Analysis, Optimization and Development of an Answer Scoring System

Master thesis by:
Iñigo Lopez-Gazpio

Supervisor:
Montse Maritxalar Anglada

Master in Analysis and Processing of Language

Department of Computer Languages and Systems
University of the Basque Country (UPV/EHU)

Donostia-San Sebastián, September 2014

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Acknowledgements

First of all, I would like to thank the IXA Natural Language Processing group for supporting this work; and, particularly, to the people that have been working with me and have taught me so much. Montse, Itziar and Oier are for me, the three of them, principal supervisors.

Last but not least I would like to thank all the people involved in open software communities for having shared with all of us such great tools that make possible works like this one. Our gratitude is the least we can give them to appreciate their work.

II

Abstract

The main contribution of this work is to analyze and describe the **state of the art performance** as regards answer scoring systems from the SemEval-2013 task, as well as to continue with the **development of an answer scoring system** (EHU-ALM) developed in the University of the Basque Country. On the overall this master thesis focuses on finding any possible configuration that lets improve the results in the SemEval dataset by using attribute engineering techniques in order to find **optimal feature subsets**, along with trying different **hierarchical configurations** in order to analyze its performance against the traditional one versus all approach. Altogether, throughout the work we propose two alternative strategies: on the one hand, to improve the EHU-ALM system without changing the architecture, and, on the other hand, to improve the system adapting it to an hierarchical configuration. To build such new models we describe and use distinct attribute engineering, data preprocessing, and machine learning techniques.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

1.1 Natural language understanding

The 1996 edition of Ethnologue listed 6703 living languages in the world [Grimes, 1996]. Eventually, the 2005 edition listed 6912 living languages [Gordon and Grimes, 2005]. Today[1], statistics offered in the Ethnologue's site lists a total amount of 7106 living languages. Obviously, languages do not vary that often, instead, the evolution of the inventory and the decisions of experts to determine what a language is and what is not a language do change more frequently. As cited by Stephen R. Anderson it turns that linguists do not have a "clear and reasonably precise notion of how many languages there are in the world ... because the very notion of enumerating languages is a lot more complicated than it might seem". Despite the differences among languages, the majority of them share a common goal: model and describe the world.

From the standpoint of computational linguistics the main challenge is and always has been to build robust and sophisticated enough models that are able to understand natural language. Such a task is extremely difficult due to the fact that machines have complete lack of common sense. To provide any machine with that background is particularly challenging as the natural language comprehension goes beyond the analysis of the words and their syntactic structure[2]. In order to build language models computational linguists have used several distinct approaches and approximations, such as: statistics based models, machine learning based models, expert knowledge based models, ... or even hybrid systems that use a certain combination of distinct techniques.

Machine learning techniques can perform important tasks by learning from examples and generalizing for new ones. These techniques can turn a relatively small amount of knowledge into a large generalizing capacity. This idea is particularly interesting because the probability of having the same exact example is very unlikely, even if large amount of learning data is used. In fact, access to data is the most critic area when dealing with machine learning, but not the only one, because data alone is not enough. Nowadays, as more data becomes available, machine learning is widely used in computer science and other fields. For instance, in the recent past much of the progress achieved in the natural language processing and in the machine learning field has been used in technology-based learning systems. The range of technology-based learning applications is wide, but, in particular, we will focus on systems that automatically grade student answers. This kind of

[1] Thursday, May 01, 2014

[2] Actually, the natural language comprehension is closely linked to the semantic representation of the general world knowledge.

systems are usually integrated in either intelligent tutoring frameworks or in reading comprehension frameworks in order to assess student on errors and personalize their formative feedback.

1.2 Contribution of the work

The Semantic Evaluation workshop (SemEval[3]) focuses on the evaluation of semantic analysis systems. The main objective of the workshop is to compare systems that are capable of analyzing various semantic phenomena that occur in texts. In the 2013 edition they proposed an **answer assessment challenge** to open questions. Actually, the task consisted on offering to the community a **student response classification** challenge based on educational data and caught the participation of different research groups around the world; including the IXA natural language processing group[4].

This master thesis focuses on **attribute engineering** techniques in order to find optimal feature subsets, as well as trying different **hierarchical configurations** in order to analyze its performance against the traditional multiclass classification approach[5]. The main contribution of the present master thesis is to:

1. Analyze and describe the **state of the art performance** as regards answer scoring systems from the SemEval-2013 task 7.

2. Optimize the **answer scoring system** developed in the University of the Basque Country. The objective of such optimization is to find any optimal feature set that lets improve the results on the described challenge.

3. Transform the architecture of the EHU-ALM system into an **hierarchical architecture** by breaking down the main task into smaller **binary subtasks**. The main objective of the experiment is to test whether a hierarchical hierarchy can help improve performance on the classification challenge.

The whole work reads as follows: *chapter* 2 analyzes and describes the whole SemEval task and also compares performance of teams that participated in the challenge. In particular, it covers the objectives and motivations of the

[3] http://alt.qcri.org/

[4] http://ixa.si.ehu.es/Ixa

[5] In a multiclass classification approach one single classifier is used to distinguish classes from each other.

task, the creation of the annotated dataset, the description and the evaluation results of the baseline systems, the analysis of the evaluation scenarios, a brief summary of all the participants and a detailed analysis of performance for all participants. It also discusses the advantages and disadvantages of domain-specific systems and the improvement flexible systems offer in order to avoid domain-specific resource gathering. *Chapter* 3 introduces the reader to the **machine learning field** and focuses on the **EHU-ALM** system in order to find a specific strategy that may improve the performance of the system. For the task of improving a classifier we have followed an iterative strategy in which **distinct attribute engineering, data preprocessing, and machine learning techniques** have been used. Actually, this chapter describes statistical tests to measure the contribution of features towards machine learning models, uses distinct feature subsets selection techniques to find relevant features among the candidate feature set, implements an equation to score the contribution of candidate feature subsets and uses learning curves to plot the performance of the new classifier. *Chapter* 4 describes in detail how hierarchical classifiers have been built out of the contingency table of the EHU-ALM system. In particular it describes the motivation and fundamentals of such a system, the **hypothesis of breaking down the main task into smaller subproblems**, the basic notions of hierarchical clustering in order to build the classifier tree and several grid-search optimizing techniques in order to train all the levels of the hierarchical configuration according to distinct criteria. We propose **two distinct hierarchical configurations**: the first one starting the classification task from the most distanced classes to the most similar ones, and the second one starting the classification task from the most similar classes to the most distances ones. Once the hierarchical configurations are built we use the same techniques as in *chapter* 3 to optimize the new systems. Chapter 4 concludes with a detailed comparison of all systems. Additionally, it describes an experiment to **reduce the dimensionality of the SemEval-2013 dataset** into three dimensions in order to visualize the feature space, thus, understand the job machine learning models face. To finish with, *chapter* 5 makes a brief summary of the whole master thesis and highlights the most important aspects seen over the work. To conclude, it exhibits different strategies to keep working on for the near and far future.

Chapter 2

Analysis of SemEval-2013 Task 7

Contents

2.1 Main task description

To **automatically grade** answers and provide **appropriate feedback** is a challenging task that, for instance, could be valuably used to **personalize the formative feedback**. Such a task is not trivial because the system must analyze and identify errors, mistakes or misconceptions based on the student answer and the reference response(s).

The approach of building domain-specific models in which systems are preloaded with a repertoire of questions and reference answers has been discussed in [Dzikovska et al., 2010], [Glass, 2001], [Aleven et al., 2001] and [Callaway et al., 2006]. This approach provides high adaptation to the objective domain, but, on the contrary, it is limited because the domain must be small enough to permit comprehensive knowledge. In contrast, to add new exercises that rely in different concepts out of the domain model is a very labor-intensive work even for small domains.

The contribution of [Dzikovska et al., 2012] is the basis to provide new corpora and new baselines in order to enable researchers continue developing specialized tools. Actually, the work mentioned permitted the **SemEval-2013 task 7** community shared challenge to happen. This task aims for effective tutorial feedback by promoting **flexible systems** in order to avoid the weaknesses of data-driven approaches.

The response analysis task addresses the problem of scoring student responses from different science domains. More specifically, given a question, an indeterminate number of correct reference answers and a student answer, the goal is to **determine the accuracy** of each answer. Student answers can be classified as correct, partially correct or incomplete, contradictory, irrelevant, and non in the domain. There are three different **evaluation scenarios** in the challenge: the unseen answer scenario, for scoring new answers to questions represented in the training data; the unseen question scenario, for scoring answers to new questions from domains represented in the training data; and the unseen domain scenario, for scoring answers to questions from new domains. The whole task is completely documented in [Dzikovska et al., 2013].

2.2 Corpora

The SemEval-2013 task 7 challenge is based on the following corpora which contain manually labeled student statements to several questions. This way, it provides an **annotated data set** that is labeled for the mentioned five re-

sponse classes[1]. The corpora has been created out of two established sources: the BEETLE corpus, a data set collected and annotated during the evaluation of the BEETLE II tutorial dialogue system [Dzikovska et al., 2010]; and the SCIENTSBANK corpus, a set of student answers to questions from sixteen science modules in the Assessing Science Knowledge assessment inventory [Nielsen et al., 2008].

The BEETLE corpus

The BEETLE corpus consists of the interactions between students and the BEETLE II tutorial dialogue system. The BEETLE II system is an intelligent tutoring engine that teaches students in basic electricity and electronics. At first, students spend three to five hours reading material, building and observing circuits in the simulator and interacting with a dialogue-based tutor. They used the keyboard to interact with the system, and the computer tutor asked them questions and provided feedback via a text-based chat interface. The data from 73 undergraduate volunteer participants at southeastern US university were recorded and annotated to form the BEETLE human-computer dialogue corpus.

The SCIENTSBANK corpus

The SCIENTSBANK corpus consists of student responses to science assessment questions. Specifically, around ten thousand answers were collected covering sixteen distinct science subject areas within physical sciences, life sciences, earth sciences, space sciences, scientific reasoning and technology. The answers came from students in grades from three to six in schools of North America and the tests were part of the Assessing Science Knowledge (ASK)[2].

Using corpus-specific annotations each student answer has been evaluated against the reference answer(s) it associates with, in order to assign **unified labels**. Labels are summarized in table 2.1 and the distribution of labels in both corpora is summarized in table 2.2.

As a result, according to scores obtained by answers, learners could be encouraged by intelligent tutoring frameworks to face questions in a different way, for instance, as annotated in table 2.3.

[1] correct, partially correct incomplete, contradictory, irrelevant, and non in the domain.
[2] Berkeley Lawrence Hall of Science.

Label	Definition
Correct	The student answer is a complete paraphrase of the reference answer
Partially	Contains information nuggets, but some parts of the answer are missing
Irrelevant	Contains domain content but does not provide the necessary information
Contra	The answer explicitly contradicts some part of the reference answer
Non_Dom	The answer does not include domain content

Table 2.1: Set of labels and their respective descriptions.

Label	BEETLE		SCIENTSBANK	
	train (%)	test (%)	train (%)	test (%)
Correct	1665 (0.42)	520 (0.41)	2008 (0.40)	2451 (0.42)
Partially	919 (0.23)	284 (0.23)	1324 (0.27)	1274 (0.22)
Irrelevant	113 (0.03)	36 (0.03)	1115 (0.22)	1548 (0.27)
Contra	1049 (0.27)	355 (0.28)	499 (0.10)	539 (0.09)
Non_Dom	195 (0.05)	63 (0.05)	23 (0.005)	23 (0.004)
Total:	3941	1258	4969	5835

Table 2.2: Distribution of labels in data.

Label	Framework feedback
Correct	Encourage the student to continue working
Partially	Encourage the student to fill the gaps of the answer
Irrelevant	Encourage the student to address relevant concepts
Contra	Emphasize there is a mistake in the student answer
Non_Dom	Indicator that the student is frustrated or confused. It may require special attention

Table 2.3: Set of labels and their respective feedback.

2.3 Baseline systems

[Dzikovska et al., 2012] established **three baseline systems**. Each one was developed using an alternative approach:

1. A straightforward majority class baseline.
2. The BEETLE II dialogue system baseline.
3. A classifier based on lexical-similarity measures.

2.3.1 Majority Baseline

The majority baseline is a classifier used to create a frontier for comparing other machine learning techniques. Straightforward majority classifiers identify the dominant class in the training data and classify test instances to whichever class it is.

2.3.2 BEETLE II system baseline

The output of the BEETLE II system was mapped into the five labels described in table 2.1 to be considered a baseline. To perform this task, they obtained the domain-specific semantic representations of student statements by means of parsers and a set of hand written rules. Finally, they matched those representations against the representations of correct statements. The BEETLE II system baseline is only available for the BEETLE corpus.

2.3.3 Lexical Baseline

The lexical baseline has been built using an implementation of a decision tree classifier. The decision tree used is the Weka [Hall et al., 2009] implementation of the C4.5 decision tree with the default set up. To train the classifier and build the baseline system they used text-similarity features obtained by means of the Text::Similarity package[3]. In total four metrics were used: the raw score of literal matchings, the F-measure score, the Lesk score and the cosine score[4]. These metrics were calculated between the student response and the reference statement(s); and between the student response and the original question. Thus, in total eight features were calculated for each student, reference and question triplet.

2.3.4 Evaluation results of baseline systems

The performance of the baseline systems was measured after holding back part of the dataset. As a consequence, the remaining split could be used as the test set of the SemEval-2013 task 7 community shared task.

As showed in table 2.2 the distribution of different classes is unbalanced in the annotated data, and therefore, [Dzikovska et al., 2012] presents several summary statistics to describe data, for instance: per-class precision,

[3]The complete documentation of the Text::Similarity package can be found in the Comprehensive Perl Archive Network (CPAN): http://search.cpan.org/dist/Text-Similarity/lib/Text/Similarity.pm

[4]Brief explanations of these attributes and more can be found in the appendix.

recall and F-score. Average evaluation scores are calculated using the micro-averaging equation and the macro-averaging equation. Given a set of classes ζ and an evaluation summary metric φ, the macro-average value of each metric across classes is defined as:

$$\varphi_{macro} = \frac{1}{N_\zeta} \sum^{\zeta} \varphi_\zeta \tag{2.1}$$

where N_ζ represents the number of classes. The micro-average value is defined as:

$$\varphi_{micro} = \frac{1}{N} \sum^{\zeta} (|\zeta| * \varphi_\zeta) \tag{2.2}$$

where N denotes the total number of instances and $|\zeta|$ denotes the cardinality of class ζ.

In brief, both are averaging measures but macro-averaging (equation 2.1) does not take the class size into account, in contrast, micro-averaging (equation 2.2) is weighted by class size. As a result, a system that performs well on most common classes will have a high micro-average score. However, the macro-average value shows the performance on small classes more appropriately in cases of unbalanced classification problems (it favors general performance across classes regardless of class size).

Performance in the BEETLE corpus

The performance of all baseline systems is summarized in table 2.4. As expected, the majority baseline obtains higher performance in the micro-average than in the macro-average, but far from those of other baselines. Particularly, the micro recall score of the majority baseline is similar to the one obtained by the BEETLE II system, but, without the shadow of a doubt at the expense of a much lower precision. In fact, the F-measure values show clearly the difference in performance between both systems. In contrast, if we compare the BEETLE II system against the lexical-similarity baseline, we observe that the F-measures are quite similar, despite the fact that the BEETLE II system performs better concerning precision, but worst concerning recall. This is not unexpected because the BEETLE II system was designed to prefer rejecting examples where classification confidence was not high, resulting in a lower recall.

Baseline	Majority			Lexical-similarity			BEETLE II		
Score	prec.	recall	F	prec.	recall	F	prec.	recall	F
Macro-average	0.09	0.20	0.12	0.44	0.46	0.45	0.62	0.39	0.46
Micro-average	0.18	0.42	0.25	0.53	0.55	0.54	0.71	0.44	0.53

Table 2.4: Baseline system performance in the BEETLE corpus.

Performance in the SCIENTSBANK corpus

The performance of all baseline systems is summarized in table 2.5. The majority system baseline obtains similar results to the ones obtained in the BEETLE corpus. These results are expected because the majority class balance is similar in both corpora as we have previously seen in table 2.2. Apart from that, we can observe that the lexical-similarity baseline outperforms the majority baseline. On the overall, the **scores from SCIENTSBANK are slightly lower than those from BEETLE**. It needs to be taken into account that the SCIENTSBANK corpus covers a wider range of science areas than BEETLE does, and as a result, it is more complicated for the decision tree to generalize in this corpus. In addition, the average answer length is considerably higher than in the BEETLE corpus, which considerably difficults the task.

Baseline	Majority			Lexical-similarity		
Score	prec.	recall	F	prec.	recall	F
Macro-average	0.08	0.20	0.11	0.29	0.29	0.29
Micro-average	0.16	0.40	0.23	0.41	0.43	0.42

Table 2.5: Baseline system performance in the SCIENTSBANK corpus.

2.4 Evaluation scenarios

The train and test set used in the SemEval-2013 task 7 challenge is described in section 2.2. As we can observe in table 2.2 the BEETLE training data had about 4000 student answers whereas the SCIENTSBANK training data had about 5000 student answers. Concerning test data, task organizers [Dzikovska et al., 2013] evaluated systems according to **three different scenarios**:

- To test how well the system generalizes towards new answers to previously seen questions (Unseen Answer scenario, UA).

- To test how well the system generalizes towards new questions of the same science domain subjects (Unseen Question scenario, UQ).
- To test how well the system generalizes towards completely new science domains (Unseen Domain scenario, UD).

Unseen Answer scenario

A set to assess system performance on answers to questions pertaining to the training data. This subset was created holding back a random set of student answers for each question in the training set.

Unseen Question scenario

A set to assess system performance on responses to previously unseen questions, but within the same domains as in the training data. This subset was created holding back all student answers to a set of randomly selected questions for each domain in the training set.

Unseen Domain scenario

A set to assess system performance on responses to previously unseen questions on topics not seen in the training data. This subset was created **only for the SCIENTSBANK corpus** holding back all three science modules out of the sixteen modules available.

Test set split and evaluation metrics

Whereas the unseen answer scenario is sufficient for the vast majority of computational linguistic applications, the unseen question and the unseen domain scenarios allow researchers to test **how well their system generalizes to far and near domains**. In addition, these scenarios implies challenging systems, because, for example: the range of things that are "irrelevant" for a given question is potentially very big and is difficult to learn.

The described scenarios allowed to evaluate system performance and their generalization capabilities more appropriately. As a consequence, the test set of the BEETLE data was divided into two test subsets (unseen answers 35% and unseen questions 65%); and similarly, the SCIENTSBANK test set was divided into three test subsets (unseen answers 9%, unseen questions 13% and unseen domains 78%). The resulting subsets are shown in table 2.6.

Notice that the test set split is also unbalanced, but in the overall it follows the same partitioning as in the training set (see table 2.7).

Label	Test BEETLE			Test SCIENTSBANK			
	UA	UQ	Total	UA	UQ	UD	Total
Correct	176	344	520	233	301	1917	2451
Partially	112	172	284	113	175	986	1274
Irrelevant	17	19	36	133	193	1222	1548
Contradictory	111	244	355	58	64	417	539
Non_Domain	23	40	63	3	0	20	23
Total:	439	819	1258	540	733	4562	5835

Table 2.6: Distribution of labels across test scenarios.

	Correct	Partially	Contradictory	Irrelevant	Non_domain	Total instances
Training set	41%	25%	17%	14%	3%	8910
Test set	42%	22%	13%	22%	1%	7093

Table 2.7: Distribution of classes in the training set and in the test set.

2.5 Conclusions of the task analysis

The SemEval-2013 task 7 challenge focused on associating student answers with their respective labels. Generally, these techniques are used to **automatically score answers to provide appropriate pedagogic feedback** and detailed explanations. Nevertheless, the challenge is not trivial because systems must perform correctly to engage students and keep their interest.

Altogether, [Dzikovska et al., 2012] designed three baseline systems for the task and one of those systems **significantly outperformed** the majority class baseline in both corpora. From the baseline results it is also noticeable the good performance obtained by the flexible lexical-similarity system. Actually, the BEETLE II baseline was a system specifically designed to parse the BEETLE corpus answers, and it just got similar results comparing to the lexical-similarity system.

Another key contribution of the authors had been to **unify the two data sets** into one large student response corpus with a common annotation scheme. The resulting corpus allowed researchers to work on the community shared task, not only that, but also they believe that the natural language capabilities needed for this task will be directly applicable to a far wider range of tasks. Taking into account the foundations described in the chapter, [Dzikovska et al., 2013] invited researchers to submit systems that generalize across domains.

2.6 Participants

Participants of the Joint Student Response Analysis were invited to submit up to **three runs**. A total of nine teams participated in the task.

1. **Celi: EDITS and Generic Text Pair Classification**

   Celi [Kouylekov, 2013] is a generic system capable of recognizing multiple semantic relationships between two texts. The system implements and harmonizes different approaches, such as: an open source package for recognizing textual entailment (RTE), text distance computations and text similarity algorithms.

2. **CNGL: Grading Student Answers by Acts of Translation**

   CNGL [Biçici and van Genabith, 2013] models the question answering assessment as a translation task. These translation tasks are modeled by computational approaches that identify translations between any two data sets with respect to a reference corpus selected in the same domain. The system uses more than 280 features to find a function f that approximates best the student answer correctness given the question and the reference answer.

3. **CoMeT: Integrating different levels of linguistic modeling for meaning assessment**

   CoMeT [Hahn and Meurers, 2013] is a combination of three types of systems in one meta classifier: an alignment-based approach which uses various levels of linguistic abstractions, a semantics-based system for meaning comparison and a variety of bag-of-words based approaches constructed out of all of the student answers in the training data.

4. **CU : Computational Assessment of Short Free Text Answers**

   CU [Okoye et al., 2013] mainly exploits shallow natural language techniques to analyze how much knowledge can be gained using text similarity measures.

5. **EHU-ALM: Similarity-Feature Based Approach for Student Response Analysis**

   EHU-ALM [Aldabe et al., 2013] is a supervised system based on syntactic and semantic similarity features. The system splits the data at training to simulate the given test scenarios.

6. **ETS: Domain Adaptation and Stacking for Short Answer Scoring**

   ETS [Heilman and Madnani, 2013] integrates item-specific n-gram features and more general text similarity measures. It uses stacking and domain adaptation to achieve the automatic scoring of short text responses.

7. **LIMSIILES: Basic English Substitution for Student Answer Assessment**

   LIMSIILES [Gleize and Grau, 2013] describes a method modeled as a paraphrase identification problem based on substitutions by basic English variants. The hypothesis is based on the idea that reducing the diversity of the vocabulary will help compare meaning of sentences. The system computes traditional lexical and semantic similarity measures over the simplified language lexicon of the data set.

8. **SOFTCARDINALITY: Hierarchical Text Overlap for Student Response Analysis**

   SOFTCARDINALITY [Jimenez et al., 2013] presents a general model for object comparison that has been used in several text applications by the author. The system is based on the recursive usage of a text overlap equation (the soft cardinality method) and a new mechanism for weight propagation.

9. **UKP-BIU: Similarity and Entailment Metrics for Student Response Analysis**

   UKP-BIU [Zesch et al., 2013] combines text similarity measures with a textual entailment engine. The system uses both technologies to extract features, and combines them in a supervised model.

2.6.1 Analysis of participants

The majority of the participants use some form of system combination approach with several components, where an additional module makes the final decision taking that information as input. Up to a point, as part of the implementation of those components, most of the systems use some kind of syntactic processing, such as: part of speech analysis, dependency analysis and constituency structure analysis; or some kind of text-to-text similarity measures, such as: Latent Semantic Analysis (LSA), Latent Dirichlet Allocation (LDA), Bag of Words based measures (BOW) and N-gram based measures. Altogether, systems taking part in the SemEval 2013 task 7 community challenge can be **classified in three sets**:

1. Systems that approach the challenge combining **text similarity techniques**.

2. Systems that approach the challenge combining **textual entailment recognizing techniques**.

3. **Hybrid systems** that use a mixture or combination of distinct approaches.

Whereas text similarity measures are usually more direct to calculate (similarity equations that output a measure score), Textual Entailment systems turn to use more complicated techniques [Magnini et al., 2014]. Doubtless, **a text entails the hypothesis if the meaning of the hypothesis can be derived from the text**. But this approach can differ slightly in student assessment systems because students often skip details that are mentioned in the question; and so, a more precise reformulation of the task is necessary. In this approach, the entailing text consists of both: the original question and the student answer [Dzikovska et al., 2013].

There is no doubt not only **different systems but also effective ones** can be obtained using distinct approaches. Nevertheless, each approach seems to reveal its own advantages and disadvantages. For instance, text similarity based systems are expected to perform better when there are multiple reference answers rather than few ones [Okoye et al., 2013]. In this article, they also suggest that stop words are important to provide context, because student answers are free text and because they usually use pronouns to reference items appearing in the question. On the contrary, the potential limits of edit distance algorithms are mentioned in [Kouylekov, 2013], although they argue that it still provides good performance for some close domain tasks. In the same way, [Heilman and Madnani, 2013] argue that

the relevance of edit-based distance approaches is not clear towards the improvement in performance, as their system did not always improved with the addition of such features. Moreover, text similarity based systems that rely on concrete units, such as: bag-of-words approaches, should perform worse in the unseen domain and unseen question scenarios because of their low level of abstraction. Actually, **the unseen domain scenario is expected to be the more challenging one**.

Concerning RTE based systems (Recognizing Textual Entailment), their objective is to recognize whether the meaning of a target statement can be inferred from another piece of text. At first, such systems might be strong enough to accurately perform the task. Conversely, some **lexicalized versions of RTE systems obtained substantially greater performance** [Zesch et al., 2013]. In other words, the addition of a set of generic features or knowledge resources contributed towards the improvement of performance while retaining the same robustness. However, **this improvement was more significant for the unseen answer scenario** rather than for the whole task scenarios. In fact, it is known that in the scenarios where the question or the domain changes lexicalized features do not have the same impact.

Participants have also try to innovate and improve their performance with new ideas, such as: modeling the semantics of a full natural language by projecting it onto a simpler English [Gleize and Grau, 2013], or grading student answers by acts of translation [Biçici and van Genabith, 2013]. In the last case, similar conclusions have been gathered concerning lexicalization, in fact, they argue that system configurations that at first did not use text similarity features, afterwards realized that the addition of lexical overlap baseline features slightly improved them. Participants have also tried to subsample the training data and divide the task into simpler resolution subtasks. For instance, [Aldabe et al., 2013] tried to use a question-type expert system.

2.7 Analysis of results

Participant results for the task are shown in the following tables and according to the following criterion: table 2.8 shows the micro F-score and the macro F-score for the best system configurations in the BEETLE corpus, table 2.9 shows the micro F-score and the macro F-score for the best system configurations in the SCIENTSBANK corpus, and finally, table 2.10 shows the mean of the micro F-score and the mean of the macro F-score for the best system configurations in the whole test set. Participant results are

shown together with baseline system performance. Baseline systems have already been described in section 2.3.

BEETLE	**Micro F**		**Macro F**	
	UA	UQ	UA	UQ
Majority baseline	0.229	0.248	0.114	0.118
Lexical baseline	0.483	0.463	0.424	0.414
CELI_run1	0.423	0.386	0.315	0.300
CNGL_run2	**0.547**	**0.469**	**0.431**	0.382
CoMeT_run1	**0.675**	0.445	**0.569**	0.300
EHU-ALM_run2	**0.566**	0.416 (run3)	**0.526**	0.370 (run3)
ETS_run1	**0.552**	**0.547**	**0.444**	**0.461**
ETS_run2	**0.705**	**0.614**	**0.619**	**0.552**
LIMSIILES_run1	**0.505**	0.424	0.327	0.280
SOFTCARDINALITY_run1	**0.558**	0.45	**0.455**	**0.436**
UKP-BIU_run1	0.448	0.269	0.423	0.285
Participants mean	**0.553**	0.446	**0.456**	0.374
Participants median	**0.552**	0.445	**0.444**	0.370

Table 2.8: Participant results in the BEETLE corpus. Higher performance than both baseline systems is showed in bold.

As we can observe in the BEETLE results table, **all of the participants performed better than the majority class baseline**. Actually, the participants mean exceeds it on average:

- 0.324[5] for the micro F-score on the UA scenario.
- 0.198 for the micro F-score on the UQ scenario.
- 0.342 for the macro F-score on the UA scenario.
- 0.256 for the macro F-score on the UQ scenario.

Notably, **the lexical baseline consisted of a harder system to beat** as the difference in performance scores 0.07, -0.017, 0.032 and -0.04 respectively. In fact, only one participant (ETS, in their first and second configuration) outperformed both baseline systems in all of the test scenarios in the BEETLE corpus.

[5]All values are calculated as follows: the mentioned micro F-score for the majority baseline is 0.229 and 0.553 for the mean of participants. The difference for those values is 0.553 - 0.229 = 0.324. Remember that micro F and macro F values are defined in the range [0,1].

As regards the unseen answer scenario, the top-3 performers are: ETS run2 (0.705/0.619), CoMeT run1 (0.675/0.569) and EHU-ALM run2 (0.566/ 0.526). With respect to the unseen question scenario, the top-3 performers are: ETS run2 (0.614/0.552), ETS run1 (0.547/0.461), and CNGL run2 (0.469/0.382). Results clearly indicate that the **macro F-score evaluation performance is lower than the micro F-score evaluation performance**[6].

SCIENTSBANK	**Micro F**			**Macro F**		
	UA	UQ	UD	UA	UQ	UD
Majority baseline	0.260	0.239	0.249	0.151	0.146	0.148
Lexical baseline	0.435	0.402	0.396	0.375	0.329	0.311
CELI_run1	0.372	0.389	0.367	0.278	0.286	0.269
CNGL_run2	0.266	0.297	0.294	0.252	0.262	0.239
CoMeT_run1	**0.598**	0.299	0.252	**0.551**	0.201	0.151
EHU-ALM_run2	**0.525(r3)**	**0.446**	**0.437**	**0.447(r3)**	**0.353**	**0.340**
ETS_run1	**0.535**	**0.487**	**0.447**	**0.467**	**0.372**	**0.334**
ETS_run2	**0.625**	0.356	**0.434**	**0.581**	0.274	**0.339**
LIMSIILES_run1	0.419	**0.456**	**0.422**	0.335	**0.361**	**0.337**
SOFTCARDINALITY_run1	**0.537**	**0.492**	**0.471**	**0.474**	**0.384**	**0.375**
UKP-BIU_run1	**0.590**	0.397(r2)	**0.407**	**0.560**	0.325(r2)	**0.348**
Participants mean	**0.496**	0.402	0.392	**0.438**	0.313	0.303
Participants median	**0.535**	0.397	**0.422**	**0.467**	0.325	**0.337**

Table 2.9: Participant results in the SCIENTSBANK corpus. Higher performance than both baseline systems is showed in bold.

The SCIENTSBANK results table shows that **all of the participants performed better than the majority class baseline**. The average participant performance exceeds the majority baseline:

- 0.236 for the micro F-score on the UA scenario.
- 0.163 for the micro F-score on the UQ scenario.
- 0.143 for the micro F-score on the UD scenario.
- 0.287 for the macro F-score on the UA scenario.
- 0.167 for the macro F-score on the UQ scenario.

[6]This result reflects that systems obtain higher accuracy on major classes.

- 0.155 for the macro F-score on the UD scenario

As expected, for the SCIENTSBANK corpus the **lexical baseline has also been more difficult to beat** as the difference in performance scores 0.061, 0, -0.004, 0.063, -0.016 and -0.008 respectively. In fact, only two participants (ETS on their first run, and SOFTCARDINALITY on their first run) outperformed both baseline systems in all of the test scenarios.

As regards the unseen answer scenario, the top-3 performers are: ETS run2 (0.625/0.581), CoMeT run1 (0.598/0.551) and UKP-BIU run1 (0.590/ 0.560). With respect to the unseen question scenario, the top-3 performers are: SOFTCARDINALITY run1 (0.492/0.384), ETS run1 (0.487/0.372), and LIMSIILES run1 (0.456/0.361). Finally, for the unseen domain scenario, the top-3 performers are: SOFTCARDINALITY run1 (0.471/0.375), ETS run1 (0.447/0.334) and EHU-ALM run2 (0.437/0.340). Results in this corpus also indicate that the **macro F-score evaluation performance is lower than the micro F-score evaluation performance**.

All test set	**Micro F mean**	**Macro F mean**
Majority baseline	0.245	0.129
Lexical baseline	0.436	0.333
CELI_run1	0.387	0.270
CNGL_run2	0.375	0.274
CoMeT_run1	**0.454**	0.312
EHU-ALM_run2	**0.471**	**0.382**
ETS_run1	**0.514**	**0.377**
ETS_run2	**0.547**	**0.428**
LIMSIILES_run1	**0.445**	0.308
SOFTCARDINALITY_run1	**0.502**	**0.389**
UKP-BIU_run1	0.418	**0.364**
Participants mean	**0.457**	**0.345**
Participants median	**0.454**	**0.367**

Table 2.10: Participant results summary on the BEETLE and on the SCIENTSBANK. Higher performance than both baseline systems is showed in bold.

As we can observe in table 2.10, **all of the participants performed significantly better than the majority class baseline**. On the contrary, only six systems outperformed the lexical baseline for the micro F-score; and only five systems for the macro F-score. Actually, **the lexical baseline has been difficult to beat** since the difference in performance between the participants mean and the lexical baseline mean is near 0.021 for the micro F-score, and near 0.012 for the macro F-score. The **top-3 performers** are the

following ones: ETS [Heilman and Madnani, 2013] (on their first and second run), SOFTCARDINALITY [Jimenez et al., 2013] (on their first run), and EHU-ALM [Aldabe et al., 2013] (on their second run).

2.8 Conclusions

Domain specific computer-based learning methods are able to assess understanding by determining when different text strings express similar concepts ([Hu et al., 2003], [Jordan et al., 2006] and [McCarthy et al., 2008]). However, these approaches **strongly depend on domain expertise and require intensive work** even for small domains. In this context, the SemEval-2013 task 7 aimed to bring together researchers in the educational domain to let them **develop systems that could operate unchanged across domains**. Such systems will only require question texts and reference answers to operate. As a consequence, the classification task at SemEval, is a challenging task that requires systems to use multiple natural language processing technology. To achieve this objective, participants either made use of text similarity metrics, adapted existing educational NLP methods, employed textual entailment engines, or built other kind of hybrid systems. Task results show that all of the systems significantly outperformed the majority class baseline. On the contrary, beating the lexical baseline was more challenging and non trivial. However, in most of the scenarios the **top performing systems significantly outperformed the lexical baseline**, sometimes by a large margin.

Regarding to corpus specific results, **performance on the SCIENTSBANK corpus has been lower than on the BEETLE corpus**. In fact, the participants average difference (tables 2.8 and 2.9) is around -10.3%[7] for the micro F-score on the unseen answer scenario, around -9.8% for the micro F-score in the unseen questions scenario, around -3.9% for the macro F-score in the unseen answer scenario, and around -16.3% for the macro F-score in the unseen question scenario. The **higher scores obtained in the BEETLE corpus could be a consequence of its simpler language**, due to the hypothesis that participants of the BEETLE study may have used a **simpler language** because they were aware of being talking with a tutorial dialogue system. In addition, **the BEETLE corpus contains more similar questions and answers** because it focus on a single science area. The wider set of topics covered by the SCIENTSBANK corpus and the average answer length been considerably higher (two or three sentences) are also

[7]The cited value and the following ones are calculated as follows: BEETLE values is 0.553, SCIENTSBANK value is 0.496, thus, 0.553 - 0.553*0.103 = 0.496.

probable reasons to clarify the achieved differences. As expected, the **performance on the unseen answer scenario was significantly higher** than on the unseen question scenario and on the unseen domain scenario, as shown in table 2.11. In addition, performance on the BEETLE unseen question scenario and on the SCIENTSBANK unseen question and unseen domain scenario was much more varied.

Performance	**Micro F**		**Macro F**	
	UQ vs UA	UD vs UA	UQ vs UA	UD vs UA
B lexical baseline	-4%		-2.3%	
SB lexical baseline	-7.6%	-8.9%	-12.2%	-17%
B participants	-19.3%		-17.9%	
SB participants	-18.9%	-20.9%	-28.5%	-30.8%

Table 2.11: Performance difference between the unseen answer scenario and the other scenarios. SB refers to the SCIENTSBANK corpus; and B refers to the BEETLE corpus.

Not only the results show that The Joint Student Response Analysis challenge has proven to be a **useful interdisciplinary task**; but also, participants have shown that **there is value in using different approaches of natural language processing** features to judge the validity of free answers from a realistic educational-domain dataset. For instance, "experimental results showed that recognizing textual entailment approaches can work meaningfully in these settings, despite the fact that the educational notion of 'expressed' and the RTE notion of 'entailed' are slightly different" [Dzikovska et al., 2013]. Nevertheless, there is no doubt that **systems need to make use of deeper semantic features to improve** their performance. Consequently, there is still a significant opportunity to improve student response assessment systems, which reflects the interesting challenges that such tasks present to the computational linguistics field.

Chapter 3

Analysis and Optimization of the EHU-ALM System

Contents

3.1 Description of the system

Section 2.1 enumerated some of the advantages and disadvantages of domain-specific models and the consequent improvement **flexible systems** offer in order to avoid the labor-intensive work of domain-specific resource gathering. Keeping on this premise the goal of the EHU-ALM system is to be **robust enough across domains and scenarios** so that no domain-specific data is required when shifting to new domains.

EHU-ALM [Aldabe et al., 2013] is a **supervised system** based on **syntactic and semantic similarity features** capable of classifying student answers into different categories. The system combines a variety of measures to compute text similarities and applies those measures between the question sentence, the reference answer(s) and the student answer to calculate features. The syntactic and semantic similarity features[1] can be grouped into the following sets:

1. Text overlap features (marked with cyan).
2. WordNet-based lexical features (marked with magenta).
3. Graph-based, Corpus-based and predicate-argument features (marked with green).

The system uses both training datasets indistinctly by mixing the instances from the BEETLE corpus and the SCIENTSBANK corpus[2]. In addition, the system is able to distinguish between seen and unseen questions in order to adequate to different test scenarios; as well as question types by means of simple heuristics. The configuration of the system is organized according to **three different configurations** and each configuration defines a framework to explore different alternatives to approach the answer scoring task:

1. Generic framework.

   Training examples are grouped so that each question and its answers are in the same fold. This configuration optimizes the unseen question scenario.

[1] See Appendix I for a brief summary of all the attributes used in the system.

[2] The SemEval task clearly distinguishes the BEETLE and the SCIENTSBANK scenarios, in contrast, developers of the EHU-ALM system used an approach in which they join both datasets in order to build a more flexible system. Throughout this work we will go on with this criteria.

2. Unseen framework.

   Training examples are grouped so that answers to the same question can occur in different folds. This configuration optimizes the unseen answer scenario.

3. Question-type expert framework.

   Training examples are grouped depending on the type of question (how, what and why). This configuration lets train question-type expert classifiers.

All of the system configurations use the same feature set and the same Support Vector Machine implementation [Chang and Lin, 2011] to train models. Models where built under Weka [Hall et al., 2009] using stratified cross-validation sets and a grid search technique was used to optimize the values of the classifier (regularization and gamma) to maximize the macro F evaluation measure (see equation 2.1). The usage of the macro F-measure favors to avoid bias towards majority classes when the distribution of classes is not balanced in the training data.

The results of the EHU-ALM run configurations are further described in [Dzikovska et al., 2013] and [Aldabe et al., 2013], but also summarized in table 2.10. Results showed that the **strategy is appropriate to build flexible systems** that perform competitively on all the test scenarios. All of the three distinct run configurations used under the EHU-ALM architecture ranked differently based on the evaluation scenario, but despite being below the best system all the runs scored **above the median and outperformed the average results obtained by participants**.

3.2 Beyond the flat system

Throughout developing and testing the baseline system only the LibSVM classifier has been used. However, there might be a different set of configurations that could help model the data better. In all, machine learning is about generalizing from examples and automatically learning patterns from data, but unfortunately, it is not trivial to find the specific strategy that would perform best in a certain dataset. Even a simpler classifier could perform more accurately than a more complex one. Actually, complex classifiers are quite attractive, but also more complicated to tune and harder to use and get good results. That is why **machine learning is not a one-shot task**, moreover, **the process of collecting valuable attributes** (attribute engineering) **can be much harder than the process of learning itself**. Consequently,

an effective machine learning task requires an iterative strategy where classifiers, data analysis, results analysis and continuous adaptation of strategies are involved.

3.2.1 Attribute engineering

Attribute engineering has been an **open research line** in many areas apart from machine learning [Liu and Yu, 2005], such as in: statistical pattern recognition, data mining, information extraction, optical character recognition ... Much of its success relies on the fact that **data preprocessing is essential to obtain good results**. In fact, attribute engineering techniques can significantly reduce the number of selected valuable attributes (removing irrelevant, redundant or noisy ones) and successfully **speed up and improve performance**.

As regards the flat system[3] we will start analyzing the contribution of each attribute performing an **ablation test** on the whole feature set. The primary objective of such a task is to **determine the contribution of each attribute** in order to obtain the most valuable attribute set. Figure 3.1 shows the results of the ablation test. The attribute name located in the left side of the figure refers to the attribute that has been removed, so that the graph reflects the difference obtained in the F-score between the baseline system with all of the features and the baseline system with all except that feature.

Despite the fact that difference in performance is quite similar (most of the differences are lower than 4e-3) results clearly show that removing one attribute at a time does not improve the system significantly, actually, it only improves performance slightly in three situations (question_lesk is removed, question_wn_lch is removed and question_wn_resnik is removed). Table 3.1 shows the resulting accuracy when cross validating the flat system and the system without the mentioned attributes[4].

System	Micro F-score	Micro Precision	Micro Recall
Flat system	0.571	0.568	0.577
Reduced system	0.569	0.566	0.576

Table 3.1: Micro F-score results between the flat system and the reduced system.

[3]The flat system is the system that makes use of all of the attributes. The raw EHU-ALM system with no modifications.

[4]See Appendix I for a brief summary of all the attributes used in the system.

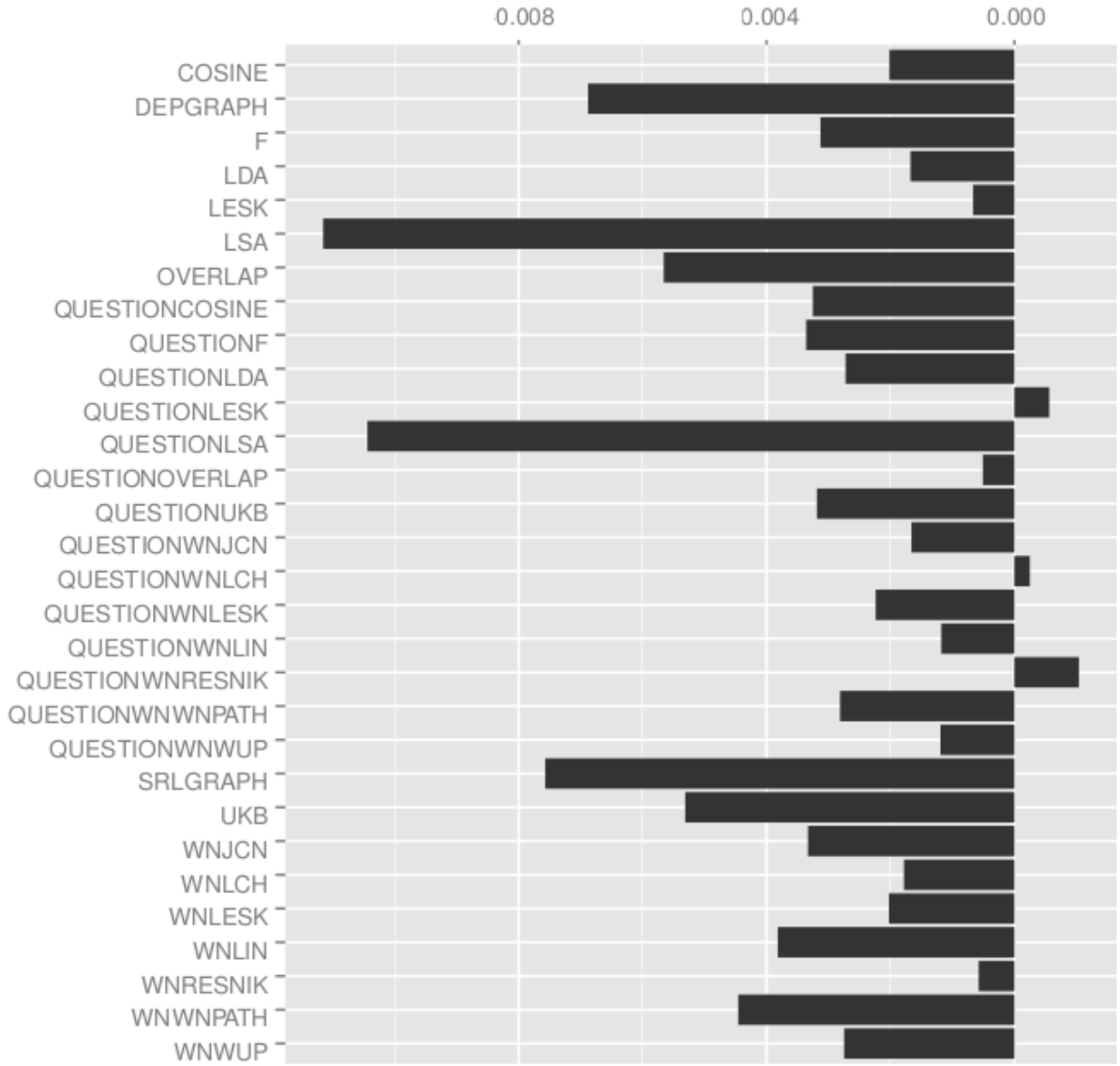

Figure 3.1: Ablation test results on the baseline system.

It seems that further analysis might be necessary to analyze the **correlation among features**. In fact, correlation analysis can result more complex than it could seem at first because features that are irrelevant in combination can gain relevance in isolation and vice-versa, another possibility is that there might be groups of inter-correlated features.

The **Pearson product-moment correlation coefficient** measures the linear dependence (correlation) between a pair of variables[5]. It outputs a value in the range [-1,+1] where a total positive correlation is quantified as +1, and a total negative correlation is quantified as -1. As expected, the value 0 indicates no correlation. The *Pearson product-moment correlation* is given by:

$$\rho(X,Y) = \frac{Cov(X,Y)}{\sigma_X \sigma_Y} \tag{3.1}$$

where Cov is the *covariance* and σ is the *standard deviation*. The objective of performing such experiment is to be able to **identify the best attributes among the candidate feature set**. Figures 3.2, 3.3 and 3.4 show the correlation analysis between all features organized in groups: figure 3.2 shows the correlation among text overlap features; figure 3.3 shows the correlation among WordNet-based features; and figure 3.4 shows the correlation among corpus-based, graph-based and dependency-based features. To interpret the figures we can take into account that points dispersed in the space with no sense might be indicative of absence of correlation, whereas graphics showing clearly lines of points might be indicative that correlation is present (positive or negative correlation may vary depending on the slope of the line).

[5]This measure is widely used in statistical sciences.

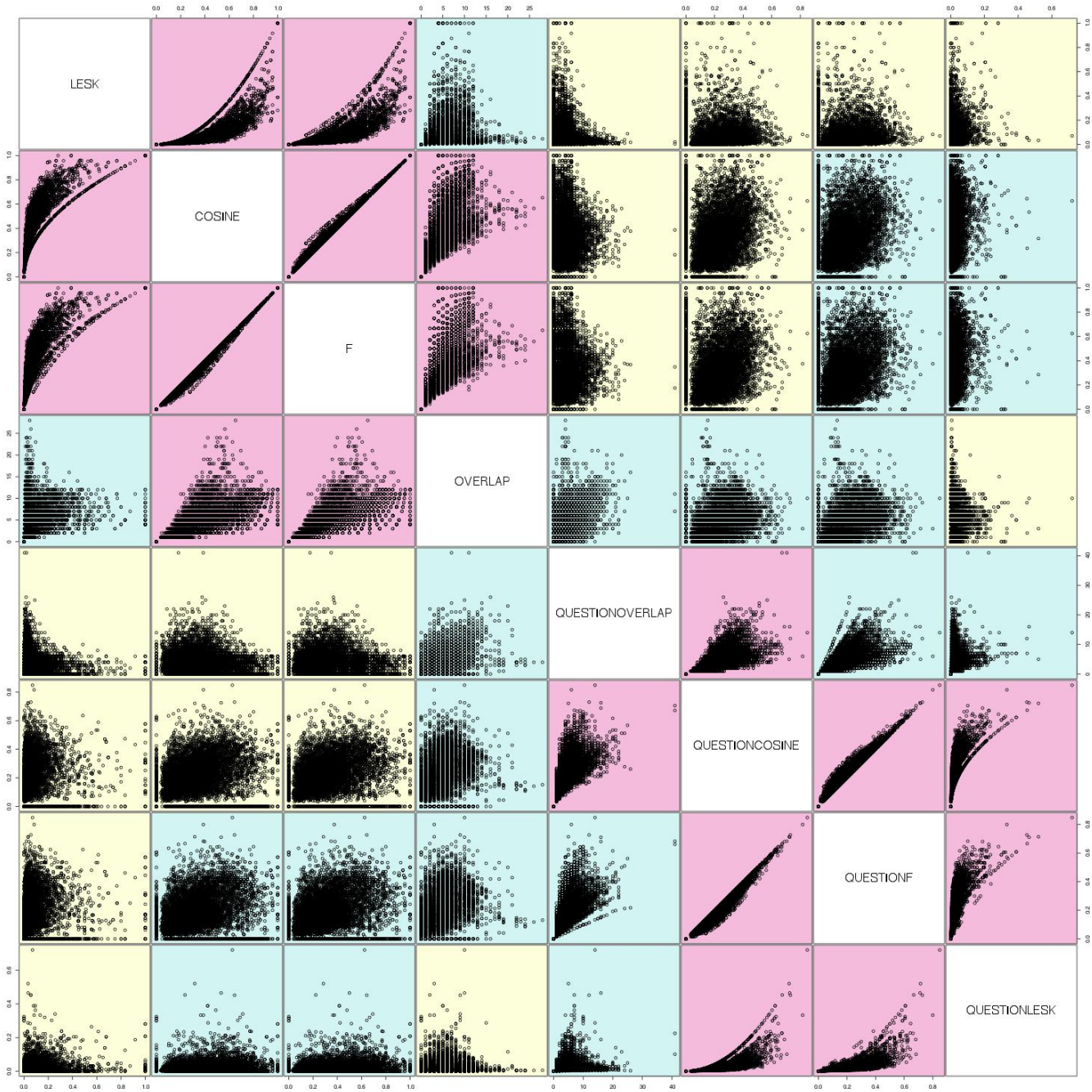

Figure 3.2: Correlation among text overlap features in the baseline system.

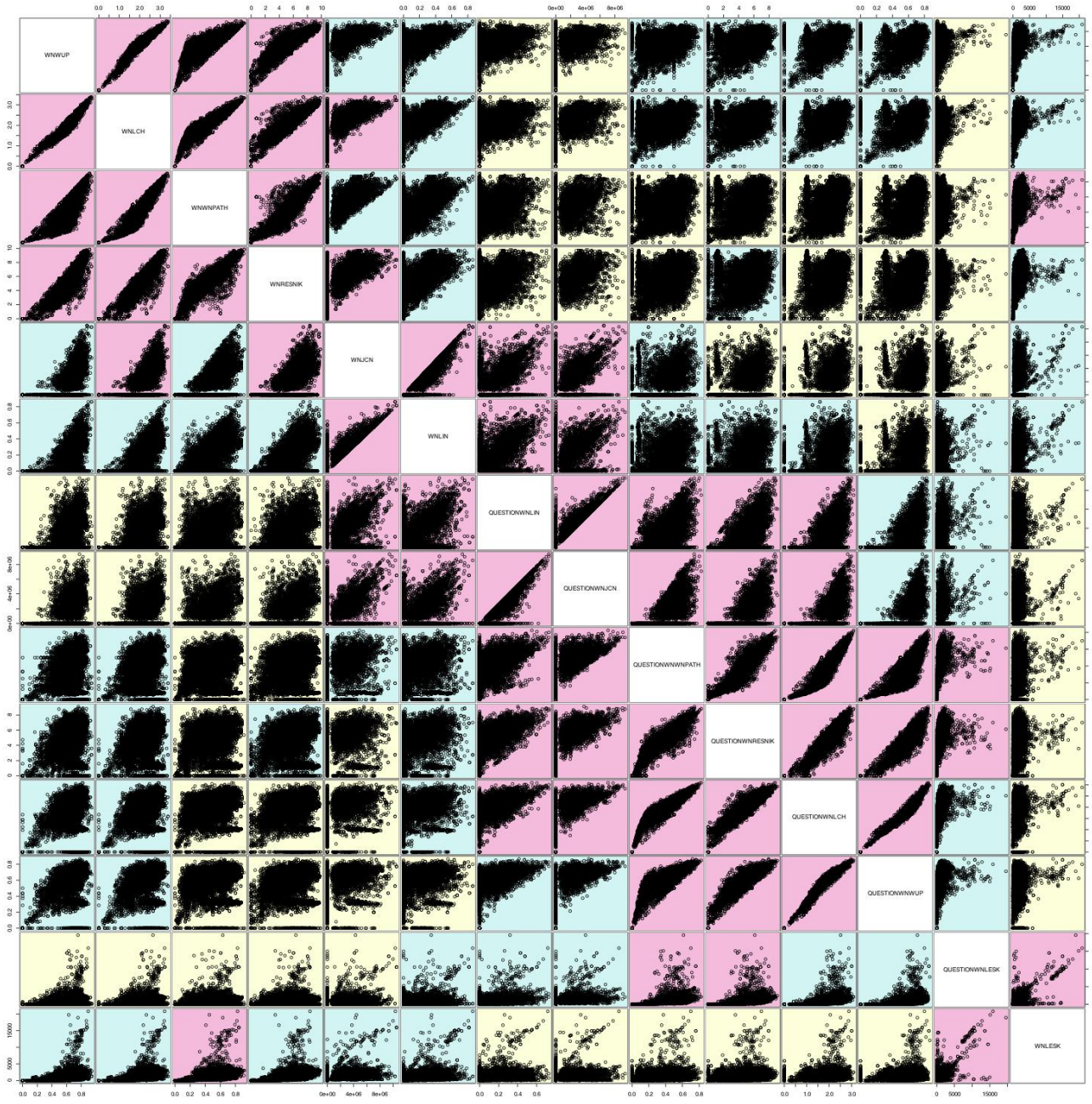

Figure 3.3: Correlation among WordNet-based features in the baseline system.

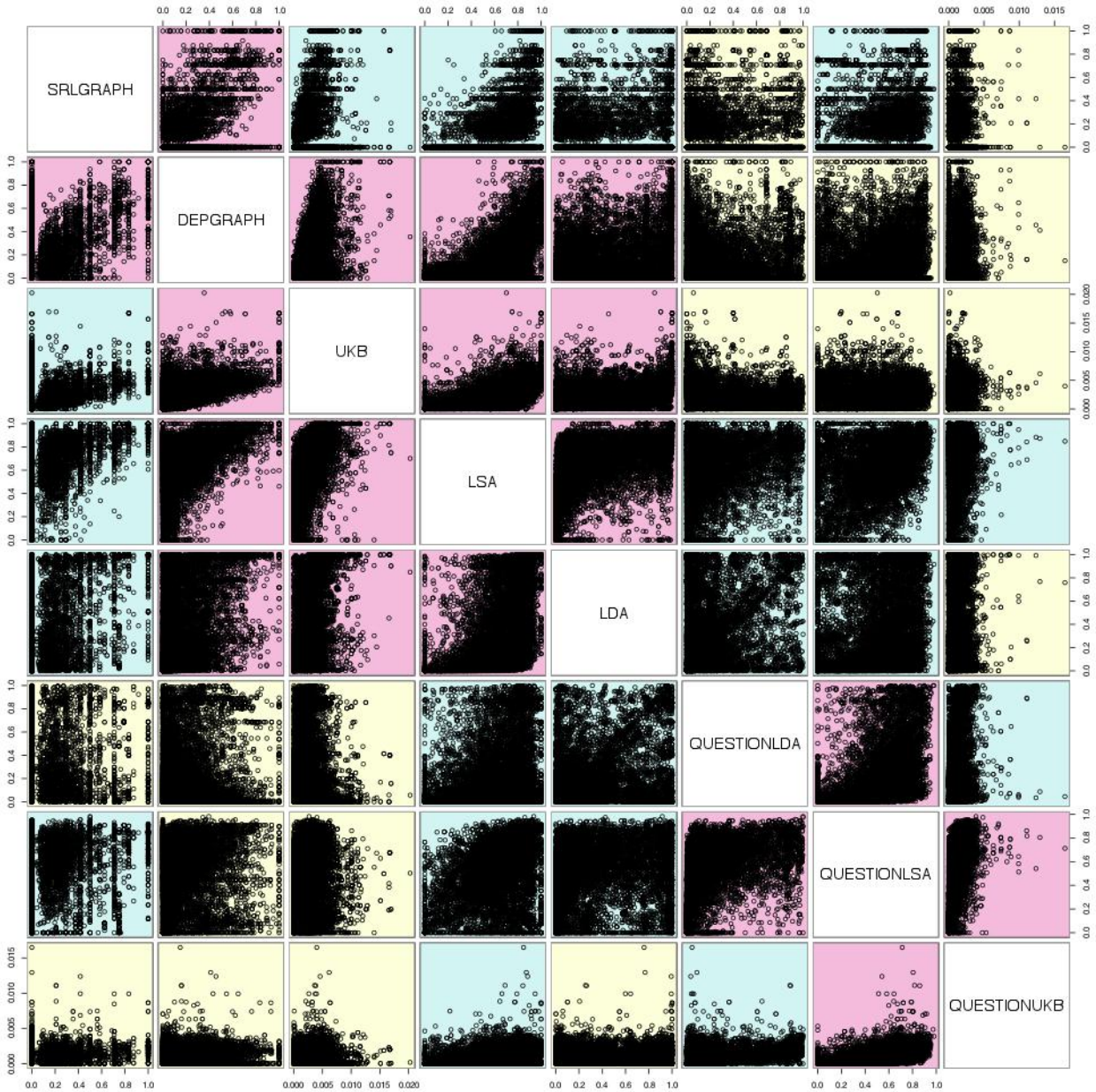

Figure 3.4: Correlation among corpus-based, graph-based and dependency-based features in the baseline system.

Results clearly show that **the correlation level between features is quite high**. This means we are providing the classifier with **repeated and probably non-useful or even noisy data**. Concerning each feature set

individually, it is noticeable that:

1. There is a high correlation among text overlap-based features.
2. There is a high correlation among WordNet-based features. Sometimes excessive.
3. Corpus-based, graph-based and dependency-based attributes are not too correlated[6].

Consequently, the high correlation observed between features makes it necessary to perform a **feature subset selection** step, in order to conveniently **select the most convenient features**.

3.2.2 Feature subset selection

Feature subset selection is a process in which **relevant features** are selected from the original subset to construct a new model. The main objective of feature subset selection techniques is to **get rid of redundant or irrelevant features**. Redundant or irrelevant features are the ones that do no provide more information than the existing one or provide no useful information at all. To find the optimal feature subset could be a very time-consuming task as the **search space is usually intractable** when the feature number is high. Additionally, the availability of a high number of feature selection algorithms makes it difficult to select the most suitable one. Among the advantages feature selection algorithms can provide, the most important ones are the following:

1. As the feature number decreases, the model gains **interpretability**.
2. As the feature number decreases, the model needs a **shorter training time**.
3. Selecting valuable attributes may help **reduce overfitting** (Simplification of the model).

Overfitting is a common word in the machine learning area, but unexpectedly, it can come in different forms. The most effective way to analyze overfitting is to decompose it into two groups: **bias and variance**. High bias symptoms (underfitting problem) are related to the problem of **learning wrong things**. This could happen, for instance, if a classifier is not able

[6]It was expected because the attributes are more diverse and come from completely distinct sources.

to produce a valid output because the feature set used to model the problem is not adequate. High variance symptoms (overfitting problem) appear when the classifier has **learnt too much** and, as a consequence, it is **not able to generalize efficiently** at test time. There are many methods to combat overfitting, such as: adding a regularization parameter or using cross validation sets to train model parameters; but using feature subset selection techniques is also valid. In summation, there is no doubt that one of the most important factor in machine learning is the features used: with the optimal feature set learning is easy, even if simple models are used.

A **feature subset selection algorithm** can typically be seen as a combination of two components: on the one hand, **a search technique** for selecting new candidate feature subsets, and, on the other hand, an **evaluation criterion** to score the optimality of the selected subsets. The basic steps of feature selection algorithms are further described in [Liu and Yu, 2005].

Concerning the EHU-ALM baseline system, we are going to apply different feature subset selection models to find optimal feature subsets:

1. Filter-based models.

2. Wrapper-based models.

Filter-based models make use of general characteristics of the data to determine the rank or relevance of features, for instance, filter-based models can be based on: mutual information, Pearson product-moment correlation coefficient, inter/intra class distance, ... Filter-based models are normally less computationally intensive than wrappers-based ones, but, on the contrary, they output a feature ranking that is not tuned to a specific classifier. **Wrapper-based models** use the attached classifier's evaluation criterion to evaluate performance. As a consequence, not only the search is much more suited to the task, but also, it turns to be more expensive computationally.

Figure 3.5 shows the results of applying a filter-based method on the baseline system. The search technique is based on Weka's Cfs RankSeach. This method evaluates "the worth of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them". To evaluate the real optimality of different subsets we used the SVM classifier and **recursively removed the worst scored attribute** at each iteration. Following the same strategy, but based on a different method, figure 3.6 shows the results of applying the Weka's Infogain Ranker. This time, the method "evaluates the worth of an attribute by measuring the gain ratio with respect to the class".

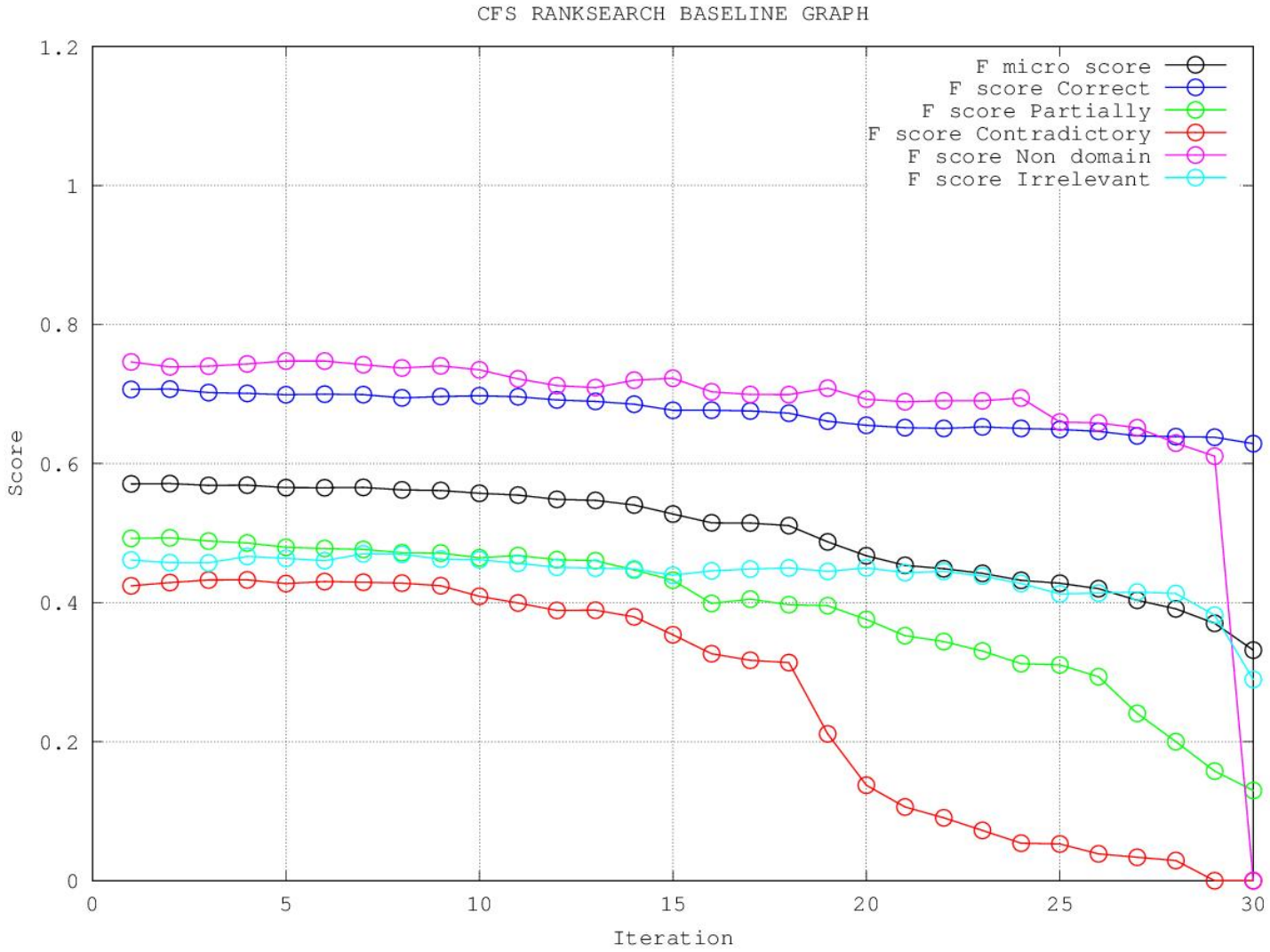

Figure 3.5: Results of the filter-based Cfs Ranksearch method. The worst scored attribute is recursively removed at each iteration.

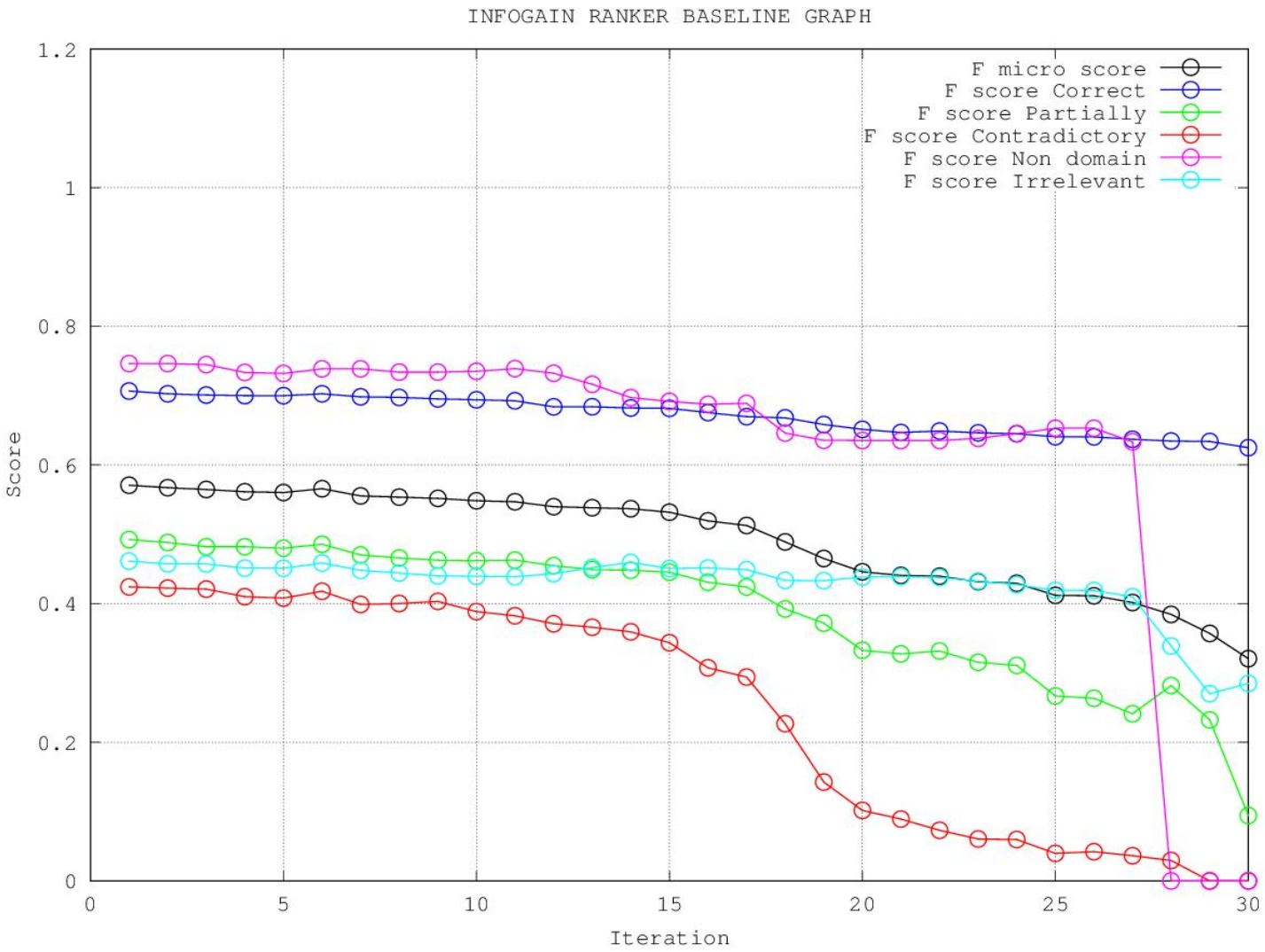

Figure 3.6: Results of the filter-based Infogain Ranker. The worst scored attribute is recursively removed at each iteration.

Both figures clearly indicate that the system obtains the **best results on the correct and on the non-domain classes**. In contrast, the performance on the **contradictory class is significantly lower**. Concerning the feature subset selection method, it is necessary to remark that in both runs no attribute subset has been able to beat the flat system[7]. However, it is surprising that the system almost maintains the performance[8] while keeping on removing attributes. It seems that some attributes **don't provide the system too relevant information** to model the task. On the contrary, it is noticeable how important some attributes are for some classes, as the F-score of the class drops suddenly when removing them. For instance, iteration 29 on figure 3.5 shows a huge drop for the non in the domain class' F-score.

In most cases filter-based models are good choices to identify redundancy, irrelevant or noisy data, because, they are unbiased and fast to compute. Nevertheless, the choice of the evaluation metric used heavily influences the results obtained. To get an alternative diagnostic, we have also used a wrapper-based model. As we have previously described, **wrapper-based methods** train distinct models for each feature set, use the accuracy of the classifier to evaluate the subset, and, as a consequence, are more computationally intensive but usually provide better results.

The wrapper method used is a combination of the SVM classifier (the optimality scorer) and a genetic algorithm (the search technique). **Genetic algorithms** are based on the genetic evolutive process of living organisms and are widely used to solve optimization and search problems. The key idea is that genetic algorithms simulate the principles of natural selection and survival of the strongest theory [Darwin, 1968]. By imitation of the process, genetic algorithms are able to **create solutions that iteratively evolve** towards optimal solutions. Results obtained with the wrapper method are shown in table 3.2.

System	Micro F-score	Micro Precision	Micro Recall	Removed Atts
Flat system	0.571	0.568	0.577	-
Wrapper-based model	0.57	0.568	0.577	1,2,5,11,16,20

Table 3.2: Wrapper-based feature subset selection results using genetic algorithms.

The accuracy of the system remains unbeaten, there is no doubt that results are almost identical. However, the optimal feature subset is not the same in both systems. Whereas the flat system makes use of all features, the

[7]Notice that slope of polynomials always go down.
[8]Slope is close to 0.

wrapper-based model has discarded five of them[9].

3.2.3 Selecting the optimal feature subset

Up to now we know which one the best feature subset is according to the results of the wrapper-based method. On the contrary, for the filter-based methods we only know the results obtained for each iteration, but not the optimal iteration.[10] Thus, we somehow need a method to **score and compare each iteration** and select the optimal one. The optimal iteration will show us which one is the optimal feature subset. For this task, we propose an iteration scorer (see equation 3.2) that evaluates the worth of an iteration according to **three distinct criterion**:

1. Removed attributes

   One of the main objectives of performing FSS is to **reduce the problem of correlation** observed among attributes by removing them, but also to build a simpler model that could **generalize more efficiently**. That is why we would positively score the number of attributes that have been removed from the model.

2. Derivative Slope

   The derivative measures the **sensitivity to change**. If we fit a polynomial to the micro F-Score points obtained out of each iteration (figures 3.5 and 3.6) we can analyze the slope of the derivative to obtain interesting information out of that data. We will score the function according to the slope of the first derivative. Consequently, increasing slope is favored versus decreasing slope[11].

3. Difference with respect to best value

   We can't forget our main goal, to **optimize the F-Score.** If the F-Score obtained in an iteration is far away from the best score among all iterations, it must be penalized.

[9] Attribute number 1 is the ID and it is not considered an attribute. See Appendix I for a brief summary of all the attributes used in the system.

[10] Notice that as mentioned previously each iteration removes the worst scored attribute recursively, but the iteration number has nothing to do with the number of the attribute that actually has been removed.

[11] The more positive/negative the slope is, the more it is favored/penalized.

Mixing up all the factors, the resulting equation we propose reads as follows:

$$Score = \frac{Removed\_Attributes * Slope\_of\_Derivative}{F\_Difference} \tag{3.2}$$

$$Removed\_Attributes = log(ar) \tag{3.3}$$

where *ar* is the number of attributes removed in the iteration under analysis.

$$Slope\_of\_Derivative = sigmoid(p'') \tag{3.4}$$

where *p"* is the value of the slope of the first derivative and *sigmoid* is the sigmoid function[12]. Notice that increasing slopes would score values in the range (0.5,1], decreasing slopes would score values in the range [0,0.5) and absence of slope would score 0.5.

$$F\_Difference = e^{best\_f - actual\_f} \tag{3.5}$$

where *best_f* is the best F-Score among all iterations and *actual_f* is the score of the iteration under analysis. Notice that a zero difference would score 1 (no penalization) whereas the value goes increasing exponentially as the difference increases.

To try alternative scoring factors we have used different penalizing parameters[13] in equations 3.4 and 3.5 when calculating the results. The penalizing parameter must be chosen cautiously because the **penalization grows up exponentially**. Results are summarized in table 3.3 and from the results of both systems we conclude that **the more the penalization factor is increased the more conservative results are obtained**. Notice that at each iteration we were recursively removing one more attribute, consequently a more conservative result might **discard less attributes in favor of a more complex model**. Taking into account that the best micro F-Score obtained for the flat system is 0.57 we can conclude that with a high penalization parameter the scorer tends to select the zone that we would intuitively choose from figures 3.5 and 3.6. Actually, the zone in which removing a nice quantity of attributes (> 12) the micro F-Score loose is minimal

[12]The sigmoid function is only used to limit the return range to a [0,1] interval.

[13]As the penalizing factor increases the penalizing effect increases.

FSS filter	Penalizing factor	Best scored iteration	Micro F-Score
Cfs Ranksearch method	1	26	0.42
	3	18	0.51
	5	13	0.55
	7	13	0.55
	9	13	0.55
FSS filter	**Penalizing factor**	**Best scored iteration**	**Micro F-Score**
Infogain Ranker	1	27	0.40
	3	15	0.53
	5	15	0.53
	7	14	0.54
	9	14	0.54

Table 3.3: Scoring filter-based iterations in the baseline system.

(< 0.03). Finally, if we **visually join the results** of the three FSS systems and use a **voting majority method** to make the final decision we obtain the following:

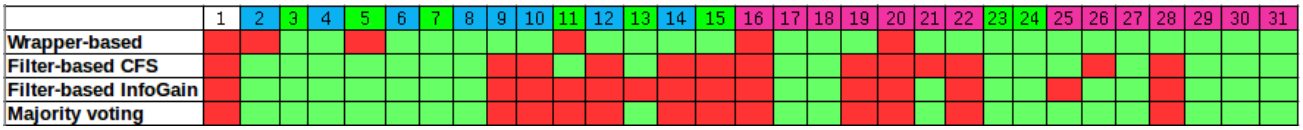

Figure 3.7: Making the feature subset final decision using a majority voting system.

Red square means discard the attribute and green square means select the attribute. Accordingly, we will remove the following attributes from the flat system: 1,9,10,11,12,14,15,16,19,20,22 and 28: the ID attribute, 4 text overlap attributes, 5 WordNet based attributes and 2 graph and corpus based attributes. The majority of the attributes removed belong to the WordNet-based lexical similarity group, actually, this result is not surprising because as we analyzed in figure 3.3 the correlation among WordNet-based attributes was high. In addition, indistinctly of the attribute grouping, most of the attributes removed are the ones calculated between the student answer and the question. At first, it seems that **features calculated among student answers and reference answers are more relevant** than those calculated among student answers and question sentences. Consequently, we will use the selected feature subset for upcoming experiments as regards the EHU-ALM system.

3.2.4 Testing the optimal feature subset. The learning curve

A **learning curve is a graphical representation of the performance** of certain model under some scenario. Learning curves are quite interesting to plot because apart from been **useful to test models** they are useful to perform **sanity checks**[14]. Actually, out of learning curve plots it is possible to interpret whether the model is working correctly or not. For instance, it is possible to diagnose if the algorithm is suffering from bias or variance problems. After such an analysis has been made, most of the times it is easier to treat the problem. Treating machine learning problems is not trivial because each kind of problem defines a **different scenario in which certain solutions are valid or not**. For example, under high bias scenarios getting more training data is not likely to help whereas under high variance scenarios probably will. Another possible example is that under high bias scenarios making a more complex model will likely help fit more closely to data whereas under high variance scenarios it would worsen the problem. To plot learning curves we will use **three distinct sources**:

1. Train set Error Rate (TrER).
2. 5-fold Cross Validation Error Rate (CV5ER).
3. Test set Error Rate (TeER).

The plot itself is a function that **depends on the training set size** used to train and test each model. In other words, we will deliberately use distinct sizes of data at each iteration so that we will start with a relatively small training and testing set and we will **recursively increase its size**. Notice that the same percentage from the training set and the testing set is used for each iteration. For instance, we will use the 10% of the total available in the first iteration, the 20% of the total available in the second, and so on ... The underlying idea is that theoretically for a good performing classifier the following must happen:

1. When the training data is reduced the model is able to fit to it very closely, that is why the training error rate might be small. But, as the size of the training data grows it becomes harder and harder to fit to it that good, and consequently what we find is that the training error increases.

[14] A sanity check consists on identifying possible performance errors on classifiers.

2. When the testing data is reduced (that is also indicative that a reduced training set was used to train the model) the model is not going to generalize well and the error rate would be high. But, as the size of the dataset increases the error rate found in testing might decrease because the model is able to generalize better.

On the overall, if there are not variance or bias problems the more data available the model has the better it generalizes. Under this hypothetical scenario, we could be able to see how in the x axis limit as the training error increases and the testing error decreases **both errors tend to join**.

If bias problem exists the learning curve plot might appear significantly different. Actually, no matter what the size of the used training set is we would pretty much be suffering the same underfitting problem in every iteration. As a consequence, as the training size increases we would not notice any change on training error; in addition, it would be very high even in last iterations of the learning curve, just as high as the test error.

At the setting of a model under high variance the model would be able to fit well to training data, and even for large training sizes the model would weirdly be able to fit well to data, due to overfitting. As a consequence, the training error would be low, but, in contrast, there would be a large gap between the training error and the test error because the model would not be able to fit to test data that well.

Figure 3.8 shows the learning curve plot that corresponds to the flat system, and figure 3.9 shows the learning curve plot that corresponds to the reduced baseline system after removing the attributes mentioned in section 3.2.3.

Results show an **intuitive behavior**. At first, we can observe that figure 3.9 (reduced system) seems to be more compact than figure 3.8 (flat system). In a way, we could say that the tendency of the error curves of the reduced system are more likely to join (at least join in a closer iteration). The reason for this behavior is a clear result of been using a **simpler system**. That is, as we are using a simpler system we have let the training and cross validating error increase a little bit (the model does no longer fit that well to training instances), but, interestingly, **the simplification of the model has resulted in a lower error rate on the test set**.

The following tables summarizes the last iteration results obtained for both models. Table 3.4 summarizes the error rate related measures (the ones used to plot the last iteration of figures 3.8 and 3.9) and table 3.5 summarizes the evaluation summary metric related measures.

Results show that the hypothesis we built when designing the feature subset selection experiments have been successful: by making use of a subset

System	TrER	CV5ER	TeER
Flat system	0.15	0.42	0.58
Reduced system	0.30	0.44	0.54

Table 3.4: Error rate related results obtained in the last iteration of the learning curve for both systems.

System	Test Micro F	Test Micro Pr	Test Micro Rec
Flat system	0.41	0.41	0.42
Reduced system	0.44	0.43	0.45

Table 3.5: Evaluation summary metric related results obtained in the last iteration of the learning curve for both systems.

of the original features we have increased the generalization capacity of the model. Concerning the sanity check of both models, we can mention that whereas one curve is more dispersed than the other one (as a result it may need more iterations, that is, more data) **both of them seem to join in a future iteration. This result is indicative that neither model is not under high bias nor high variance problem.**

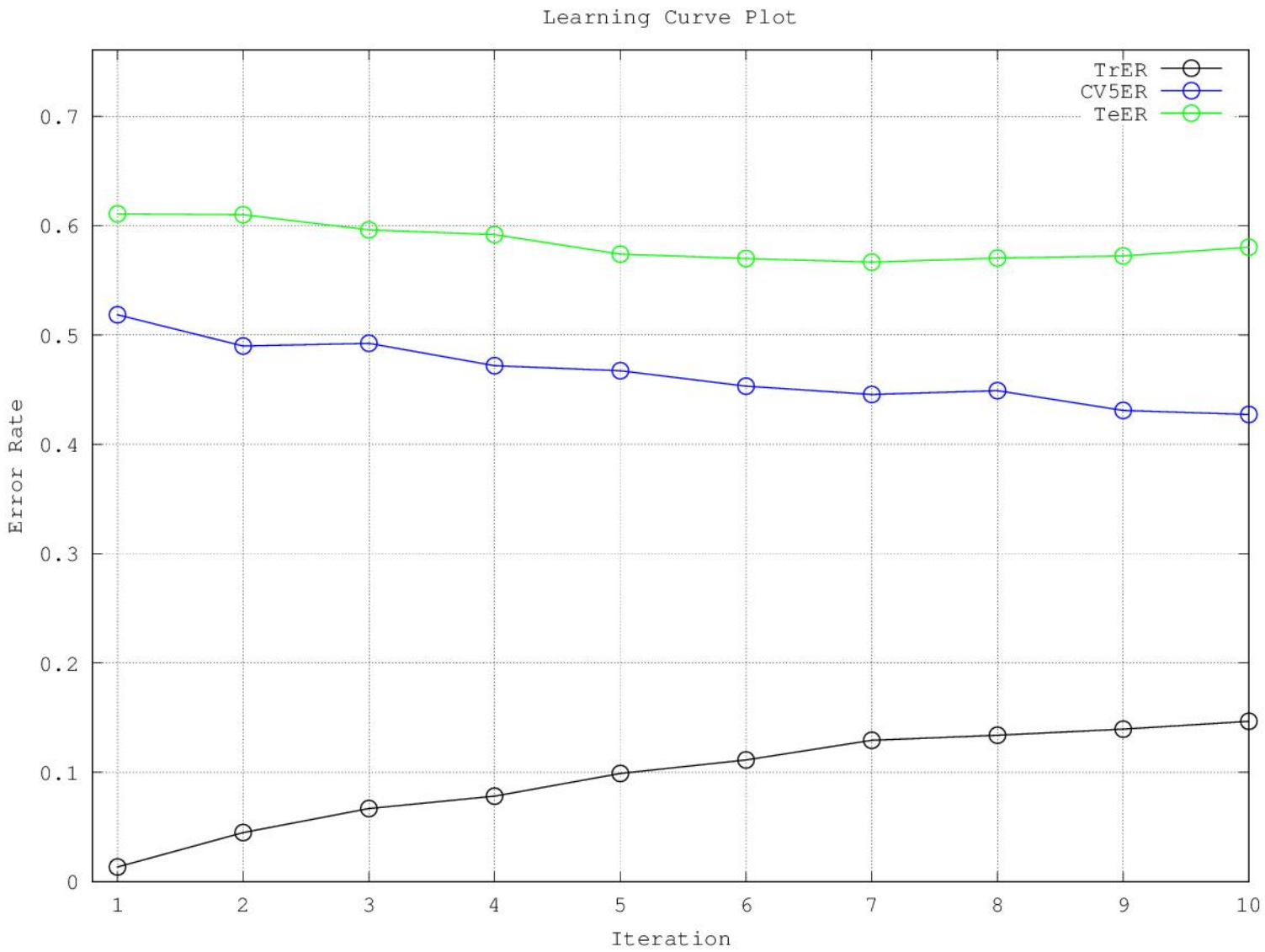

Figure 3.8: Learning curve plot of the flat system.

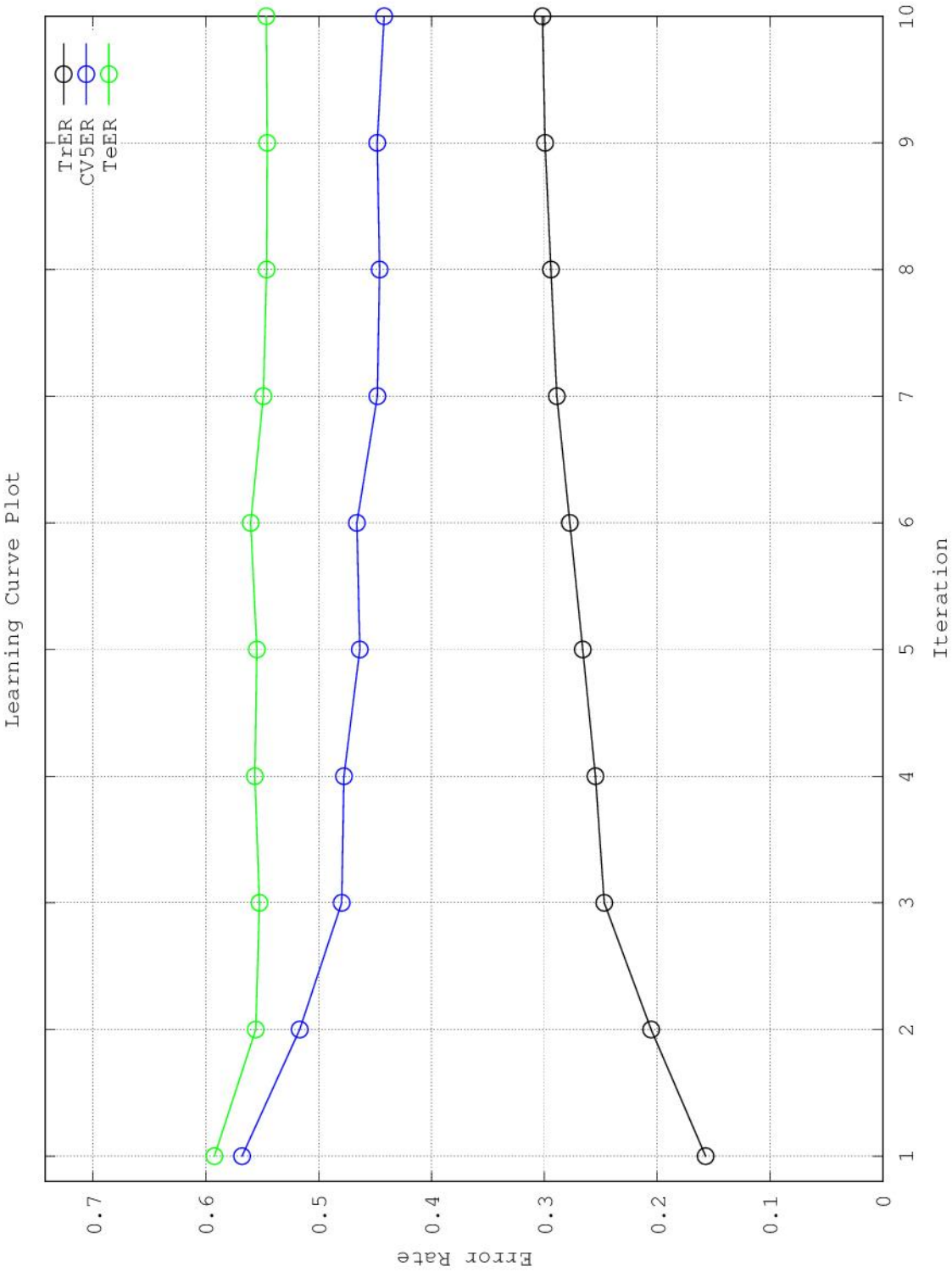

Figure 3.9: Learning curve plot of the reduced system.

3.3 Final remarks

Throughout this chapter we have described and **analyzed the EHU-ALM system**, as well as the relation among the features it uses. At first, we performed an attribute engineering step in which we measured the **contribution of each attribute** towards the system. As a result of the ablation test, we analyzed the **Pearson product-moment correlation coefficient** and we concluded that the correlation among features was excessively high. Thus, the high correlation motivated us to continue working in the attribute analysis and **get rid of redundant or irrelevant** ones. For the task we used distinct feature subset selection techniques and developed an iteration scorer to evaluate the worth of selecting or discarding each feature subset. After a majority voting the feature subset converged to a **final consensus** in which we selected the final attributes that were going to be removed from the flat system (1,9,10,11,12,14,15,16,19,20,22 and 28). That is, the ID attribute, 4 text overlap attributes, 5 WordNet based attributes and 2 graph and corpus based attributes. Concerning features we also concluded that features calculated between student answers and reference answers were more relevant than features calculated between student answers and questions. Finally, we used **learning curves** to make graphical representations and visually compare the flat and the reduced system and we observed that **the simplification of the model resulted in a lower error rate on the test set**, even the results at training where slightly worse. It is quite surprising the effort attribute engineering requires compared to the time spent doing machine learning, but, nevertheless, **attribute engineering is critical to the machine learning task**.

3.4 Future Work

In software development a project is defined as an effort that is prolongated between concrete dates. Although research projects are different with respect to software development projects[15], they booth need to terminate and as a consequence, several times there have not been **sufficient time to perform all experiments** (or even we think of great experiments just at the end of a project). That is why it is important to define the future work.

Related to the EHU-ALM system, the following experiments and ideas are the ones left as future work: to analyze if performing a feature subset selection technique but considering **groups of attributes** the system is

[15]Research projects tend to be not so defined from the beginning or may suffer more variations through their life-cycle.

able to improve (for example removing an undefined number of attributes from each group), to further analyze the **correlation** among groups and how groups affect the system (for instance we could perform an ablation test removing complete groups), to test different kind of **classifiers** and compare the results obtained (for instance to understand why an instance is classified as pertaining to some class, for this task decision trees are more understandable than other type of classifiers), to use **alternative feature subset selection** techniques, to test the system individually in the BEETLE and in the SCIENTSBANK datasets (at the same time we could analyze the **linguistic difficulty level** of each dataset), to **implement new features** of top performing systems of the SemEval task that could possibly contribute to the system, such as:

- General features ([Heilman and Madnani, 2013]).

  Intercept Term: This feature saves the classifier from having to use other features to map the class distribution of the dataset.

- N-gram based features ([Heilman and Madnani, 2013] and [Okoye et al., 2013]).

  Word-gram and N-gram coverage of the reference answer by the student answer.

- Language-model based features ([Biçici and van Genabith, 2013]).

  Perplexity: This feature measures the fluency of the sentences according to language models.

- Textual-entailment features ([Kouylekov, 2013] and [Zesch et al., 2013]).

  A text entails the hypothesis if the meaning of the hypothesis can be derived from the meaning of the text. The EOP framework is a good choice for this job[16].

- Edit-based features ([Heilman and Madnani, 2013] and [Biçici and van Genabith, 2013]).

  Features such as BLEU or TER that are able to measure the distance between texts.

- Negation features ([Gleize and Grau, 2013]).

  Features that are able to explicitly handle negation, such as the polarity of sentences. Some approximations have been computed using the Stanford parser negation marks.

[16]From http://hltfbk.github.io/Excitement-Open-Platform/ "The EXCITEMENT Open Platform (EOP) is an open source software platform containing state-of-the-art algorithms for recognizing texual entailment relations."

Chapter 4

Hierarchical Approach for the EHU-ALM System

Contents

4.1 Motivation

In computer science the divide and conquer paradigm is widely used, such as for the design of recursive algorithms. This technique works by **breaking down a complex problem into many subproblems** of the same type but known to be less complex or even simple enough to be solved directly. Keeping with this idea emerges the following uncertainty: is there any task-decomposition which lets break down the main task and **simplify the answer scoring problem?** From table 2.10 we know that the baseline system performs fine, and indeed, it scored among the three best systems of the SemEval-2013 task 7. However, the breaking down and simplification of the problem may help optimize the accuracy obtained.

4.2 The fundamentals

To test our intuition we first need to clear the following unknown: how could the main task be simplified? Our hypothesis is that if we were able to somehow divide the classification task into smaller classification tasks the process would be easier. In fact, we think that the perspective of addressing the problem as a **meta-decision tree** can be appropriate to deal with it. In this scenario, instead of dealing with a 5-way classification task at once, **each node of the tree would be a SVM expert classifier** responsible for making a **binary classification**. For example: the first level in the tree would be responsible for classifying and instance as "non in the domain" or "in the domain" and so forth. According to this result we would recursively decide whether to **go down through the tree or go to the leaf and return the class** value. To go on with idea it is critical to specify how to combine the classes so that they follow the described hierarchy.

Hierarchical clustering is a widely known technique used for cluster analysis. The aim of this technique is to build a hierarchy of clusters according to some similarity measure and a linkage function. **The linkage function** is responsible for merging clusters recursively according to some distance-based method. Thus, if we consider the training data classes as clusters the result of the hierarchical clustering (the **dendrogram**) would tell as how to set up the levels of the meta-decision tree. Nevertheless, to successfully apply any hierarchical clustering method, we first need to calculate the cluster distance matrix[1].

The **cluster distance matrix** (Ω) is a square and symmetric matrix in which each element at position $\Omega(i, j)$ determines the distance between

[1] In our case the distances between classes.

classes i and j. This information can approximately be gathered out of the **contingency table** of the baseline system. But we must proceed cautiously, because the contingency table does not provide that information directly. The contingency table provides the relation between the classifier's prediction and the gold standard accuracy; and in any way it provides the distance between classes. That is why we **can not directly apply the hierarchical clustering to the contingency table**, instead, we need to compute some kind of similarity out of it. For such a task we need to take each class pair and compute the relative frequency between the system's prediction and the gold standard, from both directions. That is, having two classes i and j we need to compute two things:

1. Out of all instances that really pertain to class i, how many times the classifier predicts them as pertaining to class j.

2. Out of all instances that really pertain to class j, how many times the classifier predicts them as pertaining to class i.

In other words, we need to compute the average relative frequency among the symmetric positions in the contingency table. This result is **not a valid mathematical metric** and it does not reflect pure similarity, but it somehow reflects the key idea we are looking for: if two classes are similar that means that the classifier makes several mistakes when classifying instances from and to those classes. **The higher the similarity is the higher the confusion of the classifier**. The following table (table 4.1) shows the interclass[2] and intraclass distance[3], where the distance is computed as:

$$Distance = 1 - Similarity \tag{4.1}$$

[2]The interclass distance is the distance computed between classes.
[3]The intraclass distance is the distance computed inside a class.

(1)	(2)	(3)	(4)	(5)	
0.32279	0.69112	0.78112	0.75395	0.99630	(1) = Correct
	0.67715	0.82863	0.85619	0.97261	(2) = Partially correct incomplete
		0.62311	0.87096	0.93081	(3) = Irrelevant
			0.87919	0.99161	(4) = Contradictory
				0.15116	(5) = Non domain

Table 4.1: Distance values computed out of the contingency table. Intraclass distances are the ones in bold

From table 4.1 we can observe that **as expected intraclass distances are mostly lower than interclass distances**. Apart from that, it is quite noticeable the high distances obtained among the non domain and all other classes. In addition, the **perplexity of the contradictory class** is quite high, indeed, the intraclass distance is sometimes higher than the interclass distance; this result might be indicative that the model has big confusion as regards this class.

Once obtained the distance matrix it is trivial to apply an agglomerative hierarchical clustering method. The resulting dendogram will show how clusters have been **merged following a bottom up strategy**. Figure 4.1 describes itself how the most similar classes are the correct and partially correct incomplete ones. The resulting cluster is then merged with the contradictory class, and, after that, it combines with the irrelevant class. Finally, the non in the domain class is joined to the cluster which terminates the linkage process.

Up to this point, we can clearly define **two different configurations** to break down the main task into smaller nuggets.

1. Start the binary classification task **from the most distanced classes** to the most similar ones (Hierarchy configuration one).

2. Start the binary classification task **from the most similar classes** to the most distanced ones (Hierarchy configuration two).

Figure 4.2 shows the graphical flow-chart of the described hierarchy configurations.

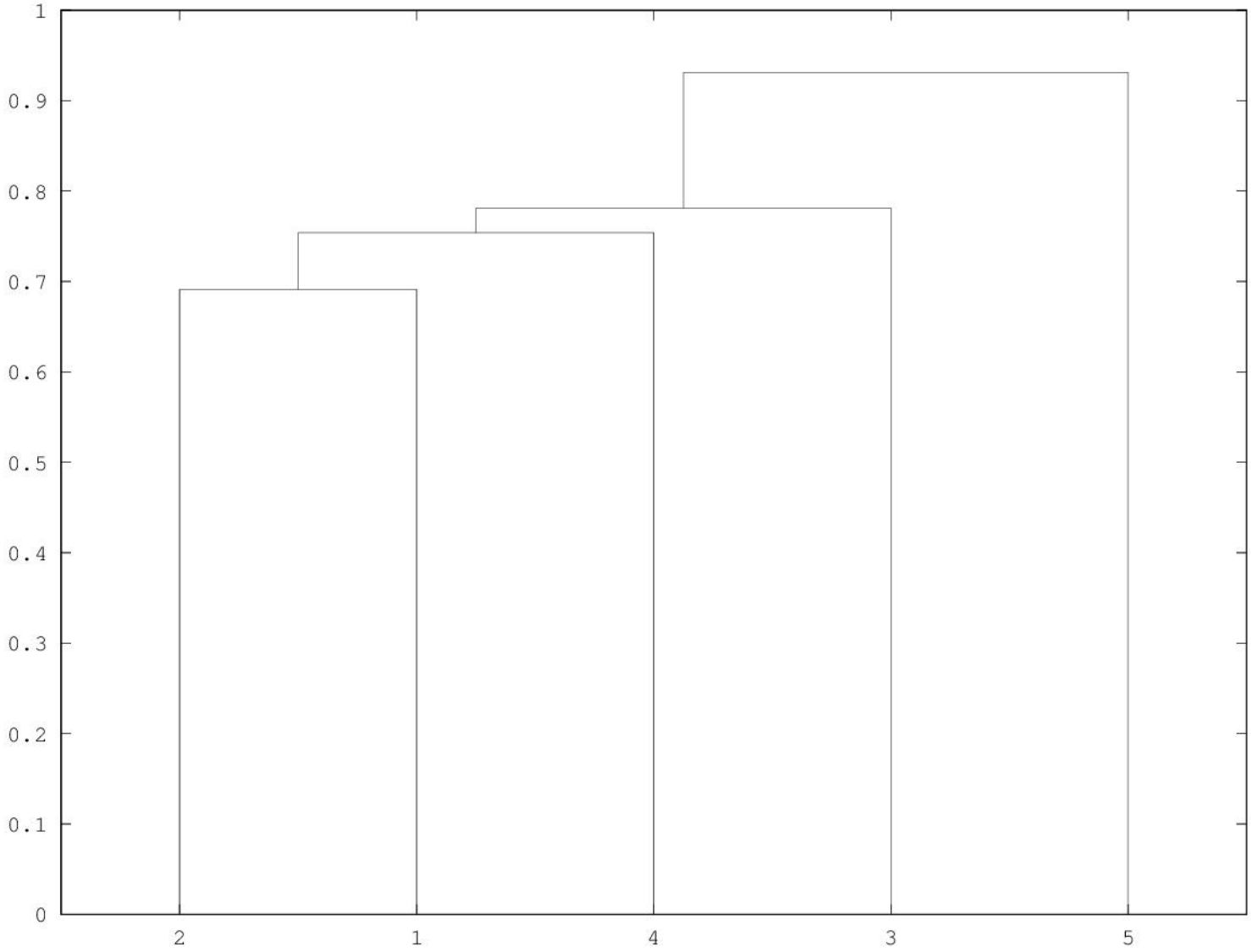

Figure 4.1: Agglomerative hierarchical clustering result.(1) Correct. (2) Partially correct incomplete. (3) Irrelevant. (4) Contradictory. (5) Non domain.

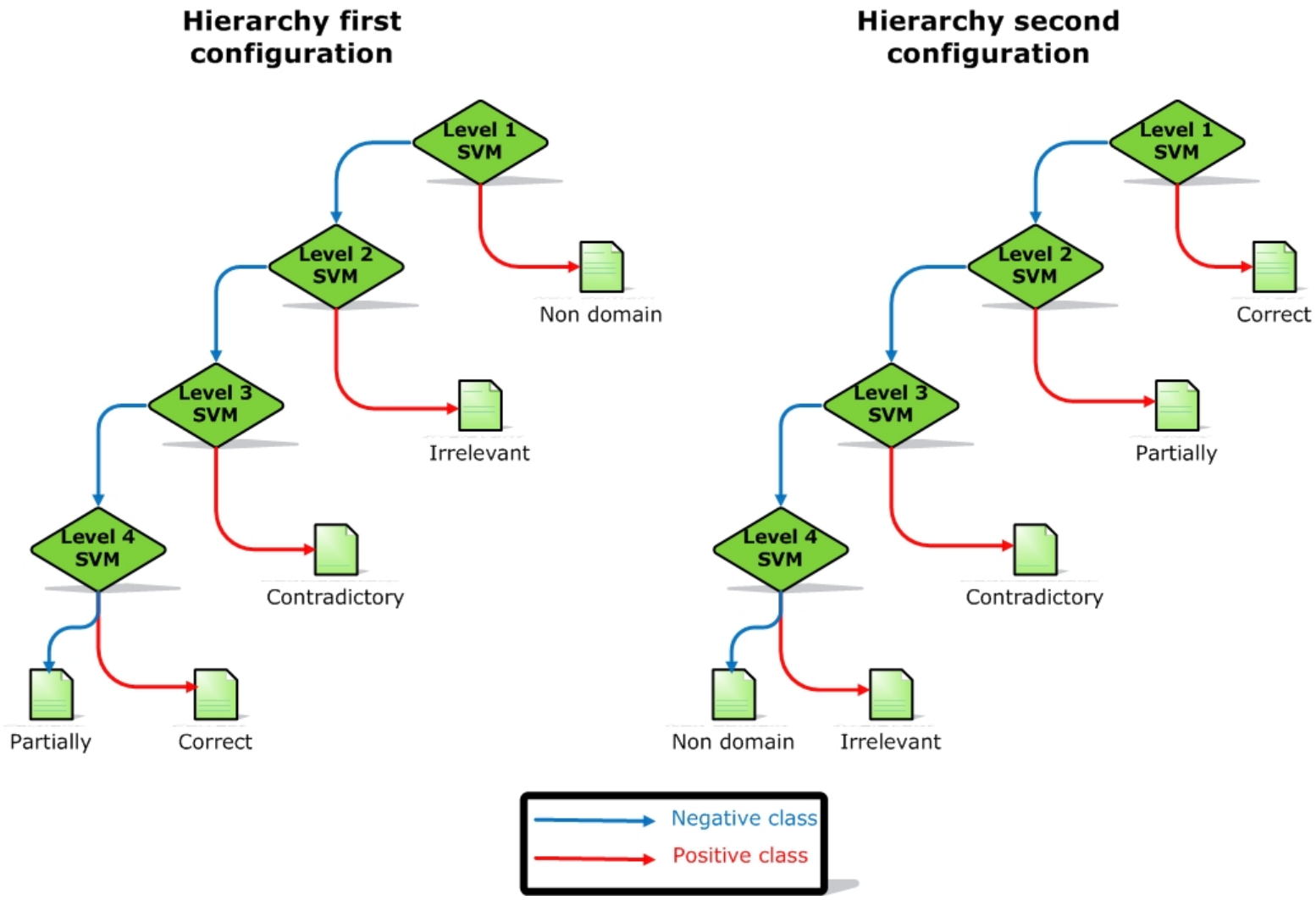

Figure 4.2: Hierarchy configurations to break down the main task.

4.3 Result analysis

In figures 4.1 and 4.2 we show how to combine different classes into a hierarchical architecture. At first, it seems that the hierarchy obtained **correlates well with the logical intuition**. Actually, it is feasible to think that the correct class is the most similar to the partially correct one, that the newly formed group is the most similar to the contradictory[4], and, finally, that the irrelevant and non in the domain classes are the most distanced ones. There are lots of distinct configurations to perform clustering, and each configuration is able to use a wide range of different parameters. That is way our objective throughout this analysis is not to find the optimal way to perform clustering but to test the hypothesis **whether the classification task could be simplified** using the results of the baseline system.

To evaluate each hierarchy configuration we will perform different experiments. Each experiment will optimize the four SVM classifiers of the hierarchy using a 5-fold cross validation technique but following distinct strategies:

F-score maximization

The F-score is a popular metric to measure the accuracy, specially for tasks consisting of **imbalanced data**. The F-score is given by:

$$F_{score} = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4.2}$$

Thus, maximizing the F-score we maximize the relation between precision and recall. Precision is given by:

$$Precision = \frac{TP}{TP + FP} \tag{4.3}$$

and recall is given by:

$$Recall = \frac{TP}{TP + FN} \tag{4.4}$$

That is, the precision metric evaluates the accuracy according to the elements predicted as pertaining to that class. In contrast, the recall metric

[4]Contradictory answers explicitly contradict some part of the reference answer, actually, they could be mostly the same as a correct answer plus the addition of a negative polarity element; in brief, this is way simple text similarity features are not the optimal way to analyze sentences with negative polarity.

evaluates the accuracy rate according to the real elements pertaining to that class. The following table will help understand equations 4.2, 4.3 and 4.4.

		Predicted Class	
		Positive Class = +	Negative Class = -
Real Class	+	TP	FN
	-	FP	TN

Table 4.2: Abstract contingency table for a binary classification task.

Positive class recall maximization

The objective of this optimizing strategy is to **minimize the error propagation through the tree**. For instance, looking at the first level of the second hierarchy configuration, the SVM classifier must classify instances into one of the following sets: correct or other. Without doubt, instances classified as correct would stay as a leaf in the tree, return the class value and increase accuracy, but, what happens with instances that really pertaining to the correct class are classified as other? For sure, those instances go down the tree and increase the error propagation. The error propagation in the tree at each level is given by the **instances that really pertaining to the positive class are classified as negative**. Thus, the error propagation in the tree is given by the FN rate (see table 4.2) and can be modeled by the recall evaluation measure or the false negative rate measure. We have previously analyzed the recall evaluation measure (equation 4.4); conversely, the false negative rate measures out of all the instances in exception of the negative ones the instances that have been incorrectly classified as pertaining to the negative class. It turns to be that for the binary case the false negative rate is given by:

$$FNR = \frac{FN}{TP + FN} = 1 - Recall \tag{4.5}$$

Thus, it seems to be equivalent to maximize the recall or to minimize the false negative rate. In conclusion, **both ways are equivalent to implement the error propagation minimization strategy**.

Cost-sensitive F-score maximization

Continuing with the error propagation minimization strategy, but from an alternative point of view, **cost-sensitive classifiers are able to penalize**

distinct kind of errors according to a cost matrix. The key idea is to consider that some type of misclassification errors (the ones that produce error propagation in the tree, as explained previously) may be worse. By means of a **grid search technique** we are able to optimize the cost sensitive classifier's cost matrix in order to minimize the error propagation in the tree.

In any manner, experiments will follow this procedure: **training data will be divided** into five folds and used to **train and test each one of the four classifiers**. Thus, following a **5-fold cross validation strategy** each classifier will be **trained and optimized independently**. Nevertheless, each classifier is responsible only for making a binary classification, so before training each one of the levels, **data must be mapped into two classes**. To analyze the results in more detail we will calculate by level results[5] as well as global results[6].

4.3.1 Hierarchy by level results

First configuration of the hierarchy

Tables 4.3, 4.4 and 4.5 show the results for the first configuration of the hierarchy by levels. Results for **all the strategies share similar results**. Some little improvement can be noticed in the case of the macro recall for the experiment maximizing the positive recall. Nevertheless, further analysis revealed us that the improvement on the positive class' recall drove the system to the decline of the precision. In all, we can conclude that **no experiment seems to be better than other** as all of them reach similar accuracy.

F-score max	Micro F	Macro F	Micro Pr	Macro Pr	Micro Rec	Macro Rec
Level 1	0.988	0.867	0.987	0.889	0.988	0.849
Level 2	0.856	0.680	0.850	0.723	0.867	0.657
Level 3	0.795	0.667	0.790	0.710	0.811	0.650
Level 4	0.745	0.727	0.744	0.732	0.747	0.724

Table 4.3: Results of the F-score maximization experiment for the first configuration of the hierarchy.

[5]Independent results for each level.
[6]Results for the entire hierarchy configuration.

Recall max	Micro F	Macro F	Micro Pr	Macro Pr	Micro Rec	Macro Rec
Level 1	0.986	0.851	0.986	0.833	0.985	0.871
Level 2	0.857	0.7	0.853	0.710	0.861	0.683
Level 3	0.787	0.676	0.787	0.677	0.788	0.676
Level 4	0.742	0.726	0.742	0.727	0.743	0.724

Table 4.4: Results of the positive class recall maximization experiment for the first configuration of the hierarchy.

Cost-sense	Micro F	Macro F	Micro Pr	Macro Pr	Micro Rec	Macro Rec
Level 1	0.987	0.863	0.987	0.867	0.987	0.859
Level 2	0.855	0.666	0.852	0.748	0.873	0.634
Level 3	0.799	0.669	0.797	0.727	0.818	0.646
Level 4	0.744	0.728	0.744	0.729	0.745	0.726

Table 4.5: Results of the cost-sensitive F-score maximization experiment for the first configuration of the hierarchy.

Second configuration of the hierarchy

Tables 4.6, 4.7 and 4.8 show the results for the second configuration of the hierarchy by levels. For the second configuration too, **all of the strategies employed obtained similar results**. It could be mentioned that the values for the cost-sensitive configuration are slightly lower than others, and, contrary to the first hierarchy configuration, this time, the recall maximizing experiment does not show any significant improvement in the recall values. Nevertheless, it is worth to mention that in the second hierarchy configuration the **drop between the micro and the macro evaluation measures is much lower** than in the first hierarchy configuration. This result, surely is caused because at mapping time the **second hierarchy configuration makes a much more balanced training data at each level**, whereas in the first hierarchy configuration the training data is much more unbalanced (table 2.2 summarizes the class distribution in the training data).

F-score max	Micro F	Macro F	Micro Pr	Macro Pr	Micro Rec	Macro Rec
Level 1	0.762	0.752	0.764	0.761	0.765	0.748
Level 2	0.699	0.692	0.698	0.693	0.700	0.690
Level 3	0.780	0.780	0.783	0.782	0.780	0.782
Level 4	0.960	0.921	0.960	0.924	0.960	0.918

Table 4.6: Results of the F-score maximization experiment for the second configuration of the hierarchy.

Recall max	Micro F	Macro F	Micro Pr	Macro Pr	Micro Rec	Macro Rec
Level 1	0.762	0.752	0.764	0.761	0.765	0.748
Level 2	0.699	0.692	0.698	0.693	0.700	0.690
Level 3	0.780	0.780	0.782	0.781	0.780	0.781
Level 4	0.959	0.919	0.959	0.926	0.959	0.911

Table 4.7: Results of the positive class recall maximization experiment for the second configuration of the hierarchy.

Cost-sense	Micro F	Macro F	Micro Pr	Macro Pr	Micro Rec	Macro Rec
Level 1	0.754	0.742	0.758	0.757	0.758	0.737
Level 2	0.695	0.687	0.694	0.690	0.696	0.685
Level 3	0.778	0.778	0.780	0.780	0.778	0.779
Level 4	0.955	0.911	0.955	0.915	0.955	0.907

Table 4.8: Results of the cost-sensitive F-score maximization experiment for the second configuration of the hierarchy.

4.3.2 Hierarchy global results

First configuration of the hierarchy

Table 4.9 shows the global results for the first configuration of the hierarchy. Results clearly indicate that although all of the optimizing strategies reach nearby values **none of them beats the baseline significantly**. Let's see what happen with the second configuration.

	Micro F	Macro F	Micro Pr	Macro Pr	Micro Rec	Macro Rec
F-score max	0.560	0.553	0.563	0.586	0.572	0.537
Recall max	0.561	0.554	0.560	0.551	0.563	0.559
Cost-sense	0.557	0.546	0.570	0.590	0.571	0.534
Baseline	0.571	0.566	0.568	0.565	0.578	0.572

Table 4.9: Global results for the first configuration of the hierarchy and the baseline.

Second configuration of the hierarchy

Table 4.10 shows the global results for the second configuration of the hierarchy. In this case we are again in the same scenario, none of the strategies implemented is able to significantly outperform the baseline. Our main goal was to test whether a hierarchical clusterization of the data was able to help in the classification task. Results show that the **objective remains unachieved**. Additionally, the different strategies used to optimize the tree have obtained quite similar results. Actually, in many cases **improvement on an evaluation measure has turned to the deterioration of another**. In all, our analysis has concluded that the **error propagation happening in every level of the tree is high** and that error propagation turns to worsen the final accuracy.

	Micro F	Macro F	Micro Pr	Macro Pr	Micro Rec	Macro Rec
F-score max	0.568	0.565	0.579	0.550	0.563	0.590
Recall max	0.568	0.566	0.579	0.551	0.563	0.591
Cost-sense	0.560	0.561	0.578	0.545	0.555	0.591
Baseline	0.571	0.566	0.568	0.565	0.578	0.572

Table 4.10: Global results for the second configuration of the hierarchy and the baseline.

4.4 Beyond the hierarchy

Similarly to the attribute engineering work done for the baseline system, we have also perform this step for the hierarchical configurations described previously[7]. Once again, the objective is to test whether there exists any **optimal attribute subset that can help model the data** in a better manner. In fact, the intuition is to think that once we have (presumably) decomposed the main task into smaller subtasks, each subtask will require **a different set of attributes**, or even new ones.

4.4.1 Feature subset selection

In this section we will perform the same experiments as we did for the baseline system (see section 3.2.2 for complete description on them). That is, we will run the following **feature subset selection algorithms**:

1. A filter-based method using Weka's Cfs RankSeach.
2. A filter-based method using Weka's Infogain Ranker.
3. A wrapper-based method using Weka's Genetic Search.

Notice that the described experiments will only be carried out for the **second configuration** of the hierarchy (see figure 4.2). As explained previously, training data is much more balanced when mapping the instances to the binary scenario and that is why we give preference to this configuration.

Figures 4.3 to 4.6 show the results of applying the Weka's filter-based Cfs RankSeach algorithm to the second hierarchy configuration. Similarly, figures 4.7 to 4.10 show the results of applying the Weka's filter-based Infogain Ranker algorithm to the second hierarchy configuration.

Although results from both filter-based methods remain similar, they are **quite more optimistic than the ones obtained for the baseline system** (figure 3.5 and 3.6). Actually, the drops observed in the F-score values as we remove attributes is much more smoothed than the one observed for the baseline system, specially for the contradictory class. Concerning the first level, it is clearly the level that most suffers from attribute removal, even though, removing the worst scored attributes does not affect the system too much. Considering the second and the fourth level, it is noticeable that after **fifteen attributes are removed the system almost maintains**

[7]Notice that we are only going to reuse the methods, not the previous optimal feature set, actually, as we have decomposed the main task into smaller subtasks, for this configuration we do not know which the optimal subset is.

the same accuracy. This factor is even more noticeable in the fourth level. Moreover, from level three graph we can notice that there might **not be any attribute specially useful to model contradictory** instances as no attribute removal seems to significantly deteriorate the performance of the system.

As regards the **wrapper-based genetic experiment**, table 4.11 shows results (wrapped-based genetic F-maximization strategy) for the second hierarchical configuration. It is obvious that results remain very similar to the ones seen in table 4.6, 4.7 and 4.8, in contrast, several attributes have been eliminated from each level.

System	Micro F	Micro Prec	Micro Rec	Removed Atts
Wrapper Lvl 1	0.752	0.756	0.757	1,6,9,11,18,19,20,28
Wrapper Lvl 2	0.699	0.699	0.701	1,5,16,18,20,23,25,30
Wrapper Lvl 3	0.786	0.788	0.786	1,3,5,6,12,15,20,23,25,30
Wrapper Lvl 4	0.965	0.965	0.965	1,2,4,9,10,11,15,16,17,21,22,23,24,26,27,28,29,30

Table 4.11: Wrapper-based feature subset selection results using genetic algorithms.

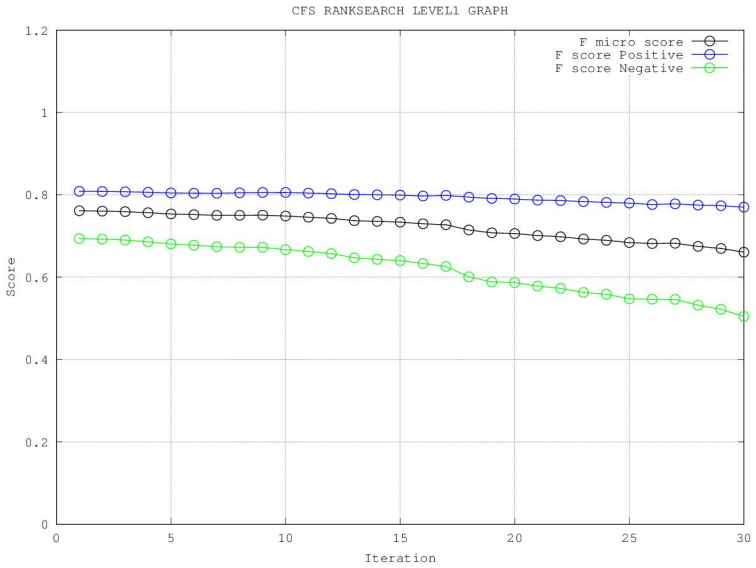

Figure 4.3: Level 1 Cfs Ranksearch results.

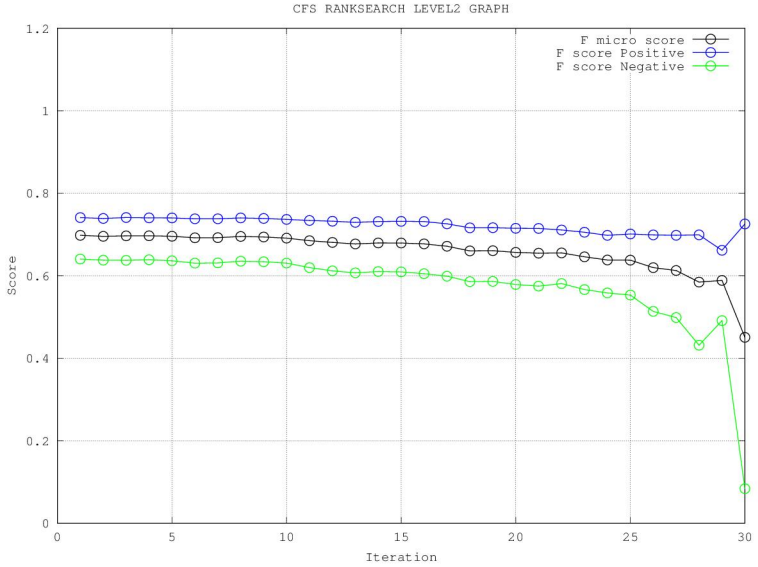

Figure 4.4: Level 2 Cfs Ranksearch results.

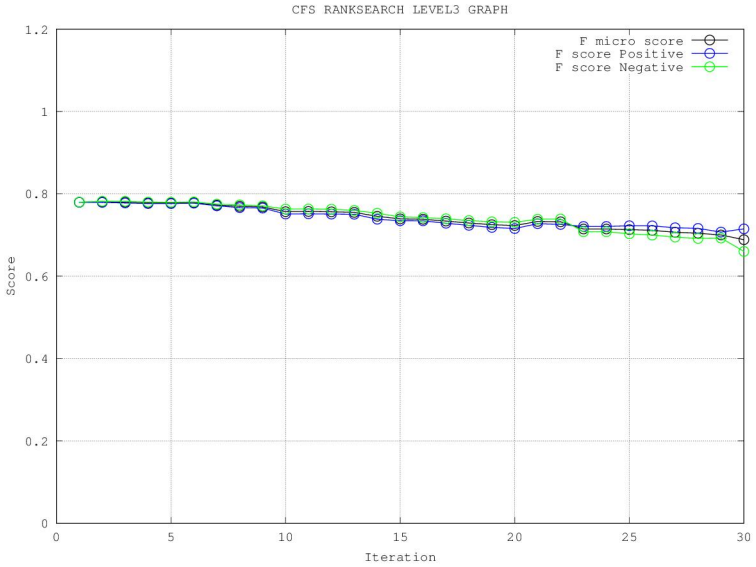

Figure 4.5: Level 3 Cfs Ranksearch results.

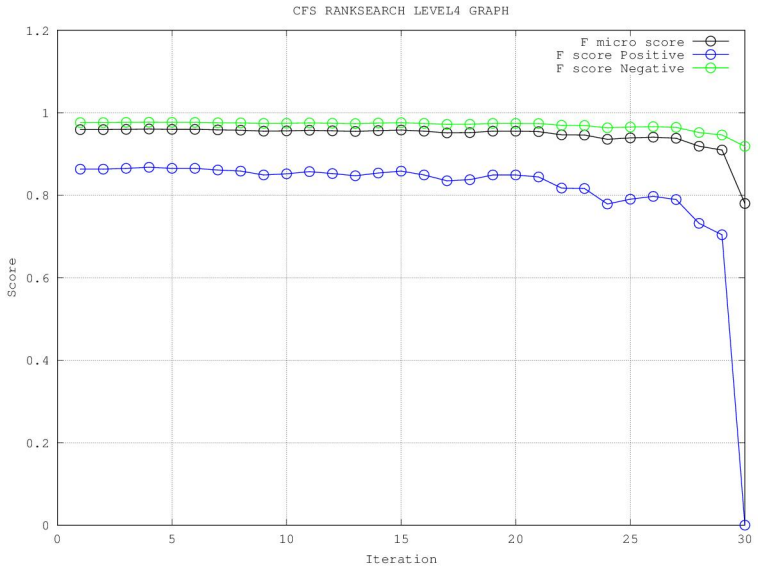

Figure 4.6: Level 4 Cfs Ranksearch results.

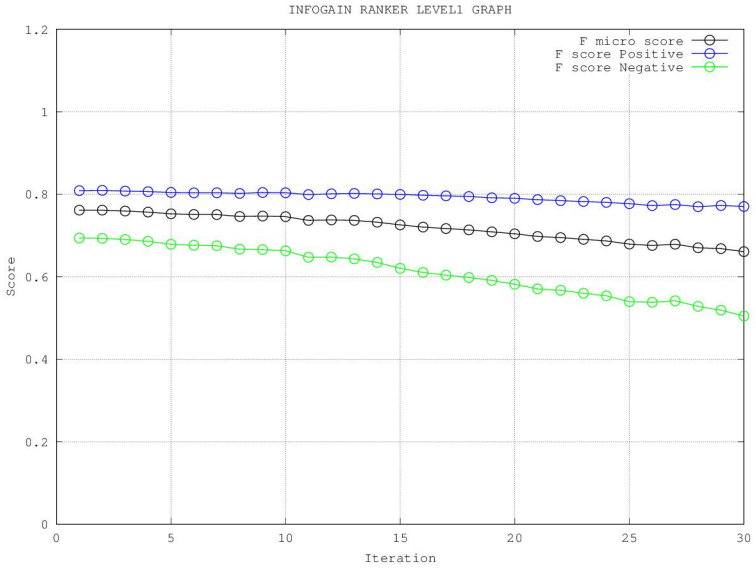

Figure 4.7: Level 1 Infogain Ranker results.

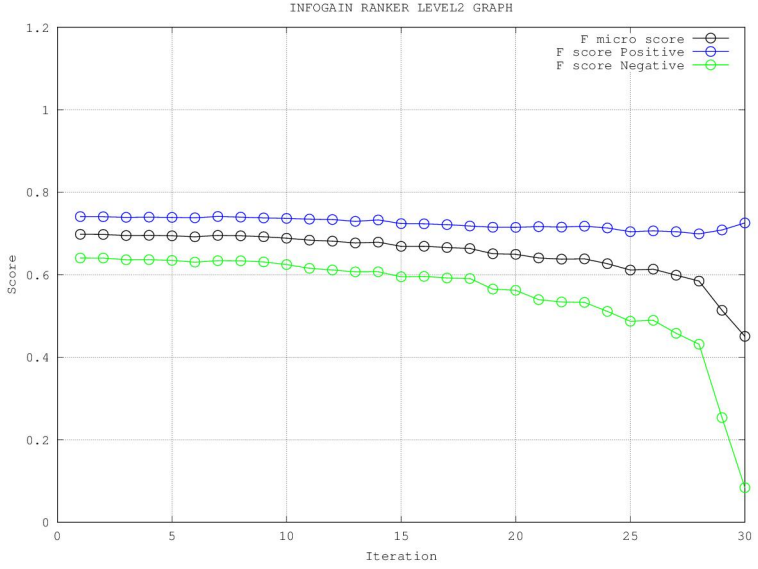

Figure 4.8: Level 2 Infogain Ranker results.

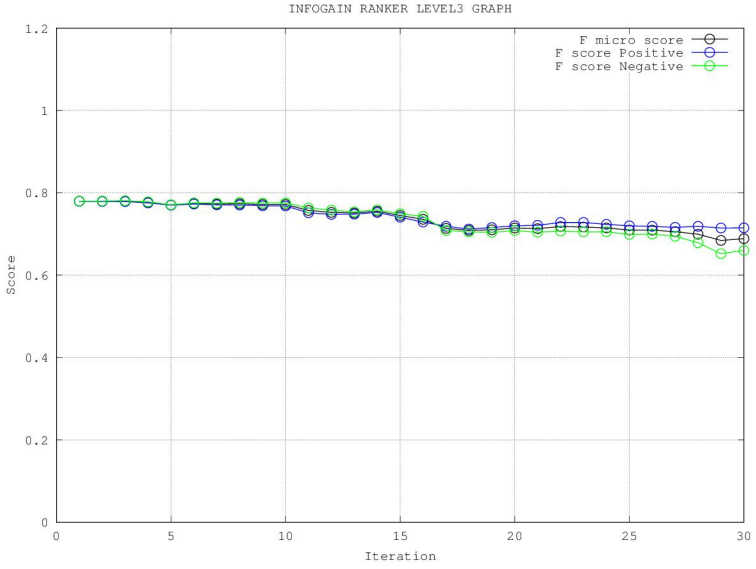

Figure 4.9: Level 3 Infogain Ranker results.

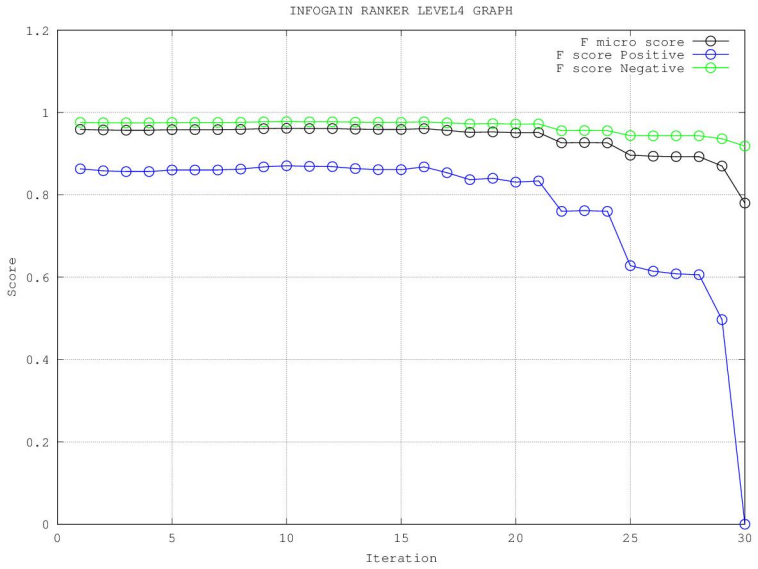

Figure 4.10: Level 4 Infogain Ranker results.

4.4.2 Selecting the optimal feature subset

Following the same criteria as for the baseline system (see section 3.2.3), we will use equation 3.2 to **score and compare** each iteration from figures 4.3 to 4.10 and select the **optimal feature set**. Results obtained for filter-based methods are resumed in tables 4.12, 4.13, 4.14 and 4.15. Each table resumes the results for each one of the levels of the hierarchy.

FSS filter	Penalizing factor	Best scored iteration	Micro F-Score
Cfs Ranksearch method	1	30	0.66
	3	27	0.68
	5	17	0.72
	7	17	0.72
	9	15	0.73
FSS filter	**Penalizing factor**	**Best scored iteration**	**Micro F-Score**
Infogain Ranker	1	30	0.66
	3	27	0.68
	5	18	0.71
	7	13	0.74
	9	13	0.74

Table 4.12: Scoring filter-based iterations in the first level of the hierarchical system.

FSS filter	Penalizing factor	Best scored iteration	Micro F-Score
Cfs Ranksearch method	1	25	0.63
	3	22	0.65
	5	16	0.68
	7	16	0.68
	9	16	0.68
FSS filter	**Penalizing factor**	**Best scored iteration**	**Micro F-Score**
Infogain Ranker	1	26	0.61
	3	18	0.66
	5	18	0.66
	7	18	0.66
	9	14	0.68

Table 4.13: Scoring filter-based iterations in the second level of the hierarchical system..

FSS filter	Penalizing factor	Best scored iteration	Micro F-Score
Cfs Ranksearch method	1	29	0.70
	3	22	0.73
	5	22	0.73
	7	22	0.73
	9	13	0.75
FSS filter	**Penalizing factor**	**Best scored iteration**	**Micro F-Score**
Infogain Ranker	1	30	0.68
	3	26	0.70
	5	14	0.75
	7	14	0.75
	9	14	0.75

Table 4.14: Scoring filter-based iterations in the third level of the hierarchical system.

FSS filter	Penalizing factor	Best scored iteration	Micro F-Score
Cfs Ranksearch method	1	27	0.93
	3	27	0.93
	5	21	0.95
	7	20	0.96
	9	20	0.96
FSS filter	**Penalizing factor**	**Best scored iteration**	**Micro F-Score**
Infogain Ranker	1	28	0.89
	3	21	0.95
	5	21	0.95
	7	21	0.95
	9	16	0.96

Table 4.15: Scoring filter-based iterations in the fourth level of the hierarchical system.

In the overall, results show the same tendency as the one seen for the baseline system. The increase of the **penalization parameter** makes the iteration scorer approximate more to a conservative result in which it discards a fewer number of attributes. Taking into account that reference micro F-Scores obtained are: 0.76, 0.69, 0.79 and 0.96 respectively for each level of the hierarchical system (go to tables 4.6, 4.7 and 4.8 for more information), and the results obtained are: 0.73, 0.68, 0.75 and 0.96 as regards the most conservative ones; we can conclude that there is **not much loose in favor of simpler models**. The following figures show a graphical representation

of the attributes that will be discarded from each level. For the task, we will use the **majority voting method** as we have previously done.

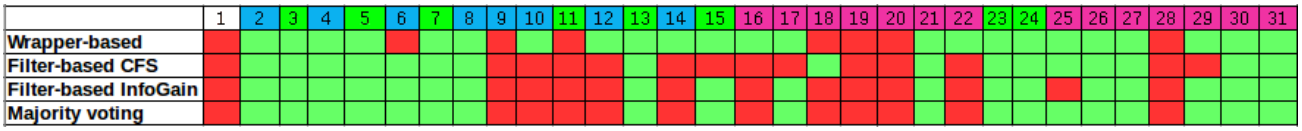

Figure 4.11: Making the feature final decision in the first level of the second configuration of the hierarchy.

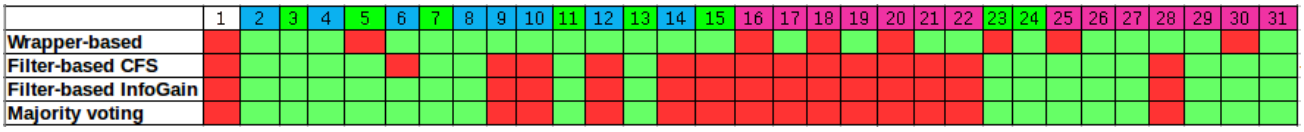

Figure 4.12: Making the feature final decision in the second level of the second configuration of the hierarchy.

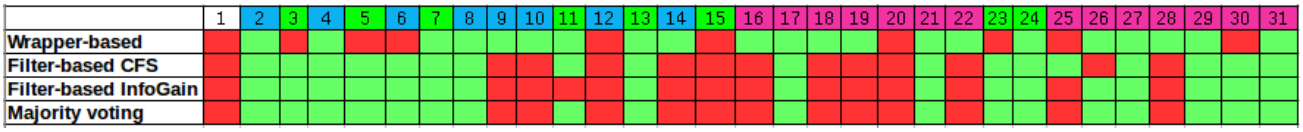

Figure 4.13: Making the feature final decision in the third level of the second configuration of the hierarchy.

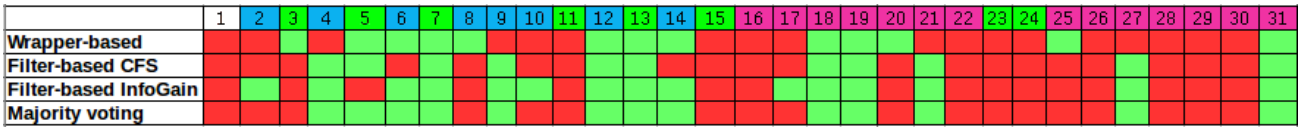

Figure 4.14: Making the feature final decision in the fourth level of the second configuration of the hierarchy.

Red square means discard the attribute and green square means select the attribute. Accordingly, we will remove the following attributes from each level of the hierarchy:

- Level 1.

  Removed attributes: 1, 9, 10, 11, 12, 14, 16, 18, 19, 20, 22 and 28: the ID attribute, 4 text overlap attributes, 6 WordNet based attributes and 1 graph and corpus based attributes.

- Level 2.

  Removed attributes: 1, 9, 10, 12, 14, 15, 16, 17, 18, 19, 20, 21, 22 and 28: the ID attribute, 4 text overlap attributes, 8 WordNet based attributes and 1 graph and corpus based attributes.

- Level 3.

  Removed attributes: 1, 9, 10, 12 , 14, 15, 16, 18, 19, 20, 22, 25 and 28: the ID attribute, 4 text overlap attributes, 7 WordNet based attributes and 1 graph and corpus based attributes.

- Level 4.

  Removed attributes: 1, 2, 3, 8, 10, 11, 15, 16, 17, 20, 22, 23, 24, 25, 26, 28, 29 and 30: the ID attribute, 3 text overlap attributes, 9 WordNet based attributes and 5 graph and corpus based attributes.

From these results we can highlight that the most penalized features are the ones based on WordNet, followed by the features based on text overlap. As we previously mentioned in section 3.2.1 the correlation among text overlap features and among WordNet features was high, that is way feature subset selection techniques have penalized more this feature groups. In contrast, graph-based, corpus-based and predicate-argument based features are more disperse between each other (they come from distinct sources) and are not highly correlated, that is way they are not penalized that much. It is also noticeable that in the same way than it was for the baseline system, this time too, for each level of the hierarchy and indistinctly of the attribute grouping **the majority of the attributes removed are the ones calculated between the student answer and the question sentence**.

4.4.3 Testing the optimal feature subset. The learning curve

For the hierarchical configuration too we will use the same technique to **graphically represent its performance**. Remember that **learning curve plots** are very interesting to interpret whether the model is working correctly or not, and, in addition, to perform sanity checks of the model under analysis. Following the same criteria than in section 3.2.4, we will use **three distinct sources** to plot the curves:

1. Train set Error Rate (TrER).
2. 5-fold Cross Validation Error Rate (CV5ER).
3. Test set Error Rate (TeER).

To plot the learning curves we will start with a relatively small training set and we will recursively increase its size in each iteration. Figure 4.15 shows the learning curve plot that corresponds to the flat hierarchical configuration, and figure 4.16 shows the learning curve plot that corresponds to the reduced hierarchical configuration. As regards the reduced system, the attributes mentioned in the previous section have been removed from the corresponding levels.

The plots obtained for the hierarchical configuration of the system are quite similar to the ones obtained for the baseline system (see section 3.2.4). Actually, figure 4.16 (reduced system) is more compact than figure 4.15 (full system) just the same as for the baseline system. Remember that the reduced system plot is more compact due to the fact the **usage of simpler models yields a minor adaptation to data**, which increases the training error rate. We can also mention that, on the overall, the tendency of the curves follow the correct behavior: on the one hand, concerning the training error rate, it is quite low on first iterations but as the size of the training data grows it increases because the model is not able to fit that well. On the other hand, as regards the testing data, it is quite high on first iterations and it gets smaller as the training data increases. This last comment is more noticeable in the cross validation error rate because as we increase the training data the testing error rate does not reflect as much decrease as we would expect. It seems that for test instances the classifier does not significantly improve its generalizing capacity and maintains similar results. As regards the feature subset selection methods employed we can conclude that the **removal of attributes does not negatively affect the system**, actually, the test error rate is a little lower in the reduced system.

Table 4.16 and 4.17 summarize the last iteration results obtained for all models (we have added the results of the baseline system to facilitate comparisons). Table 4.16 summarizes the error rate related measures and table 4.17 summarizes the evaluation summary metric related measures:

System	TrER	CV5ER	TeER
Hierarchy flat	0.17	0.44	0.58
Hierarchy reduced	0.35	0.48	0.57
Baseline flat	0.15	0.42	0.58
Baseline reduced	0.30	0.44	0.54

Table 4.16: Results obtained in the last iteration of the learning curve for all of the systems.

System	Test Micro F	Test Micro Pr	Test Micro Rec
Hierarchy flat	0.42	0.45	0.41
Hierarchy reduced	0.43	0.46	0.41
Baseline flat	0.41	0.41	0.42
Baseline reduced	0.44	0.43	0.45

Table 4.17: Results obtained in the last iteration of the learning curve for all of the systems.

For the hierarchical configuration the **simplification of the model has not resulted in a significant improvement** (+0.01 on the test micro F-Score) although several attributes have been eliminated from each level. At first, it is noticeable that both hierarchical configurations obtain **higher precision scores** than the scores of the baseline flat and reduced systems, but then they compensate that with **lower recall values**. It is also noticeable that for all systems the training and cross validation error rates of reduced systems are higher due to the fact of being using simpler models that do not fit that much to data. To conclude with, it seems that **in the overall the best results are the ones of the baseline reduced system** (see section 3.2.4).

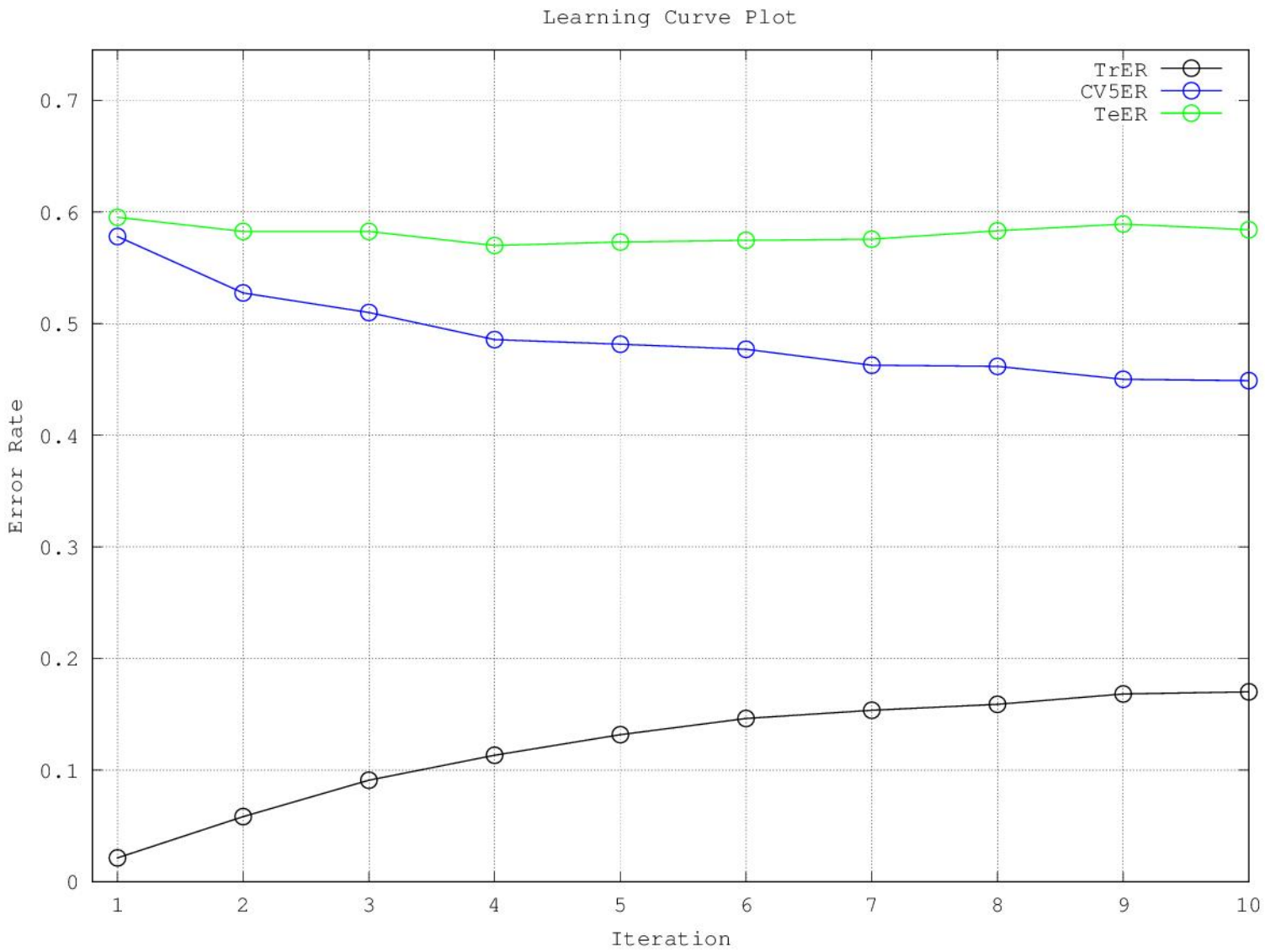

Figure 4.15: Learning curve plot of the flat hierarchical configuration.

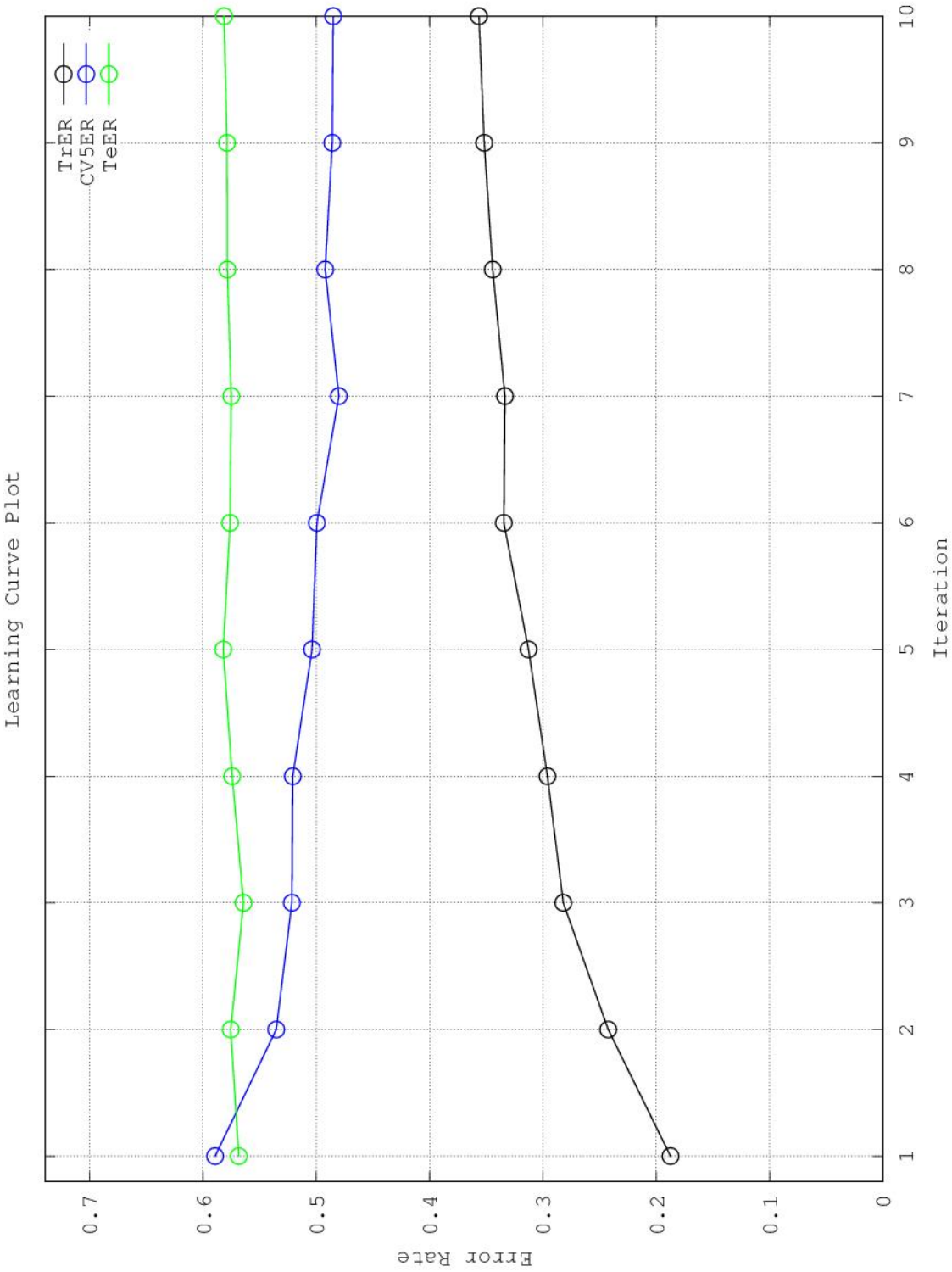

Figure 4.16: Learning curve plot of the reduced hierarchical configuration.

4.5 Final remarks

Throughout this chapter we have been working with an alternative strategy in which the main goal has been to **simplify the main classification task**. To build such a system we have used a divide and conquer approach to break down the main problem into distinct **binary subtasks**. After that, we computed some kind of similarity measure out the contingency table and by means of **hierarchical clustering** techniques we have built two alternative hierarchical configurations. Each hierarchical meta-decision tree consists of four SVM classifiers, but each one trained to be responsible for a single binary decision. Thus, in order to classify a new instance, it must go down the tree until it reaches a leaf.

To continue with, we tried different experiments in order to **optimize the resulting systems and reduce the error propagation** but we obtained quite similar results with all of the methods employed (F-Score maximization, positive class recall maximization and cost-sensitive F-Score maximization). For the feature subset selection algorithms and further experiments we have only used the second configuration of the hierarchy, as both configurations obtained similar results, but, the class distribution of the second is better balanced. We have used the same scorer and majority voting method as for the baseline system to score distinct feature subsets for each level of the hierarchical configuration; in order to obtain the **optimal attribute set** that could help model the data in a better manner. As regards features we concluded that **features calculated between student answers and reference answers were more relevant** than those computed between student answers and question sentences, and, finally, we have used learning curve plots to graphically visualize its performance and compare the results with the ones of the baseline system.

We have also seen that the **removal of several attributes does not negatively affect the system**, but, unfortunately, the baseline system's results remains unbeaten, however, **the hierarchical configuration obtains quite similar results**. The reason for achieving similar results is not completely solved, but it could be one of the following ones (or a combination of them):

- Classes can be globally partitioned with no major problem. Making binary decisions is not simpler than making a 5-way decision.
- Feature space is not distributable[8]. The system requires new attributes in order to improve.

[8]Instances from different classes can not be separated one from another.

- The error propagation in the meta-decision tree is too high. Even obtaining good results at each of the four levels independently, final accuracy is deteriorated when combining all levels.

Altogether, even if performing feature subset selection techniques each system (flat and hierarchical) has distinct optimal feature subsets, there is no doubt that the origin of features is identical at the initial point. This fact could be quite determinant for having different systems with similar results; and that is why we performed a **Kappa test** to measure the inter-agreement between the baseline system and the hierarchical system. We obtained a value of K = 0.744, which confirms that the **agreement level between both systems is quite high** (7362 instances out of 8910 classified exactly the same manner).

4.5.1 Graphical representation of data in a 3 dimensional latent space

Motivated by the fact that the hierarchy has only obtained similar results with respect to the baseline system, we mentioned that classes could be **globally partitioned with no major problems** making it unnecessary to simplify the task. But this is (for the moment) impossible to visualize in a feature space with so many attributes, that is way we will use principal component analysis[9] to reduce the dimensionality of the problem to a 3 dimensional latent space and visually inspect the space. For the task we will be using the Waffles framework [Gashler, 2011].

PCA is a dimensionality reduction technique that aims to compress data[10]. The key idea about PCA is that it projects data into a new N-dimensional space[11]. Usually, this latent space will be lower than the original space, for instance, in this experiment we will reduce a 31-dimensional space to a 3-dimensional space in order to plot it and visualize it in 3D. The **principal component vectors** of a dataset aim to **minimize the projection error** of the data while retaining as much variance as possible; and, actually, the number of principal components to use is chosen according to the **reduction needed and the variance that we want to retain** in the new latent space.

[9] PCA was invented in 1901 by Karl Pearson

[10] As a consequence of compressing data it is also used to speed up algorithms.

[11] Sometimes most of the information is gathered from a few but meaningful dimensions.

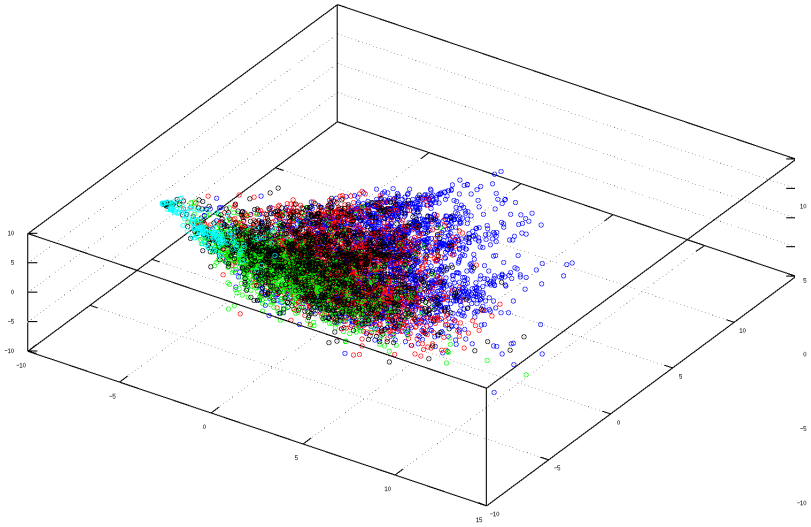

Figure 4.17: Scatter3 plot for the 5-way task. View 1.

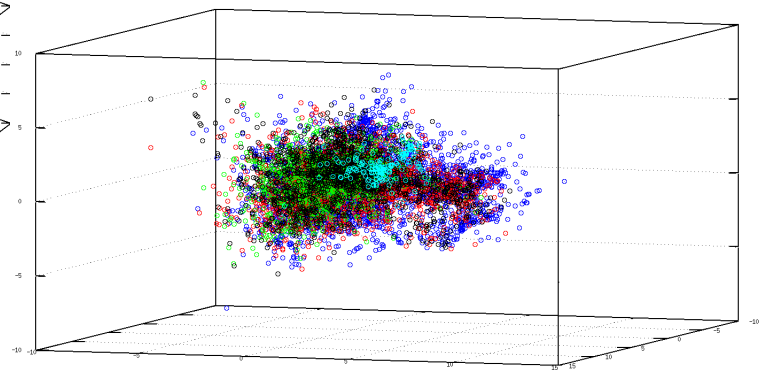

Figure 4.18: Scatter3 plot for the 5-way task. View 2.

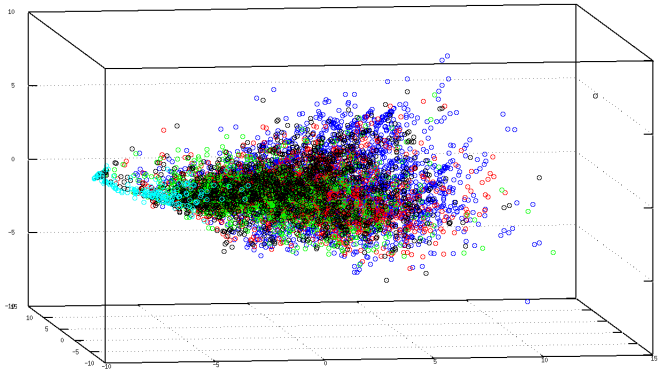

Figure 4.19: Scatter3 plot for the 5-way task. View 3.

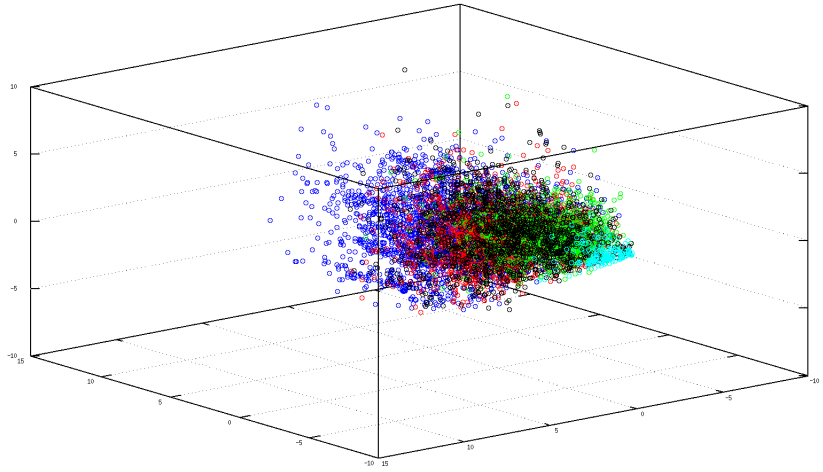

Figure 4.20: Scatter3 plot for the 5-way task. View 4.

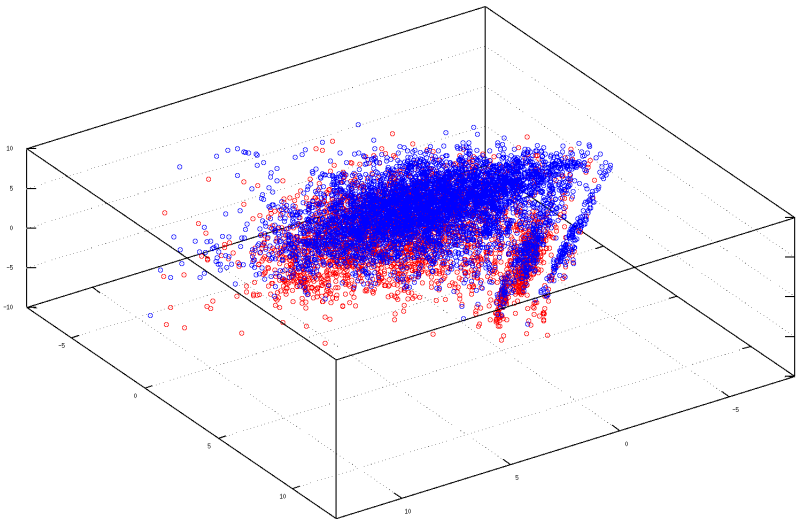

Figure 4.21: Scatter3 plot for the first level of the hierarchy.

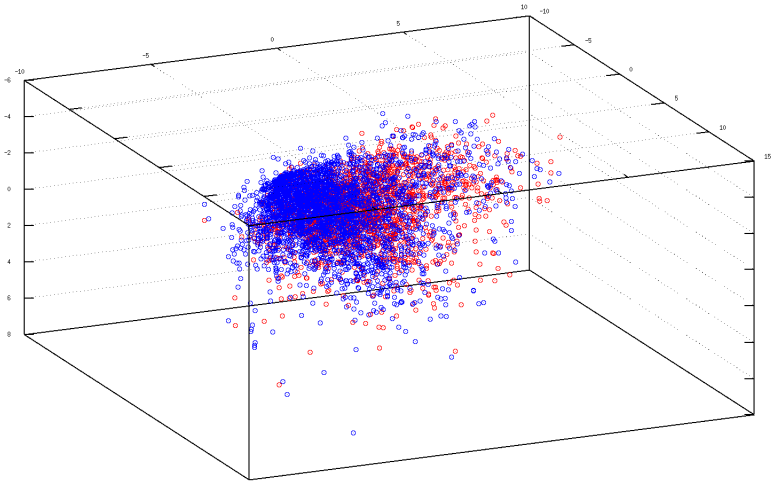

Figure 4.22: Scatter3 plot for the second level of the hierarchy.

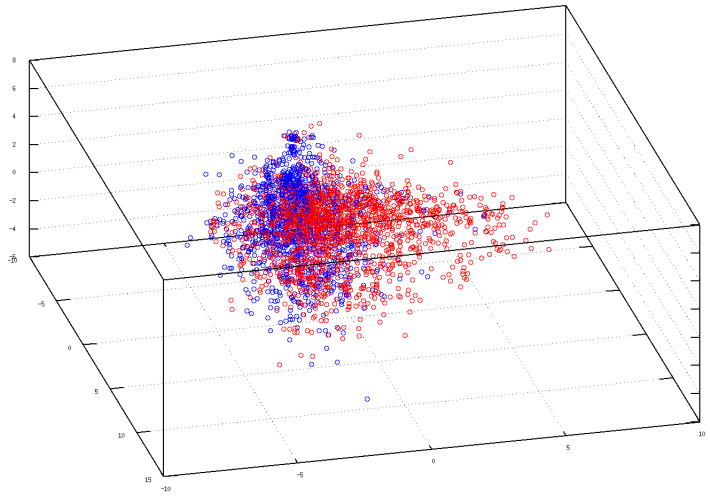

Figure 4.23: Scatter3 plot for the third level of the hierarchy.

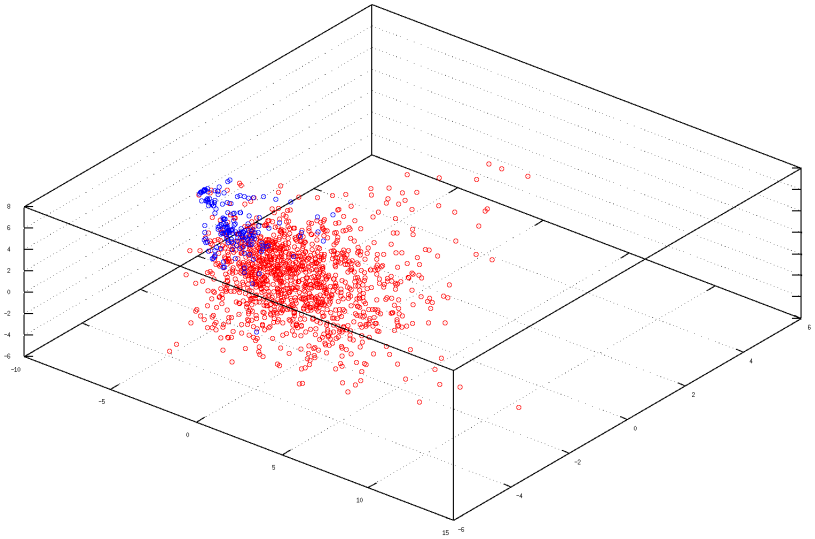

Figure 4.24: Scatter3 plot for the fourth level of the hierarchy

Figures 4.17 to 4.20 show the 5-way data from the Joint Student Response Analysis task reduced to a **3-dimensional latent space**. That is, the 3-dimensional projection that tries to retain the maximum variance. A different color has been used to represent each class, colors are as follows:

1. Correct: blue
2. Partially correct: red
3. Contradictory: black
4. Irrelevant: green
5. Non domain: cyan

Figures 4.21 to 4.24 show the 5-way data reduced to a 3-dimensional latent space but as seen by each level of the hierarchy. That is, each figure shows the **binary partition** discussed previously (see the second hierarchy configuration described in figure 4.2). The colors used are the same ones as the colors of the figure: red color for the positive class and blue color for the negative class.

1. Level 1 - Correct(+); rest(-)
2. Level 2 - Partially correct(+); rest(-)
3. Level 3 - Contradictory(+); rest(-)
4. Level 4 - Irrelevant(+); Non domain(-)

It is noticeable that **the 5-way task seems to be not clearly distributable**, but unfortunately the same effect happens in some levels of the hierarchy. Actually, correct, irrelevant and non in the domain instances seem to be the clearest ones to partition, and, on the contrary, partially correct and contradictory instances seem to be the most confusing ones. The thing is that the **same effect is observed in the hierarchical implementation**, that is: from level 1, level 3 and level 4 we could somehow draw a separation plane (maybe level 1 is more questionable). However, it is **not easy to separate instances** drawn in level 3, just as it is not easy to separate black instances from the 5-way figures. They just look as completely overlapped.

In the overall, this figures are provided just to visualize the data and help guess what is happening in there; and not to make big decisions out of them. Using just three principal components is not the optimal way to reduce this dataset, but to visualize data.

4.6 Future work

Related to the hierarchical configuration of the EHU-ALM system, the following experiments and ideas are the ones left as future work: to analyze **distinct configurations to subdivide the task**, such as: combinations of one versus all approaches or boosting approaches, to analyze if performing a feature subset selection technique but considering **groups of attributes** level results are able to improve, to further analyze the correlation among groups of features and how groups affect each level of the system, to test **different kind of classifiers for each level**, to use **alternative feature subset selection** techniques, to test the system individually in the BEETLE and in the SCIENTSBANK datasets, to **implement new features** of top performing systems of the SemEval task that could possibly contribute to the system, in fact, it is feasible to think that different kind of attributes might contribute in a different way for each level, for instance: features that explicitly handle negation seem to be critical in the level where we classify contradictory instances.

Chapter 5

Final Remarks Concerning the Master Thesis Project

Contents

5.1 Brief summary

Through this master thesis we have been working on an **answer scoring task** from the standpoint of **machine learning**. We have seen that machine learning can perform important tasks by learning from examples and **generalizing** for new ones. Supported by these techniques we focused on the Joint Student **Response Analysis** challenge. The task mainly consisted of automatically scoring short answers to asses student understanding. Initially, we discussed the advantages and disadvantages of **domain-specific models** and how the SemEval task promoted **flexible systems** in order to avoid the weaknesses[1] of data-driven approaches. Actually, the task reiterated two important factors: to perform the first steps towards open-domain systems and to continue developing techniques able to compare meaning between sentences.

Not only we have analyzed the task itself, but also we have described all of the **systems** that took part in the task. In addition, we compared the results obtained by participants and we saw **different alternatives** to approach the scoring challenge. In brief, we summarized distinct approaches as pertaining to one of the following three sets: systems that approached the challenge combining **text similarity** techniques, systems that approached the challenge combining **textual entailment** techniques or **hybrid systems** that used a mixture or combination of techniques. The task resulted to be challenging but results showed that systems achieved relatively high performance.

After that, we focused on **EHU-ALM**, which is a supervised system based on syntactic and semantic similarity features, and, on the overall, this master thesis has followed a machine learning approach whose goal has been to **continue with its development**. In order to find the strategy that would perform better in the SemEval dataset we defined **two alternative strategies**: on the one hand, to improve the system without changing the architecture, and, on the other hand, to improve the system using an hierarchical configuration. To define and build such new models we have used **distinct attribute engineering, data preprocessing, and machine learning techniques**, such as: linear dependency correlation measurements and tests to score the contribution of attributes, filter-based and wrapper-based feature selections through the candidate feature space, hierarchical agglomerative clustering techniques to find new partitions of the training data according to distances between classes, and also, we have developed a **feature scorer** to find the optimal feature set out of all possible candidates.

[1] To collect labeled data is a very time consuming task.

This way, out of the EHU-ALM flat system we developed a **reduced version that used several less attributes than the original system**. Moreover, out of the contingency table of the flat EHU-ALM system we developed an **hierarchical configuration** for the EHU-ALM system, and, to finish with, out of the flat hierarchical system we built a **reduced hierarchical system** that used several less attributes. Finally, we used **learning curve plots** to make graphical representations of resulting models and evaluate their performance. As a result, through this master thesis we have analyzed in detail how the **one versus all strategy has obtained better results** whereas the hierarchical strategy has only managed to somehow equalize them. In summation, throughout this work we have seen how important attribute engineering is to machine learning because **attribute engineering** techniques can significantly reduce the number of optimal attributes by removing irrelevant, redundant or noisy ones, speed up algorithms and even improve performance. In particular, **the simplification of the flat system resulted in a lower error rate** on the test set, even the results at training where slightly worse.

5.2 Technology-based learning systems

Humans develop several learning skills. **Reading comprehension** is one of them and it is, among others, one of the abilities responsible for the correct development of competencies throughout life. Reading comprehension exercises are often linked to either open or multiple choice questions that **assess the understanding**. The usefulness of multiple choice questions in the **formative assessment** of individuals have been questioned in [Davies, 2002] and in [Conole and Warburton, 2005]. On the contrary, they are widely used for the **summative assessment**. Conversely, building answers to open questions plays a critical role on the **consolidation of knowledge** [Karpicke and Roediger, 2008]. In this context, the development of computational linguistic applications that are able to **automatically analyze and assess learners** is an open field of research with many unsolved topics.

Knowledge measuring systems are able to assess understanding by automatically evaluating answers. As a result, it would be trivial to **elaborate personalized student models** making it possible to know what and how much each individual knows. Thus, evaluators could **efficiently and effectively educate** learners or reinforce topics that are not mastered by personalizing the formative feedback. Actually, such tools clearly open dif-

ferent research lines[2]. Among them **reading comprehension** frameworks and **dialogue systems**.

Up to this moment we have done some work concerning reading comprehension frameworks, for instance, [Lopez-Gazpio and Marichalar, 2013b] describes a **collaborative web application prototype** in which evaluators are able to upload texts and manage reading comprehension exercises. That is, out of uploaded texts the framework gives the opportunity to create, manage and share question-based exercises. As a consequence, users are able to work on reading comprehension based on exercises provided by tutors. The web application is also able to **generate questions automatically** following some techniques summarized in [Lopez-Gazpio, 2014]. In addition, it is also capable of checking learner answers based on simple distance-based techniques. In fact, for the near future we plan to **upgrade the answer scoring system with the ideas contributed by this work**. The full framework is described in detail in [Lopez-Gazpio and Marichalar, 2013a]. For the following years, we hope to keep working on similar tasks, such as:

- continue with the development of the answer scoring system(s) described in this master thesis[3].
- explore new semantic similarity features, such as, textual entailment and integrate it in the answer scorer.
- improve the collaborative reading comprehension web application and promote its usage.
- collect new question-answer corpora and generalize the short scoring system to Basque.
- analyze clusters of student answers.
- analyze the internal structure of student responses beyond text-similarity features.
- analyze the viability of scoring answers to open questions from a text-document without a reference answer.

In all, our objective is to continue developing techniques and systems that would be able to give learners **personalized feedback automatically scoring their performance** in order to let them learn better.

[2]Both strategies simulate the human-human tutoring strategy.
[3]Previous chapters describe this work in detail.

Appendix I

Brief Attribute Summary

The following enumeration lists (order is relevant) all of the features used in the baseline system. For further documentation please read [Aldabe et al., 2013]. The syntactic and semantic similarity features can be grouped into the following sets:

1. Text overlap features (marked with cyan).
2. WordNet-based lexical features (marked with magenta).
3. Graph-based, Corpus-based and predicate-argument based features (marked with green).

1. ID

   This is not really an attribute. It is an identification number that uniquely distinguishes each observation in the dataset. This attribute is always removed for every system.

2. COSINE

   Text overlap measure computed using the Perl Text::Similarity package. The cosine measure is given by $raw\_score/sqrt(precision+recall)$. Computed between the student answer and the related reference answer.

3. DEPGRAPH

   Depgraph measures the similarity by analyzing the overlap of the predicates and their associated semantic arguments. Computed between the student answer and the related reference answer.

4. F

   Text overlap measure computed using the Perl Text::Similarity package. The F-measure is given by $2 * precision * recall/(precision + recall)$. Computed between the student answer and the related reference answer.

5. LDA

   LDA measures the similarity between the resulting vectors associated with each text in the latent space (50 topics were estimated). Computed between the student answer and the related reference answer.

6. LESK

   Text overlap measure computed using the Perl Text::Similarity package. The Lesk parameter measures the square sum of the length of

phrasal matches. Computed between the student answer and the related reference answer.

7. LSA

   LSA measures the similarity between the resulting vectors associated with each text in the latent space (100 topics were estimated). Computed between the student answer and the related reference answer.

8. OVERLAP

   Text overlap measure computed using the Perl Text::Similarity package. Overlap measures the number of overlapping words. Computed between the student answer and the related reference answer.

9. QUESTIONCOSINE

   Same as COSINE but computed between the student answer and the related question.

10. QUESTIONF

    Same as F but computed between the student answer and the related question.

11. QUESTIONLDA

    Same as LDA but computed between the student answer and the related question.

12. QUESTIONLESK

    Same as LESK but computed between the student answer and the related question.

13. QUESTIONLSA

    Same as LSA but computed between the student answer and the related question.

14. QUESTIONOVERLAP

    Same as OVERLAP but computed between the student answer and the related question.

15. QUESTIONUKB

    Same as UKB but computed between the student answer and the related question.

16. QUESTIONWNJCN

    Same as WNJCN but computed between the student answer and the related question.

17. QUESTIONWNLCH

    Same as WNLCH but computed between the student answer and the related question.

18. QUESTIONWNLESK

    Same as WNLESK but computed between the student answer and the related question.

19. QUESTIONWNLIN

    Same as WNLIN but computed between the student answer and the related question.

20. QUESTIONWNRESNIK

    Same as WNRESNIK but computed between the student answer and the related question.

21. QUESTIONWNPATH

    Same as WNPATH but computed between the student answer and the related question.

22. QUESTIONWNWUP

    Same as WNUP but computed between the student answer and the related question.

23. SRLGRAPH

    SRLGRAPH measures the role of syntax studying graph subsumption. Computed between the student answer and the related reference answer.

24. UKB

    UKB measures the similarity between the resulting Personalized PageRank Vectors (PPV) associated with each text when performing a random walk over the knowledge base. Computed between the student answer and the related reference answer.

25. WNJCN

    WNJCN measure is computed using the Perl WordNet::Similarity package. WordNet JCN measures the semantic relatedness of word senses based on a combination of edge counts in the WordNet 'is-a' hierarchy and the information content value. Computed between the student answer and the related reference answer.

26. WNLCH

    WNLCH measure is computed using the Perl WordNet::Similarity package. WordNet LCH measures the number of edges between the senses in the 'is-a' hierarchy of WordNet scaled by the maximun depth of the hierarchy. Computed between the student answer and the related reference answer.

27. WNLESK

    WNLESK measure is computed using the Perl WordNet::Similarity package. WordNet LESK measures the semantic similarity proportionally to the extent of overlap words of their dictionary definitions (Extended Gloss Overlap algorithm). Computed between the student answer and the related reference answer.

28. WNLIN

    WNLIN measure is computed using the Perl WordNet::Similarity package. WordNet LIN measures the semantic similarity from the information content of the concepts in WordNet and the 'Similarity Theorem'. Computed between the student answer and the related reference answer.

29. WNRESNIK

    WNRESNIK measure is computed using the Perl WordNet::Similarity package. WordNet RESNIK measures the semantic relatedness of word senses using an information content based measure. Computed between the student answer and the related reference answer.

30. WNPATH

    WNPATH measure is computed using the Perl WordNet::Similarity package. WordNet PATH measures the semantic relatedness of word senses by counting nodes in the noun and verb WordNet 'is-a' hierarchies. Computed between the student answer and the related reference answer.

31. WNWUP

    WNUP measure is computed using the Perl WordNet::Similarity package. WordNet UP measures the semantic relatedness of word senses by using the edge counting method. Computed between the student answer and the related reference answer.

References

[Aldabe et al., 2013] Aldabe, I., Maritxalar, M., and de Lacalle, O. L. (2013). Ehu-alm: Similarity-feature based approach for student response analysis.

[Aleven et al., 2001] Aleven, V., Popescu, O., and Koedinger, K. R. (2001). Towards tutorial dialog to support self-explanation: Adding natural language understanding to a cognitive tutor. In *Proceedings of Artificial Intelligence in Education*, pages 246–255. Citeseer.

[Biçici and van Genabith, 2013] Biçici, E. and van Genabith, J. (2013). Cngl: Grading student answers by acts of translation.

[Callaway et al., 2006] Callaway, C., Dzikovska, M., Matheson, C., Moore, J., and Zinn, C. (2006). Using dialogue to learn math in the leactivemath project. In *Proceedings of the ECAI Workshop on Language-Enhanced Educational Technology*, pages 1–8.

[Chang and Lin, 2011] Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[Conole and Warburton, 2005] Conole, G. and Warburton, B. (2005). A review of computer-assisted assessment. *Research in learning technology*, 13(1).

[Darwin, 1968] Darwin, C. (1968). On the origin of species by means of natural selection. 1859. *Murray, London*.

[Davies, 2002] Davies, P. (2002). There's no confidence in multiple-choice testing,.....

[Dzikovska et al., 2010] Dzikovska, M. O., Bental, D., Moore, J. D., Steinhauser, N. B., Campbell, G. E., Farrow, E., and Callaway, C. B. (2010). Intelligent tutoring with natural language support in the beetle ii system.

In *Sustaining TEL: From Innovation to Learning and Practice*, pages 620–625. Springer.

[Dzikovska et al., 2012] Dzikovska, M. O., Nielsen, R. D., and Brew, C. (2012). Towards effective tutorial feedback for explanation questions: A dataset and baselines. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 200–210. Association for Computational Linguistics.

[Dzikovska et al., 2013] Dzikovska, M. O., Nielsen, R. D., Brew, C., Leacock, C., Giampiccolo, D., Bentivogli, L., Clark, P., Dagan, I., and Dang, H. T. (2013). Semeval-2013 task 7: The joint student response analysis and 8th recognizing textual entailment challenge. In *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval 2013), in conjunction with the Second Joint Conference on Lexical and Computational Semantcis (* SEM 2013), Atlanta, Georgia, USA, June. Association for Computational Linguistics*.

[Gashler, 2011] Gashler, M. S. (2011). Waffles: A machine learning toolkit. *Journal of Machine Learning Research*, MLOSS 12:2383–2387.

[Glass, 2001] Glass, M. (2001). Processing language input in the circsim-tutor intelligent tutoring system. In *Artificial Intelligence in Education*, pages 210–221.

[Gleize and Grau, 2013] Gleize, M. and Grau, B. (2013). Limsiiles: Basic english substitution for student answer assessment at semeval 2013.

[Gordon and Grimes, 2005] Gordon, R. G. and Grimes, B. F. (2005). *Ethnologue: Languages of the world*, volume 15. SIL international Dallas, TX.

[Grimes, 1996] Grimes, B. F. (1996). Ethnologue: Languages of the world. dallas, texas: Summer institute of linguistics.

[Hahn and Meurers, 2013] Hahn, N. O. R. Z. M. and Meurers, D. (2013). Comet: Integrating different levels of linguistic modeling for meaning assessment.

[Hall et al., 2009] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18.

[Heilman and Madnani, 2013] Heilman, M. and Madnani, N. (2013). Ets: Domain adaptation and stacking for short answer scoring.

[Hu et al., 2003] Hu, X., Cai, Z., Louwerse, M., Olney, A., Penumatsa, P., and Graesser, A. (2003). A revised algorithm for latent semantic analysis. In *Proceedings of the 18th international joint conference on Artificial intelligence*, pages 1489–1491. Morgan Kaufmann Publishers Inc.

[Jimenez et al., 2013] Jimenez, S., Becerra, C., Gelbukh, A., Bátiz, A. J. D., and Mendizábal, A. (2013). Softcardinality: Hierarchical text overlap for student response analysis.

[Jordan et al., 2006] Jordan, P. W., Makatchev, M., Pappuswamy, U., VanLehn, K., and Albacete, P. L. (2006). A natural language tutorial dialogue system for physics. In *FLAIRS Conference*, pages 521–526.

[Karpicke and Roediger, 2008] Karpicke, J. D. and Roediger, H. L. (2008). The critical importance of retrieval for learning. *science*, 319(5865):966–968.

[Kouylekov, 2013] Kouylekov, M. (2013). Celi: Edits and generic text pair classification. In *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval 2013)*.

[Liu and Yu, 2005] Liu, H. and Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering. *Knowledge and Data Engineering, IEEE Transactions on*, 17(4):491–502.

[Lopez-Gazpio, 2014] Lopez-Gazpio, I. (2014). Bota galdera! *Elhuyar: zientzia eta teknika*, (307):48–50.

[Lopez-Gazpio and Marichalar, 2013a] Lopez-Gazpio, I. and Marichalar, M. (2013a). Seneko: galderak automatikoki sortuz testuak lantzeko aukera ematen duen aplikazioa.

[Lopez-Gazpio and Marichalar, 2013b] Lopez-Gazpio, I. and Marichalar, M. (2013b). Web application for reading practice. In *IADAT-e2013: Proceedings of the 6th IADAT International Conference on Education*, pages pp–74. IADAT-e2013. ISBN: 978-84-935915-3-3.

[Magnini et al., 2014] Magnini, B., Zanoli, R., Dagan, I., Eichler, K., Neumann, G., Noh, T.-G., Pado, S., Stern, A., and Levy, O. (2014). The excitement open platform for textual inferences.

[McCarthy et al., 2008] McCarthy, P. M., Rus, V., Crossley, S. A., Graesser, A. C., and McNamara, D. S. (2008). Assessing forward-, reverse-, and average-entailer indices on natural language input from the intelligent tutoring system, istart. In *FLAIRS Conference*, pages 165–170.

[Nielsen et al., 2008] Nielsen, R. D., Ward, W., Martin, J. H., and Palmer, M. (2008). Annotating students' understanding of science concepts. In *LREC*.

[Okoye et al., 2013] Okoye, I., Bethard, S., and Sumner, T. (2013). Cu: Computational assessment of short free text answers-a tool for evaluating students' understanding.

[Zesch et al., 2013] Zesch, T., Levy, O., Gurevych, I., and Dagan, I. (2013). Ukp-biu: Similarity and entailment metrics for student response analysis.