INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Um sistema gerador de respostas enganadoras para questões de exame

JOÃO RUI MACEDO FONTES
Outubro de 2019

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

A misleading answer generation system for exam questions

João Rui Fontes

A dissertation submitted in partial fulfillment of the requirements for the degree of Master of Science, Specialisation Area of Information and Knowledge Systems

Supervisor: Nuno Escudeiro

Porto, October 13, 2019

Abstract

University professors are responsible for teaching and grading their students in each semester. Normally, in order to evaluate the students progress, professors create exams that are composed of questions regarding the subjects taught in the teaching period. Each year, professors need to develop new questions for their exams since students are free to discuss and register the correct answers to the various questions on prior exams. Professors want to be able to grade students based on their knowledge and not on their memorization skills. Each year, as discovered by our research, professors spend over roughtly 2:30 hours each year for a single course only on multiple answer questions sections. This solution will have at its core a misleading answer generator that would reduce the time and effort when creating a Fill Gap Type Questions through the merger of highly biased lexical model towards a specific subject with a generalist model. To help the most amount of professors with this task a web-server was implemented that served as an access to a exam creator interface with the misleading answer generator feature. To implement the misleading answer generator feature, several accessory programs had to be created as well as manually edditing textbooks pertaining to the question base topic. To evaluate the effectiveness of our implementation, several evaluation methods were proposed composed of objective measurements of the misleading answers generator, as well as subjective methods of evaluation by expert input. The development of the misleading answer suggestion function required us to build a lexical model composed from a highly biased corpus in a specific curricular subject. A highly biased model is probable to give good in-context misleading answers but their variance would most likely be limited. To counteract this the model was merged with a generalist model, in hopes of improving its overall performance. With the development of the custom lexical model and the server the professor can receive misleading answers suggestions to a newly formed question reducing the time spent on creating new exams questions each year to assess students' knowledge.

Keywords: Natural Language Processing (NLP), Golang, Automatic Question Generation (AQG), Neural Networks (NN)

Acknowledgement

I would like to thank professor Nuno Escudeiro for his help, wisdom and pacience in the development of this thesis. I would like to thank my parents for their unconditional support as well as Joana Elisa Alves for her conditional support. I also thank professor Ana Almeida, Fernanda Delindro and to all professors that contributed to the development of this thesis.

Contents

List of Figures

List of Tables

List of Acronyms

AQG	Automatic Question Generation.
CBOW	Continuous Bag-of-Words.
HTML	HyperText Markup Language.
HTTP	Hypertext Transfer Protocol.
ISEP	Instituto Superior de Engenharia do Porto.
MT	Machine Translation.
NER	Named Entity Recognition.
NLP	Natural Language Processing.
NN	Neural Network.
PDF	Portable Document Format.
POS	Part-Of-Speech Tagging.
QFD	Quality Function Deployment.
SG	Skip Gram.
SGD	Stochastic Gradient Descent.
SINFE	Sistemas de Informação Empresariais.
SRL	Semantic Role Labeling.
UI	User Interface.
UML	Universal Modeling Language.

Chapter 1

Introduction

This document presents the work plan and schedule for the Master's Thesis in Informatics Engineering entitled "A misleading answer generation system for exam questions" and developed in Instituto Superior de Engenharia do Porto under supervision of professor Nuno Escudeiro. This work addresses the field of Area of Information and Knowledge Systems. In this document it will be reported the project assignments, the schedule and project design and its main results. Finally it is important to mention that this work was conducted under the advise of an expert (Professor) who also represents the end-user.

1.1 Motivation

The act of learning in formal education is inherently linked to teaching. The formal way of education requires the teacher to set goals for their students to achieve in a set amount of time. The closer the student got to stipulated goals in the given period the better the grade the student should receive.

School teachers, college and university professors alike are responsible for teaching and grading their students in each semester. Tests or exams are the most common evaluating tools used in these institutions. Both tests and exams are composed of questions regarding the subjects the tutor (professor or teacher) taught in the teaching period. We can define many types of exam questions: *True or False statement questions (A.2), Fill Gap questions (A.3), Multiple Answer questions (A.4)*[1]*, Essay questions (A.5)* to name a few [1, 2]. These can be accompanied by images, code, schematics, equations or other resources. There are many ways one can present a question but no matter the type or format of the question posed the underlying objective is to challenge the students knowledge in a given subject.

Yearly, in each Curricular Unit's, new students enter, some students stay and other students graduate or pass. Students who passed or tried to pass the Curricular Unit's exam are free to discuss and register the correct answers to the various questions on that exam. They tend to form groups and share their knowledge about the exam questions with the newer students. With this knowledge, some students only make the effort of memorizing the questions and correct answers from latter exams. This creates a problem. Tutors want to grade students based on their knowledge of the Curricular Unit's various subjects and students want to maximize their grade on the Curricular Unit's exam with the least amount of effort.

In the end, the responsibility of correctly determining the students knowledge falls to the tutor. This struggle lead most tutors to create new questions to their exams *every single*

[1] Fill Gap Questions tend to fall on the category of Multiple Answer questions however, in this document we make this distinction.

year. In some cases, as we will analyze in Chapter 3, this translates to a workload of 30+ extra out of work hours in creating new multiple questions each year. With this in mind, in this thesis, we propose the development of a intelligent tool that can help tutors by reducing the workload on the creation of new multiple answer questions in exams.

1.2 Problem Statement

As disclosed in previous Section 1.1, in universities, it is the majority of the professorship, as will be disclosed by our questionnaire in Chapter 3, to create exams. There are many tasks involved in the creation of an exam. In a series of interviews with a university professor we were able to extract and document his exam creation process. A summary of the exam creation process is detailed in the last paragraph of Section 2.1.

As result of the interviews, the most emphasized functionality request for an exam creator assistant was an evaluator that could determine the degree or percentage of similarity in previous exam questions. There are many online websites such as the "Online exam builder" [2], "Easy Test Maker" [3], "Flex Quiz" [4] and "Test Moz" [5] that report can store, edit, export exam versions and other functions that help tutors with their work.

In the literature, in the most research projects, there are some with promising results in the field of education regarding Automatic Question Generation (AQG) topic. When asking the professor on his feelings towards a similar solution the response was clear. The professor claimed that editing a proposed exam question from his peers was a much harder job than one that he would create one himself from the start. However, the suggestion of misleading answers to multiple answer questions seemed a much better prospect. At worst the professor felt that it would not interfere with his work, at best it would save him some time.

1.3 Objective

The main goal of this project would be primarily to support professors in the creation of exams by reducing the effort and time spent on developing new questions. To achieve this goal, the work described in this report has the following major objectives:

Phase 1 - Inception:

- Analysis and comprehension of the exam creation process in Portuguese universities;
- Identify the end-user (costumer) main and technical requirements;
- Identify and analyze current applications and solutions related to the presented problem, by performing state-of-the-art research;

Phase 2 – Elaboration and Implementation:

- Research regarding machine learning and Neural Network algorithms;
- Plan the design, method of implementation and evaluation of the previously obtained requirements;

[2] https://www.onlineexambuilder.com/
[3] https://www.easytestmaker.com/
[4] https://www.flexiquiz.com/
[5] https://testmoz.com/

- Guarantee the access and constant communication to a database of exam question-s/answers;
- Development and implementation of a application to support professors in the creation of exams through a misleading answer generation system;

Phase 3 – Validation and Conclusion:

- Analysis and evaluation of the success and performance rate of developed program through validation tests next to university professors;
- Assess the accuracy of the created models;
- Project protocols and model tester setups to verify adequacy of range of subtopics present in given misleading answers.

1.4 Hypothesis

In recent years research shows that Neural Network (NN) perform better than other methods in the field of Natural Language Processing (NLP) as it will be described in Chapter 2. By creating a custom Lexical Model derieved from books within a particular discipline of learning using these methods we believe we are able to assist the professors work in exam creation by improving the time and effort spent in creating new multiple answer questions.

1.5 Document Structure

This document is composed by seven chapters. In this chapter it is given a summary of the context and motivation for problem's interpretation, and the methodology to address its solution. In Chapter 2, is presented a State-of-the-Art, a highlight the Problem Context as well as a deeper analysis on the Fill-Gap Question Type. Chapter 3 presents a Value Analysis for our program to identify the customer requirements. Chapter 4 and 5 describes the program, the information gathering and manipulation in the making of the misleading answer extraction. In Chapter 6 - Evaluation is described the process to analyze the performance and success of the program: testing hypotheses, evaluate metrics, and other proposed approaches to test the usability of the program. Finally, the main results and a overview of the project until this phase will be given in Chapter 7. Future work and further directions will also be discussed in this section.

Chapter 2

Technical Review

As highlighted in Chapter 1 we seek to find an answer to reduce the efforts of professors in creating exams through an intelligent technological solution. Towards this goal it is fundamental to research the most relevant topics, available solutions and technologies regarding the underlying subject. Also, the problem must be well studied to be well understood and explored. Therefore, in this chapter, we will be discussing the context of the problem and we will present the state-of-art.

2.1 Problem Context

In universities, professors in the end of the semester most often present an exam to their students in order to test their knowledge in each Curricular Unit. In an exam, each question is tailored to address the different topics of a particular Curricular Unit. In this thesis we defined four types of questions commonly found in exams. To reiterate we have: *True/False, Fill-Gap, Multiple Answer and Essay type questions.*[1]

The Essay type questions give the student the liberty to express himself but the correction of this type of question is harder to automatize than the others. By contrast True/False, Fill-Gap and Multiple Answer type questions don't give the liberty for the student to express themselves but are much easier to correct automatically [3].

Here, we pair Fill-Gap questions and Multiple Answer questions due to their similar structure. We divide these two types in the following parts: *the query, the answer and the false/misleading answers*. In these types of questions the professor is the sole editor of each part. In order to highlight the students knowledge the correct answer has to be "disguised" by false/misleading answers. As evidenced by our questionnaire in Chapter 3 the reported difficulty in making these questions tends to be relatively low but the amount of new questions that teachers make each year can be up to 30 making the proposed solution in this thesis valuable. This creates the opportunity of creating an intelligent system that helps the professor in the creation of these types of questions.

When interviewing a university professor, about his question creation process, he first checks his exam planed structure for topics to evaluate. After picking a topic he first creates the query for that particular topic with the respective correct answer followed by the misleading answers. The professor knows the underlying difficulties of his students therefore the exam questions created pose a test to the students knowledge. The professor sometimes consults his exam question database to reuse questions from latter exams. When asked specifically if a AQG system seemed like a good asset to consider as a functionality for a exam assistant

[1] Examples present in AppendixA

the response was negative. The reasoning behind this was since the professor knew the topics his students struggled with most it was easy for him to come up with a new question. When asked directly if he perceived a misleading question generator to Multiple Answer Questions to be an interesting and useful functionality the response was positive. This lead to the focus of misleading question generator as the main goal of this thesis.

2.2 Fill-Gap Questions Analysis

We classify Fill-Gap Type Questions as questions with a gap to fill. These gaps are usually underlined by an indeterminate number of "_" characters anywere on query [4]. As seen on Figure 2.1 there can be different variations within the same Fill-Gap Type questions such as different numbers of gaps, number of words per gap and different semmantic roles within the phrase.

Questions 2 and 4 contain more than one gap. In the possible answer section of these questions[2] the delimiter that seperates the answers for the respective gaps is in this example the character ",". In these variations, the character seperator must first be identified, as it could change depending on the users choice. The character seperator should also be consistent throughout all possible answers while also matching the number of gaps with the number of answers.

The misleading answer suggestion will be heavily influenced by the correct answer. Questions 1, 2 and 3 are cleary different from each other. Even though the concept is the same, questions 1 and 2 answers are grammar objects representing a concept while question 3 is the definition of the object present in the query. Question 4 is simmilar to question 2, having both more than one gap to fill, however the correct answers are of differing semantical values.

These variations within Fill Gap Type Questions were accounted for in the programs' design, as described on Chapter 4.

2.3 State of the Art

In this section it will be described a brief state-of-the-art and insight about language processing and solutions/technologies to achieve a program for misleading answer. Here we define Language Processing as the scientific field in computer engineering with its focus on being able somehow to understand and or assist humans through the use of text. The study of this topic and its understanding will be of great importance for the development of this project. Therefore, here will be stated some of the most relevant systems and technologies in this area.

2.3.1 Information Gathering

The first need for Language Processing came with the purpose of facilitating communication between people, in other words, translation between languages [5]. Machine Translation (MT) is a process through which a computer is able to translate words from different languages. The advantages of MT spurred international interest and gave rise to the first

[2] the lines containing the a), b), c) etc.

1) _____ is (are) data that have been organized to have meaning and value to a recipient.

a) Insights
b) Information
c) Knowledge
d) Experience
e) Wisdom

Answer: b

2) As a manager in your company, you expect to retrieve _____ from operational systems and present it as _____ to your boss, who will react based on his _____.

a) knowledge, information, data
b) information, data, knowledge
c) data, information, knowledge
d) data, information, experience
e) information, experience, wisdom

Answer: c

3) A protocol is _____.

a) A device that handles the switching of voice and data in a local area network.
b) A standard set of rules and procedures for the control of communications in a network.
c) A communications service for the connection of devices in a local area network.
d) The main communications channel in a wide-area network.
e) Synonymous with network interface card.

Answer: b

4) For most companies, the Web _____ the threat that new competitors will enter the market by _____ traditional barriers to entry.

a) decreases, increasing
b) increases, decreasing
c) increases, increasing
d) decreases, decreasing

Answer: b

Figure 2.1: Different Variations of Fill-Gap Type Questions

computer applications using thesaurus for Language Translations [6]. With these new applications came new challenges. The possibility of making summaries of a complex text using the technology developed for MT was thought to be in reach. This led to the use of fixed rule systems that were capable to do summaries of a previously existent text [7, 8].

Nonetheless, the natural human language is indeed very complex and eclectic. In this way, to communicate in the same language as humans our programs need to take into account the many differences, vocabulary, culture context and dimensions of natural language. This lead to the division of NLP in multiple tasks: Part-Of-Speech Tagging (POS), Chunking, Named Entity Recognition (NER), Semantic Role Labeling (SRL), Language Models and Semantically Related Words ("Synonyms") [9]. These tasks are related to the different problems, that the scientific community came to independently tackle when, aiming towards NLP. In the last article cited Collobert and Weston [9] however it was proven that in the

field of NLP these problems were best tackled together using an algorithm inspired by the machinations present in our brains.

As science and computational power progressed, old, theoretical algorithms based on the neuron became applicable [10, 11]. NN based systems, in essence, are very simple. They do however need enormous amounts of relevant data and processing power in order to be useful. In the context of NLP, Neural Network based systems proved to be more efficient and accurate when compared to rule based systems in solving natural language processing related problems [9, 12–14]. In the last decade, NN based systems are used applications such as automatic identity recognition [13], fraud/plagiarism detection [15] and next word prediction [16]. In Collobert et al. Collobert et al. [17] a summary of the tasks mentioned above was presented and it has what the author found to be the best algorithmic implementations related to NLP problem. The paper also makes a comparison between different algorithms in terms of how good are they at finding synonyms to specific types of words. This feature, in particular, is considered to be very helpful in the proposed thesis work.

In the literature, AQG is referred to as the process in which questions are automatically generated by algorithms. Moreover in Fill-Gap Question Generation the underlying method to generate its questions is: *1. selecting a sentence, 2. identifying the part to use as gap 3. creating the misleading answers* [18–21]. All of these use SRL algorithms to know the role of the part the gap word was picked. This way the algorithm will only pick words with the same role to create misleading answers.

When processing text documents, such as wikipedia, books or Portable Document Format (PDF) files it is expected to find in the stream punctuation marks, references, titles, page numbers and so on. To optimize the relations between words to be calculated by the algorithms, pre-processing, in other words, removal or editing of the concepts mentioned before is implemented to the raw documents. As mentioned in [22], data preparation reduces a dataset, which can significantly improve the efficiency of data mining. To achieve our goal in this thesis we intend to pre-process highly specific content about a specific subject creating a corpora of text of high quality which will hopefully lead to acceptable misleading answers suggestions.

Neural Networks (NN) work simmilarly to brain matter as the name implies. In [10], the authors envisioned a mathematical structure that would come to be the basis of what NN came to be today. In computing NN can be described as a group of functions that use, at its core, a chain of linked mathematical algorithms that can be trainned to provide a close optimum solution to a pre-determined goal. These algorithms are no different than any other generic function, however it is its simplicity that makes algorithm powerfull as it can be rigged to provide close to optimal solutions to any particular problem. As it was mentioned in the introduction of this chapter, in order to NN to be effective must be thouroughly trainned with large ammounts of raw data. If large ammounts of data for particular problem are hard to come by, information extracted through the use of a NN probably wont garner the best results.

In NLP systems, most use the NNs. In particular, the algorithm Word2Vec used in many studies found in the literature are used to predict words within a particular context. The Word2Vec algorithm is able to work using either Continuous Bag-of-Words (CBOW) or Skip Gram (SG). In both algorithms each word will be translated to a vector. The vector's angle or direction in comparison to other words will determine their simmilarity. The lower the angle between words the simmilar they are to each other.

To calculate the vector each word will have the algorithm uses a Stochastic Gradient Descent (SGD). The SGD is an iterative process that in this case, calculates the word's vector angle by aproximation to it's minimum value, meaning the angle that best represents the word in it's space determined by the data given.

CBOW and SG are two modes of building a vector space model (Lexical Model). CBOW build a vector space model taking a sequence of words in a sentence to a target word chosen by the algorithm. SG takes the target word and builds a vector space with the next words in the sequence effectively creating a model much more complex requiring more memory space to store.

2.3.2 Relevant Technology

Go, or Golang, is a programming language designed at Google and released to the public in 2009 by the prominet programmers Robert Griesemer, Rob Pike and Ken Thompson [23, 24]. Go was created with intent of being fast and improve programming productivity. Fast at compiling, at start-up with easy to understand and implement concurrent functions and inbedded networking toolkit and information sharing with a web-site dedicated to search Golang community created packages as well as a maven style importing service.

In the following Chapters we will be detailing functions and variables in this programming language. As such, in order to better understand the details of the elements presented a brief summary and explation to what the Golang language entails is presented.

The Golang data structure most simmilar to Java Classes are structs. Golang differenciates functions from methods, being functions "class free" in java terms and methods bound to structs. Encapsulation, that is the restriction of access, is delimited to objects outside the objects package and is defined by case (upper case being public, lower case private) of the first letter of either method or variable in Golang. Golang calls its parallel processing "packets" *goroutines*. The authors gave it a different name from threads, coroutines and processes since goroutines hide many complexities of thread creation and managment while also being light in stack space allocation. To start a goroutine for a function the word "go" followed by the function name is the only code needed ("go list.Sort()" for example). Communication between goroutines can be done using channels, that can act as semaphores or structures provided they are appropriatly implemented with concurrent access safe strutures.

The implementation of the server side program as well as all of the auxilliary functions mentioned in Chapter 5 exept for the model_creator was written in Golang due to the properties mentioned above.

Chapter 3

Value Analysis

In order to stipulate the value of the suggested system and search for further functionalities for the project, a questionnaire was created and distributed to several university professors of different curricular subjects and scientific fields. We gathered 59 answers in total. The anonymity was guaranteed in the questionnaire process. Some professors chose to answer the questionnaire with the interviewer present in order to clarify possible doubts that would come up during the filling process. It is important to note that this might lead to a bias on the answers given particularly on **Q6** described bellow. Nonetheless the clarifications requested were answered carefully as not to influence the professors answer to the posed question.

The questionnaire was divided in sections. Depending on the answers given in the first couple of steps **Step3** was not shown. The questionnaire followed the state diagram in Figure 3.1 and the questions asked were the following (translated from Portuguese):

Step# Number of the step in the Questionnaire

Q1 Is the professor responsible for creating exams?

Q2 Do the exams usually contain Multiple Answer Questions?

Q3 On average, how many Multiple Answer Questions do you use on exams?

Q4 On average, how many Multiple Answer Questions do you reuse from former exams?

Q5 Usually, on average, how much time does the professor take creating a new Multiple Answer Question?

Q6 How hard is it to come up with misleading answers to exam questions?

Premise Imagine that you possess a program that when typing a new multiple question query and correct answer it automatically gave a misleading answer suggestion.

Q7 How useful do you consider this function?

Q8 Keeping in mind the premise above, what additional/complementary functionalities would you suggest to enhance its value?

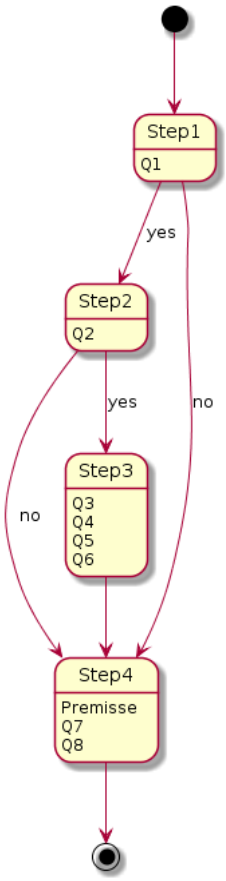

Figure 3.1: Questionnaire Question/Answer State Diagram

3.1 Questionnaire Answer Analysis

From the 59 answers received by professors, 54 were responsible for creating exams and 30 of them usually had a Multiple Answer Question section. Some answers had to be pruned since they contained ambiguous answers [1] and others adjusted[2] in an honest attempt to represent the professors input. In the end, 25 questionnaires in total were valid for analysis.

In the following paragraphs we will present the results taken from the questions in **Step3** and **Step4** making brief analysis where we found it most relevant.

By taking the average number of multiple answer questions teachers reportedly use in exams (**Q3**) and subtracting the average number teachers use multiple answer questions from later exams (**Q4**) we get the "Average New Multiple Answer Questions by Exam" metric (**M1**). This as shown on Figure 3.2.

[1] Example: In **Q4** the answer "Not sure" was pruned
[2] Example: The answer "4 to 5 minutes" to question **Q5** was changed to "4,5" minutes

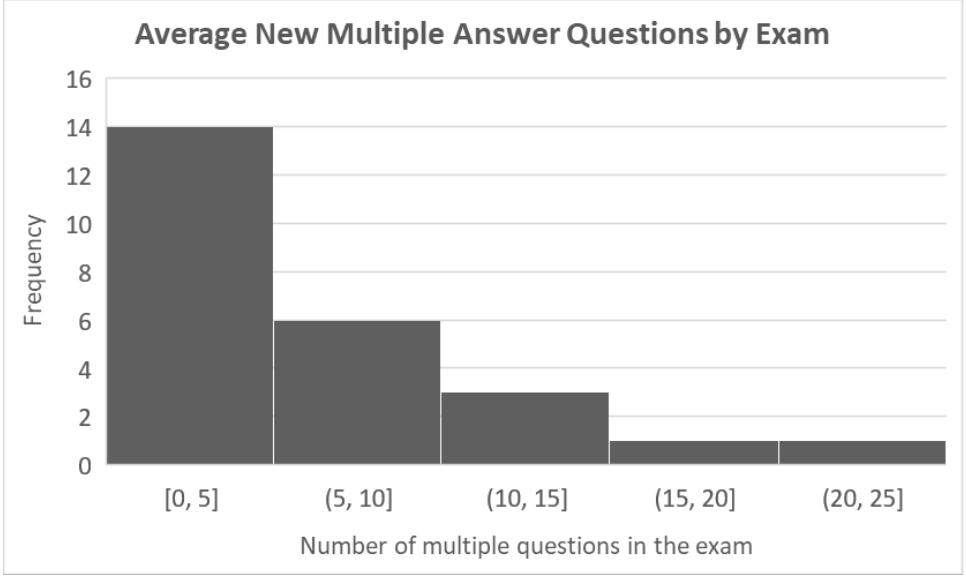

Figure 3.2: Average new Multiple Answer questions created for exams by professors (**M1**)

In **Q5** we asked for the average time the professor took when creating a new Multiple Answer Question. The question posed was not given a time unit to standardize the answers. In hindsight this could have been prevented, however most professors marked their answers with the "minutes" unit in their answers. Therefore the resulting answers to **Q5** where standardized with the measurement unit "average minutes per new multiple answer question" Figure 3.3.

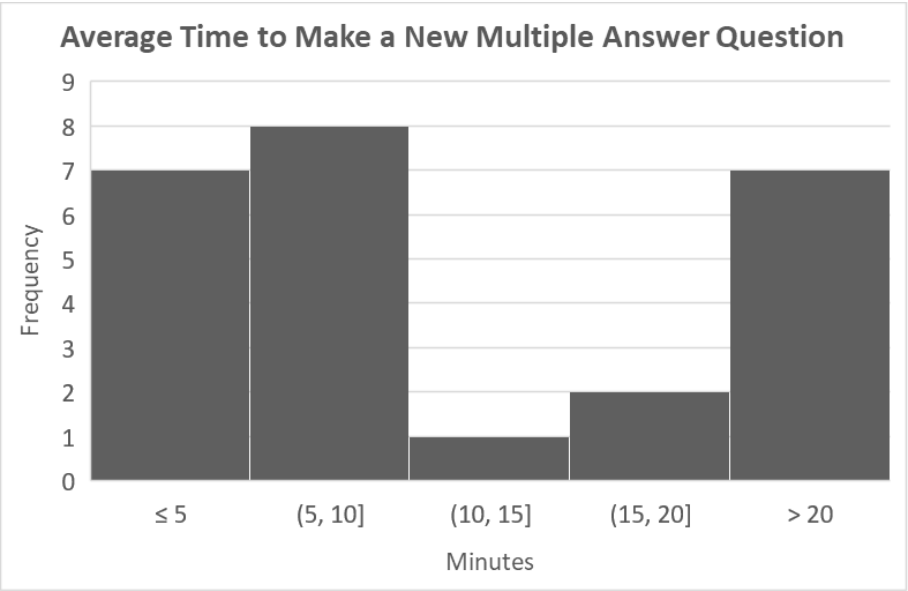

Figure 3.3: Histogram of the average time professors take making new Multiple Answer Questions (**Q5**)

By multiplying **M1** with the later (**Q5**) we get the "Total average minutes per exam on new multiple answer questions" (**M2**) as shown in Figure 3.4. If we consider that professors

have to deliver at the maximum three exams per course (normal exams, recourse and special season) we can calculate the "Total average minutes per course on new multiple answer questions in exams" (**M3**) by multiplying **M2** by three as shown in Figure 3.5.

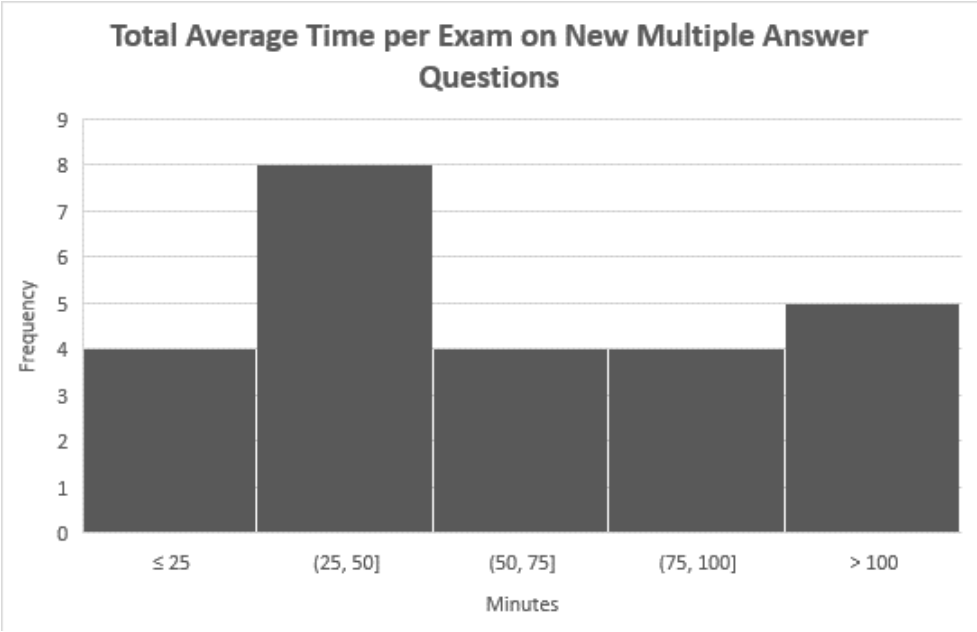

Figure 3.4: Histogram of the time spent by professors in each exam creating new multiple answer questions in minutes (**M2**)

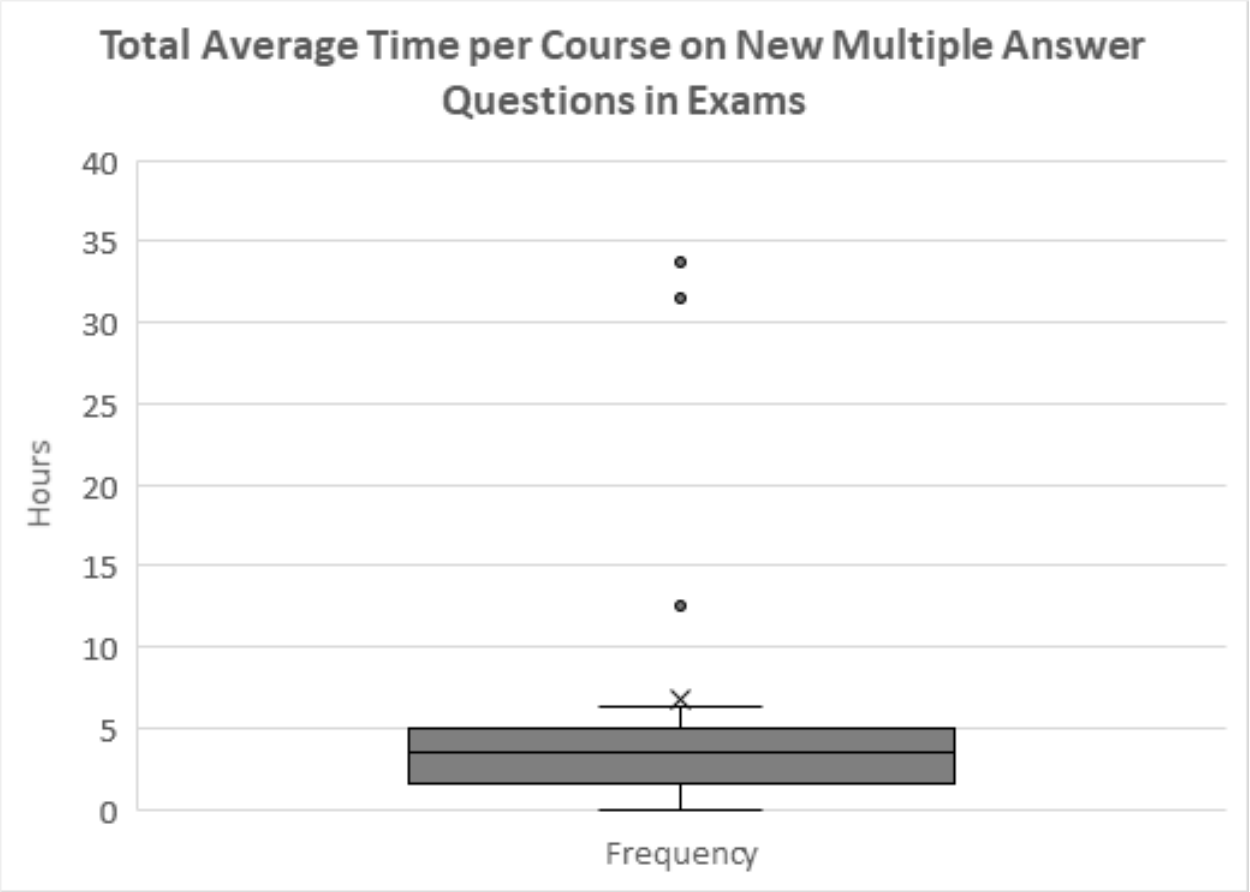

Figure 3.5: Boxplot of the time spent by professors in each course creating new multiple answer questions (**M3**)

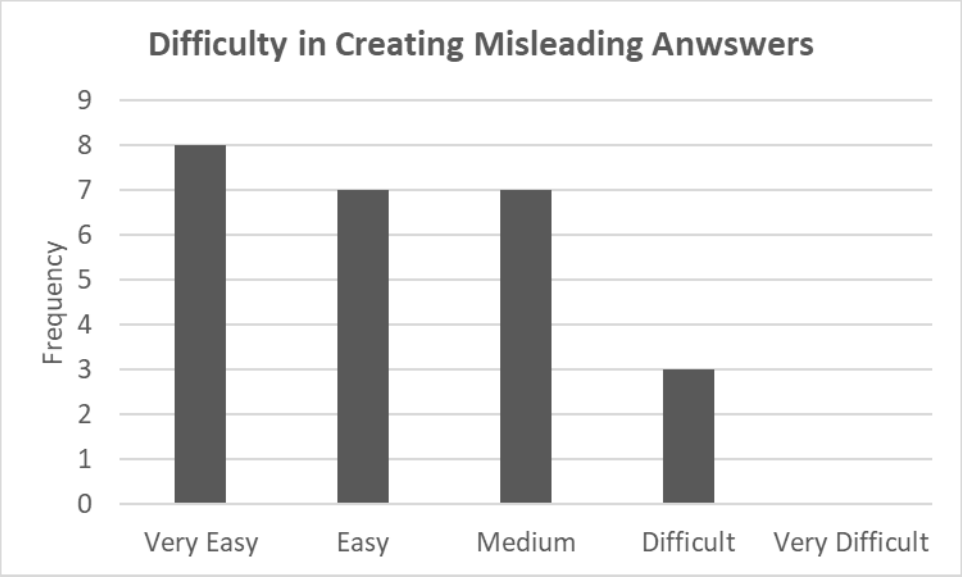

Figure 3.6: Bar graph of the reported difficulty in creating Multiple Answer Questions (**Q6**)

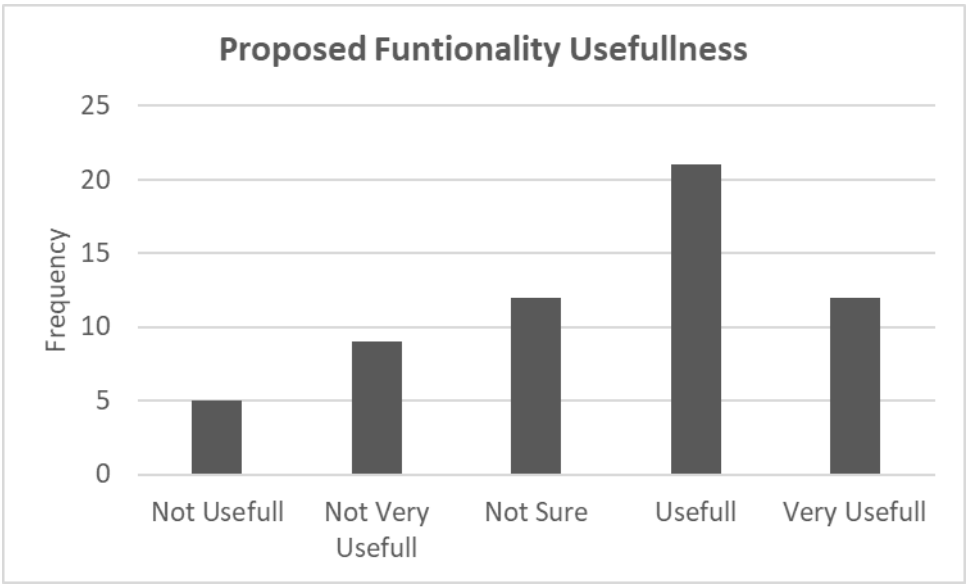

Figure 3.7: Bar graph of the reported perceived usefulness of a Misleading Answer Generator to Multiple Answer Questions (**Q7**)

Looking at Figure 3.3 we can see clear dispersion between number of minutes a teacher spends on creating a new multiple answer question. Half of them spend an average of about five minutes creating a new multiple answer question where as the other half spends more than double the amount of time. However when looking at Figure 3.4 we can see that the overall instances average out when taking into account the number of new multiple answer questions used in exams. We can interpret this by saying that effort made by teachers when creating new multiple answer questions is best analyzed when taking into account the number of questions they produce in each exam.

In Figure3.5 we have a boxplot depicting the total average hours a professor normally takes in creating new multiple answer question with a calculated average of 2,78 and a standard deviation of 1,57. These values can be interpreted as the standard values for the number of hours spent by teachers creating multiple answer questions each year for a single course.

As the **Q6** suggests, the purpose of this question was to know the inherent difficulty of creating misleading answers. In Figure 3.6 we can infer that professors do not find it hard to create new misleading answers to new multiple answer questions. However, this contrasts with the answers given to **Q7**. In Figure 3.7 is represented a clear support for the proposed main function of the application. This only comes to reinforce the calculations done in the previous paragraph. Although it is easy to create the new multiple answer questions for exams the sheer amount of time taken for this task each semester makes the case for the proposed functionality.

3.2 Quality Function Deployment

In **Q8** we asked what additional/complementary functionalities should the program have. **Q8** was presented as an open answer question in the questionnaire, therefore in order to evaluate and group these answers, they were interpreted, organized and their frequency registered as seen in Table 3.1. In order to turn these requisites into functionalities that

we can analytically deduce their relative priorities we will use a summarized version of the Quality Function Deployment (QFD) model diagram[25].

The aim of the quality function deployment method is to set targets to be achieved for the engineering characteristics of a product, in this particular case, an informatics software program, such that it satisfies customer requirements. This method starts with the identification of the customers and their own views of requirements and desired product attributes. However, not all the identified product attributes will be equally important to customers. Thus, we will need to perceive and establish the relative importance of those attributes by allocate relative weights to the set of customer-specified program attributes (Customer Priority). Moreover, customers cannot specify their requirements in terms of the program's engineering characteristics, so we defined Technical Requirements followed its directed relationship which will not all be of equal value – some characteristics will have a strong influence on some attributes, while other characteristics might only have a weak influence. This relationship was done by checking through the cells of the QFD matrix.

The top of the existing matrix (triangular shaped roof form) enables a systematic check of the interactions between the technical requirements, and whether these interactions are negative or positive. For example, as seen of Figure 3.8 Interactive Text Interface as a positive correlation with both Grading/Evaluation Mode and Staging Exam Section. Finally, Importance Rating is assessed through an absolute weight point of view, following this formula:

$$W_j = \sum_{i=1}^{n} R_{ij} c_i \tag{3.1}$$

The ratings are calculated by summing down each column the product of the customer importance rating and the value assigned to the correlation symbol. In the formula, W is the Importance Rating value, R is the relationship value assigned and c is the customer priority/importance.

Table 3.1: Extracted Customer Requirements and Frequencies

Customer Requirements	**Frequency**
Automatic answer grading	5
Fraud Detection	1
Definition of number of misleading answers	1
Degree of similarity between previous questions	4
Editable misleading answers	2
Exam Version Generation	1
Relevant misleading answers	2
Regulation of similarity when compared to the correct answer	3
Suggestion of questions from previous exams when given a "theme"[3]	1

When analyzing the Customer Requirements we then suggested the following Technical Requirements presented on Table 3.1.

[3]this should be accompanied by the date the question was last used during an exam

Table 3.2: Suggested Technical Requirements

Technical Requirements	**Details**
Interactive Text Interface	All text boxes detaining exam relevant information should be easily editable
Advanced Question Options	A settings' display pertaining different options relative to the activity context at hand
Grading/Evaluation Mode	A grading/evaluation stage that displays the students answers and results
Staging Exam Section	An interactive overview stage of an exam where the teacher can edit questions and publish
Focus Detection Program	An automatic facial focus expression to combat fraud/copy detection from other students

In Figure3.8 we take the Costumer Requirements from Table3.1 and the Technical Requirements from Table3.2 and combine them to implement a QFD model diagram. By filling the relationship matrix, with the most appropriate tags, we can now calculate the Importance Rating.

Starting with the **Interactive Text Interface** requirement, we classified it with the strongest relationship with the **Editable misleading answers** requirement. The main purpose of the **Interactive Text Interface** is to be a tool which the professor interact with the underlying program. This tool not only gives him the ability to edit the information about the question it but also should give him feedback on the how the question relates to the other questions already in the database. This last functionality description is the justification for the strong relationship with the **Degree of similarity between previous questions** customer requirement. When manually altering the misleading questions through the **Interactive Text Interface** the user will be also guiding the program by giving him different misleading questions giving this technical requirement a fair relationship with the **Regulation of similarity** when compared to the correct answer requirement.

The **Advanced Question Options** does not have a "Strongest" relationship with none of the Customer Requirements due to its nature. The **Advanced Question Options** is envisioned as a context sensitive settings tool that gives the professor options when defining the number of misleading answers, regulating the misleading answers/correct answers similarity thus leading to more relevant misleading answers suggestion or giving the option of suggesting past exam questions when composing an exam.

The **Grading/Evaluation Mode** technical requirement is what enables the **Automatic answer grading** customer requirement. This functionality proceeds the **Staging Exam Section**. After the teacher plans and implements his exam accordingly he will distribute these exams to its students. When the students finish their exams the teacher is interested in an function that automatically grades the students' multiple answer questions.

The **Staging Exam Section** is where the professor has an overview of the whole exam. This function has the most synergy with the suggestion of previous questions. Since the question already exists, the professors' concern in this context should only be the placement of the question in the overall structure of the exam. The **Staging Exam Section** should allow the creation of exam versions as per Customer Requirements. In this overview the interface should also give the degree of similarity between the questions already on the database but also within the exam itself.

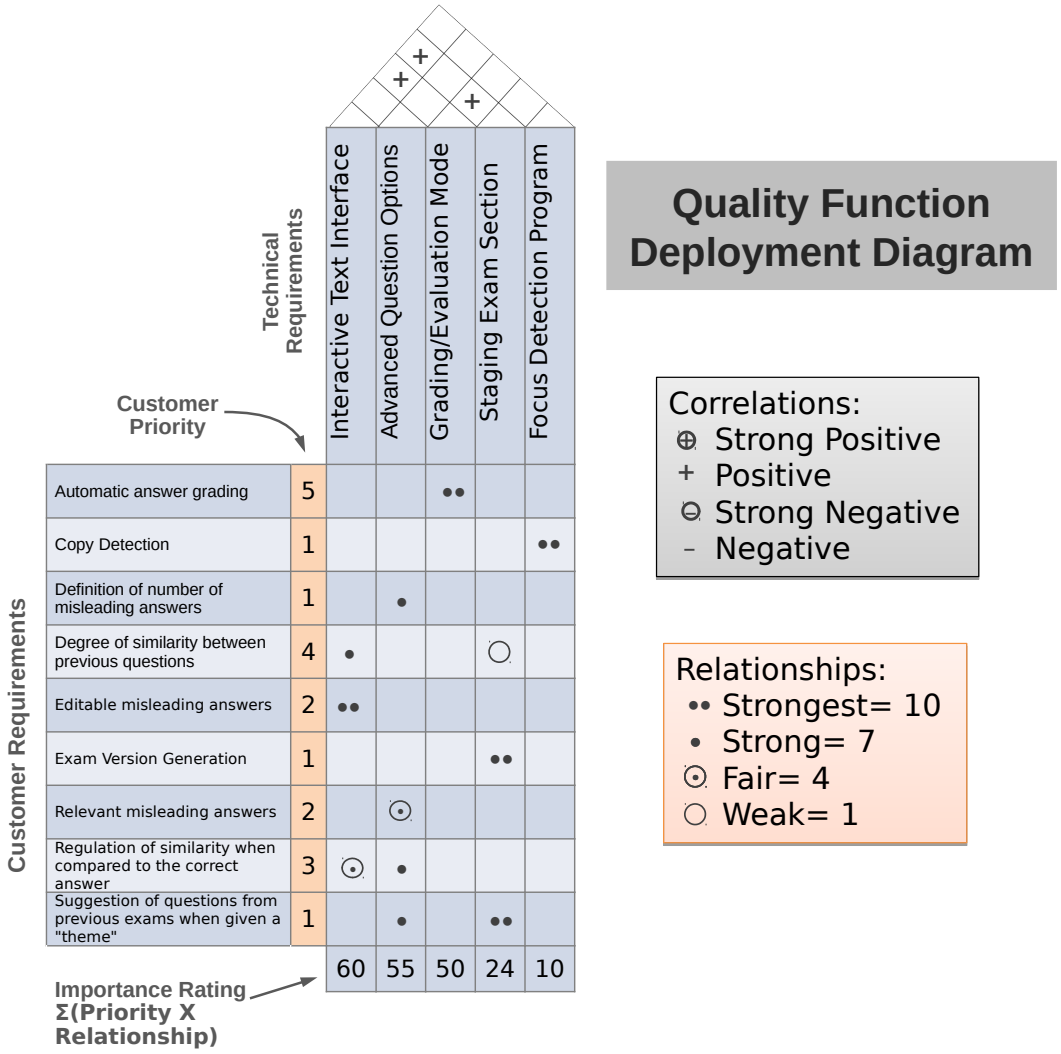

Figure 3.8: Quality Function Deployment Diagram depicting the priority of each Technical Requirement

In order cope with the **Fraud Detection** customer requirement we suggest a Focus Detection Program. As said before, the purpose of exams is to test the students knowledge on a certain topic. To make sure that it is the students' own wits answering the exam questions and ensuring that he is not being assisted by a foreign method we propose a program that would be capable of detecting such forms of assistance. This would most likely require the combination of eye tracking software already freely available on the internet[26, 27], monitor manipulation software and operating system verification to deter virtual machine users who would try to cheat the system and integration with some other sensors to deter other possible cheating methods. When taking the exam the program would most likely have to make a presence test. This test would order the student to make a random amount of actions before the test in order to establish that the camera is actually detecting a student and its not being tricked with an movie running in front of the camera for example. This method to some would seem cumbersome and other methods of tackling cheating in universities are being discussed[28]. However due to the scope of this project, this functionality will not be implemented.

Table 3.3: Technical Requirement Priority Assessment

Technical Requirements	**Importance Rating**	**Priority**
Interactive Text Interface	60	1
Advanced Question Options	55	2
Grading/Evaluation Mode	50	3
Staging Exam Section	24	4
Focus Detection Algorithm	10	5

By taking the calculated Importance Rating and ordering the results in a descending fashion we get the relative Priorities of the aforementioned technical requirements as seen on Table 3.3. Looking at Table 3.3 it is clear that an Interactive Text Interface is most important feature from the pooling. As such, in this thesis special care will be taken in consideration to provide the user with a reliable and Intuitive Text Interface.

Chapter 4

Implementation

This project's end goal is to deliver an application that can aid university professors in their work by facilitating and reducing the time spent in the development of new exam questions. In the second paragraph of Subsection 1.2 we reference programs that acomplish this goal by giving professors a platform that can easily stage and create different versions for exams, create, edit and search for older exam questions and many other functions. The application not only gives structural support to the utilities mentioned but also is capable of holding crucial data essencial for the the retrival of misleading answers to fill-gap exam questions. In this chapter we will present the underlying support structure the main thesis function, starting from the analysis phase, followed by the design and implementation.

4.1 Analysis

To grasp a better understanding of the problem domain we have interviewed university professors, the final users of our application, and also school teachers from the linguistic area for an informed overview of the grammar realations between words in Portuguese writing. The domain model in Figure 4.1 depicts the underlying concepts and their relationships to each other and a flowchart (Figure B.1) detailing the steps needed for the program to suggest Misleading Answers to Fill-Gap type Questions.

4.1.1 Data Modeling

The domain model shown in Figure 4.1 depicts the core concepts implemented in the program. The color range of this figure was chosen to represent the following criteria:

Blue Envisioned class structures (Java Classes)

Purple Exam specific structures

Yellow The Question structure, key to either the program and the professors work.

Red Enumeration/Description structures.

Green Relevant attributes.

White Support data.

Orange Lexical Model specific structures/data.

The arrows in Figure 4.1 represent the relations between the concepts and follow the Universal Modeling Language (UML) standard.

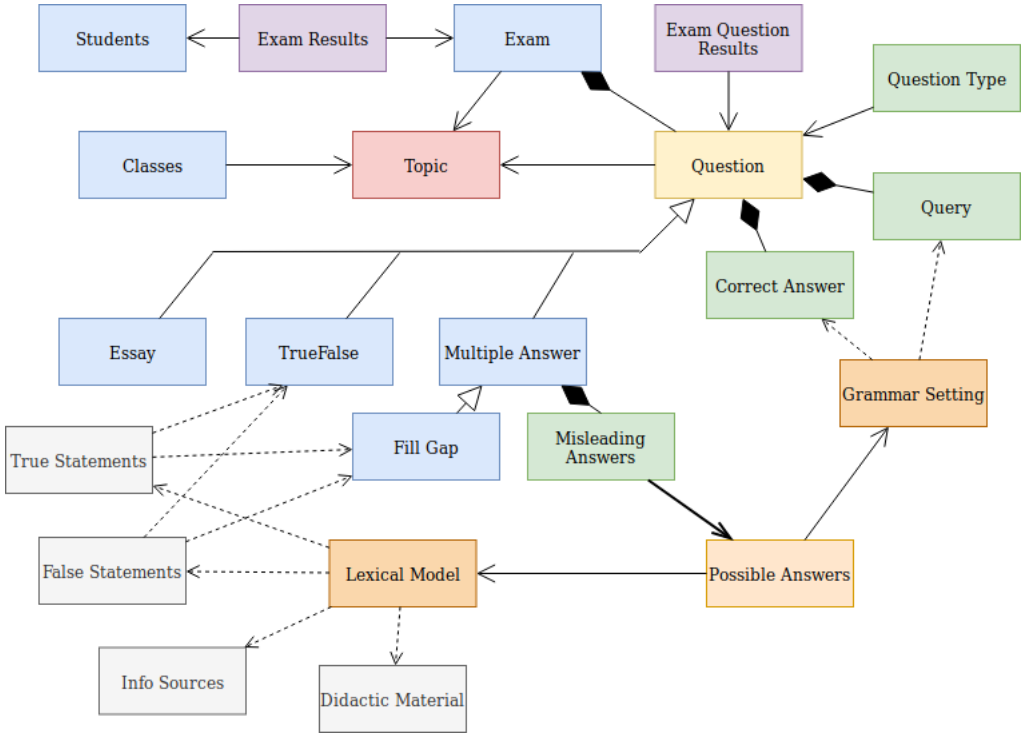

Figure 4.1: Domain model of the relevant concepts

The Question structure is key in either the program structure and in the professors work. A Question is composed of a Correct Answer and a Query both self descriptive of what information they represent. The Question Type is an attribute of the Question. This attribute is important since with it becomes easier to discern the inherent question type within the program. Questions have an inherent Topic. The Topic domain represents a university course theme or class topic. As the name implies, the Class domain represents the class/study session bettween professor and student. Both Class and Question are linked to the Topic domain for this reason. Questions can be of the following types: Essay, TrueFalse and Multiple Answer. Multiple Answer type questions have misleading answers and some will be of the sub-type Fill Gap. With the TrueFalse and Fill Gap type questions we are able to produce True and False Statements. True Statements are obtained from TrueFalse questions from its query if its answer is true. False Statements are obtained from the same source with the opposite answer. True Statements are obtained from Fill Gap questions by placing the correct answer in the matching gap in its query. False Statements are obtained from Fill Gap questions by filling its misleading answers in the gap.

Individually True Statements, False Statements, Info Sources and Didactic Material are necessary to create a Lexical Model. A Lexical Model, in this thesis context, is a binary file created from the elements previously mentioned using an algoritm that gives Possible Answers for new Misleading Answers. Possible Answers ganered via the Lexical Model can be filtered using the Grammar Setting. A Grammar Setting is a combined set of grammatical parameters given to every word in a specific sentence. Extracting the Grammar Setting from the Correct Answer and the Query allows the program to filter Possible Answers giving the user a better batch of possible Misleading Answers.

4.1.2 Misleading Extraction Flowchart

Figure B.1 is a Activity Diagram depicting the steps towards the attainment of Misleading Anwsers from a newly created Fill-Gap type Question.

In Figure B.1, starting from the top, after the professor makes the request for suggestions of Misleading Answers, the program retrieves the following pieces of information: the newly created Query (1), the corresponding Correct Answer (2) and the Topic (3) associated. This information is then forwarded from the Browse to the Server throught a Question Packet (QP). A Question Packet is the name given to the structure in which the Browser and the Server will base their communications on for future ease of use. This information is then recieved in the Server, identified on the flowchart by the Misleading Answer Suggestion group. Here the information is shared to the Information Gathering and the Candidate Misleading Answer Collection group. The Information Gathering group, like the name suggests is responsible for gathering more information to obtain aditional Misleading Answers.

In the Information Gathering group the leftmost sequence of activities focus on using the union bettwen the Answer (2) and the gap present on the Fill-Gap Query (2) in order to obtain what we will call a True Statement (TS). As stated in the subsecion above the program will have a collection of True and False Statements. Using word by word comparisson, we can then skim the top N closest True Statments (CTS) to our initial True Statement (TS) from the same Topic (T). Closest True Statements (CTS) gathered are linked to their original Questions. Should these Questions be linked to other corresponding False Statments (CFS) they will then be added to our pool of Near Statements (CTS and CFS).

Looking back at Figure B.1 on the same group we have the 'Identify the "Grammar Setting" (GS) with 1 and 2'. Expanding on the previous subsection, a Grammar Setting is a combined set of grammatical parameters given to every word in a specific sentence here obtained by the Query (1) and the Correct Answer (2) using Semantic Role Labeling and Part-of-Speach Tagging techniques.

Using the identified Grammar Setting (GS) we then select the words with the same Grammar Setting from the pool of collected Near Statements (CFS and CTS) as stated on the leftmost activity of the Candidate Misleading Answer Collection group. The alternate activity of the same group represents the method of obtaining Misleading Answer candidatades using a thesaurus and the Correct Answer (2).

After aggregating bought collections we then prune repeating answers and those equal to the original Answer (2). Using a Lexical Model, we then calculate the level of proximity of the words collected giving us an ordered list that we can then present to the professor.

4.2 Design

With a frame and road map of how the program should be built we could proceed to the next phase. In this section we present the database structure derived from the Domain Model, the technologies used chosen to be implemented and the frontend design.

4.2.1 Data Model

To be able to store exam information and support the main functions of the program the following database model structure was designed in the fashion presented in Figure B.2.

On the right side of the Figure B.2 exept for True_Statements and False_Statments tables that pertain to the information covered in Subsection 4.1.2 are the tables that are most critical to the professor's work. The Exam, Exam_Structure, Student and Exam_Result nomenclature directly correspond to their pourpose as well as their fields. Not surprisingly the Question table holds the professors questions. The "query", "answer" hold their respective information and the "q_type" is an integer that internally corresponds to the respective Questions' Question Type. The "created" field is the Questions' creation date and the "last_used" field is a reference to the last exam that the Question was used.

As discovered by the interviews to professors about their work method it is almost essencial to get information about the questions performance after the exams' correction. In particular, the percentage of students that answered correctly to the question, a metric represented by the "correct_percentage" field, and the discrimination index that roughly translates to the correlation between good students and correct answer selection represented as the "discrimination_index" field.

The pourpose of the Raw_Misleading_Answer table is to keep the Misleading Answers to the Multiple Answer type questions. Teachers every so often use a type of answer that does not pertain to the same context of the answer. "Neither of the above" or simmilar is an example of these "off context" answers in Multiple Answer Questions. Keeping in mind the main objective of this thesis, these answers, although possibly misleading, if not distinguished from "in context" answers would be used by the algorithm creating an unwanted bias towards these answers. These answers however are still relevant, therefore the reason for the name Raw_Misleading_Answer and the FG_Misleading_Field_Relations described further bellow.

As described in Subsection 4.1.2, True Statements and False Statements are generated from Fill-Gap type questions that are then stored in the bottomost left tables with the matching names in Figure B.1.

At the top left corner of Figure B.2 we find the tables most associated with this thesis objective. These tables serve to keep a record of the Misleading Answers suggested by the program and the answers kept or used in Fill-Gap Type Questions. In this way it becomes possible to continually improve the suggestion of Misleading Answers due to the professors feedback on either accepted answers and the discarted ones. Not only that, but it can also be used to add Possible Answers to the pool given by the Lexical Model and/or adjusting the order the answers are presented to the professor. This implicit relevance feedback is a key input to learn and improve the accurac of our application.

Starting with the FG_Misleading_Field_Relations, this table serves to keep a record of the Fill-Gap Misleading Answers as well as their Grammar Setting and Topic. Since every Question has a Topic and the delivery of good Misleading Answers depends on its context, or in this case the Topic the FG_Misleading_Field_Relations table is connected not to the Topic and Question tables individually but to a Question_Topic table that ensures the mentioned dependency.

Rembembering the analysis done to the different variations of Fill-Gap Questions in Section 2.2 we know that some Fill-Gap Questions have more than one gap to fill. With this in mind, looking at the fields in the FG_Misleading_Field_Relations table from the top we have "gap_order" that corresponds to the gap number in the Fill-Gap Query, "ma_id" that corresponds to the de-facto misleading answer in the question and the "word_order" that corresponds to the word order should the misleading answer have more than one word. The other

fields are foreign keys to the Question_Topic, Grammar_Setting and FG_Used_Words tables.

The Generated_Misleading_Answers table keeps the record of the answers generated and the order they were ranked by the Lexical Model in relation to the Fill-Gap Question. It also stores whether the answers were accepted by the teacher, whether the answers were edited and the original answer if the prior case is true. The way it relates to a Fill-Gap Question is through the FG_Misleading_Field_Relations table. Data is stored in this table following the entry of a new Fill-Gap Question with an misleading answer requisition. Regardless of acceptance or editing of the answers sugested the Misleading Answers to the Question are saved on the FG_Misleading_Field_Relations table. Either edited, accepted or not accepted suggestions are stored in the Generated_Misleading_Answers table through the "accepted" field flag, the "edited" field flag, the "original_answer" field that saves the unedited answer and the "generated_misleading_answer" that saves the de-facto suggestion no matter the conditions prior.

Left of the FG_Misleading_Field_Relations table we have the Grammar_Setting table and FG_Used_Words that keep the combined SRL and POS index of a question and an index of the words used in the Misleading Answers respectively.

Finnaly, towards the bottom left side we have the Tag, Topic and Class group tables. Class refers to the class or lecture the teacher planned, and is related to a Topic. An Info_Source contains information pertaining either to web-pages or text and its pourpose is to store data that can potencially be used to train Lexical Models. The Tag and Tag_Question tables are meant to hold additional, low level or meta-information about the Question. These tables where designed to be used as filter attributes with which both the professor and the program can more easily search and group the Questions at will.

4.2.2 Implementation Process

The proposed solution requires database from where we will be able to extract and analyze multiple questions and misleading answers. This database should contain a vast selection of several exam questions containing Fill Gap type questions. The Fill Gap type question is a type of multiple answer question that has a gap on query. The student is supposed to choose an alternative that completes the sentence in a way that makes it a true statement, in other words, the correct answer.

In Fill Gap type questions the teacher writes a true statement on a Curricular Unit's topic. After this, he chooses a part of the statement to take out. The part taken out will be the correct answer and the now incomplete statement becomes the query. The gap in the sentence should represent a challenge to the students' knowledge in regards to the curricular unit's field. However, the statement that fills the gap does not play the same semantic role in every question. This means that in order for our program to generate grammatically correct misleading questions it must correctly identify what semantic role does the true statement play in the question. Not only this but it also has to identify what are the best in topic alternatives to suggest to the professor.

To accomplish the proposed goals we present the following sequence of actions according to the premises above:

1. First we will select a set of questions from the Fill Gap question type. This set will be characterized by a the semantic role the gap plays and the type of content the correct

answer statement has. We will call the combination of a specific semantic role and a specific type of content a "setting";

2. From this set we will take a single question which will be dubbed "Test Question". The query and correct answer from this question will be used to test our program till a satisfactory result of misleading answers can be generated;
3. With this in place our goal will be to generate some or all of the already stated misleading answers but most importantly it should also generate other semantically correct and valid misleading answers.

From the knowledge gathered from Chapter 2 the implementation of a NN algorithm seems the most promising way to be able to achieve the proposed goals. To this end we will create a secondary database containing true and false statements from the exam database. Using all the other Fill Gap type questions regardless of the setting we will combine the misleading answers from the question to create false statements and create true statements by combining the correct answer with the query. Using the secondary database we will feed our NN algorithm to create a lexical model that should be capable of suggesting valid misleading answers to the "Test Question".

If this system is not able to produce acceptable results we will add an engineering lexicon with the same curricular unit (in the same idiom as the questions) in an attempt to improve the algorithm's performance. Ideally, if the original exam question database has information regarding the quality of the underlying questions, it should be possible to make use of this information to improve the lexical model by giving a weight to the better classified questions.

After the design and implementation of the code that has revealed itself to be adequate to the creation of valid misleading answers to the "Test Question" settings we would use the same process to generate misleading answers to questions with different settings.

It will be also necessary to implement a code that allows the application to recognize different settings within the fill-gap type questions giving valid, professor approved, misleading questions.

4.3 The Server

The goal of this work is to provide easier access and manipulation to the professors exam questions database and at the same give a time saving function in exam question creation and development. With this goal in mind we designed the program to be a web-server making it possible to professor to access and manipulate their data via an internet connected browser. This architecture, althought less secure than a stand-alone aplication, allows for potencially global access to the platform from every professor provided with an internet connection. In informal discussions with professors it was noted the importance of security in regards to exam question assistance programs. In Section 7.2 we present a possible solution that would help to improve the confidence of professors when opting to use this platform throught the use of security oriented applications.

The servers program struture is represented in Figure 4.2.

The server archictecture itself is fairly simple. The endpoints located in the main "server.go" file that only allow for the following actions by the user: Reciving Questions and recieving Misleading Answer suggestions to a Fill-Gap question.

The Question model (Question.go) and Data Access Object (DAO.go) along with a test file for the Question model (question_test.go) and a utility function repository (standard_utils.go) files are located the "smart_exam_editor" package. These files were split from the main "server" package since auxiliary programs described in Chapter 5 required information already developed hence their separation. The DAO.go is Golang source code file that its functions serve as an interface between the database and the main program. Due to many changes during the database development for this thesis main functionality the DAO.go can only connect to the implemented database, deployed on the authors home server, and insert Question related data into the database.

The Question.go file encapsulates the information and functions for the Question structure for the program. The structure contains the fields for alocation of the concepts already discussed such as Query, Answer, Question Type (q_type), Tags (tags), Topics (topics) and Multiple Answer (m_answer). Adding to these there are the Id, Answer Context (Answer_Context), Negative_query, Single_sentence and Sketchy_query concepts that will be decribed shortly as well as some of the structures used in the concepts detailed before.

The Query and Answer fields are simple strings, the Topics and Tags are a map with both key and value strings and a array of strings respectivly pertaining to the matching concepts. The Id is an integer pertaining to the index number in the database and the Question Type is a struct pertaining to four types of Questions described in Subsection 4.1.1.

The reason behind the Multiple Answer structure (map[string]bool) and the Answer_Context field is the same as the one mentioned for the Raw_Misleading_Answer table referenced in the fourth paragraph of the Subsection 4.2.1. The Answer_Context is a flag that signals if the answer is "off-topic" in regards to the Answer. The Multiple Answer key refers to the Question Misleading Answer and the boolean value refers concept previously mentioned.

The Negative_query boolean signals if the query is a negative statement and the Single Sentence signals if the query is a single sentence. The field Sketchy_query is a boolean that is accerted true by the "testUnusualQuery" function if any of the following strings are found in the query ("all of the", "All ", "Are all", "Which", "which", "except"). The motive for the extraction of these concepts pertain to the information levied from professor Fernanda Delindro (Section 5.1). The syntactical relevancy of these types of statements is important to professors and the development of NLP algorithms alike, thus justifying their implementation.

When starting the program it first checks to see if it can communicate with the lexical model. In this set up the lexical model is held in a seperate Hypertext Transfer Protocol (HTTP) server that the main server makes requests to obtain the Misleadin Answers to the professors questions. This is possible due to the Word2Vec Golang package [1]. This package is used in this thesis to create the HTTP server as well as to establish the communications between the main and the Lexical Model holder server. It was possible to load the Lexical Model directly in the main server, however, depeding on the size of the lexical model, loading and retriving the misleading answers from it could delay the processing speed at which the main program ran. The decoupling of the program in two different services promotes cohesion and decopling while ensuring more control and modularity when alocating resources to these services.

[1] https://github.com/sajari/word2vec

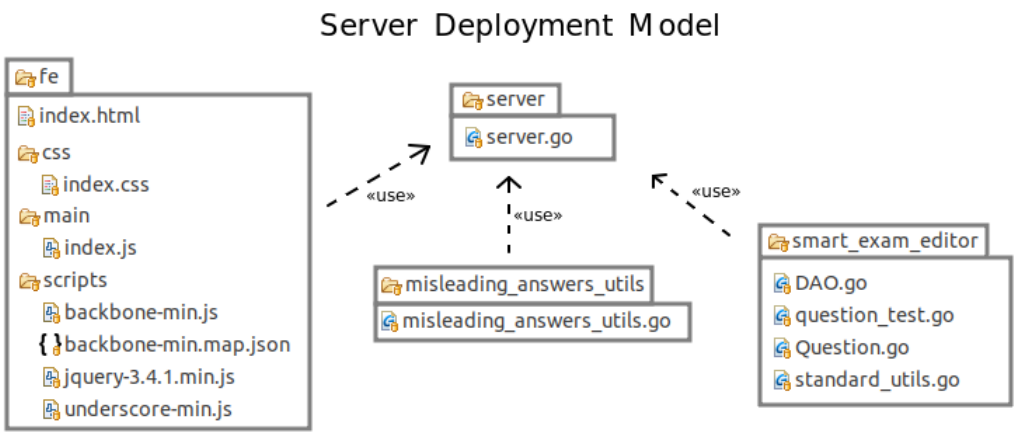

Figure 4.2: Deployment model depicting the artifacts used by the server program

4.4 Backbone.js

Backbone.js is a JavaScript library that aides front-end programming. By using its Model, View and Event structures it is easier to decouple the information the user wants to manipulate from the way it is presented and what group or single actions should trigger a behaviour from the program. This is advantageous for a long term program development since Backbone.js's correct implementation requires the programmer to develop his front-end logic to split information manipulation from information presentation. The library also makes use of stateless communications protocol functions that reduce server energy costs through the low overhead of browser-server communications as well as imbedded event tracking that essencially facilitates the design of the user's experience for the programmer. We choose to use this library having in mind the possibility of long term development for the program. By implementing this library from the start we are making sure that we are committing to a good set of front-end engineering practices and saving time on future developmental changes. In this work Backbone.js was used to support the front-end behaviour by logically separating the browser's User Interface (UI) in to several structures. The main method of working with the main Backbone.js strutures is simmilar to the logic of Java based systems. In the program we have several Backbone.js Views extended structures that encapsulate sections of code that represent error messages[2], exam question display,[3], the misleading answers display[4] and the main View that has the function of coordenating messages between different the different views within the site and finally the missleading answer suggestion view. Exept for the main view every one of the later views are tied to HTML templates and added dinamically to the main page when needed. This template based Backbone.js extended view allows for further modularity and efficiency in the development of the UI since it allows the programmer to re-use interface modules without having to duplicate code.

Figure 4.3 shows the main view of the program. On the left side of the main view we have the professor's question manipulation where the teacher can freely edit the main parts of a question present on the database. On the right side is the server's feedback where on the main view the misleading answer suggestions are shown and the user can select

[2] an example of an error message on the top right side Figure 4.3
[3] the left side of Figure 4.3
[4] the right side of Figure 4.3

which missleading answers are acceptable to add to the selected question. The plus sign button, present below the misleading answers section, only appears to the users on fill-gap type questions. After clicking this button the user is presented with misleading answer suggestions from the server. The decision of adding the Backbone.js library to the project was time consuming. In hindsight, it was not essencial or even needed to have the library integrated however the motive behind the development of this program was to have a proof of concept that could help teachers with their work. It was with the latter motive in mind that lead to the decision of integrating the library with the program.

Figure 4.3: Main View showing a Fill-Gap question and sugestions

Chapter 5

Information Gathering

In the previous chapter we detailed the structure of the main program where the professors interact with exam related information. This chapter's objective is to build upon the base structure and basic functions already covered and present the work most relevant to the title of this thesis.

In Chapter 2 we infered that NN oriented NLP technologies are proven to be the most accurate when dealing with problems of the nature of this thesis' project. To deliver suggestions of misleading answers to the user the program needs a Lexical Model. There are many non-topic specific Lexical Models already available to the public. These Lexical Models tend to give good "human-like" suggestions to any given word regardless of context. In the following sections we will present information, the work process and results gathered towards building this main thesis topic solution.

5.1 Question Crafting

In order to understand more what constitutes a good exam question and gain more insight into the work involved in creating exam questions we had a meting with professor Fernanda Delindro. Fernanda Delindro has 36 years of working as a Portuguese teacher, 24 years as a internship coordinator for highscool teachers and is a co-author of the books "Preparar a Prova Final 2020 - Português - 9.º Ano" and "Preparar o Exame Nacional 2020 - Português - 12.º Ano". Statements made in Section 1.1 regarding Multiple Answer type questions were based on her insights. The knowledge imparted by professor Fernanda Delindro in the following paragrafs is based in the Portuguese language, however we believe that this information transcends idioms and should be thought out as guidelines in question crafting.

When first formulating an exam a professor should know what concepts the questions should be about and to what purpose do the questions serve. The professor should also be aware of the thought process each student must have depending on the question type, be it identification ("What is..." type questions), memory based, intrepertation ("Why is..."), description of a subject based on the students knowledge and others.

When building an exam the professor should use questions with a balanced assortment of difficulty levels and type. By doing so it should become easier to gauge the knowledge level to the determined topic of every student. The time taken to answer each question counts as grading tool however students often lag when switching between questions specially if the questions cover different topics. The teacher should also be aware of questions contain the answers in the query or other questions present in the same exam.

Each question should have an unambigous answer or clearly detail the task the student should do. "Good questions" serve the professors pourpose, thus they should be well understood by the student. They should also be given a clear context, orient and drive the informed student's thought basing itself on the knowledge passed down during the teaching period.

"What", "When", "Whom" questions should be carefully pondered as they rely on memory and not on logic or racional. Questions relying on number of answers such as "Name a few...", "Give examples..." and "What are..." should be bounded to give a clear correction criteria. Professors should be weary of questions that can be correct by simply answering "Yes" or "No". Conversely, "Comment on ..." or "Give your opinion about..." and "In a few short lines..." or "Briefly..." type questions should be also avoided as the terms for successfull completion on behalf of the student are not clearly defined and consequently hard to gauge.

Lastly, carefull consideration should be given regarding the punctuation and the verb of the query. The verb in the query is what determines the action of the student. Correctly matching the professor's intent for the question with the apropriate verb give the teacher explicit expectations for the answer and an implicit gauge of the students knowledge of the underlying subject.

5.2 Questions Extraction Process

To serve as a test base for this thesis a database containing more than 1600 exam questions was graciously given by professor Ana Almeida. Professor Ana Almeida is the current director of the course Sistemas de Informação Empresariais (SINFE) at Instituto Superior de Engenharia do Porto (ISEP). Both course and database fall under the Enterprise Information Systems "Topic", and inherit the question types detailed on Figure 4.1 (Essay, True/False, Multiple Answer and Fill Gap). The questions came in a .docx format and in order to be more easly handled by software program were inserted into a database. As stated before, the questions were given in a .docx format and split up in several different files. Each file contained a different number of questions bettewen themselfs, and first glance they seemed to always start with a header detailing their chapter, a True/False question title followed by several questions of the same type, Multiple Answer title with the same pattern and finally Essay type questions. Each question was marked by a number and before the next question it had a footer composed of a "Title", "Learning Objective", "Section Reference", "Bloom's Taxonomy" and "Difficulty". True/False question types and Multiple Answer questions had a answer section giving either "true" or "false" or the letter containing the correct answer. By analising several Multiple Answer Questions sections it was quickly discovered that the possible answers (misleading answers mix with the correct answer) whould vary in number. Finally, Essay questions did not have a answer section.

Since it would probably take more time to ensure that every question would follow the same struture we used a tracer code approach [29]. The extraction program was implemented with these rules in mind however not all files followed the exact same structure above. To prevent After some debugging the extractor program had the structure represented in Figure B.3.

The extraction process functions iteratively for every file containing the raw format questions. The program uses the headers for each section of type questions (example: "Question Type: True/False") to split the text file in the apropriate sections for processing. Each section

had a different number of questions for each file which meant that the iterative process of question extraction was designed to check for the final line of text before anything else.

The True/False and Essay type questions exatraction process never needed repair following the first run. The Multiple Answer type questions was not. Firstly, Fill Gap type questions were present but not explicitly identified as such and were mixed with Multiple Answer type questions bellow the "Question Type: Multiple Choice" header. Secondly, the correct answer for the query was a letter that referenced an answer in the possible answer section. The possible answer section only exists within the Multiple Answer type questions section and consisted of four to six answers one of which was the correct answer. Thirdly, in the middle of some files inside Multiple Answer type question section was a fifth type of questions named Multiple Selection. Multiple Selection type questions were questions that there could be more than one possible answer to the query. A decision was made not to extract this question type however its "unpredictable" presence within the raw question files made the extraction process logic for the Multiple Answer type questions more complex.

As stated before, regardless of Question Type, every question was succeeded by a string of meta-information. This was helpfull since it made possible to isolate the extraction of meta-information in a single function to be used for every section. With hindsight we can state that the meta-information in the ".txt" files was the least consistent. Grammatical errors aside that were corrected manually as they were found, the meta-information was only consistent in it's key terms already referenced above ("Title", "Learning Objective", "Section Reference", "Bloom's Taxonomy" and "Difficulty"). In order to correctly extract this information the order and existance of these key terms had to be discovered prior to extraction. This meta-information is seen as potencially usefull to professors as they could be used as filters when searching for specific Questions and so it was kept under the Tag and Tag_Question in the database as seen on Figure B.2.

5.3 Didactic Material

In the Subsection 4.1.1 we describe this thesis Domain Model. There we briefely described the Didactic Material relation to the Lexical Model. In this Section we will further elaborate on the efforts to extract better results in the custom Lexical Model througth the retrival and sanatization of "lexical" data from techincal books on the topics of relevance.

As stated in the literature present in Chapter 2 most recent AQG or Automatic Question Generation processes use books to enhance their results. In accordance with these findings we used the books Fundamentals of Business Process Management [30], Management Information Systems: Managing the Digital Firm [31], Management Information Systems, Sixth Edition [32] and Principles of Information Systems [33] in attempt to enhance our results when working with database questions. Every book, averging at about 580 pages, in the set contained highly specific, grammatically correct and truthfull content on Enterprise Information Systems "Topic".

As the popular saying in computer science goes: "Garbage in, garbage out". Thus, by using only the core contents of those books, specifically the essays and the glossary terms, the objective was to keep the garbage out. This decision ment that the books contents would have to be sanatized. When working with text content, books included, normally there are references, quotes and simbols within. These strings are not only present throughout a single corpus but across multiple documents creating an unwanted bias towards these

specific terms. These terms were automatically removed by a sanatizer program described in Section 5.5 , however in these books there were other pieces of unwanted content such as image discriptions, example case studies, indexes, forewords, headers and footers as well as poorly formated text as a result of unorderly pdf text indexation. This work effort was done by manually reviewing the text files and comparing them to the original books.

5.4 Info Sources

Another repository of "lexical" data that was extracted to gather better results in the custom Lexical Model is the internet, specifically Wikipedia. Wikipedia has over five million articles[1] and is a reputable data aggregator with a plethora of different subjects. The usage of this repository in order to collect more information on this thesis main topic it was considered important therefore we considered the options available for this procurement. Wikipedia regularly provides "data dumps" due to popular research demand, however these big collections of information were deemed too expansive and broad for this thesis purpouse. Ultimatly, it was decided that it was best to use a crawler[2] in order to extract Wikipedia articles deemed relevant to our topic at hand.

Following the suggestion of supervisor Nuno Escudeiro an R coded web crawler was offered to be used as the Wikipedia article extractor. Unfortunalty this program had several depencies on now updated packages and after some attempts at updating the original code to fit the dependency changes this option was abandoned. In the end, a custom crawler was developed in order to extract the content of on topic Wikipedia articles.

To this end, a list of fifty one sub topics from SINFE courses files was detailed and matched to Wikipedia articles. This initial link list was used by the custom crawler to extract the original links and links directly linked to those pages. Like the books in the last Section, the lexical data extracted from Wikipedia also included unwanted text. Althought the crawler was implemented to ignore large sections of Wikipedia articles it did not outright remove those excerpts. In Figure 5.1 is a cropped and highlighed screenshot of the Business intelligence Wikipedia article. This figure highlighs in green the section of wanted content and in red the portions of unwanted content.

The custom crawlers' design was tailored to extract content from Wikipedia pages. For a more controlled extraction of content, the crawler was implemented in two seperate stages: link retrieval and content extraction. The crawler runs both stages in sequence by default althought it can run either one independently if provided with the necessary parameters. The crawlers' first stage is link retrieval. When provided with a file containing a list of valid Wikipedia links it adds these to a concurrent ready dictionary structure. Depending on the "depth", meaning the hierarchical distance between the original links, the program concurrently extracts the contents from each link, filters its content for wikepedia articles and finally adds the links to the aformentioned structure. The program repeats this cycle iteratively till it reaches the desired depth and finally saving the extracted links to a file. Due to the number of concurrent requests made to the Wikipedia page the server denied the response to the crawler. To combat this a recover from panic function that saved the overall progress in case of an denied response was added and each thread was awarded a random waiting time period.

[1] in English. Sourced from www.wikipedia.org

[2] a web crawler is a program that searches the web for websites

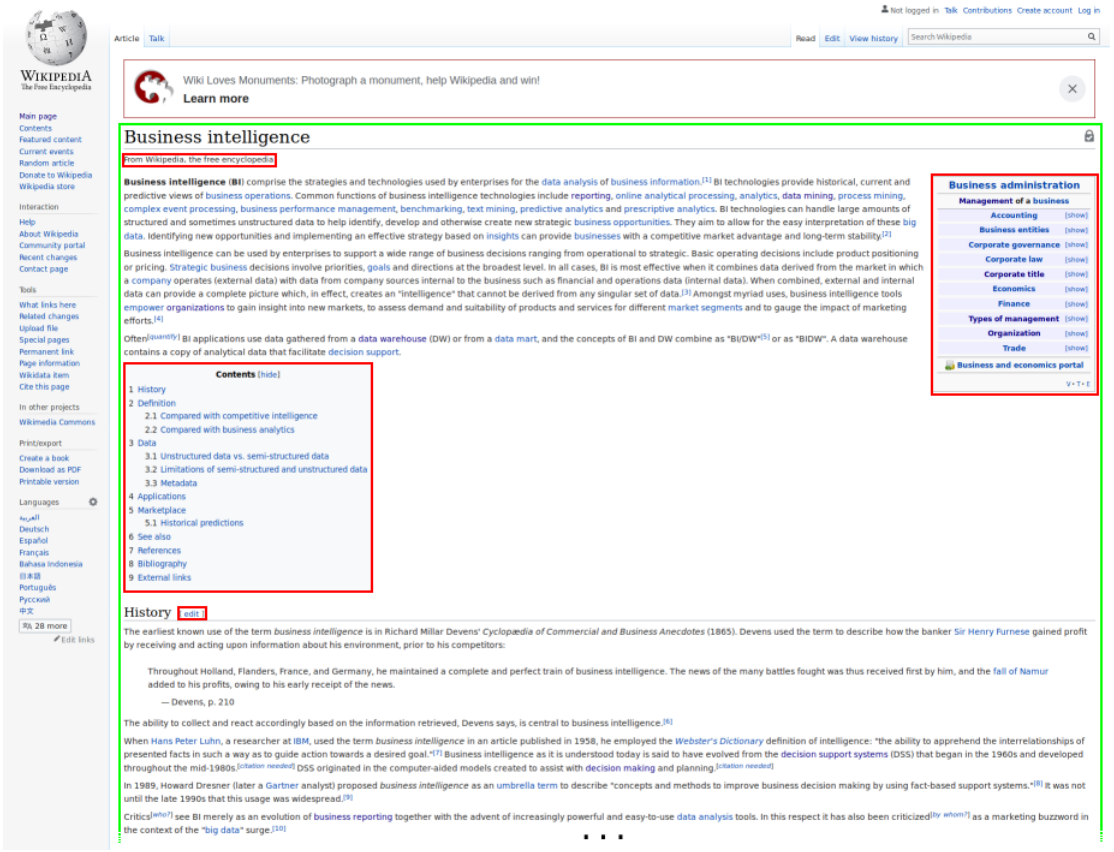

Figure 5.1: Inside the green lines are contained the core part of this Wikipedia article. The red squares represent sections of unwanted content contained inside the the core part of the webpage.

Taking into consideration the Wikipedia service denial error from the link retrival process, the content extractor was implemented to Wikipedia articles iteratively. By analysing the HyperText Markup Language (HTML) scruture of a number of different Wikipedia articles the many unwanted sections of content were listed and coded into an XPath query that automatically removed said content. XPath is a query language for selecting nodes from an XML document. Due to inconsistencies in the bottom area reserved for references in Wikipedia articles a standard algorithmic aproach was implemented to remove this unwanted content. Finnaly, after filtering, each content was saved to a file, named after the Wikipedia articles' subject/title.

The retrieved content only contained the essay present in the Wikipedia article althought it's still riddled with the same type of unwanted content referenced in Section 5.3. In the next section we will cover the custom sanatization program used to remove the references, quotes and simbols that would certainly decrease the quality of the custum Lexical Model.

5.5 Sanatizer

A sanatizer is a program that removes unwanted excerpts of text from documents to ensure better results from the data mining algorithms. When manually editing the books described on Section 5.3 and reviewing the Wikipedia articles in the last section a record was kept of the unwanted text patterns to be removed by the sanatizer. These text patterns were

then translated to regular expressions that were implemented in the sanatizer program. The expressions were divided into three main categories: remove, alter and special. As the nomenclature suggests, the remove type expressions removed pieces of unwanted text from the document. Alter type expressions only kept the first group of the match from the whole and the special expressions were treated indeplendently as some were meant to add or remove specic excerpts of text within the expression. Table 5.1 presents some examples of expressions used in the sanatizer program.

Table 5.1: Sanatizer Expressions

Type Expressions	**Code**	**Target Text**
Remove	\(x?(i[xv]|v?i{0,3})\)	Roman Numerals
Remove	\(Source: (.*\n?.*){1,2}\)	"(Source: ABC)"
Alter	\n\w\. (\w)	Numerical Listings
"Or" Expression	(\w+) ?/ ?(\w+)	Convert "/" between words to "or"

Both books and articles mentioned in the previous sections were processed by the sanatizer ensuring better results from the future custom Lexical Model as referenced in the state of the art.

5.6 Lexical Models

To create the custom Lexical Models we used the Gensim API [34] functions. As stated in their web-page: "Gensim is a free Python library designed to automatically extract semantic topics from documents, as efficiently (computer-wise) and painlessly (human-wise) as possible."

Gensim has a number of different algorithms that generate lexical models. In particular the Word2Vec algorithm was used to create the "GoogleNews-vectors-negative300" Lexical Model. It was created using a 3 billion running word corpus[3] from the Google News app.

The three billion words of corpora and the ambigous bias towards news articles made this model the candidate model for our base Lexical Model to compare our custom models to.

Moreover, in order for the main program to communicate with the Lexical Models several additional functions had to be implemented. Fortunately, a package with such functions already existed[4] and was therefore used in this thesis.

How was the test planned?

Theoretically, in order to objectively know if the new Lexical Models where improving when in comparison to the Google Model we first conceptualized the test. To test the models performance we gathered thirty questions with the the following criteria:

- The question has to be of Fill-Gap type.
- The query has to have only one gap.
- The answer had to be a single word.

[3] https://github.com/mmihaltz/word2vec-GoogleNews-vectors
[4] https://github.com/sajari/word2vec

These questions are acompanied by their determined correct answer, the original misleading answers and added possible other misleading answers. Some of the misleading answers contained more than one word. They were added to the test since they still constituted a possible misleading answer.

With the correct answer we then feed it to the Lexical Model we intend to test. The Lexical Model then gives us a list of words with a float number representing its calculated distance between the word that was given and the correct answer. We call this number the score and the higher the number the higher the correlation.

To evaluate objectively efficency of the Lexical Models we first determine the following parameters: the precision and the recall. We define the recall as the percentage of answers in the Match list in the possible answer list. The precision is the percentage of anwsers present in the top N Matches list in the possible answers list where N correponds to the number of answers in the possible answer list.

We can then cross compare the average results of these tests to determine which models give better Misleading Answers to the composed test.

How were the Lexical Models made?

In order to effectively compare the custom models to the base model (Google News) the custom models had to be calculated using the the same algorithm. The Gensim API packages containing the Word2Vec algorithm are implemented in Python language. The "gensim_model_creator.py" file holds the functions used to create the different Lexical Models.

Using the sanatized corpora of books and Wikipedia articles and giving the Word2Vec algorithm different iteration values, effectivelly controlling the bias towards words in the corpus, we could then produce comparable Lexical Models to our base Lexical Model.

Due to the reduced number of words in the individual corpora we feared that the custom Lexical Models produced would not give adequate results. To combat this we used the "intersect_word2vec_format" function that, essencially, averaged the score values from the custom lexical model with that of another, in this case, the Google News model.

The "model_tester" program produces a JSON file detailing the name of the model tested, the name given to the question agregation, the individual results to the questions and finally the average precision and recall values detailed above.

Chapter 6

Evaluation

To test the aforementioned hypothesis (Section 1.4) we propose a set of evaluation methods and a evaluation metric. These evaluation methods gather user feedback from the professors use and experience when working with the implemented solution. Towards this goal we will base our research hypothesis in three indicators: (1) the accuracy and recall of answer generated by the Lexical Model using pre-arranged parameters, (2) the reduction in the time required to create a new multiple answer question section, particularly, for the Fill-Gap Type Questions as well as (3) a simmilar test to the first using experts for these last two.

The Lexical Model will be trained in a particular Curricular Unit's topic. This means that only professors with the same field of knowledge as the trained algorithm will be invited to test the proposed main functionality.

In the following sections it will be described the evaluation process for the implemented solution.

6.1 Model Tester Setup

As stated Section 5.6 the lexical models were built using the Gensim API.

The parameters chosen to build the lexical models through the were:

size The size parameter represents the dimentions present in the words' vector. In essence, the bigger the corpus size the more dimensions should the vector have. The vector dimentions (size) was set to 300 due to our base model (Google News) size parameter. This enabled the intersection of both models as explained in the last paragraphs of Section 5.6.

window The window represents the number of words the algorithm considers when calculating the word vectors. It was set as 5, the same used in the base model.

min_count The min_count parameter is an integer that represents the minimum frequency count in individual words present in the corpus in order for the algorithm to process them. The value chosen was 1 since which meant that every word was represented in the vector space.

alpha The alpha parameter is the learning rate of the algorithm. The learning rate is a parameter that determines the extent that newly acquired information overrides old information, in this case the relation to words. The lower the rate, the longer it takes, iteration wise, to override information but more precisely, due lower rate the information changes. The alpha was set to 0.001 in an effort to gather better results.

iter The iter parameter determines the iterations the algorithm repeats before concluding its process. The more iterations it has the more bisased the model becomes. Since it is established that the model will be intersected with the base model and apha value for the learning rate is 0.001 which was lower than the one used in the base model we wanted the pre-intersected model to be overfitted towards the custom c. To this end the chosen value was 256. This number was chosen due to perliminary tests to the Book corpus where values in 2^n, n ranging from 3 to 8 was executed, being the 256 iterations to garner the best results.

This test is composed of 30 questions from the SINFE topic questions database. The questions selected covered every subtopic[1] present in the database with the parameters described in 5.6. Each question was composed of 5 to 8 possible answers that the Lexical Model results would be matched against, giving resulting in the precision and recall values. The adequacy of range of subtopics present and the possible answers to the answer and questions are to be approved by an expert.

6.2 UI Evaluation

In the first part of the evaluation the selected professors will be asked to complete a selection of objectives. The following objectives' purpose is to test the basic functionalities and the applications' intuitive interface.

1. Create an Exam Plan
2. Create a Question
3. Select a Previous Exam Question
4. Create Exam Versions
5. Export Exam

At the start of the test a timer will be set running. Each completed objective will have its time registered along will the total test time. These will be compared to the test done by the author in order to objectively access the intuitiveness of the functionalities proposed. After the completion of the proposed tests a questionnaire accessing the users likability to the functions tested will be distributed.

6.3 Expert Evaluation

To test the implemented program the invited professors will be asked beforehand to describe their own normal exam creation process with a similar questionnaire used in Section 3. Although the answers will be subjective to the professors own perspectives we can compare these answers with the ones made before to verify if the professors fall into the norm previously assessed. Depending on the number of available professors, each one will be asked to write a number of fill gap queries and correct answer pairs and the proximate time they took to write them.

After the test, time reduction will be measured by the difference of the average time required to draft a new exam with and without using the program. The mesurements will be the

[1] As mentioned in 4.2.1 the questions were acompanied by meta-information. The subtopic was derived from this information.

accuracy and recall described in Section 5.6 and also adding the percentage of accepted answers generated by the program without editing, the percentage of edited answers and the percentage of answers not accepted by the professors.

Professors in this test will be asked to complete the collected fill gap questions with misleading answers using the developed software. We will ask professors to perceive this test as a real exam proposition and relay the available options in the process. Each fill gap question will have a total of four misleading questions making the tally of five answers for the students to choose from in a real exam setting. In this test we will gather the following data:

- Misleading answers' chosen to each question and by what method
- Number of accepted misleading answers from the program's suggestion in total
- Number of new misleading answer suggestions requested in total
- Number of misleading answers suggestions edited in total
- Number of misleading answers written by hand in total
- Time took complete each fill gap question
- Overall time to make the test

From these tests we can then ascertain the success of the proposed main functionality with a point system derived from the following set of criteria:

Table 6.1 represents the total chosen misleading answers proposed by the program and corresponding points attributed.

Table 6.1: Percentage of picked and unaltered answers by the professor

Percentage Chosen	**Points**
More than 75%	4
From %50 to 75%	3
From %25 to 50%	2
From %25 to 5%	1
Less than 5%	0

Table 6.2 represents the percentage of total additional questions requested with respective points.

Table 6.2: Percentage of requested additional generated answers in total

Percentage Chosen	**Points**
Less than 25%	3
From %25 to 50%	2
From %50 to 75%	1
More than 75%	0

Table 6.3 represents the percentage of total eddied questions requested with respective points.

Table 6.3: Percentage of edited answers in total

Percentage Chosen	Points
Less Than 25%	0
From 25% to 50%	-1
From 50% to 75%	-2
More than 75%	-3

Table 6.4 represents the percentage of total hand written questions requested with respective points.

Table 6.4: Percentage of hand written answers in total

Percentage Chosen	Points
Less Than 5%	0
From 5% to 15%	-4
From 15% to 25%	-8
More than 25%	-12

To get the total creation question time we will add the time the time professors spent creating the query and true answer with the time spent choosing the misleading answers. With this number we will make the comparison to the values gathered in Chapter 3. Like in Table 6.1, Table 6.5 will give a set of points to the following criteria.

Table 6.5: Time Difference from the Norm Spent Creating the Multiple Answer questions

Norm - Gathered Results Norm	Points
Lower than zero	5
Approximately zero	0
More than zero	-5

By summing up all the points in the previous tables we can assess the programs' success.

Table 6.6: Grading Metrics

Total Points	**Grade**
+10	Complete Success
10 to 5	Minor Success
5 to 0	Shows Promise
-0	Failure

Chapter 7

Conclusions

In this Chapter analysing the tests implemented and their results and as well as making an overview of this thesis' work. Section 7.1 only covers the analytical tests refered in Section 6.1. The expert consulted to review que possible answers as well as the question topic range was Professor Ana Almeida. Due to unvailabillity of experts to participate in the tests described in Section 6.2 and 6.3 the results and conclusions to these could not be harnessed.

7.1 Result Analysis

All models were built with the prior parameters with differing corpora. The corpora tested came from the sanatized books and the handpicked wikipedia articles extracted using the crawler. The number of links gathered from the crawler with the depth parameter at 1 was 147787 wikipedia articles. After analysing a sample of the articles it was concluded that the information present did not pertain to the topic at hand. When atempting to process the depth 1 articles an error relating to lack memory persisted. Attempts at fixing the problem through iterative processing was analysed but not implemented.

Table 7.1 and Table 7.2 shows the results from the testing of the three models created for the two test bases. Both test contain suggested possible answers to the respective questions. Table 7.2 however was reviewed expert, prunning an ammount of possible answers in every question. All models were created using the sanatized versions of the books described in Section 5.3, the handpicked links and a corpus composed of both contents. Of the three, the Books model gives thet best average results to followed by Books_Links and Links models when comparing to the base model.

Table 7.1: Unreviewed Test Results

Models	**Average Precision**	**Average Recall**
Books	0.1333	0.1635
Books_Links	0.1300	0.1549
Links	0.0733	0.0811
GoogleNews-vectors-negative300	0.0433	0.0467

Althought the difference between in both precision and recall is small between the Books and Books_Links it worthy of expliciting that the Book model had a better performance then the agreggate. These results were to be expected since a generalist model would be out performed by models created specifically for a specific task. However, both corpus covered

Table 7.2: Expert Reviewed Test Results

Models	Average Precision	Average Recall
Books	0.0621	0.1022
Books_Links	0.0621	0.0851
Links	0.0379	0.0655
GoogleNews-vectors-negative300	0.0206	0.0425

the same topics difering only in quality and quantity of information. The corpus for the Links model had less 3 megabytes compared to the Books. When creating a model from a smaller corpus of words the underlying model should be overfitted to the underlying corpus, arguably improving the results for test of this nature. This was not the case. Therefore, these results corroborate the assersion that feeding a Lexical Model with a higher quality of information towards a specific topic improves its performance within the same topic.

7.2 Overview and Future Work

The work realized in this thesis was done in accordance with its title. With the development of the custom Lexical Models and the Server the professor can receive misleading answers suggestions to a newly formed question within the SINFE. When reviewing Figure B.1 the process of obtaining Misleading Answers takes a shortcut after "Sends Question Packet [QP] to server" directly to the "Misleading Answer Rank" section. It extracts its misleading answers along with its rank from the Lexical Model, removes its suggestions equal to the answer finally returning the misleading answers.

Admittedly, although not fully functional, the database section of the program was crucial to deepen the understanding of the mechanisms behind the Lexical Model algorithm creation and as a visual prototype for the model_tester program. During development priorities had to be shifted towards the construction of the Lexical Model, the main method of obtaining the misleading answer suggestions. Now that the main feature is functional striving to make the database fully functional should not be difficult.

In the development of this thesis better NLP algorithms were discovered. The paper [35] describes a method of obtaining distractors to gap-fill questions through various types of algorithms. From the concepts descriptions, the distractors are analogous to this thesis Misleading Answers and gap-fill questions to our Fill-Gap questions. By using the algorithms and methodologies provided from this thesis it could improve our models' performance. There were also news about a new state-of-the-art Lexical Model [36] that could reportedly "...generate realistic text in a variety of styles from news articles to fan fiction, based off some seed text."[1] If true, it could open possibillity towards a question suggestion generator, a simmilar program to this thesis, but focused on delivering the question suggestions themselfs rather than only the misleading answer suggestion.

Although this thesis only covers one specific Topic, the programs' structure was designed to service professors in other areas of education arround the world. However, serving a global market exposes the program, and consequently the professors work to very serious security risks. The Open Web Application Security Project (OWASP) is an international

[1] https://towardsdatascience.com/openais-gpt-2-the-model-the-hype-and-the-controversy-1109f4bfd5e8

organization that strives to ensure safe access to online aplications throught security program solutions. Notabily, the Enterprise Security API (ESAPI) is quote "... a free, open source, web application security control library that makes it easier for programmers to write lower-risk applications"[2]. By correctly implementing this API in the program it should go a long way towards gaining confidence with our users making this programs' information accumulate organically. The information could then be inderectly shared by other professors increasing the programs' service to the user overall.

[2]https://www.owasp.org/index.php/Category:OWASP_Enterprise_Security_API

Bibliography

[1] Cesar Ortega-Sanchez. "Written exams: How effectively are we using them?" In: *Procedia-Social and Behavioral Sciences* 228 (2016), pp. 144–148.

[2] John B Biggs. *Teaching for quality learning at university: What the student does*. McGraw-hill education (UK), 2011.

[3] Thomas M Haladyna. *Developing and validating multiple-choice test items*. Routledge, 2004.

[4] Halka Capkova, Jarmila Kroupova, and Katerina Young. "An Analysis of Gap Fill Items in Achievement Tests". In: *Procedia-Social and Behavioral Sciences* 192 (2015), pp. 547–553.

[5] John Hutchins. "The history of machine translation in a nutshell". In: *Retrieved December* 20 (2005), p. 2009.

[6] Jonathan Slocum. "A Survey of Machine Translation: Its History, Current Status, and Future Prospects". In: *Comput. Linguist.* 11.1 (Jan. 1985), pp. 1–17. issn: 0891-2017. url: http://dl.acm.org/citation.cfm?id=5615.5616.

[7] Peter Turney. "Learning Algorithms for Keyphrase Extraction". In: *Inf. Retr.* 2 (May 2000), pp. 303–336. doi: 10.1023/A:1009976227802.

[8] Rada Mihalcea and Paul Tarau. "TextRank: Bringing Order into Text." In: July 2004.

[9] Ronan Collobert and Jason Weston. "A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning". In: *Proceedings of the 25th International Conference on Machine Learning*. ICML '08. Helsinki, Finland: ACM, 2008, pp. 160–167. isbn: 978-1-60558-205-4. doi: 10.1145/1390156.1390177. url: http://doi.acm.org/10.1145/1390156.1390177.

[10] James A Anderson. "A simple neural network generating an interactive memory". In: *Mathematical biosciences* 14.3-4 (1972), pp. 197–220.

[11] Gail A. Carpenter, Stephen Grossberg, and David B. Rosen. "Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system". In: *Neural Networks* 4 (1991), pp. 759–771.

[12] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. "A Convolutional Neural Network for Modelling Sentences". In: *ACL*. 2014.

[13] Ronan Collobert and Jason Weston. "Fast Semantic Extraction Using a Novel Neural Network Architecture". In: Jan. 2007.

[14] Y. Lecun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (Nov. 1998), pp. 2278–2324. issn: 0018-9219. doi: 10.1109/5.726791.

[15] Chao Liu et al. "GPLAG: Detection of software plagiarism by program dependence graph analysis". In: vol. 2006. Jan. 2006, pp. 872–881. doi: 10.1145/1150402.1150522.

[16] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), p. 436.

[17] Ronan Collobert et al. "Natural Language Processing (almost) from Scratch". In: *CoRR* abs/1103.0398 (2011).

[18] G. Kumar, R. E. Banchs, and L. F. D'Haro. "Automatic fill-the-blank question generator for student self-assessment". In: *2015 IEEE Frontiers in Education Conference (FIE)*. Oct. 2015, pp. 1–3. doi: 10.1109/FIE.2015.7344291.

[19] Tianlin Zhang, Pei Quan, et al. "Domain specific automatic Chinese multiple-type question generation". In: *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE. 2018, pp. 1967–1971.

[20] Pei Quan et al. "Automatic Chinese Multiple-Choice Question Generation for Human Resource Performance Appraisal". In: *Procedia Computer Science* 139 (2018). 6th International Conference on Information Technology and Quantitative Management, pp. 165–172. issn: 1877-0509. doi: https://doi.org/10.1016/j.procs.2018.10.235. url: http://www.sciencedirect.com/science/article/pii/S1877050918319045.

[21] Michael Flor and Brian Riordan. "A Semantic Role-based Approach to Open-Domain Automatic Question Generation". In: *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*. 2018, pp. 254–263.

[22] Shichao Zhang, Chengqi Zhang, and Qiang Yang. "Data preparation for data mining". In: *Applied Artificial Intelligence* 17.5-6 (2003), pp. 375–381. doi: 10.1080/713827180.

[23] Google. *The Go Programming Language*. Jan. 2009. url: https://golang.org/.

[24] Daniel Whitenack. *Machine Learning With Go*. Packt Publishing, 2007.

[25] Jiju Antony. "Design for Six Sigma: a breakthrough business improvement strategy for achieving competitive advantage". In: *Work Study* 51.1 (2002), pp. 6–8.

[26] Adrian Voßkühler. "OGAMA description (for Version 2.5)". In: *Berlin: Freie Universität Berlin, Germany. Fachbereich Physik* (2009).

[27] Edwin S Dalmaijer, Sebastiaan Mathôt, and Stefan Van der Stigchel. "PyGaze: An open-source, cross-platform toolbox for minimal-effort programming of eyetracking experiments". In: *Behavior research methods* 46.4 (2014), pp. 913–921.

[28] Jon Scott. *To tackle student cheating, we need to reimagine university assessment*. Ed. by The Guardian. [Online; posted 13-June-2018]. Sept. 2018. url: https://www.theguardian.com/higher-education-network/2018/jun/13/to-tackle-student-cheating-we-need-to-reimagine-university-assessment.

[29] David Thomas Andrew Hunt. *The Pragmatic Programmer*. Addison Wesley, 1999. isbn: 978-0-2016-1622-4.

[30] Marlon Dumas et al. *Fundamentals of Business Process Management*. Springer Publishing Company, Incorporated, 2013. isbn: 9783642331428.

[31] Kenneth Laudon, Jane Laudon, and Ahmed Elragal. *Management Information Systems: Managing the Digital Firm*. May 2013. isbn: 978-0-136-07846-3.

[32] Effy Oz. *Management Information Systems, Sixth Edition*. 6th. Boston, MA, United States: Course Technology Press, 2008. isbn: 9781423901785.

[33] R. Stair and G. Reynolds. *Principles of Information Systems*. Available Titles Skills Assessment Manager (SAM) - Office 2010 Series. Cengage Learning, 2009. isbn: 9780324665284. url: https://books.google.pt/books?id=TlQCvdWQkfEC.

[34] Radim Rehurek and Petr Sojka. "Software framework for topic modelling with large corpora". In: *In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Citeseer. 2010.

[35] Girish Kumar, Rafael Banchs, and Luis D'Haro. "RevUP: Automatic Gap-Fill Question Generation from Educational Texts". In: Jan. 2015, pp. 154–161. doi: 10.3115/v1/W15-0618.

[36] Alec Radford et al. "Language Models are Unsupervised Multitask Learners". In: (2019).

Appendix A

Exam Questions Examples

13) A blogger is a company that operates a wiki.

Answer: false

Title: Testbank Question 8.13
Learning Objective: LO 8.1: Describe six Web 2.0 tools and the two major types of Web 2.0 sites.
Section Reference: 8.1 Web 2.0
Difficulty: Medium

Figure A.1: An example of the original question structure.

Question Type: True/False

Information technology (IT) refers to the collection, processing, storage, analyzing, and dissemination of information.

Answer: _____

Figure A.2: An example of a true or false answer question type.

Question Type: Fill-Gap

Q: _____ collects, processes, stores, analyzes, and disseminates information for a specific purpose.

a) Information Technology
b) Network System
c) Information System
d) Information Network

Figure A.3: An example of a fill gap answer question type.

Question Type: Multiple Choice

Q: Supply chain systems are which type of information system?

a) departmental information systems
b) enterprisewide information systems
c) interorganizational information systems
d) end-user computing systems
e) individual information systems

Figure A.4: An example of a multiple answer question type.

Question Type: Essay

Q: Differentiate between information systems and information technology.

Figure A.5: An example of an essay question type.

Appendix B

Implementation Figures

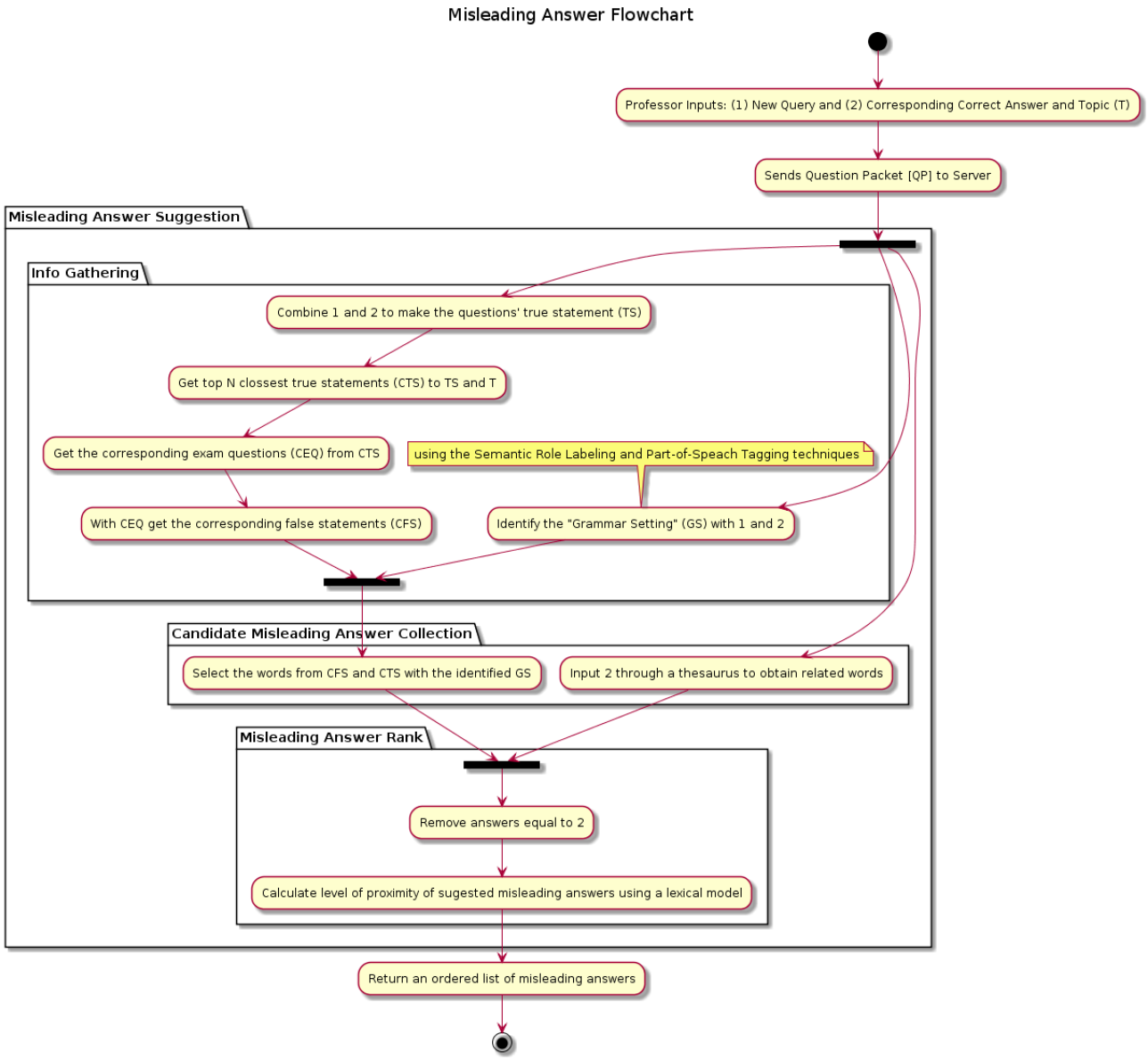

Figure B.1: Flowchart detailing the process for aquiring misleading answer suggestions

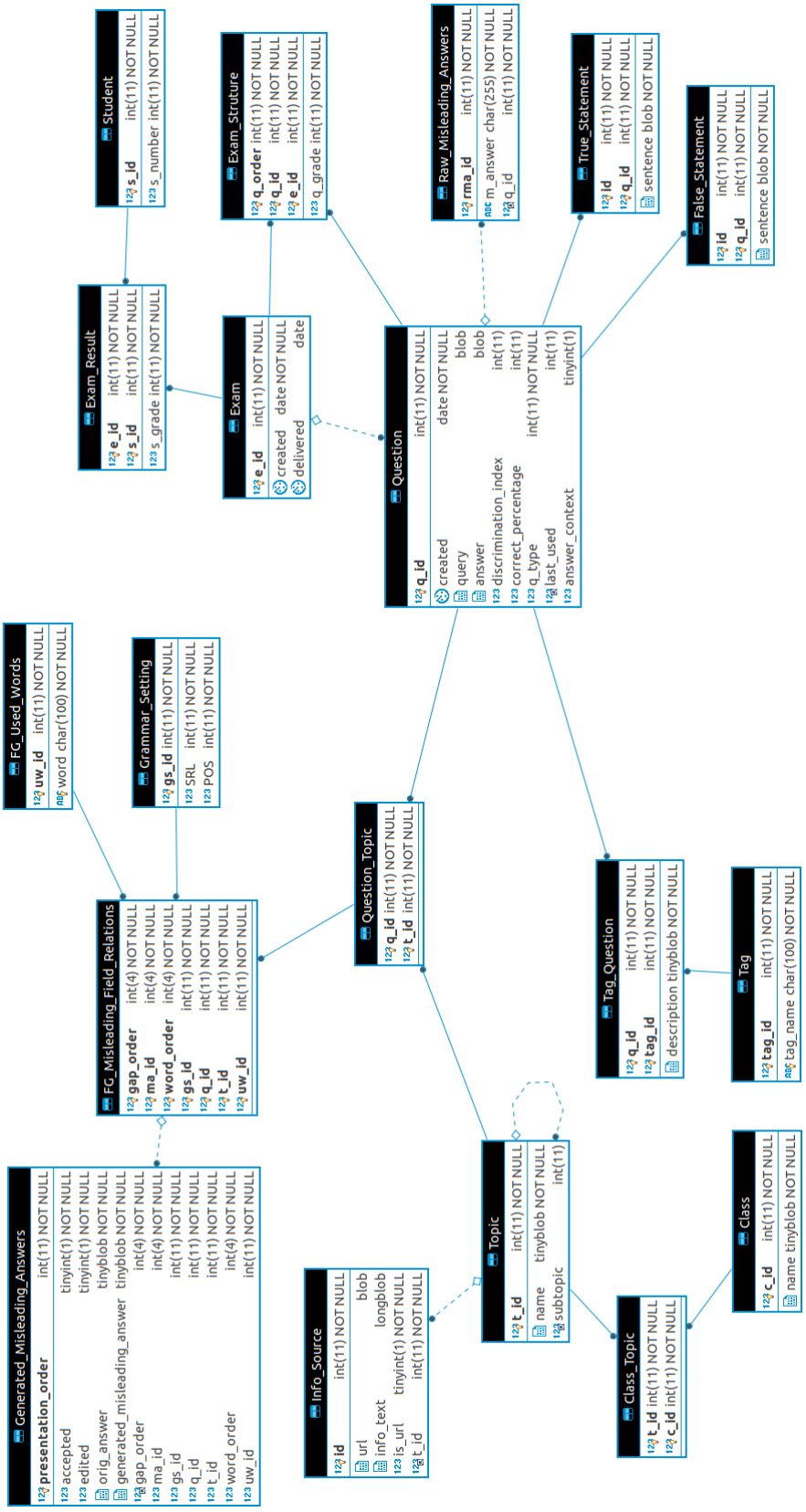

Figure B.2: Database Model

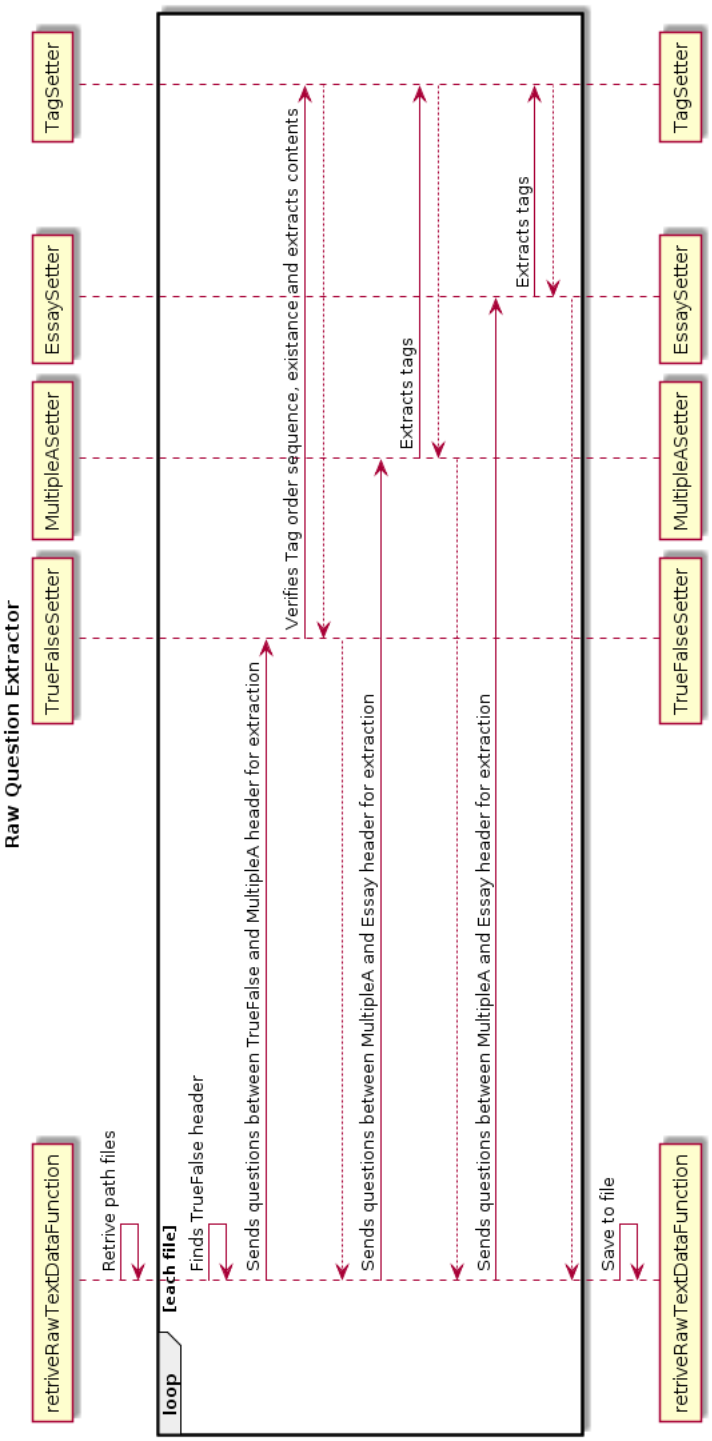

Figure B.3: Sequence diagram of the raw text question extraction process