0.836
	fastText	0.8861	0.88	0.8769	0.874
CS	GloVe	0.5808	0.5771	0.6036	0.6021
	fastText	0.656	0.6448	0.7608	0.7576
WS	GloVe	0.7539	0.7439	0.7881	0.7838
	fastText	0.7539	0.7439	0.7881	0.7838
SS	GloVe	0.599	0.4527	0.6127	0.4537
	fastText	0.599	0.4527	0.6127	0.4537
WS + CS	GloVe	0.7312	0.7222	0.7858	0.7833
	fastText	0.7517	0.7427	0.7767	0.7741
WS + SS	GloVe	0.7494	0.7391	0.7927	0.7882
	fastText	0.7494	0.7391	0.7927	0.7882

SS + S6	GloVe	0.861	0.8534	0.8405	0.8354
	fastText	0.8792	0.872	0.8747	0.8716
CS + S6	GloVe	0.8496	0.8416	0.8428	0.8384
	fastText	0.8792	0.8732	**0.8792**	**0.8763**
SS + CS	GloVe	0.6127	0.6078	0.599	0.5973
	fastText	0.6628	0.6532	0.7676	0.7642
WS + S6	GloVe	0.8633	0.856	0.8519	0.8478
	fastText	0.8861	0.8791	0.8747	0.8716
WS + SS + CS	GloVe	0.7334	0.7233	0.7881	0.7855
	fastText	0.7494	0.7414	0.7744	0.7712
WS + SS + S6	GloVe	0.8428	0.8345	0.8473	0.8426
	fastText	0.8883	0.882	0.8747	0.8716
WS + CS + S6	GloVe	0.8587	0.8505	0.8519	0.8478
	fastText	0.8815	0.8746	0.8769	0.8738
SS + CS + S6	GloVe	0.8473	0.8406	0.8451	0.8404
	fastText	0.8952	0.8891	0.8747	0.8716
SS + WS + CS + S6	GloVe	0.8587	0.8508	0.8496	0.8451
	fastText	**0.8974**	**0.8921**	0.8769	0.8738

Table 21. Accuracy and Macro-F1 results on SemEval Beetle test set with reference-based features

The combination of semantic similarity and word similarity features, cosine similarity and S6, using InferSent with fastText, provided the top accuracy of 89.74% as well as the best macro average f1-score of 89.21% using RFC. We notice however that semantic similarity, cosine similarity and S6 embeddings combination obtains almost the same performance.

In terms of precision, for the correct label, the best model included semantic similarity, cosine similarity and S6 embeddings (91%) on RFC while for the incorrect label any combination containing S6 embeddings achieved the highest precision on SVM (93%). The highest f1-score for both correct (87%) and incorrect (92%) label was obtained by combining all four reference-based features on RFC.

4.5 SemEval SciEntBank Dataset Results

4.5.1 Response-based models

Next, the accuracy and macro average f1-score results for the SciEntBanks dataset are presented on Table 22. Ngrams with RFC obtain the highest results for accuracy and macro average f1-score on each feature set. Overall, for this dataset using response-based feature, the combination of n-grams provided the highest accuracy of 70.74% as well as the highest macro average f1-score of 69.17%.

		DUM		**SVM**		**RFC**	
Feature Set	**Configuration**	**ACC**	**Macro Avg F1**	**ACC**	**Macro Avg F1**	**ACC**	**Macro Avg F1**
Ngrams	Unigrams	0.5685	0.3625	0.6556	0.6513	0.6963	0.6846
	Bigrams	0.5685	0.3625	0.6870	0.6742	0.7000	0.6803
	Trigrams	0.5685	0.3625	0.6278	0.5651	0.6352	0.5500
	Ngrams	0.5685	0.3625	0.6907	0.6867	**0.7074**	**0.6917**
EU	Spotlight	0.5685	0.3625	0.6056	0.5817	0.6148	0.5690
	TAGME	0.5685	0.3625	0.6500	0.6483	0.6815	0.6619
	Intersection	0.5685	0.3625	0.5907	0.5415	0.6000	0.5271

	Union	0.5685	0.3625	0.6593	0.6574	**0.6870**	**0.6699**
EM	Spotlight	0.5685	0.3625	0.6185	0.5828	0.6333	0.5852
	TAGME	0.5685	0.3625	0.6630	0.6588	0.6778	0.6659
	Intersection	0.5685	0.3625	0.6130	0.5731	0.6315	0.5848
	Union	0.5685	0.3625	0.6556	0.6533	**0.6833**	**0.6709**
EE	Spotlight	0.5685	0.3625	0.6093	0.5761	0.6130	0.5589
	TAGME	0.5685	0.3625	0.5759	0.5724	**0.6611**	**0.6280**
	Intersection	0.5685	0.3625	0.5815	0.5317	0.6074	0.5293
	Union	0.5685	0.3625	0.5778	0.5747	0.6519	0.6202
ME	Spotlight	0.5685	0.3625	0.6056	0.5581	0.6315	0.5605
	TAGME	0.5685	0.3625	0.6278	0.6230	0.6889	0.6606
	Intersection	0.5685	0.3625	0.6111	0.5588	0.6278	0.5577
	Union	0.5685	0.3625	0.6389	0.6342	**0.6981**	**0.6761**

Table 22. Accuracy & macro average f1-score on SemEval SciEntBank test set with response-based features

Finally, results obtained with each classifier per label on the test set are presented on Tables 23 to 24. The obtained baseline for this analysis obtained with DUM is a precision of 57% and f1-score of 72% for the '*incorrect'* label. The highest precision for the correct label (75%) was obtained using trigrams and RFC (Table 24). The highest f1-score was obtained using the combination of n-grams and SVM (65%) as shown on Table 23. For the incorrect label, the combination of n-grams (74% with SVM) provided the highest precision and the top f1-score 76% was obtained with several features combined with RFC: bigrams, ngrams and mention embeddings (TAGME and Union).

Correct	**Incorrect**

Feature Set	Conf.	Precision	Recall	F-score	Precision	Recall	F-score
Ngrams	Unigrams	0.6	0.63	0.61	0.71	0.67	0.69
	Bigrams	**0.66**	0.57	0.61	0.7	0.78	**0.74**
	Trigrams	**0.66**	0.29	0.4	0.62	**0.89**	0.73
	Ngrams	0.63	**0.67**	**0.65**	**0.74**	0.71	0.72
EM	Spotlight	0.59	0.38	0.46	0.63	0.8	**0.7**
	TAGME	**0.6**	0.64	**0.62**	0.71	0.68	**0.7**
	Intersection	0.58	0.36	0.44	0.62	**0.81**	**0.7**
	Union	0.59	**0.67**	**0.62**	**0.72**	0.65	0.68
EE	Spotlight	**0.57**	0.38	0.46	0.62	0.78	**0.69**
	TAGME	0.51	0.56	0.53	**0.64**	0.59	0.61
	Intersection	0.53	0.3	0.38	0.6	**0.8**	0.68
	Union	0.51	**0.57**	**0.54**	**0.64**	0.58	0.61
EU	Spotlight	0.56	0.42	0.48	0.63	0.74	0.68
	TAGME	0.58	0.67	0.62	0.72	0.63	0.67
	Intersection	0.55	0.3	0.39	0.6	**0.81**	**0.69**
	Union	**0.59**	**0.68**	**0.63**	**0.73**	0.64	0.68
ME	Spotlight	0.58	0.32	0.41	0.61	0.82	0.7
	TAGME	0.57	0.6	0.58	0.68	0.65	0.67
	Intersection	**0.6**	0.31	0.41	0.62	**0.84**	**0.71**
	Union	0.58	**0.61**	**0.59**	**0.69**	0.66	0.68

Table 23. SVM results per label on SemEval SciEntBank test set with response-based features

		Correct			Incorrect		
Feature Set	**Conf.**	**Precision**	**Recall**	**F-score**	**Precision**	**Recall**	**F-score**

Ngrams	Unigrams	0.67	**0.58**	**0.62**	**0.71**	0.78	0.75
	Bigrams	0.71	0.52	0.6	0.7	0.83	**0.76**
	Trigrams	**0.75**	0.23	0.35	0.62	**0.94**	0.75
	Ngrams	0.7	0.56	**0.62**	**0.71**	0.82	**0.76**
EM	Spotlight	0.64	0.34	0.44	0.63	**0.86**	**0.73**
	TAGME	0.64	**0.57**	0.6	**0.7**	0.76	**0.73**
	Intersection	0.63	0.34	0.45	0.63	0.85	0.72
	Union	**0.65**	**0.57**	**0.61**	**0.7**	0.77	**0.73**
EE	Spotlight	0.6	0.3	0.4	0.62	0.85	0.71
	TAGME	**0.67**	**0.42**	**0.52**	**0.66**	0.84	**0.74**
	Intersection	0.62	0.23	0.34	0.6	**0.89**	0.72
	Union	0.65	**0.42**	0.51	0.65	0.83	0.73
EU	Spotlight	0.6	0.33	0.43	0.62	0.83	0.71
	TAGME	**0.67**	0.51	0.58	**0.69**	0.81	0.74
	Intersection	0.59	0.24	0.34	0.6	**0.87**	0.71
	Union	**0.67**	**0.53**	**0.59**	**0.69**	0.8	**0.75**
ME	Spotlight	0.69	0.27	0.38	0.62	**0.91**	0.74
	TAGME	**0.72**	0.46	0.56	0.68	0.86	**0.76**
	Intersection	0.67	0.27	0.38	0.62	0.9	0.73
	Union	0.71	**0.51**	**0.59**	**0.69**	0.84	**0.76**

Table 24. RFC results per label on SemEval SciEntBank test set with response-based features

4.5.2 Reference-based models

Table 25 presents the results of the reference-based classifiers. Both, top highest accuracy and macro average f1-score are highlighted in bold. We achieved our best results in terms of accuracy

(77.22%) and macro average f1 (76.86%) score by combining word similarity features and both sentence embeddings features, CS and S6 using InferSent with GloVe, on SVM.

		RFC		SVM	
Feature	**Model**	**ACC**	**Macro Avg F1**	**ACC**	**Macro Avg F1**
S6	GloVe	0.7222	0.7022	0.7537	0.7483
	fastText	0.7518	0.7339	0.7537	0.7483
CS	GloVe	0.5351	0.5265	0.55	0.5441
	fastText	0.6018	0.5902	0.6481	0.647
WS	GloVe	0.6333	0.6071	0.6314	0.6207
	fastText	0.6333	0.6071	0.6314	0.6207
SS	GloVe	0.6222	0.5827	0.6296	0.6068
	fastText	0.6222	0.5827	0.6296	0.6068
WS + CS	GloVe	0.5777	0.5627	0.6296	0.6185
	fastText	0.6259	0.6064	0.6518	0.6551
WS + SS	GloVe	0.6092	0.5939	0.6351	0.6172
	fastText	0.6092	0.5939	0.6351	0.6172
SS + S6	GloVe	0.7277	0.7096	0.7592	0.7549
	fastText	0.7388	0.7185	0.7537	0.7483
CS + S6	GloVe	0.7314	0.7118	0.7537	0.7483
	fastText	0.737	0.7156	0.7555	0.75
SS + CS	GloVe	0.5777	0.5657	0.6296	0.606
	fastText	0.5792	0.5631	0.6407	0.6356
WS + S6	GloVe	0.7296	0.7101	0.7666	0.7631

	fastText	0.7388	0.7209	0.7518	0.7478
WS + SS + CS	GloVe	0.587	0.5726	0.6425	0.6262
	fastText	0.6574	0.6444	0.6537	0.6426
WS + SS + S6	GloVe	0.7296	0.7118	0.7611	0.758
	fastText	0.7481	0.73	0.7518	0.7483
WS + CS + S6	GloVe	0.7092	0.6886	**0.7722**	**0.7686**
	fastText	0.7388	0.722	0.7518	0.7478
SS + CS + S6	GloVe	0.7388	0.7209	0.7592	0.7549
	fastText	**0.7555**	**0.7379**	0.7537	0.7483
SS + WS + CS + S6	GloVe	0.7296	0.7089	0.7629	0.7597
	fastText	0.737	0.7169	0.7537	0.75

Table 25. Accuracy and Macro-F1 results on SemEval SciEntBank Test Set with reference-based features

Results per label (Annex II) show that the combination of semantic similarity, cosine similarity and S6 embeddings, or S6 embeddings alone, is the best option in terms of precision for the correct label on RFC (80%). For the incorrect label, the highest precision was obtained with the combination of word similarity, cosine similarity and S6 embeddings with SVM (81%). In terms of f1-score, word similarity, cosine similarity and S6 embeddings work better for the correct label on SVM (74%), while semantic similarity, cosine similarity and S6 embeddings lead to better results for the incorrect label (81%) on RFC.

4.6 Feature Combination and Stacking

In an attempt to obtain better results for our classification task, we combined the best performing response-based and reference-based models. Table 26 summarizes the best possible combinations of features in the reference-based models on the two SemEval datasets and INSIGHT dataset.

Feature	Model	Classifier	Dataset	ACC	Macro Avg F1
WS + SS + S6	fastText	SVM	INSIGHT	93.48%	92.83%
WS + CS + S6	GloVe	SVM	SciEntBank	77.22%	76.86%
WS + CS +SS + S6	fastText	RFC	Beetle	89.74%	89.21%

Table 26. Top performing models on the reference-based approach

For response-based models, we implemented our three proposed combinations (precision boost, f1-score boost and accuracy boost) to try to increase our performance. First, for each label, we select the top performing features in terms of precision, f1-score or accuracy. Then, we combine (concatenation of features) top models that were trained using the same classification algorithm or we stack them in a meta-LR classifier if they performed the best using different classification algorithms.

Tables 27 to 29, present the results of top performing models together with their combination or stacking on INSIGHT, Beetle and SciEntBank.

	Feature	Conf.	Classifier	ACC	Macro Avg F1
Precision Boost	Ngrams	Ngrams	RFC	87.44%	85.11%
	EU	TAGME	SVM	83.6%	82.31%
	Stacking	-	LR	**88.94%**	**87.48%**
F1-score Boost	Ngrams	Unigrams	RFC	89.12%	87.53%
	EU	Union	RFC	87.08%	85.1%
	Concatenation	-	RFC	**89.18%**	**87.67%**
Accuracy Boost	EM	Union	RFC	88.65%	86.96%
	Ngrams	Unigrams	RFC	89.12%	87.53%
	Concatenation	-	RFC	**89.30%**	**87.81%**

Table 27. Stacking/combination of response-based models on INSIGHT dataset (10FCV)

	Feature	Conf.	Classifier	ACC	Macro Avg F1
Precision Boost	Ngrams	Unigrams	RFC	**78.81%**	**78.01%**
	EU	TAGME	SVM	70.84%	70.84%
	Stacking	-	LR	78.13%	77.23%
F1-score Boost	Ngrams	Unigrams	RFC	**78.81%**	**78.01%**
	ME	Union	RFC	74.94%	74.41%
	Concatenation	-	RFC	77.44%	76.50%
Accuracy Boost	Ngrams	Unigrams	RFC	**78.81%**	**78.01%**
	ME	Union	RFC	74.94%	74.41%
	Concatenation	-	RFC	77.44%	76.50%

Table 28. Stacking/combination of response-based models on SemEval Beetle test set

	Feature	Conf.	Classifier	ACC	Macro Avg F1
Precision Boost	Ngrams	Trigrams	RFC	63.51%	54.99%
	EU	Union	SVM	65.93%	65.74%
	Stacking	-	LR	**67.03%**	**66.09%**
F1-score Boost	Ngrams	Ngrams	SVM	69.07%	68.67%
	ME	Union	RFC	69.25%	66.83%
	Stacking	-	LR	**70.92%**	**69.78%**
Accuracy Boost	Ngrams	Ngrams	RFC	**70.74%**	**69.16%**
	ME	Union	RFC	69.25%	66.83%
	Concatenation	-	RFC	70.37%	67.59%

Table 29. Stacking/combination of response-based models on SemEval SciEntBank test set

On the INSIGHT dataset (Table 27), the concatenation of unigrams and union of entity mentions from TAGME and DBpedia Spotlight on a RFC provided the highest accuracy of 89.30% and a macro average f1-score of 87.81%.

For Beetle (Table 28), base RFC models trained using unigrams always obtained the best accuracy and macro average f1-score (78.81% and 78.01%).

As for the SciEntBank dataset, Table 29 shows that the top resulting model is a stack of an SVM classifier trained on the combination of n-grams (unigrams, bigrams and trigrams) and the union of ME with a RFC classifier. We obtained a top accuracy of 70.92% and macro average f1-score of 69.78%.

For the final combination of the response-based and reference-based models, we selected the best performing models in terms of macro average f1-score (due to comparison reasons with the state of the art) and stacked them into a LR meta-classifier trained on the labels or stacked class-probabilities of the first level classifiers. Tables 30 to 32 present the results of the final combinations on each dataset.

Features / Combination	**Conf.**	**Classifier**	**ACC**	**Macro Avg F1**
Accuracy Boost	Concatenation	RFC	89.30%	87.81%
WS + SS + S6	fastText	SVM	**93.48%**	**92.83%**
Stack	Probability	LR	92.13%	91.14%
Stack	Normal	LR	90.62%	89.13%

Table 30. Stacking of response-based and reference-based models on INSIGHT dataset (10FCV)

Both stacks for the INSIGHT dataset produced a lower accuracy and macro average f1-score than the highest performing base classifier (combination of word similarity, semantic similarity and S6 embedding from the reference-based approach using SVM) as shown in Table 30. Among the two

stacks, training the meta-classifier on class-probabilities provided a better result with a 92.13% accuracy and 91.14% macro average f1-score, than training by using the output labels of base classifiers.

Features / Combination	**Conf.**	**Classifier**	**ACC**	**Macro Avg F1**
Ngrams	Unigrams	RFC	78.81%	78.01%
SS + WS +CS + S6	fastText	RFC	89.74%	89.21%
Stack	Probability	LR	**90.20%**	**89.64%**
Stack	Label	LR	88.83%	88.30%

Table 31. Stacking of response-based and reference-based models on SemEval Beetle Test Set

For the Beetle dataset (Table 31), by stacking unigrams and RFC for the response-based model and the combination of SS, WS, CS and S6 embedding for the reference-based model, we created a meta classifier that slightly outperforms all our previous classifiers with an accuracy of 90.20% and a macro average f1-score of 89.64%.

Features / Combination	**Conf.**	**Classifier**	**ACC**	**Macro Avg F1**
F1-score Boost	Stacking	LR	70.92%	69.78%
WS + CS + S6	GloVe	SVM	**77.22%**	**76.86%**
Stack	Probability	LR	73.88%	72.64%
Stack	Label	LR	70.92%	69.44%

Table 32. Stacking of response-based approach and reference-based approach on SemEval SciEntBank Test Set

Finally, for the SciEntBank dataset, we stacked the F1-boost meta-classifier from the response-based approach and the top reference-based model (SS, WS combined with CS and S6

embedding). This new meta-classifier provided an accuracy of 70.92% and 73.88% (trained on labels and probabilities respectively) and 69.44% and 72.64% macro average f1-score (Table 32). These results are lower than those of our best base classifier (reference-based combination of semantic similarity, word similarity, CS and S6) with a 77.22% accuracy and 76.86% macro average f1-score.

4.7 SemEval State-of-the-art Comparison

In this section, we compare our top performing models with state-of-art models trained on the on both SemEval datasets (Dzikovska et al., 2013).

While for SciEntBank the use of reference-based features did the best with a macro average f1-score of 76.86%, for Beetle the stacking of response-based and reference-based approaches provided the best macro average f1-score of 89.64%. Top results were achieved mostly thanks to the S6 embeddings representation feature and by adding features based on semantic web annotations for Beetle (along with word similarity and cosine similarity) and word similarity for SciEntBank.

In Table 33, we show our top results and ngrams as baselines, along with the state-of-art results. Each of the top classifiers results are included in both datasets to compare their performance. These results are presented in terms of macro average f1-score.

Team/Classifier	**SciEntBank**	**Rank**	**Team/Classifier**	**Beetle**
WS + CS + S6 with SVM and GloVe	***76.86%***	1st	***Stack Response and Reference-based models with LR trained on probabilities***	***89.64%***
CoMet	76.80%	2nd	***WS + CS + S6 with SVM and GloVe***	***84.78%***

ETS2	76.20%	3rd	CoMet	83.30%
Stack Response and Reference-based with LR trained on probabilities	***72.64%***	4th	ETS2	83.30%
UKP-BIU	72.60%	5th	ETS1	80.20%
SoftCardinality	71.50%	6th	CNGL	80.00%
ETS1	70.50%	7th	BaseLine-Lexical	78.80%
Ngrams with RFC	***69.17%***	8th	***Ngrams with RFC***	***78.01%***
BaseLine-Lexical	61.70%	9th	CU	77.80%
CU	60.30%	10th	SoftCardinality	77.40%
CNGL	59.10%	11th	LIMSIILES	72.30%
CELI	58.80%	12th	CELI	64.00%
LIMSIILES	58.30%	13th	UKP-BIU	60.80%
BaseLine-Majority	36.20%	14th	BaseLine-Majority	37.5

Table 33. Ranking of top performing results on SemEval2013 datasets in terms of macro average f1-score

For SciEntBank, we obtained the 1st position with a slight increase in macro average f1-score of 76.86% with a reference-based model using word similarity features in combination with answer embedding features (cosine similarity and S6 embedding). As for the SemEval Beetle dataset, we outperform the first system CoMet by ~ 5 percentage points using a stack of response and reference-based models (89.64%). Additionally, both top configurations performed better than the given baselines on each dataset.

Even though we obtained better results for the SemEval SciEntBank dataset, more recent ASAG pipelines have achieved better results than ours. As mentioned before, the S6 embeddings were extracted using the formula presented by Saha in (Saha, Dhamecha, Marvaniya, Sindhgatta, &

Sengupta, 2018). In their paper, they apply it to SciEntBank extracting sentence embeddings using the InferSent method (with pre-trained embeddings from GloVe) to train a Multinomial Logistic Regression classifier and then evaluate it using a test set of unseen answers (same test set used in our experiments). The predictive power of this formula alone is that of 74.81% in terms of accuracy and 74.22% in terms of macro average f1-score. In comparison, our best result with this feature (using InferSent with GloVe as well) was an accuracy of 75.37% and 74.83% with an SVM classifier. Nonetheless, they were able to outperform our top result (word similarity with cosine similarity and S6 on SVM) by including syntactic features and obtained an accuracy of 79.26% and macro average f1 of 78.58% against our 77.22% accuracy and 76.86% macro average f1-score.

Chapter 5. Discussion

In this chapter, we discuss how our classifiers and approaches perform across datasets. Then, we discuss our limitations and explore possible future work to improve even further the predictive power of classifiers for the ASAG task.

5.1 Classification Algorithms

Assessing the best classification algorithm in general is a difficult task due to the variety of results. Most top performing models for INSIGHT and SemEval SciEntBank are trained using Support Vector Machine. However, RFC models trained on the SemEval Beetle dataset are often the best. Table 34 presents the frequency of top performing models trained with each specific classification algorithm per dataset and approach.

	INSIGHT			**Beetle**			**SciEntBank**			**Total**		
Approach	**RFC**	**LR**	**SVM**	**RFC**	**LR**	**SVM**	**RFC**	**LR**	**SVM**	**RFC**	**LR**	**SVM**
Reference	4	1	11	6	1	9	1	1	14	11	3	34
Response	7	1	0	7	1	0	6	2	0	20	4	0

Table 34. Number of top performing models trained on each classifier per dataset

While the response-based approach seems to work better with RFC in all datasets, the use of reference-based features seems to improve with SVM with one exception on SemEval Beetle where it did almost as well as RFC.

5.2 Response-based and Reference-based Models

Reference-based features were clearly a better predictor of correct and incorrect answers than response-based features. Results obtained using models trained on the INSIGHT dataset showed

that reference-based features, specifically the combination of word similarity, semantic similarity and S6 embedding (92.83% macro average f1-score), perform notably better than the top response-based feature set (87.81% macro average f1-score obtained by accuracy boost) for this dataset by almost 5 percentage points.

On the Beetle dataset, following the response-based approach, we achieved the best accuracy and macro average f1-score with unigrams (78.81% and 78.01% respectively). In contrast, with referenced-based models and a combination of semantic similarity features, word similarity features, cosine similarity and S6 embedding, we achieved 89.74% accuracy and 89.21% macro average f1-score (using fastText pre-trained word embeddings for InferSent). In both cases, RFC was the best classification algorithm.

The same conclusion on the superiority of reference-based models holds on the SciEntBank, but with different features. The response-based models in the f1-score boost combination (n-grams/RFC and entity mention union /SVM) obtained 70.92% accuracy and 69.78% macro average f1-score. In comparison, the reference-based approach obtained a 77.22% accuracy and 76.86% macro average f1-score. The combination/stacking did not really improve the overall performance (lower performance on SciEntBank, slight increase on Beetle).

5.2.1 Semantic Similarity vs Word Similarity

In the reference-based approach, we introduced two feature sets: semantic similarity features and word similarity features. In both features, we calculated the similarity between model answers and students' answers based on items (words, entity URIs or entity mentions) overlap or Jaccard index. For the INSIGHT dataset, we obtained slightly better results training our multiple classifiers with only semantic similarity features (URI overlap, mention overlap, URI Jaccard index, mention Jaccard index) than word similarity features (word overlap, word Jaccard index).

On SemEval Beetle and SciEntBank we obtained the opposite result, with a noticeable better performance of our classifiers trained on word similarity features than the ones trained on semantic similarity features.

Now, we look at how well similarity features performed when combined with other features from the reference-based approach.

The addition of semantic similarity features to cosine similarity generated better performing models for the INSIGHT dataset. In contrast, word similarity features blended better with cosine similarity, producing more accurate models, on both SemEval datasets.

While the addition of semantic similarity to S6 embeddings provided some improvement for SciEntBank, it caused a decrease in the performance of some classifiers on the INSIGHT dataset and Beetle. As for the addition of word similarity to S6 embeddings, they were beneficial in both SemEval datasets, mainly with S6 embeddings generated with GloVe, but the performance decreased when S6 embeddings were generated with fastText.

Overall, the use of word similarities as stand-alone feature provided better results than semantic similarities. Nonetheless, semantic similarity features were part of the final top performing models in two out of three classifiers.

5.2.1 Embedding Representations

We introduced three different embedding representations of student answers: mention embeddings, entity embeddings and S6.

Both embedding representations from the response-based approach use items (URI or mention) level embeddings, extracted from GloVe or Zhou's Wiki2Vec model, that are later averaged to generate a single embedding representation for an answer. We found out that the use of mention

embeddings led to a slightly better representation of students' answers and model answers on all datasets providing a higher top accuracy and macro average f1-score than entity embeddings.

In the reference-based approach, sentence level embeddings, specifically S6 embedding, generated by InferSent provided better results when using the fastText pre-trained model for the retrieval of initial word embeddings on INSIGHT and Beetle. On SciEntBank, the use of GloVe pre-trained word embeddings on InferSent yielded to a better performance in terms of accuracy and macro average f1 score than fastText.

Within our two approaches, response-based an reference-based, S6 embeddings with the use of fastText in InferSent, provided a more accurate representation than word level embeddings (entity embeddings and mention embeddings).

Overall, S6 embedding proved to be the most critical feature for the discrimination between correct and incorrect answers in our three datasets. Models that used S6 embeddings are always at the top of our rankings.

5.3 Stacking Efficiency

One of our main hypotheses was that combining response-based and feature-based approaches into a single classifier would produce better results.

The propose combinations within the response-based approach (precision boost, accuracy boost and f1-score boost) yielded different results depending on the dataset used. Each of them improved the performance of the base classifiers for the INSIGHT dataset (Table 24). In contrast, none of them outperformed their base classifier on Beetle (Table 25). On SciEntBanks, we managed to obtain slightly better results with precision boost and f1-score boost but not with accuracy boost (Table 26).

As for reference-based models, we performed all possible combinations of features. For the INSIGHT dataset, the combination of word similarity, semantic similarity and S6 provided the best result on SVM. SciEntBank obtained its highest result by combining word similarity with cosine similarity and S6 on SVM as well. Similarly, SemEval Beetle achieved its highest reference-based result combining semantic similarity, word similarity, cosine similarity and S6 embeddings on a RFC classifier. For this approach, combination of features always provided better results than stand-alone feature sets.

Finally, the stacking of top performing models from each approach in terms of macro average f1-score only produced better results than base classifiers on Beetle. For the other two remaining datasets, reference-based models still outperformed the stacking model. It is worth mentioning that the use of probabilities for training the LR meta classifier (compared to plain labels) always provided higher accuracy and macro average f1-score.

5.4 Limitations

In this work, we can cite several limitations. An initial limitation is the availability, size and unstandardized format of short answers to open-ended questions datasets. Each dataset contains questions based on different objectives. Questions from the INSIGHT dataset were designed to generate meaningful answers from students. In contrast, some questions from the SemEval datasets rely on external corpora such as graphs, images and charts; and specifically, for the SemEval SciEntBank, there were cases of questions with multiple possible correct answers. Additionally, SemEval Beetle contained cases with follow up questions to other questions asking for longer explanations on the subject matter.

Another limitation regarding format is the grading scheme. On INSIGHT, we reduced the original 18 labels into two labels (correct, incorrect) to allow for a binary classification task while SemEval provided such labels for both datasets on its two-way task.

In terms of potential implementation issues of our approach, we find three major factors: availability of semantic Web annotators, memory consumption and execution time. As mentioned before, the context in which we attempt to solve the ASAG task is that of MOOCs, meaning that we aim for a reliable scalable system. By querying both TAGME and DBpedia Spotlight through HTTP requests we are dependent on their availability. Then, querying these APIs is not only time consuming but also requires considerable memory space. Also, the training time of our classifiers might cause an initial delay in execution time depending on the dataset size.

We consider the difficulty to generalize, given the lack of consistency of top performing features on each dataset, as a limitation. We believe this is due to many factors such as the different sizes of datasets and unbalanced labeling, quality of questions and students' answers, dataset domain, etc.

The necessity of fast and effective assessment of student knowledge is a limitation to manual feature engineering using domain knowledge. This could have improved our performance but damaged the possibility of generalizing our pipeline.

Finally, we used three pre-trained models in total: Stanford's GloVe model trained on Wikipedia dumps and Gigaword5 for Entity Mentions, Zhou's Wiki2Vec model for Entity URIs, and both GloVe and fastText (also trained on Wikipedia) models as initial word embeddings for the computation of sentence embeddings using InferSent. While these models have been widely used, they might not be the best models for some specific domains.

5.5 Future Work

We consider the following points worth to be explored as possible avenues for improvement of our current pipeline.

The adaptation of various elements of our pipeline to a course domain can be promising to generate better fitted models. For instance, we could generate our own word and sentence embedding models using the entire corpus of class material e.g. class text books, slides, exams, notes, etc. The resulting embeddings would not only be within the domain but would also hopefully capture the context in which these elements are used.

We can also extend this train of thought to TAGME and DBpedia Spotlight since both Semantic Web annotators provide ways to filter out annotations. TAGME's API has the possibility to specify the retrieval of the DBpedia category (consequently Wikipedia category) to which each entity belongs to. We can remove the annotations from categories that are not within a pre-established set of categories deemed to be important for the domain. As for DBpedia Spotlight, its API offers a parameter for the input of SPARQL queries to restrict the extraction of annotations to specific DBpedia categories.

Lastly, the implementation of Artificial Neural Networks presents the most promising future work. RNN have been used in multiple NLP tasks such as semantic matching and sentiment analysis and have proven to be superior in terms of performance. We ran initial Bi-LSTM architectures but due to time constraints we were not able to include them in this thesis.

Chapter 6. Conclusion

In this thesis, we compared several feature sets, classification algorithms, approaches and their combinations for the automated short answer grading task. We applied our proposed pipeline to three different datasets of different sizes and domains.

Within response-based features, on average, the combination of traditional n-grams (mix of unigrams, bigrams and trigrams) or unigrams alone proved the best feature for precision. The same configurations for each dataset holds up for accuracy and macro average f1-score, making n-grams the most prominent feature set for this approach with either unigrams or n-grams. The combination of features or stacking of best performing models within this approach, based on f1-score and precision, improved the results for most datasets.

On the reference-based approach, S6 embeddings were the most prominent feature with the highest accuracy and macro average f1-score across all individual features. Even more important, they were part of all top performing models based on feature combinations.

Overall. reference-based features performed better than response-based features.

While the combination of both approaches seemed a promising venue, we only managed to further improve the results for one dataset. The resulting model uses a Logistic regression meta-classifier with two RFC first level classifiers, one trained on unigrams and the second one on the combination of semantic similarity features, word similarity features, cosine similarity and S6 embeddings. Still, we consider this step worth taking as an exploratory measure for possible improvement in performance.

From all three embedding representations proposed across approaches (S6 embeddings, mention embeddings and entity embeddings), we found out that S6 embeddings were better than word level embeddings (mention embeddings and entity embeddings).

As for which classification algorithm works the best with our features, our results show that RFC was the best option for response-based features and, for reference-based, SVM did better except for one dataset where the stacking of classifiers into a LR classifier produced the best model.

We can also conclude that semantic features based on semantic Web annotations positively – but slightly - affected the performance of our top classifier on two datasets, except in one dataset. While semantic similarity features blended well with other features and were part of the top performing models in most datasets, word similarity features alone usually performed better.

In summary, reference-based features proved to have more predictive power than response-based features. Particularly, S6 embedding representation presented in (Saha, Dhamecha, Marvaniya, Sindhgatta, & Sengupta, 2018) was the most contributing factor to the results obtained in this thesis. The effects of semantic similarity features based on semantic annotations seem to be dependent on the dataset. While their use alone performed poorly in the reference-based approach, they generally improved the overall performance of the best classifiers when combined with the S6 embedding representation.

References

Alvarado Mantecon, J. G., Abdi Ghavidel, H., Zouaq, A., Jovanovic, J., & McDonald, J. (2018). A Comparison of Features for the Automatic Labeling of Student answers to Open-ended Questions. *11th International Conference on Educational Data Mining*, (pp. 55-65). Buffalo.

Bär, D., Biemann, C., Gurevych, I., & Zesch, T. (2012). UKP: Computing Semantic Textual Similarity by Combining Multiple Content Similarity Measures. *6th International Workshop on Semantic Evaluation, in conjunction with the 1st Joint Conference on Lexical and Computational Semantics* , (pp. 435-440).

Bär, D., Zesch, T., & Iryna, G. (2011). A Reflective View on Text Similarity. *International Conference on Recent Advances in Natural Language Processing* , (pp. 515-520).

Basu, S., Jacobs, C., & Vanderwende, L. (2013). Powergrading: a Clustering Approach to Amplify Human Effort for Short Answer Grading. *Transactions of the Association for Computational Linguistics, 1*, 391-402.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001, May). *The Semantic Web*. Retrieved from Scientific American: https://www.scientificamerican.com/article/the-semantic-web/

Bojanowski, P., Edouard, G., Joulin, A., & Mikolov, T. (2016). Enriching Word Vectors with Subword Information. *CoRR*.

Boston, C. (2002). The Concept of Formative Assessment. *Practical Assessment, Research and Evaluation, 8*(9).

Breiman, L. (2001). Random Forests. *Machine Learning*, 5-32.

Burrows, S., Gurevych, I., & Stein, B. (2014). The Eras and Trends of Automatic Short Answer Grading. *International Journal of Artificial Intelligence in Education, 25*, 60-117.

Burstein, J., Leacock, C., & Swartz, R. (2001). Automated Evaluation of Essays and Short Answers. *5th CAA Conference.* Loughborough University.

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., & Bordes, A. (2017, September). Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. *CoRR, abs/1705.02364*, 670-680.

Cortes, C. (1995). Support-Vector Networks. *Machine Learning*, 273-297.

Daiber, J., Jakob, M., Hokamp, C., & Mendes, P. N. (2013). Improving Efficiency and Accuracy in Multilingual Entity Extraction. *9th International Conference on Semantic Systems.*

Daniel, J. (2012). Making Sence of MOOCs: Musing in a Maze of Myth, Paradox and Possibility. *Journal of Interactive Media in Education*.

Dessus, P., Lemaire, B., & Vernier, A. (2000). Free-Text Assessment in a Virtual Campus. *3rd International Conference on Human System Learning* , (pp. 61-76). Paris.

Dzikovska, M. O., Moore, J. D., Steinhauser, N., Campbell, G., Farrow, E., & Callaway, C. B. (2010). Beetle II: a system for tutoring and computational linguistics experimentation. *ACL 2010 System Demonstrations*, (pp. 13-18).

Dzikovska, M., Nielsen, R., Brew, C., Leacock, C., Giampiccolo, D., Bentivogli, L., . . . Dang, H. T. (2013). SemEval-2013 Task 7: The Joint Student Response Analysis and 8th Recognizing Textual Entailment Challenge. *Conference on Empirical Methods in Natural Language Processing* (pp. 263-274). Association for Computational Linguistics.

Ferragina, P., & Scaiella, U. (2010). TAGME: on-the-fly annotation of short text fragments (by wikipedia entities). *19th ACM International Conference on Information and Knowledge Management* .

Gabrilovich, E., & Markovitch, S. (2007). Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis . *Proceedings of The Twentieth International Joint Conference for Artificial Intelligence*, (pp. 1606-1611). Hyderabad, India.

Gagnon, M., Zouaq, A., & Jean-Louis, L. (2013). Can we use Linked Data Semantic Annotators for the Extraction of Domain-Relevant Expressions? *22nd International Conference on World Wide Web* (pp. 1239-1246). ACM.

Gikandi, J. W., Morrow, D., & Davis, N. E. (2011). Online formative assessment in higher education: A review of the literature. *Computers & Education, 57*(4), 2333-2351.

Heilman, M., & Madnani, N. (2012). ETS: Discriminative Edit Models for Paraphrase Scoring. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation* (pp. 529-535). Association for Computational Linguistics.

Heilman, M., & Madnani, N. (2013). 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), (pp. 275-279).

Hollands, F. M., & Devayani, T. (2014). *MOOCs: Expectations and reality: Full report.* New York: Center for Benefit-Cost Studies of Education, Teachers College, Columbia University.

Horrocks, I., Patel-Schneider, F., P., & Harmelen, F. v. (2003). From shiq and rdf to owl: The making of a web ontology language. *Web Semantics, 1*, 7-26.

Hou, W.-J., Tsao, J.-H., Li, S.-Y., & Chen, L. (2011). Automatic Assessment of Students' Free-Text Answers with Different Levels. *International Journal on Artificial Intelligence Tools*, 327–347.

Khurana, D., Koli, A., Khatter, K., & Singh, S. (2017). Natural Language Processing: State of The Art, Current Trends and Challenges. *CoRR, abs/1708.05148*.

Klein, R., Kyrilov, A., & Tokman, M. (2011). Automated assessment of short free-text responses in computer science using latent semantic analysis. *16th Annual Joint Conference on Innovation and Technology in Computer Science Education* (pp. 158-162). ACM.

Leacock, C., & Chodorow, M. (2003). C-rater: Automated Scoring of Short-Answer Questions. *Computers and the Humanities, 37*(4), 389-405.

Ledeneva, Y., & Sidorov, G. (2010). Recent Advances in Computational Linguistics. *Informatica (Slovenia), 34*, 3-18.

Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., . . . Bizer, C. (2015). DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web, 6*(2), 167-195.

Li, Y., & Yang, T. (2017). Word Embedding for Understanding Natural Language: A Survey. In Y. Li, T. Yang, & S. Srinivasan (Ed.), *Guide to Big Data Applications* (1st ed., Vol. 26). Springer International Publishing.

Madnani, N., Loukina, A., & Cahill, A. (2017). A Large Scale Quantitative Exploration of Modeling Strategies for Content Scoring. *12th Workshop on Innovative Use of NLP for Building Educational Applications*, (pp. 457-467). Copenhagen.

McCarthy, P. M., & Jarvis, S. (2010). MTLD, vocd-D, and HD-D: A validation study of sophisticated approaches to lexical diversity assessment. *Behavior Research Methods*.

McDonald, J., Bird, R., Zouaq, A., & Moskal, A. (2017). Short answers to deep questions: supporting teachers in large-class settings. *Journal of Computer Assisted Learning, 33*(4), 306-319.

McDonald, J., Knott, A., Zeng, R., & Cohen, A. (2011). Learning from student responses: A domain-independent natural language tutor. *Australasian Language Technology Association Workshop*, (pp. 148-156).

Mikolov, T., Le, Q. V., & Sutskever, I. (2013). Efficient Estimation of Word Representations in Vector Space. *CoRR*.

Nielsen, R. D., Ward, W., & Martin, J. H. (2008a). Learning to assess low-level conceptual understanding. *21st Intl. FLAIRS Conference*, (pp. 427–432).

Osisanwo, F., Akinsola, J., Awodele, O., Hinmikaiye, J. O., Olakanmi, O., & Akinjobi, J. (2017). Supervised Machine Learning Algorithms: Classification and Comparison . *International Journal of Computer Trends and Technology (IJCTT), 48*(3), 128-138.

Page, E. (1966). The imminence of grading essays by computer. *Phi Delta Kappan, 47*(5), 238-243.

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). BLEU: a Method for Automatic Evaluation of Machine Translation. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics.*

Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. *Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1532-1543). Association for Computational Linguistics.

Reilly, E., Stafford, R., Williams, K., & Corliss, S. (2014). Evaluating the validity and applicability of automated essay scoring in two massive open online courses. *The International Review of Research in Open and Distributed Learning, 15*(5).

Rodriguez, O. C. (2012). MOOCs and the AI-Stanford Like Courses: Two Successful and Distinct Course Formats for Massive Open Online Courses. *Journal of Open, Distance and E-Learning*.

Roy, S., Bhatt, H. S., & Narahari, Y. (2016). Transfer Learning for Automatic Short Answer Grading. *The 22nd European Conference on Artificial Intelligence (ECAI)*, (pp. 1622-1623).

Saha, S., Dhamecha, T., Marvaniya, S., Sindhgatta, R., & Sengupta, B. (2018). Sentence Level or Token Level Features for Automatic Short Answer Grading?: Use Both. *International Conference on Artificial Intelligence in Education.*

Sakaguchi, K., Heilman, M., & Madnani, N. (2015). Effective feature integration for automated short answer scoring. *Conference of the North American Chapter of The Association for Computational Linguistics: Human languate Technologies*, (pp. 1049-1054). Denver.

Schonlau, M., Guenther, N., & Sucholutsky, I. (2017). Text mining with n-gram variables. *Stata Journal, 17*(4), 866-881.

Scouller, K. (1998). The influence of assessment method on students' learning approaches: Multiple choice question examination versus assignment essay. *Higher Education, 35*(4), 453-472.

Shermis, M. D. (2015). Contrasting State-of-the-Art in the Machine Scoring of Short-Form Constructed Responses. *Educational Assessment*, 46-65.

Siemens, G. (2004). Connectivism: A Learning Theory for the Digital Age. *International Journal of Instructional Technology and Distance Learning, 2*, 3-10.

Sil, A., Shelton, A., Ketelhut, D. J., & Yates, A. (2012). Automatic Grading of Scientific Inquiry. *The 7th Workshop on Building Educational Applications Using NLP* (pp. 22-32). Montreal: Association for Computational Linguistics.

Statistics Canada. (2017). *Table 37-10-0018-01 Postsecondary enrolments, by registration status, institution type, status of student in Canada and sex.* Retrieved from Statistics Canada: https://www150.statcan.gc.ca/t1/tbl1/en/tv.action?pid=3710001801

Xiong, Y., & Suen, H. K. (2018). Assessment approaches in massive open online courses: Possibilities, challenges and future directions. *International Review of Education, 64*(2), 241-263.

Xu, Z. E., Chen, M., Weinberger, K. Q., & Sha, F. (2013). An alternative text representation to TF-IDF and Bag-of-Words. *CoRR*.

Yuan, L., & Powell, S. (2013). *MOOCs and Open Education: Implications for Higher Education.* Retrieved from Centre for Educational Technology & Inoperability Standards.: https://publications.cetis.org.uk/wp-content/uploads/2013/03/MOOCs-and-Open-Education.pdf

Zhang, Y., Shah, R., & Chi, M. (2016). Deep Learning + Student Modeling + Clustering: a Recipe for Effective Automatic Short Answer Grading. *The 9th International Conference on Educational Data Mining (EDM)*, (pp. 562-567).

Zhou, H., Zouaq, A., & Inkpen, D. (2017). DBpedia Entity Type Detection Using Entity Embeddings and N-Gram Models. *International Conference on Knowledge Engineering and the Semantic Web*, (pp. 309-322).

Annex I

This section presents the paper published by the author (in team effort with Dr. Amal Zouaq, Dr. Jenny McDonald, Dr. Jelena Jovanovic and Mr. Hadi Abdi Ghavidel) of this thesis in the proceedings of the 11th International Conference of Educational Data Mining conference in 2018 in Buffalo, NY (Alvarado Mantecon, Abdi Ghavidel, Zouaq, Jovanovic, & McDonald, 2018).

A Comparison of Features for the Automatic Labeling of Student Answers to Open-ended Questions

Jesus Gerardo Alvarado Mantecon
University of Ottawa
800 King Edward Avenue.
Ottawa, Canada.
+1 613 668 7214
jalva061@uottawa.ca

Hadi Abdi Ghavidel
University of Ottawa
800 King Edward Avenue.
Ottawa, Canada.
+1 613 890 1389
habdi018@uottawa.ca

Amal Zouaq
University of Ottawa
800 King Edward Avenue.
Ottawa, Canada.
+1 613 562 5800 ext.6227
azouaq@uottawa.ca

Jelena Jovanovic
University of Belgrade
Jove Ilica 154.
11000 Belgrade, Serbia.
+381 11 3950 853
jelena.jovanovic@fon.bg.ac.rs

Jenny McDonald
University of Auckland
Private Bag 92019.
Auckland, NZ.
+64 9 9238140
j.mcdonald@auckland.ac.nz

ABSTRACT
The automatic evaluation of text-based assessment items, such as short answers or essays, is an open and important research challenge. In this paper, we compare several features for the classification of short open-ended responses to questions related to a large first-year health sciences course. These features include a) traditional n-gram models; b) entity URIs (Uniform Resource Identifier) and c) entity mentions extracted using a semantic annotation API; d) entity mention embeddings based on GloVe, and e) entity URI embeddings extracted from Wikipedia. These features are used in combination with classification algorithms to discriminate correct answers from incorrect ones. Our results show that, on average, n-gram features performed the best in terms of precision and entity mentions in terms of f1-score. Similarly, in terms of accuracy, entity mentions and n-gram features performed the best. Finally, features based on dense vector representations such as entity embeddings and mention embeddings obtained the best f1-score for predicting correct answers.

Keywords
Short open-ended responses, N-gram models, Entity URIs, Entity Mentions, Entity embeddings, Mention embeddings.

1. INTRODUCTION

Due to the growth of Massive Open Online Courses (MOOCs) and increased class sizes in traditional higher education settings, the automatic evaluation of answers to open-ended questions has become an important challenge and one which has yet to be fully resolved. On the other hand, it has been shown that open-ended assessments are better able to capture a higher level of understanding of a subject than other machine-scored assessment items [24]. Still, MOOCs usually rely on multiple-choice questions since the evaluation of open-ended assessments requires more resources in massive online courses [32]. The human effort required to manually evaluate students' answers has escalated with the spread of large-scale courses that enroll several hundred, or even thousands of students. To tackle this challenge, we analyze textual responses to a set of open-ended questions designed to encourage deep responses from students. We explore the use of vector space models (VSMs) that represent each answer with a real-valued vector, and evaluate those models

on the task of classifying student responses into correct and not-correct. In particular, we examine and evaluate different feature sets that can be automatically derived from students' answers and used to represent those answers as vectors in a high dimensional space. The examined features do not require handcrafting based on the particularities of specific questions. Our main objective is to examine and compare the predictive power of different text features, automatically extracted from a corpus of answers to open-ended questions, on multiple classification algorithms.

We build VSMs using different text representations that result in either a sparse VSM (e.g., n-gram based VSM) or a dense VSM (e.g., VSM based on word embeddings). For sparse VSMs, we explore traditional n-gram features (unigrams, bigrams, trigrams, and n-grams that combine all of the previous features). We also investigate the usefulness of semantic annotations of students' responses for the classification task. Semantic annotation adds machine-readable meaning in the form of entities [21]. Hence, it enables the association of students' answers with vectors of domain-specific entities. Semantic annotators often rely on open Web-based knowledge bases such as DBpedia [13], an RDF representation of Wikipedia's semi-structured content. For example, given the entity *Aorta,* identified by a semantic annotator, we obtain its associated Web resource from DBpedia (http://dbpedia.org/page/Aorta), which further links to other related entities and properties from DBpedia. We make use of two semantic annotators: DBpedia Spotlight [19] and TAGME [6]. We query each annotator with the students' responses to obtain entities mentioned in the response. For each entity, we take the entity label and use it as *entity mention feature*, whereas the entity's Uniform Resource Identifier (URI) is used as *entity URI* feature.

To build a dense VSM, we rely on the entity mentions identified through semantic annotation and pre-trained word and entity embeddings. In particular, we retrieve vector representations of entity mentions using a GloVe model pre-trained on Wikipedia dumps [23]. Thus, our fourth feature set consists of entity mention embeddings based on GloVe. Finally, we represent entity URIs using a Wiki2Vec model trained on Wikipedia dumps to obtain another dense VSM. Hence, entity URI embeddings extracted from Wikipedia constitute our fifth feature set.

Given the short length of most answers and large vocabularies providing sparse vectors, we decided to include the last two sets of features to produce dense vector representations. In fact, dense vectors have shown an increase in performance for several natural language processing tasks [15]. Both GloVe [23] and Word2vec models [20] learn vector representations of words (called word embeddings) based on context. In total, we compare five types of features (n-gram, entity mentions, entity URIs, mention embeddings and entity embeddings) to train classification models to automatically label each student answer as correct or incorrect.

The rest of the paper is structured as follows: In Section 2, we present related work on automatic short answer grading. Then, we introduce our methodology, including the corpus description, our analysis pipeline and an in-depth description of our features. Section 5 describes the results of our experiments followed by the analysis of the effect of feature selection on our classifiers in Section 6. Finally, we discuss our findings and conclude the paper in Section 7 and 8.

2. RELATED WORK

One of the hot topics in the field of educational data mining is automatic short answer (response) grading (ASAG). In general, there are two kinds of ASAG approaches: response-based and reference-based [27]. In this paper, we analyze students' answers based on the response-based approach, which focuses only on students' answers. In contrast, reference-based ASAG also rely on the comparison of the student answer to the model answer.

Burrows et al. [4] classified all types of approaches to ASAG into five categories (eras): Concept mapping [8, 10, 12], Information

extraction [5], Corpus-based methods [11], Machine learning, and Evaluation [28, 30]. In the Machine Learning approach, which is the approach followed in this study, the trend is to build models (supervised or unsupervised) through data mining and natural language processing techniques in order to assess students' answers.

ASAG systems can also be categorized into semi-automatic (teacher-assisted) and fully-automatic systems. In semi-automatic systems, students' answers are processed (clustered) to facilitate the grading process. For example, Basu [1] applied k-medoids clustering to students' answers to ease the grading process. In another work, Jayashankar [9] proposed an integration of data mining and word clouds to help teachers evaluate student answers through visualization.

Fully-automatic systems produce grades for each student, with or without additional feedback. Several features are considered in training these systems: lexical features (e.g. word length), syntactic features (e.g. sentence length and part-of-speech), semantic features (e.g. semantic annotations and triples), discursive features (e.g. referential expressions), statistical features (e.g. language modelling like n-grams and embeddings), and similarity features (e.g. cosine similarity).

McDonald et al. [17, 18] evaluated Naive Bayes and Max Ent classifiers using a number of features like bag of words, word length, and word and character n-grams. Madnani et al. [14] used these types of features in combination with triples to examine the performance (accuracy) of 8 different classifiers and regressors (linear and nonlinear). In another work, Riordan et al. [26] combined n-gram features, answer length, and word and character embeddings to compare the performance of SVM (as a baseline) with neural architectures. In several approaches, features based on the similarity between the students' responses and the teacher's response were used together with n-grams. For example, Sakaguchi et al. [27] used stacked generalization [31] to integrate response-based and reference-based models. In particular, Sakaguchi et al. first built a classifier based on sparse response-based features (e.g. character n-gram and word n-gram); the obtained predictions were combined with dense reference-based features (e.g. BLEU [22]) to build another stacked classifier. Both classifiers were built as support vector regression (SVR) models. Zhang et al. [33] compared Deep Belief Networks (DBN) [2] to five classifiers such as Naive Bayes and Logistic Regression. The classifiers were trained on features extracted from three models, namely the Question model (e.g. question difficulty), the Student model (e.g. probability that a student learned a concept based on the student's past performance), and the Answer model (e.g. length difference between student answer and model answer). The DBN performed better than the other classifiers in terms of accuracy, precision, and F-measure, but not recall. Roy et al. [25] developed an ASAG system that can grade answers in different domains. They relied on an ensemble classifier of student answers (question-specific approach) and a numeric classifier based on the similarity score between the model answer and students' answers (question-agnostic approach). Their features were words, n-grams, and similarity scores between student answers and model answer. Finally, Tack et al. [29] used ensemble learning of five classifiers based on lexical features (e.g., word length), syntactic features (e.g., sentence length), discursive features (e.g., number of referential expressions), and a number of psycholinguistic norms.

In this work, we follow the response-based approach as we build classifiers based on students' answers. Our approach differs from previous works in that we carry out ASAG (and more specifically classification) by comparing six classifiers trained with both sparse vector representations (based on n-grams and entities) and dense vectors representations (GloVe, Word2Vec). One additional difference is the use of semantic annotations (entity mentions and entity URIs) to build some of our vector space models. Finally, the features used in this work do not necessitate a huge feature engineering effort as they come directly from text or from the use of

a semantic annotation API and an embedding model.

3. METHODOLOGY

We first give a description of the corpus used in our experiments, then we detail our overall approach as well as the metrics used in the evaluation phase. This is followed by an in-depth explanation of our features.

3.1 Corpus Description

Our data set is extracted from a corpus of student short-answer question (SAQ) responses drawn from a first-year human biology course (McDonald [16]). Among multiple elements in our data set, our experiments are based only on the labeled student responses to the survey and model answers (expected answers to the questions). Student SAQ responses and associated metadata were collected through a dialog system.

From the initial data set, we selected a sub-set of student answers based on the following criteria:

- Answers from the year 2012 only as this year is the one with the highest participation; out of 15,758 answers collected over 4 years, 7,548 originate from 2012.
- Out of the 42 different unique questions, we only use 6 questions that provide a reasonable number of responses as well as lengthy (deep) responses. We avoided questions that do not encourage answers that display deep understanding of the topic (e.g., yes-no questions, calculation questions or multiple choice questions).

The questions asked are designed to encourage deep responses from students [3]. The students are expected to explain or describe the knowledge obtained during the course in their own words rather than giving answers by the book. Table 1 presents the questions used in the study and their expected answers.

Table 1. Survey questions

ID	Question	Model Answer
Q.1	HR or heart rate is the number of times the heart beats each minute. A normal adult HR is around 72 beats/min. How would you check someone's HR?	You could measure their pulse.
Q.2	What is the pulse?	The pulse is a pressure wave or a pulsatile wave generated by the difference between systolic and diastolic pressures in the aorta.
Q.3	Inotropic state is a term that is sometimes used to describe the contractility of the heart. Can you describe what is meant by contractility?	Contractility is the force or pressure generated by the heart muscle during contraction.
Q.4	If you were 'building' a human being and you wanted to position receptors in the body to monitor blood pressure, where would you put them?	You'd probably want to put them near vital organs and at the main outflow from the heart. It turns out that the main human baroreceptors are located in the carotid sinuses and aortic arch.
Q.5	What feature of artery walls allows us to feel the pulse?	Artery walls are thick and strong and not very compliant
Q.6	Can you explain why you cannot feel a pulse in someone's vein?	You cannot feel a pulse in veins because the blood flow in veins is not pulsatile

The resulting sub-set amounts to 1,876 answers from 218 students to 6 questions. Note that not all students answered all the questions. Completing responses was voluntary, which accounts for the variability in the number of responses received to each question. In addition, the nature and quality of the responses are not necessarily representative of the class as a whole. Table 2 presents descriptive statistics on the students' answers to the selected subset of questions used in all the experiments.

Table 2. Statistics on students' answers per question

Question	Avg. words	Min. words	Max. words	Answers	Correct (%)
Q.1	6	1	36	243	65.43%
Q.2	9	1	82	422	17.54%
Q.3	6	1	31	316	33.86%
Q.4	4	1	34	151	54.97%
Q.5	3	1	27	171	25.15%
Q.6	9	1	34	361	31.86%

Each of these questions is associated with a set of students' answers. As an example, for question Q.6, we present the expected answer (i.e. Model answer), a deep response (Student Answer 1), and a simpler response (Student Answer 2):

Model Answer: You cannot feel a pulse in veins because the blood flow in veins is not pulsatile
Student Answer 1: The wave motion associated with the heart beat is stopped by the arteries and capillaries. Therefore, the vein has no pulse.
Student Answer 2: The blood flow is continuous.
Both student answers were labeled as correct by the human markers. Student Answer 1 would be considered a deeper answer than Student Answer 2, because it makes explicit the reasoning behind the answer, thus suggesting a better understanding of the topic.
The students' responses were manually evaluated by human markers with expertise in the domain of human biology. The annotators assigned a label negotiated through discussion. Such labels describe different aspects of an answer like quality of the response or correctness [16]. For example, answers may be labelled as incorrect, incomplete, and display disinterest in responding (dont-know label), among others. Further details on the labels used can be found in McDonald [16]. Table 3 displays some of those answers and the assigned labels.

Table 3. Student Answers sample

Question	Student Answer	Label
Q.5	Lack of elastic tissue	incorrect
Q.6	idk lol	dont-know
Q.4	In major arteries of the body, such as the common carotid or the aortic arch	ok
Q.3	ability to change volume	incomplete
Q.6	Ventricle contracts blood ejected into aorta, expanding vessel and increase pressure in vessel, wave of pressure cane felt is pulse	correct

For all of our experiments, we used model answer (expected answer) and student answers and re-labeled them as correct or not-correct. Correct answers comprise model answers plus all answers labeled as correct or ok. All other answers were re-labeled as not-correct. The resulting data set is composed of 65% not-correct answers and 35% correct answers.

Overall Approach

Our general approach can be described as follows:
Data pre-processing: in this step, we perform lemmatization and removal of punctuation marks and stop words (NLTK[23] stop words list) from the selected answers.
Feature extraction: We consider five types of features: n-gram, entity URIs, entity mentions, URI embeddings, and mention embeddings, which are detailed in section 4. We extract n-grams, entity URIs and entity mentions from student responses. Then, entity mentions are used to query a pre-trained GloVe model [23] to obtain mention embeddings. Likewise, entity URIs are used to query a pre-trained Wiki2Vec model [34] to obtain entity embeddings. Both GloVe and Wiki2Vec are pre-trained on Wikipedia.
Vector space model (VSM): For n-gram features, entity mentions, and entity URIs, we compute a vector representation of each answer by extracting a vocabulary from all students' answers and using TF-IDF as the relevance metric to weight each feature in an answer. As for mention embeddings and entity embeddings, we generate VSMs by averaging embeddings over all mentions or URIs appearing in an answer. The output from this step is one VSM representation of all answers for each feature type.
Classification task: we run several classification algorithms: the ZeroR algorithm as our baseline, Logistic regression, K-nearest neighbors (IBK), Decision trees (J48), Naïve Bayes, Support vector machine (SVM), and Random forest. We train each classifier using the entire data set of answers regardless of the question to which they belong. The rationale is that all answers belong to the same domain, and thus can be expected to be in a shared semantic space.

3.2 Evaluation Metrics

The evaluation is performed through 10-fold cross validation on each classifier. The metrics used for this purpose include:

- Accuracy: Percentage of correctly classified answers.

[23] https://www.nltk.org/

- Area Under the Curve (AUC): Probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance.
- Precision: Fraction of correctly classified answers within all classified instances.
- Recall: Fraction of relevant answers successfully retrieved.
- F1-score: Weighted harmonic mean of the precision and recall. It represents how precise and complete a classifier is.

4. FEATURE DESCRIPTION

4.1 N-gram Features

We create a vector representation for each answer based on n-grams. Table 4 shows some descriptive statistics on the obtained n-grams. We perform four experiments using different n-grams: unigrams, bigrams, trigrams, and the combination of all of them. To that end, four VSMs are built, one per n-gram group. Each vector holds the TF-IDF value of each item found in the answers. TF-IDF is calculated with the formula:

$$\text{tf-idf}_{i,j} = \text{tf}_{i,j} \times \left(\log \frac{1+n_d}{1+df_i} + 1\right)$$

Where $\text{tf}_{i,j}$ is the total number of occurrences of the term i in the student answer j, $\boldsymbol{n_d}$ is the total number of documents (i.e. answers) and $\boldsymbol{df_i}$ is the number of documents (i.e. answers) containing the term i.

Table 4. Total number of n-grams in answers for all questions

Answers	Unigrams	Bigrams	Trigrams
Unique	700	2114	4750
Total	6364	2589	3383

4.2 Entity URI Features

These features are based on entity URIs extracted from answers using two semantic annotators: DBpedia Spotlight and TAGME (see Sect. 1). The basic unit in the built VSM is the URI of a DBpedia resource (e.g. http://dbpedia.org/page/Baroreceptor). We send get requests to both annotators with the answers to be analyzed, and receive, for each answer, a list of entity mentions and their associated URIs. Table 5 shows statistics on the number of entity URIs and mentions (lowercase) retrieved by each of the two annotators.

Table 5. Number of entity URIs and mentions on all answers

Semantic Annotator	Entities		Mentions	
	Unique	**Total**	**Unique**	**Total**
Spotlight	143	1620	188	1620
TAGME	876	5054	806	5054

Table 6 provides an example of retrieved entity mentions and URIs for an answer to Q.2.

Table 6. Sample of retrieved entity URIs

Answer	Semantic Annotator	Mention	URI
Recoil caused by pressure in arteries	Spotlight	Recoil	dbpedia.org/page/Recoil
		Arteries	dbpedia.org/page/Artery
	TAGME	Recoil	dbpedia.org/page/Recoil
		Pressure	dbpedia.org/page/Pressure
		Arteries	dbpedia.org/page/Artery

We build a vector representation of each answer for each of the following configurations (i.e., vocabularies):

- Spotlight_URI: Set of entity URIs retrieved from all answers using DBpedia Spotlight.
- TAGME_URI: Set of entity URIs retrieved from all answers using TAGME.
- Intersection: Set intersection between the entity URIs retrieved from all answers with both tools.
- Union: Set union between the entity URIs retrieved from all answers with both tools.

This produces four VSMs based on entity URIs. The resulting VSMs use TF-IDF as the metric for estimating the value of each entity URI for each answer.

4.3 Entity Mention Features

We use the annotations retrieved by Spotlight and TAGME, selecting entity mentions as the basic units for building VSMs. A mention is a sequence of words spotted in an answer and associated to a URI. This means that an entity mention can be a unigram, but also a bigram or trigram. We compute the TF-IDF of each entity mention present in an answer to build its vector

representation. As in entity URI features, we have one vocabulary per configuration with four VSMs as the final output. The available configurations, based on mentions, used to build a vector representation of each answer, are analogous to those described for entity URIs, except that they are based on mentions (Spotlight_Mention, TAGME_Mention, Intersection and Union).

4.4 Entity Embedding Features

For this set of features, we rely on the Wiki2Vec[24] model, a Word2Vec implementation pre-trained on Wikipedia, where Wikipedia hyperlinks are replaced with DBpedia entities (URIs). The model was presented by Zhou et al. [34] and is based on 100-dimensional vectors. Word2Vec models can either learn to predict a word given its context (CBOW) or predict a context given a target word (Skip-gram) [20]. This creates a vector space in which similar words or entities are close to each other. Likewise, Wiki2Vec creates a vector space model in which similar DBpedia entities are close to each other. Given that our entity URIs reference DBpedia resources, we consider it a suitable match. For each configuration, we query the Wiki2Vec model with the entity URIs found in each answer to obtain their corresponding embeddings. Table 7 shows the percentage of entity URIs that are associated with an embedding vector in the Wiki2Vec model per configuration. We also show the percentage for the GloVe model which is presented in section 4.5.

Table 7. Coverage of entity URIs and mentions on their corresponding models (Wiki2Vec and GloVe)

Configuration	% of entity URIs in Wiki2Vec	% of entity mentions in GloVe
Spotlight_URI	97.5 %	50.46%
TAGME_URI	93.94 %	62.16%
Intersection	97.11 %	49%
Union	94.63 %	65.10%

For each configuration, we have one VSM. In each VSM, we aggregate the entity embeddings per answer by calculating the average of the entity URI vectors. This produces a single embedding that represents the answer.

4.5 Mention Embedding Features

For the mention embedding features, we rely on word embeddings, where each word is an entity mention instead of an entity URI. We use the GloVe model [23] trained using Wikipedia dumps from 2014 and build vectors with 100 dimensions (as for entity URI embeddings). Unlike Word2Vec, GloVe is a count-based model derived from a co-occurrence matrix. We query the GloVe model with the entity mentions found in each answer. The coverage of the model is given in Table 7. For each configuration, we have one VSM where each answer is represented as the average of the entity mention vectors.

5. RESULTS

For each feature set, we trained six classification algorithms, and evaluated 120 different models. Due to the space limit, we present only the top two performing classifiers (Random forest and SVM) in terms of overall accuracy for each of our feature sets. ZeroR is also included as the baseline.

5.1 N-gram Results

Table 8 shows the accuracy (ACC) and AUC obtained using n-gram features. Overall, the accuracy and AUC obtained with Random forest were always higher than with SVM. In particular, unigrams obtained the best accuracy of 88.40% as well as the highest AUC (.95) using Random forest.

Table 8. Accuracy & AUC using n-gram features

N-gram	Random Forest		SVM		ZeroR	
	ACC %	AUC	ACC %	AUC	ACC %	AUC
Unigrams	**88.40**	**.95**	84.44	.82	65.1	.50
Bigrams	81.97	.87	79.93	.73	65.1	.50
Trigrams	72.84	.68	72.12	.61	65.1	.50
N-grams	85.58	.93	84.25	.80	65.1	.50

Table 9 shows additional results for models cross-validated with n-gram features. For the

[24] https://github.com/idio/wiki2vec

correct label, our best classifier was Random forest using unigrams for the f1-score (.82) and trigrams or n-grams for best precision (.93). For the *not-correct* label, again, Random forest got the best results, using unigrams for both f1-score (.91) and precision (.88).

Table 9. Precision, recall & f1-score using n-gram features

Label	Classifier	N-gram	Precision	Recall	F1
Correct	Random Forest	Unigrams	.89	**.77**	**.82**
		Bigrams	.92	.53	.67
		Trigrams	**.93**	.24	.38
		N-grams	**.93**	.63	.75
	SVM	Unigrams	.79	**.76**	**.77**
		Bigrams	.85	.51	.64
		Trigrams	**.88**	.23	.37
		N-grams	.86	.65	.74
	ZeroR	Unigrams	0	0	0
		Bigrams	0	0	0
		Trigrams	0	0	0
		N-grams	0	0	0
Not-correct	Random Forest	Unigrams	**.88**	.95	**.91**
		Bigrams	.79	.98	.88
		Trigrams	.71	**.99**	.83
		N-grams	.83	.98	.90
	SVM	Unigrams	**.87**	.89	.88
		Bigrams	.79	.95	.86
		Trigrams	.71	**.98**	.82
		N-grams	.84	.95	**.89**
	ZeroR	Unigrams	.65	1	.79
		Bigrams	.65	1	.79
		Trigrams	.65	1	.79
		N-grams	.65	1	.79

A visible drop in recall from unigrams to trigrams (difference of .53) can be spotted for the correct label in both SVM and Random Forest. Based on the number of elements in each n-gram feature (Table 4), we observe that the amount of bigrams and trigrams is notably lower than unigrams. This can, at least partially, explain the lower recall using these features. Another noticeable result is that while the results obtained with Random Forest and SVM exceed the baseline for the *correct* label in terms of precision, recall and f1-score, the results for the *not-correct* label are closer to the baseline.

5.2 Entity Mention Results

The highest accuracy among these feature sets was achieved by Random Forest with the Union configuration (88.58%), as shown on Table 10. Again, Random forest outperformed SVM in terms of accuracy and AUC for each configuration.

Table 10. Accuracy & AUC using Entity mentions

Tool	Random Forest		SVM		ZeroR	
	ACC %	AUC	ACC %	AUC	ACC %	AUC
Spotlight Mention	78.61	.78	75.05	.67	65.1	.50
TAGME Mention	88.52	**.95**	85.22	.83	65.1	.50
Intersection	78.48	.77	75	.67	65.1	.50
Union	**88.58**	**.95**	85.34	.83	65.1	.50

Given that our Random forest classifier performed better in general for entity mentions, we based our following analysis on its results (Table 11). For the *correct* label, the use of TAGME_Mention or Union provided the highest f1-score (.83), but the use of TAGME_Mention alone provided slightly better precision (.87). On the *not-correct* label, once again, TAGME_Mention and the Union achieved the highest f1-score (.91), but this time the Union alone gave slightly better precision (.90).

Table 11. Precision, recall & f1-score using Entity mentions

Label	Classifier	Tool	Precision	Recall	F1
Correct	Random Forest	Spotlight Mention	.85	.47	.61
		TAGME Mention	**.87**	.79	**.83**
		Intersection	.84	.48	.61
		Union	.86	**.80**	**.83**
	SVM	Spotlight Mention	.77	.40	.53
		TAGME Mention	**.81**	.75	**.78**
		Intersection	.77	.40	.53
		Union	**.81**	**.76**	**.78**
	ZeroR	Spotlight Mention	0	0	0
		TAGME Mention	0	0	0
		Intersection	0	0	0
		Union	0	0	0
Not-correct	Random Forest	Spotlight Mention	.77	**.96**	.85
		TAGME Mention	.89	.94	**.91**
		Intersection	.77	**.95**	.85
		Union	**.90**	.93	**.91**
	SVM	Spotlight Mention	.75	**.94**	.83
		TAGME Mention	**.87**	.91	**.89**
		Intersection	.74	**.93**	.83

		Union	**.87**	.91	**.89**
	ZeroR	Spotlight Mention	.65	1	.79
		TAGME Mention	.65	1	.79
		Intersection	.65	1	.79
		Union	.65	1	.79

An explanation for the difference in performance between Spotlight_Mention and TAGME_Mention is the amount of mentions retrieved by each of the semantic annotators. Spotlight provided fewer annotations for the same answers than TAGME. In addition, our manual inspection of annotations revealed that TAGME tended to produce more accurate annotations than Spotlight. This suggests that higher quantity and quality of semantic annotations leads to a feature set that successfully differentiates between *correct* and *not-correct* answers.

5.3 Entity URI Results

The results presented in Table 12 show that Random forest provided highest accuracy and AUC on each configuration. The best accuracy and AUC were achieved by Random forest with TAGME_URI (86.60% and .94, respectively).

Table 12. Accuracy & AUC using Entity URIs

Tool	Random Forest		SVM		ZeroR	
	ACC %	**AUC**	**ACC %**	**AUC**	**ACC %**	**AUC**
Spotlight URI	80.55	.84	77.60	.75	60.8	.45
TAGME URI	**86.60**	**.94**	84.74	.82	65.1	.45
Intersection	77.03	.82	76.44	.74	59	.45
Union	86.50	.94	82.80	.80	63.6	.45

We notice that in terms of accuracy and AUC, TAGME_URI and Union on Random forest are slightly lower than TAGME_Mention and Union for Entity mention features.

Focusing on Random forest as the best performing classifier, we observe that for the *correct* label, the use of TAGME_URI and union of entity URIs provided the best f1-score of .80 (Table 13). In terms of precision, the union of entity URIs had a better performance (.86). For the *not-correct* label, again on Random forest, TAGME_URI and the Union configurations get better f1-score (.90). This time TAGME_URI alone provided the best precision (.88) for this label.

We observed that in some cases, the same mention was associated to different entity URIs in two different answers and that only one of the URIs was correct. When this happens, it affects the quality of the vector representation of student answers by increasing the number of URIs in the VSM vocabulary, thus making the representation even sparser.

Table 13. Precision, recall & f1-score using Entity URIs

Label	Classifier	Tool	Precision	Recall	F1
Correct	Random Forest	Spotlight URI	.82	.64	.72
		TAGME URI	.84	**.76**	**.80**
		Intersection	.77	.63	.69
		Union	**.86**	**.76**	**.80**
	SVM	Spotlight URI	.77	.62	.68
		TAGME URI	**.81**	**.73**	**.77**
		Intersection	.77	.61	.68
		Union	.80	.71	.75
	ZeroR	Spotlight URI	0	0	0
		TAGME URI	0	0	0
		Intersection	0	0	0
		Union	0	0	0
Not-correct	Random Forest	Spotlight URI	.80	.91	.85
		TAGME URI	**.88**	.92	**.90**
		Intersection	.77	.87	.82
		Union	.87	**.93**	**.90**
	SVM	Spotlight URI	.78	.88	.83
		TAGME URI	**.86**	**.91**	**.89**
		Intersection	.76	.87	.81
		Union	.84	.90	.87
	ZeroR	Spotlight URI	.61	1	.76
		TAGME URI	**.65**	1	**.79**
		Intersection	.59	1	.74
		Union	.64	1	.78

5.4 Entity Embedding Results

Among models trained using entity embeddings, the highest accuracy and AUC were achieved by Random forest with the TAGME_URI configuration, as shown in Table 14. For this feature set, we observe that Random forest has higher accuracy with TAGME_URI and Union than SVM on the same configurations; but SVM gets higher accuracy than Random forest using Spotlight_URI and Intersection. However, the AUC for Random forest is still higher than for SVM in all the configurations. We can also

observe an increase in accuracy and in AUC (although modest) for the baseline.

Table 14. Accuracy & AUC using Entity embeddings

Tool	Random Forest		SVM		ZeroR	
	ACC %	AUC	ACC %	AUC	ACC %	AUC
Spotlight URI	80.13	.86	81.13	.71	73.5	.50
TAGME URI	**82.67**	**.90**	75.46	.70	63.7	.50
Intersection	76.43	.81	79.64	.66	74.6	.49
Union	82.45	.89	80.79	.69	73.5	.50

Further inspection of the results obtained on cross-validated models (Table 15) reveals that this time, the highest results differ between classification algorithms. For the *correct* label, we obtained better f1-score with Random forest using the union of entity embeddings (.89). However, SVM provided better precision using Spotlight (.88). The *not-correct* label had both the best precision (.85 using the union of entity embeddings) and f1-score (.87 using TAGME_URI) results using Random forest.

Table 15. Precision, recall & f1-score using Entity embeddings

Label	Classifier	Tool	Precision	Recall	F1
Correct	Random Forest	Spotlight URI	.83	.92	.87
		TAGME URI	**.85**	.64	.73
		Intersection	.81	.90	.85
		Union	.82	**.97**	**.89**
	SVM	Spotlight URI	**.88**	.92	**.88**
		TAGME URI	.74	.50	.60
		Intersection	.82	.93	**.88**
		Union	.82	**.94**	**.88**
	ZeroR	Spotlight URI	.73	1	.85
		TAGME URI	0	0	0
		Intersection	**.75**	1	**.86**
		Union	.73	1	**.86**
Not-correct	Random Forest	Spotlight URI	.68	.47	.56
		TAGME URI	.82	**.93**	**.87**
		Intersection	.56	.37	.44
		Union	**.85**	.41	.55
	SVM	Spotlight URI	.70	.50	.58
		TAGME URI	**.76**	**.90**	**.82**
		Intersection	.67	.39	.50
		Union	.72	.45	.55
	ZeroR	Spotlight URI	0	0	0
		TAGME URI	**.64**	**1**	**.78**
		Intersection	0	0	0
		Union	0	0	0

Even though TAGME_URI provided better precision for the correct label, the union of entity embeddings got better f1-score and recall. The increase in f1-score can be related to the amount of entity URIs provided by the Union (set union of entity URIs from DBpedia Spotlight and TAGME). This suggests that more entities have a positive effect on performance. Similarly, as in accuracy, there was an increase in precision and f1-score on both labels for the baseline classifier.

5.5 Mention Embedding Results

Table 16 shows that when mention embeddings were used as features, SVM achieved the highest accuracy of 81.79% with the Union configuration. This is the first time that Random forest is surpassed by SVM in terms of accuracy, However, Random forest is still outperforming SVM in terms of AUC.

Table 16. Accuracy & AUC using mention embeddings

Tool	Random Forest		SVM		ZeroR	
	ACC %	AUC	ACC %	AUC	ACC %	AUC
Spotlight Mention	74.83	.74	75.83	.63	73.50	.50
TAGME Mention	80.85	.85	79.66	.76	63.69	.50
Intersection	78.50	.73	79.52	.68	74.40	.48
Union	79.80	**.86**	**81.79**	.71	73.50	.49

As in entity embeddings, the highest results differ between classification algorithms. Table 17 presents detailed results for the performance of Random forest and SVM using mention embeddings. For the *correct* label, the Random forest classifier with the union of mention embeddings had f1-score of .88 (the highest F1 value). For precision, SVM did better with either the intersection or union of mention embeddings (.83). The *not-correct* label had both best precision (.86 using TAGME_Mention) and f1-score (.83 using the union of mention embeddings) with the SVM classifier.

Table 17. Precision, recall & f1-score using mention embeddings

Label	Classifier	Tool	Precision	Recall	F1
Correct	Random Forest	Spotlight Mention	.78	.91	.84
		TAGME Mention	**.82**	.61	.70
		Intersection	.81	.93	.87
		Union	.80	**.98**	**.88**

	SVM	Spotlight Mention	.80	.90	.85
		TAGME Mention	.78	.61	.69
		Intersection	**.83**	.92	.87
		Union	**.83**	**.94**	**.88**
	ZeroR	Spotlight Mention	.73	1	.85
		TAGME Mention	0	0	0
		Intersection	**.74**	1	.85
		Union	.73	1	.85
Not-correct	Random Forest	Spotlight Mention	.55	.30	.39
		TAGME Mention	.81	**.92**	**.86**
		Intersection	.64	.36	.46
		Union	**.83**	.30	.44
	SVM	Spotlight Mention	.57	.36	.44
		TAGME Mention	**.80**	**.90**	**.85**
		Intersection	.65	.44	.52
		Union	.75	.48	.58
	ZeroR	Spotlight Mention	0	0	0
		TAGME Mention	**.64**	**1**	**.78**
		Intersection	0	0	0
		Union	0	0	0

6. FEATURE SELECTION

In this section, we describe the results obtained when applying two feature selection methods to our dataset: mean decrease impurity (MDI) and mean decrease accuracy (MDA). Both methods employ random trees to measure the importance of a feature [7]. We trained different classifiers with the selected features and compared their results to the same classifiers without feature selection.

First, we calculated the MDA and MDI scores for each feature in our data set and kept only features with scores strictly higher than 0. Negative or zero MDA/MDI values were either detrimental or unhelpful to the performance of the classifiers. Table 18 shows the number of features before and after feature selection.

Table 18. Number of remaining features with and without (WFS) feature selection

	Technique		
Features	**WFS**	**MDA**	**MDI**
N-gram	700	90	205
Entity Mention	665	99	179
Entity URI	875	109	236
Entity Embedding	100	83	84
Mention Embedding	300	161	117

Then, we trained and evaluated Random Forest, SVM, and ZeroR classifiers using each of the top performing configurations per feature set in terms of f1-score to compare the results obtained with and without feature selection. The obtained results (Figure 1) show that in most cases, feature selection led to a slight increase in the accuracy of our classifiers. Specifically, MDA improved the accuracy of the classifiers in every case by as much as 4.9 for SVM using mention embeddings as features. However, overall Random forest generally remained the best, in terms of accuracy, with and without feature selection.

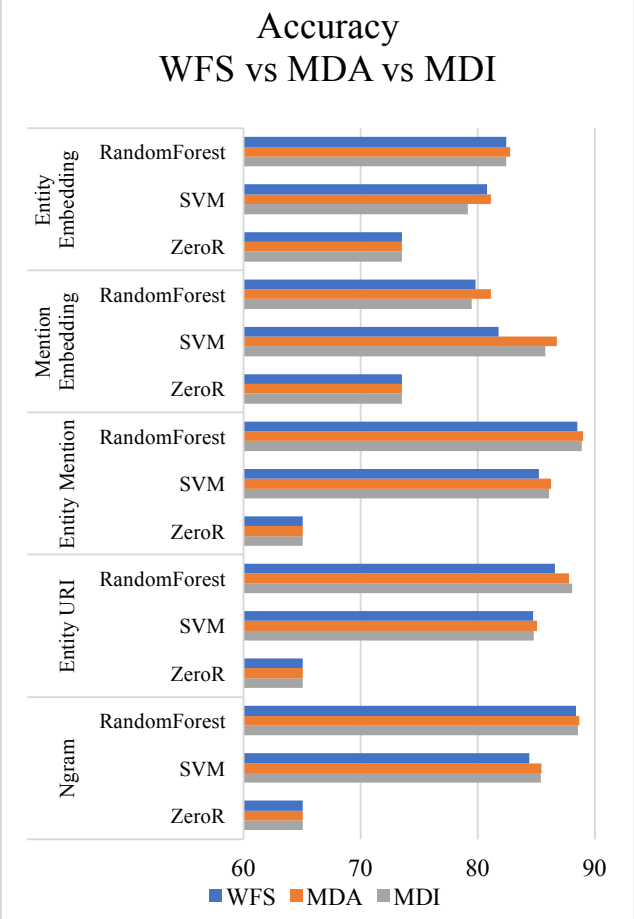


Figure 1. Accuracy without feature selection (WFS) versus MDA & MDI

7. DISCUSSION

Overall, our Random forest classifiers proved the best in terms of accuracy and AUC. The only

exception is with mention embeddings in which SVM did better in terms of accuracy by at most 1 percentage point. Therefore, we base our conclusions only on Random forest.

In terms of accuracy, there was not much difference between several feature sets as shown on Figure 2. The two best feature sets for accuracy were entity mentions with the union (88.58%) or TAGME configurations (88.52%) and n-gram features with the unigrams configuration (88.40%); these feature sets achieved the highest AUC (.95), as well.

In terms of precision (Figure 3), n-grams outperformed other feature sets for the *correct* label (.93) and entity mentions obtained the best results for the *not-correct* label using the union and TAGME configurations (.90, .89).

For F1-score (Figure 4), entity embeddings achieved the highest score for the *correct* label (.89 using the union configuration) closely followed by mention embeddings (union). Entity mentions (using the union or TAGME configurations) and unigrams did better for *not-correct* (.91) followed by entity URIs (.90 with TAGME and union) and n-grams.

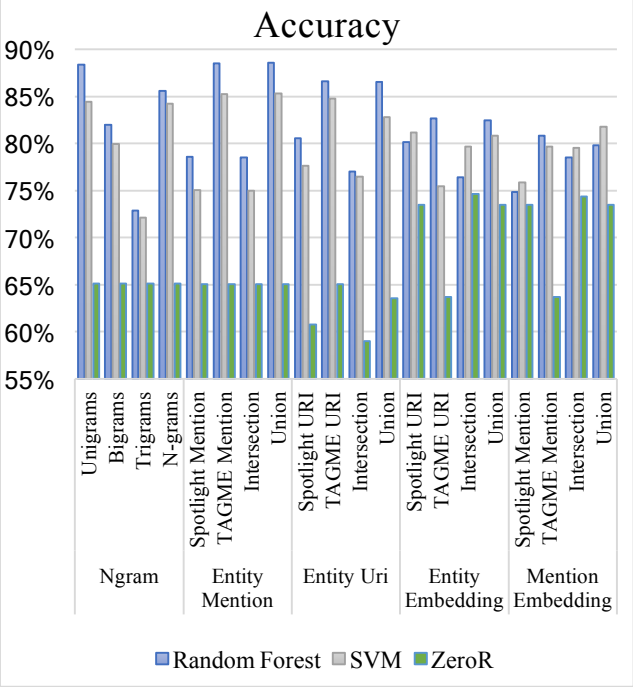


Figure 2. Accuracy results

When considering which class (*correct*, *not-correct*) we were best able to predict in terms of precision (Figure 3), we found that the detection of *correct* answers was better than *not-correct answers*, with differences ranging from .01 to .25 with Random forest. N-gram features were better at detecting *correct* answers than *not-correct* ones; while entity mentions did better for the *not-correct* (using union or TAGME) label. In 14 out of our 20 possible configurations, the classifiers were more precise in detecting *correct* answers. This is the case despite the unbalanced ratio of 35% *correct* answers and 65% *not-correct* answers used for training. When we focus on the f1-score (Figure 4) we obtain better results for the *not-correct* label. We observe that the union configuration for entity embeddings and mention embeddings is the best for *correct* answers while entity mentions (TAGME or union) followed by unigrams outperform the other features for the *not-correct* answers.

On average, unigrams are the best at differentiating between correct and not-correct labels in terms of precision while entity mentions (either with TAGME or Union) is preferred in terms of f1-score.

The best configuration based on semantic annotations depends on the considered evaluation metric. Based on accuracy, features that use mentions (entity mentions and mention embeddings) performed better with either union or TAGME. The feature sets that use URIs (entity URIs and entity embeddings) performed better with URIs obtained using TAGME. In both cases, the use of TAGME alone obtains either the best result or is very close to the highest value. For f1-score, the use of TAGME for entity mentions and entity URIs provided the same results as the union for both labels; additionally, TAGME and union are also the best configurations for both entity mentions and entity URIs. Entity embeddings and mention embeddings had their best f1-score on the *correct* label using the union, but better f1-score for *not-correct* using TAGME alone. When we average the f1-score for both labels, we obtain higher results with TAGME. The reason for very similar results with TAGME and the union is that the

annotations provided by Spotlight were often a subset of those provided by TAGME.

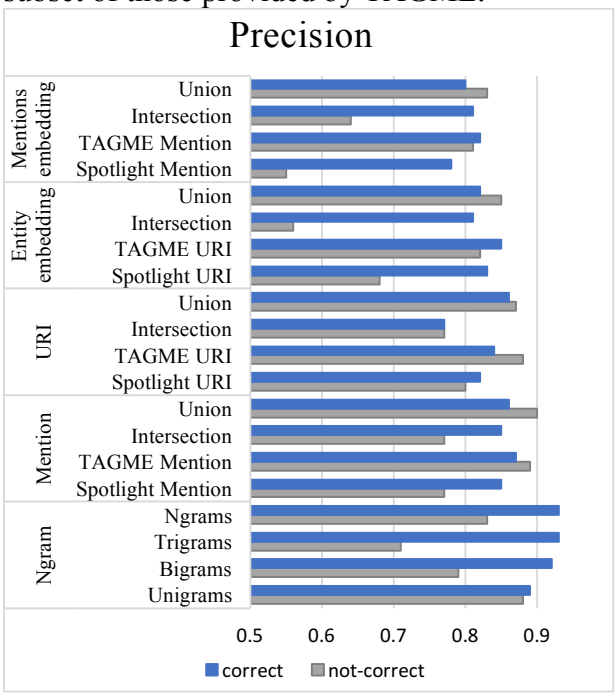


Figure 3. Precision results for Random forest

Both entity and mention embeddings performed worse than n-gram features and semantic annotations models based on accuracy. However, one interesting observation is that, for the *correct* label, entity and mention embeddings outperformed all features on f1-score (Figure 4). Entity embeddings obtained slightly better results (precision, f1-score and accuracy) compared to mention embeddings.

Our feature selection efforts show that MDI did not consistently improve the overall accuracy of our classifiers. It was the MDA feature selection technique which provided improvement in all the cases. The increase in accuracy ranged from .3% to 4.9%.

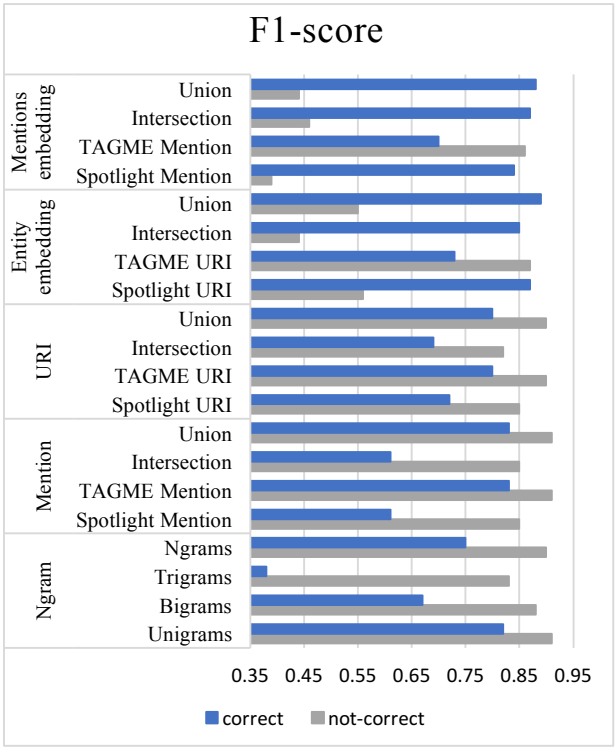


Figure 4. F1-score results for Random Forest

8. CONCLUSION

In this paper, we compared several vector-based feature sets coupled with classifiers for the ASAG task.

In general, we showed that on average, entity mention features (TAGME or union) are the top features in terms of f1-score while n-gram features (unigrams) are the best in terms of precision. For the detection of *correct* answers, we showed that n-gram features (trigrams and n-grams) and features based on embeddings (entity and mention embeddings with the union configuration) are the most effective in terms of precision and f1-score respectively. In terms of semantic annotations, TAGME provided the best accuracy for each feature with the exception of entity mentions, where the union configuration slightly outperformed TAGME alone. Finally, the MDA feature selection technique slightly improved the accuracy of all the classifiers.

One of the main limitations of this study is the unbalanced set of labeled answers available in the corpus. Another limitation is associated with the configuration of semantic annotators as we only tested the default level of confidence for

each annotator. One additional limitation, for mention embeddings specifically, is the relatively low coverage obtained using GloVe. We plan to address these limitations in future work by testing the proposed features against other available ASAG datasets. We also intend to experiment with varying the level of confidence and similar parameters of the semantic annotators. Another important step will be to exploit a combination of the current features to benefit from their respective strengths for the correct and not correct labels. Finally, we will explore other methods for response classification using additional features that exploit model answers and deep learning architectures.

9. ACKNOWLEDGMENTS

This research is supported by an INSIGHT grant from the Social Sciences and Humanities Research Council of Canada (SSHRC).

10. REFERENCES

[1] Basu, S., Jacobs, C., and Vanderwende, L. 2013. Powergrading: a clustering approach to amplify human effort for short answer grading. *Transactions of the Association for Computational Linguistics*, 1, 391-402.

[2] Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. 2007. Greedy layer-wise training of deep networks. In *Proceedings of Advances in Neural Information Processing Systems Conference*, 153-160.

[3] Biggs, J., and Tang, C. 2011. *Teaching for quality learning at university (4th ed.)*. London, UK: McGraw-Hill International.

[4] Burrows, S., Gurevych, I., and Stein, B. 2015. The eras and trends of automatic short answer grading. *International Journal of Artificial Intelligence in Education*, 25, 1, 60–117.

[5] Dessus, P., Lemaire, B., and Vernier, A. 2000. Free-text assessment in a virtual campus. In *Proceedings of the 3rd International Conference on Human System Learning,* 61-75.

[6] Ferragina, P., and U. Scaiella. 2010. TAGME: On-the-fly Annotation of Short Text Fragments (by Wikipedia Entities). In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, 1625–1628.

[7] Gilles Louppe, Louis Wehenkel, Antonio Sutera, and Pierre Geurts. 2013. Understanding variable importances in forests of randomized trees. In *Proceedings of Advances in Neural Information Processing Systems Conference*. 431–439.

[8] Heilman, M., and Madnani, N. 2013. ETS: Domain adaptation and stacking for short answer scoring. In Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: In *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval)*, 2, 275-279.

[9] Jayashankar, S., and Sridaran, R. 2017. Superlative model using word cloud for short answers evaluation in eLearning. *Education and Information Technologies*, 22, 5, 2383-2402.

[10] Jordan, S., and Mitchell, T. 2009. e-Assessment for learning? The potential of short-answer free-text questions with tailored feedback. *British Journal of Educational Technology*, 40, 2, 371–385.

[11] Klein, R., Kyrilov, A., and Tokman, M. 2011. Automated assessment of short free-text responses in computer science using latent semantic analysis. In G. Roßling, T. Naps, C. Spannagel (Eds.), In *Proceedings of the 16th annual joint conference on innovation and technology in computer science education*, 158–162. Darmstadt: ACM. (CAPS'3), 61-76.

[12] Leacock, C., and Chodorow, M. 2003. C-rater: automated scoring of short-answer questions, *Computers and the Humanities*, 37, 4, 389–405.

[13] Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., van Kleef, P., Auer, Sö. and Bizer, C. 2015. DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web Journal*, 6, 167-195.

[14] Madnani, N., Loukina, A., and Cahill, A. (2017). A Large Scale Quantitative Exploration of Modeling Strategies for Content Scoring. *In Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, 457-467.

[15] Magooda, A. E., Zahran, M. A., Rashwan, M., Raafat, H. M., and Fayek, M. B. 2016. Vector Based Techniques for Short Answer Grading. In *Proceedings of Florida Artificial Intelligence Research Society Conference* (*FLAIRS)*, 238-243.

[16] McDonald, J., Bird, R. J., Zouaq, A., and Moskal, A. C. M. 2017. Short answers to deep questions: supporting teachers in large-class settings, *Journal of Computer Assisted Learning*. 33, 4 306–319.

[17] McDonald, J., Knott, A., and Zeng, R. 2012. Free-text input vs menu selection: exploring the difference with a tutorial dialogue system. In *Proceedings of the Australasian Language Technology Association Workshop 2012*, 97-105.

[18] McDonald, J., Knott, A., Zeng, R., and Cohen, A. 2011. Learning from student responses: A domain-independent natural language tutor. In *Proceedings of the Australasian Language Technology Association Workshop,* 148-156.

[19] Mendes, P. N., Jakob, M., García-Silva, A., and Bizer, C. (2011, September). DBpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems*, ACM, 1-8.

[20] Mikolov, T., Chen, K., Corrado, G. and Dean, J. 2013. Efficient Estimation of Word Representations in Vector Space, In *Proceedings of Workshop at International Conference on Learning Representations ICLR.*

[21] Oren, E, Moller K, Scerri, S, Handschuh, S and Sintek, M 2006, What are Semantic Annotations? *Relatório técnico. DERI Galway*, 9, 62-75.

[22] Papineni, K., Roukos, S., Ward, T., and Zhu, W. J. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, 311-318.

[23] Pennington, J., Socher, R., and Manning, C. 2014. GloVe: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532-1543.

[24] Reilly, E. D., Stafford, R. E., Williams, K. M., and Corliss, S. B. 2014. Evaluating the validity and applicability of automated essay scoring in two massive open online courses, *The International Review of Research in Open and Distributed Learning*, 15, 5, 83-98.

[25] Roy, S., Bhatt, H. S., and Narahari, Y. 2016. Transfer Learning for Automatic Short Answer Grading. In *Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI)*, Hague, Netherlands, 1622-1623.

[26] Riordan, B., Horbach, A., Cahill, A., Zesch, T., and Lee, C. M. 2017. Investigating neural architectures for short answer scoring. *In Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications,* 159-168.

[27] Sakaguchi, K., Heilman, M., and Madnani, N. 2015. Effective feature integration for automated short answer scoring. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1049-1054.

[28] Shermis, M. D. 2015. Contrasting state-of-the-art in the machine scoring of short-form constructed responses. *Educational Assessment*, 20, 1, 46-65.

[29] Tack, A., François, T., Roekhaut, S., and Fairon, C. 2017. Human and Automated CEFR-based Grading of Short Answers. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, 169-179.

[30] The Hewlett Foundation. 2012. The Automated Student Assessment Prize: Short Answer Scoring. http://www.kaggle.com/c/asap-sas

[31] Wolpert, D. H. 1992. Stacked generalization. Neural Networks, 5, 2, 241–259.

[32] Yuan, L., and Powell, S. 2013. MOOCs and open education: Implications for higher education. Centre for Educational Technology & Inoperability Standards. Retrieved from http://publications.cetis.ac.uk/wp-content/uploads/2013/03/MOOCs-and-Open-Education.pdf

[33] Zhang, Y., Shah, R., and Chi, M. 2016. Deep Learning + Student Modeling + Clustering: a Recipe for Effective Automatic Short Answer Grading. In *Proceedings of the 9th International Conference on Educational Data Mining* (*EDM)*, 562-567.

[34] Zhou, H., Zouaq, A., and Inkpen, D. 2017. DBpedia Entity Type Detection Using Entity Embeddings and N-Gram Models. In *Proceedings of International Conference on Knowledge Engineering and the Semantic Web*, 309-322.

Annex II

This section present extended results describing performance per label (correct and incorrect) for the reference-based combinations on each dataset (INSIGHT, SemEval Beetle and SciEntBank).

Top results in terms of precision and f1-score are highlighted in bold for each label.

		Correct			Incorrect		
		Precision	Recall	F1-score	Precision	Recall	F1-score
S6	GloVe	0.88	0.95	0.92	0.89	0.77	0.82
	fastText	0.88	**0.97**	0.92	0.93	0.76	0.84
CS	GloVe	0.74	0.74	0.74	0.52	0.52	0.52
	fastText	0.75	0.75	0.75	0.54	0.53	0.54
WS	GloVe	0.73	0.87	0.79	0.62	0.4	0.48
	fastText	0.73	0.87	0.79	0.62	0.4	0.48
SS	GloVe	0.78	0.94	0.85	0.81	0.49	0.61
	fastText	0.78	0.94	0.85	0.81	0.49	0.61
WS + CS	GloVe	0.78	0.8	0.79	0.61	0.58	0.6
	fastText	0.84	0.84	0.84	0.7	0.69	0.7
WS +SS	GloVe	0.78	0.89	0.83	0.72	0.53	0.61
	fastText	0.78	0.89	0.83	0.72	0.53	0.61
SS + S6	GloVe	0.88	0.95	0.91	0.89	0.76	0.82
	fastText	**0.89**	**0.97**	**0.93**	**0.94**	0.78	**0.85**
CS + S6	GloVe	0.88	0.95	0.91	0.88	0.75	0.81
	fastText	**0.89**	**0.97**	**0.93**	0.93	0.78	**0.85**
SS + CS	GloVe	0.83	0.83	0.83	0.69	0.69	0.69
	fastText	0.84	0.84	0.84	0.7	0.7	0.7

WS + S6	GloVe	0.88	0.96	0.92	0.92	0.75	0.82
	fastText	**0.89**	**0.97**	**0.93**	**0.94**	0.78	**0.85**
WS + SS + CS	GloVe	0.83	0.86	0.85	0.72	0.68	0.7
	fastText	0.86	0.88	0.87	0.77	0.72	0.75
WS + SS + S6	GloVe	0.88	0.95	0.91	0.89	0.76	0.82
	fastText	**0.89**	**0.97**	**0.93**	0.93	0.76	0.84
WS + CS +S6	GloVe	0.88	0.96	0.92	0.9	0.75	0.82
	fastText	**0.89**	**0.97**	**0.93**	0.93	0.77	0.84
SS + CS + S6	GloVe	0.88	0.95	0.91	0.89	0.76	0.82
	fastText	**0.89**	**0.97**	**0.93**	0.93	**0.79**	**0.85**
SS + WS + CS + S6	GloVe	0.88	0.96	0.92	0.91	0.75	0.82
	fastText	**0.89**	**0.97**	**0.93**	0.93	0.78	**0.85**

Table Annex II - 1. RFC results per label on INSIGHT dataset (10FCV) with reference-based features

		Correct			Incorrect		
		Precision	Recall	F1-score	Precision	Recall	F1-score
S6	GloVe	0.93	0.92	0.92	0.85	0.87	0.86
	fastText	**0.95**	0.94	**0.95**	0.9	**0.91**	0.9
CS	GloVe	0.74	0.74	0.74	0.52	0.52	0.52
	fastText	0.7	0.53	0.6	0.4	0.57	0.47
WS	GloVe	0.71	0.71	0.71	0.47	0.47	0.47
	fastText	0.71	0.71	0.71	0.47	0.47	0.47
SS	GloVe	0.75	0.8	0.78	0.58	0.51	0.54
	fastText	0.75	0.8	0.78	0.58	0.51	0.54
WS + CS	GloVe	0.71	0.71	0.71	0.47	0.47	0.47
	fastText	0.7	0.68	0.69	0.43	0.45	0.44

WS +SS	GloVe	0.75	0.81	0.78	0.59	0.51	0.54
	fastText	0.75	0.81	0.78	0.59	0.51	0.54
SS + S6	GloVe	0.94	0.93	0.93	0.87	0.88	0.88
	fastText	**0.95**	**0.95**	**0.95**	**0.91**	0.9	0.9
CS + S6	GloVe	0.93	0.92	0.92	0.85	0.87	0.86
	fastText	**0.95**	**0.95**	**0.95**	0.9	**0.91**	0.9
SS + CS	GloVe	0.76	0.81	0.78	0.59	0.51	0.55
	fastText	0.76	0.81	0.78	0.59	0.51	0.55
WS + S6	GloVe	0.93	0.92	0.92	0.85	0.88	0.86
	fastText	**0.95**	**0.95**	**0.95**	0.9	**0.91**	0.9
WS + SS + CS	GloVe	0.75	0.81	0.78	0.59	0.51	0.54
	fastText	0.75	0.81	0.78	0.59	0.51	0.55
WS + SS + S6	GloVe	0.94	0.93	0.93	0.87	0.88	0.88
	fastText	**0.95**	**0.95**	**0.95**	0.9	**0.91**	**0.91**
WS + CS +S6	GloVe	0.93	0.92	0.92	0.85	0.88	0.86
	fastText	**0.95**	**0.95**	**0.95**	0.9	**0.91**	**0.91**
SS + CS + S6	GloVe	0.94	0.93	0.93	0.88	0.88	0.88
	fastText	**0.95**	**0.95**	**0.95**	**0.91**	0.9	**0.91**
SS + WS + CS + S6	GloVe	0.94	0.93	0.93	0.87	0.88	0.88
	fastText	**0.95**	**0.95**	**0.95**	**0.91**	**0.91**	**0.91**

Table Annex II - 2. SVM results per label on INSIGHT dataset (10FCV) with reference-based features

		Correct			Incorrect		
		Precision	Recall	F1-score	Precision	Recall	F1-score
S6	GloVe	0.86	0.82	0.84	0.89	0.91	0.9
	fastText	0.88	0.82	0.85	0.89	0.93	0.91

CS	GloVe	0.48	0.61	0.54	0.68	0.56	0.62
	fastText	0.57	0.6	0.58	0.72	0.7	0.71
WS	GloVe	0.69	0.69	0.69	0.79	0.79	0.79
	fastText	0.69	0.69	0.69	0.79	0.79	0.79
SS	GloVe	0.5	0.1	0.17	0.61	0.93	0.74
	fastText	0.5	0.1	0.17	0.61	0.93	0.74
WS + CS	GloVe	0.66	0.69	0.67	0.78	0.76	0.77
	fastText	0.69	0.7	0.69	0.8	0.78	0.79
WS +SS	GloVe	0.69	0.69	0.69	0.79	0.79	0.79
	fastText	0.69	0.69	0.69	0.79	0.79	0.79
SS + S6	GloVe	0.85	0.79	0.82	0.87	0.91	0.89
	fastText	0.89	0.8	0.84	0.88	0.93	0.9
CS + S6	GloVe	0.84	0.78	0.81	0.86	0.9	0.88
	fastText	0.87	0.82	0.85	0.89	0.92	0.9
SS + CS	GloVe	0.51	0.62	0.56	0.71	0.6	0.65
	fastText	0.57	0.62	0.6	0.73	0.69	0.71
WS + S6	GloVe	0.85	0.8	0.82	0.87	0.91	0.89
	fastText	0.9	0.81	0.85	0.88	**0.94**	0.91
WS + SS + CS	GloVe	0.66	0.68	0.67	0.78	0.77	0.78
	fastText	0.68	0.72	0.7	0.8	0.77	0.79
WS + SS + S6	GloVe	0.82	0.77	0.8	0.85	0.89	0.87
	fastText	0.89	0.82	0.85	0.88	**0.94**	0.91
WS + CS +S6	GloVe	0.86	0.78	0.82	0.86	0.91	0.89
	fastText	0.89	0.81	0.85	0.88	0.93	0.9
SS + CS + S6	GloVe	0.82	0.8	0.81	0.87	0.88	0.87

	fastText	**0.91**	0.82	0.86	0.89	**0.94**	**0.92**
SS + WS + CS + S6	GloVe	0.85	0.78	0.82	0.86	0.91	0.89
	fastText	0.9	**0.84**	**0.87**	**0.9**	**0.94**	**0.92**

Table Annex II - 3. RFC results per label on SemEval Beetle test set with reference-based features

		Correct			Incorrect		
		Precision	Recall	F1-score	Precision	Recall	F1-score
S6	GloVe	0.78	0.84	0.81	0.89	0.84	0.86
	fastText	0.81	**0.9**	0.85	**0.93**	0.86	0.89
CS	GloVe	0.5	0.83	0.63	0.8	0.45	0.58
	fastText	0.67	0.81	0.73	0.85	0.73	0.79
WS	GloVe	0.71	0.81	0.75	0.86	0.78	0.81
	fastText	0.71	0.81	0.75	0.86	0.78	0.81
SS	GloVe	0.67	0.07	0.12	0.61	**0.98**	0.75
	fastText	0.67	0.07	0.12	0.61	**0.98**	0.75
WS + CS	GloVe	0.69	0.85	0.76	0.88	0.75	0.81
	fastText	0.68	0.84	0.75	0.87	0.74	0.8
WS +SS	GloVe	0.71	0.81	0.76	0.86	0.78	0.82
	fastText	0.71	0.81	0.76	0.86	0.78	0.82
SS + S6	GloVe	0.78	0.83	0.81	0.88	0.85	0.86
	fastText	0.81	**0.9**	0.85	**0.93**	0.86	0.89
CS + S6	GloVe	0.78	0.85	0.81	0.89	0.84	0.86
	fastText	**0.82**	**0.9**	**0.86**	**0.93**	0.86	**0.9**
SS + CS	GloVe	0.5	0.83	0.62	0.8	0.44	0.57
	fastText	0.68	0.81	0.74	0.85	0.74	0.79
WS + S6	GloVe	0.79	0.86	0.82	0.9	0.85	0.87

	fastText	0.81	**0.9**	0.85	**0.93**	0.86	0.89
WS + SS + CS	GloVe	0.69	0.85	0.76	0.88	0.75	0.81
	fastText	0.68	0.82	0.74	0.86	0.75	0.8
WS + SS + S6	GloVe	0.79	0.84	0.82	0.89	0.85	0.87
	fastText	0.81	**0.9**	0.85	**0.93**	0.86	0.89
WS + CS +S6	GloVe	0.79	0.86	0.82	0.9	0.85	0.87
	fastText	0.81	**0.9**	0.85	**0.93**	0.86	0.89
SS + CS + S6	GloVe	0.79	0.84	0.81	0.89	0.85	0.87
	fastText	0.81	**0.9**	0.85	**0.93**	0.86	0.89
SS + WS + CS + S6	GloVe	0.79	0.85	0.82	0.89	0.85	0.87
	fastText	0.81	**0.9**	0.85	**0.93**	0.86	0.89

Table Annex II - 4. SVM results per label on SemEval Beetle test set with reference-based features

		Correct			Incorrect		
		Precision	Recall	F1-score	Precision	Recall	F1-score
S6	GloVe	0.75	0.54	0.62	0.71	0.86	0.78
	fastText	**0.8**	0.57	0.66	**0.73**	**0.89**	0.8
CS	GloVe	0.46	0.46	0.46	0.59	0.59	0.59
	fastText	0.54	0.5	0.52	0.64	0.68	0.66
WS	GloVe	0.6	0.44	0.51	0.65	0.78	0.71
	fastText	0.6	0.44	0.51	0.65	0.78	0.71
SS	GloVe	0.6	0.36	0.45	0.63	0.82	0.71
	fastText	0.6	0.36	0.45	0.63	0.82	0.71
WS + CS	GloVe	0.51	0.45	0.48	0.62	0.67	0.64
	fastText	0.58	0.47	0.52	0.65	0.75	0.69
WS +SS	GloVe	0.55	0.48	0.51	0.64	0.71	0.67

	fastText	0.55	0.48	0.51	0.64	0.71	0.67
SS + S6	GloVe	0.75	0.55	0.64	0.72	0.86	0.78
	fastText	0.78	0.55	0.64	0.72	**0.89**	0.79
CS + S6	GloVe	0.77	0.55	0.64	0.72	0.87	0.79
	fastText	0.79	0.54	0.64	0.72	**0.89**	0.79
SS + CS	GloVe	0.51	0.48	0.49	0.62	0.65	0.64
	fastText	0.51	0.45	0.48	0.62	0.68	0.65
WS + S6	GloVe	0.76	0.55	0.64	0.72	0.87	0.79
	fastText	0.77	0.56	0.65	0.72	0.87	0.79
WS + SS + CS	GloVe	0.52	0.47	0.49	0.63	0.68	0.65
	fastText	0.62	0.54	0.58	0.68	0.75	0.71
WS + SS + S6	GloVe	0.75	0.56	0.64	0.72	0.86	0.78
	fastText	0.79	0.57	0.66	**0.73**	**0.89**	0.8
WS + CS +S6	GloVe	0.73	0.52	0.61	0.7	0.85	0.77
	fastText	0.76	0.57	0.65	**0.73**	0.87	0.79
SS + CS + S6	GloVe	0.77	0.56	0.65	0.72	0.87	0.79
	fastText	**0.8**	**0.58**	**0.67**	**0.73**	**0.89**	**0.81**
SS + WS + CS + S6	GloVe	0.77	0.54	0.63	0.71	0.88	0.79
	fastText	0.78	0.55	0.64	0.72	0.88	0.79

Table Annex II - 5. RFC results per label on SemEval SciEntBank test set with reference-based features

		Correct			Incorrect		
		Precision	Recall	F1-score	Precision	Recall	F1-score
S6	GloVe	0.72	0.7	0.71	0.78	**0.79**	0.79
	fastText	0.72	0.7	0.71	0.78	**0.79**	0.79
CS	GloVe	0.49	**0.77**	0.6	0.69	0.38	0.49

	fastText	0.58	0.69	0.63	0.72	0.62	0.67
WS	GloVe	0.58	0.54	0.56	0.67	0.7	0.68
	fastText	0.58	0.54	0.56	0.67	0.7	0.68
SS	GloVe	0.59	0.45	0.51	0.65	0.77	0.7
	fastText	0.59	0.45	0.51	0.65	0.77	0.7
WS + CS	GloVe	0.58	0.53	0.55	0.66	0.7	0.68
	fastText	0.6	0.6	0.6	0.69	0.69	0.69
WS +SS	GloVe	0.59	0.48	0.53	0.66	0.75	0.7
	fastText	0.59	0.48	0.53	0.66	0.75	0.7
SS + S6	GloVe	0.72	0.73	0.72	0.79	**0.79**	0.79
	fastText	0.72	0.7	0.71	0.78	**0.79**	0.79
CS + S6	GloVe	0.72	0.7	0.71	0.78	**0.79**	0.79
	fastText	0.72	0.7	0.71	0.78	**0.79**	0.79
SS + CS	GloVe	0.59	0.45	0.51	0.65	0.77	0.7
	fastText	0.58	0.61	0.59	0.69	0.67	0.68
WS + S6	GloVe	0.72	0.75	0.73	0.8	0.78	0.79
	fastText	0.71	0.73	0.72	0.79	0.77	0.78
WS + SS + CS	GloVe	0.6	0.5	0.55	0.66	0.75	0.7
	fastText	0.61	0.55	0.58	0.68	0.73	0.71
WS + SS + S6	GloVe	0.71	0.75	0.73	0.8	0.77	0.79
	fastText	0.7	0.73	0.72	0.79	0.77	0.78
WS + CS +S6	GloVe	**0.73**	0.75	**0.74**	**0.81**	**0.79**	**0.8**
	fastText	0.71	0.73	0.72	0.79	0.77	0.78
SS + CS + S6	GloVe	0.72	0.73	0.72	0.79	**0.79**	0.79
	fastText	0.72	0.7	0.71	0.78	**0.79**	0.79

SS + WS + CS + S6	GloVe	0.71	0.75	0.73	0.8	0.77	0.79
	fastText	0.71	0.73	0.72	0.79	0.77	0.78

Table Annex II - 6. SVM results per label on SemEval SciEntBank test set with reference-based features