Learning Semantic Patterns for Question Generation

Hugo Patinho Rodrigues

CMU-LTI-20-013

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

*Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in Language and Information Technologies*

TÉCNICO LISBOA

UNIVERSIDADE DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

Learning Semantic Patterns for Question Generation

Hugo Patinho Rodrigues

Supervisor:	Doctor Maria Luísa Torres Ribeiro Marques da Silva Coheur
Co-Supervisor:	Doctor Eric Nyberg

Thesis approved in public session to obtain the PhD Degree in Computer Science and Engineering

Jury final classification: Pass with Distinction

2020

TÉCNICO LISBOA

Carnegie Mellon University

UNIVERSIDADE DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

LANGUAGE TECHNOLOGIES INSTITUTE
CARNEGIE MELLON UNIVERSITY

Learning Semantic Patterns for Question Generation

Hugo Patinho Rodrigues

Supervisor: Doctor Maria Luísa Torres Ribeiro Marques da Silva Coheur
Co-Supervisor: Doctor Eric Nyberg

Thesis approved in public session to obtain the PhD Degree in Computer Science and Engineering

Jury final classification: Pass with Distinction

Jury

Funding Institution

Fundação para a Ciência e Tecnologia

2020

Abstract

Question Generation (QG) is the Natural Language Processing (NLP) task dedicated to the automatic generation of questions from raw text. It can be useful in many different scenarios, from educational settings, in which the generation of questions can eliminate a huge burden on professors and instructors in creating them to assess their students, to populating the knowledge base of a conversational agent. In this thesis we present a pattern-based system, GEN, that automatically performs the task of QG. Given an information source and a set of seeds constituted of question/answer/sentence triplets, GEN outputs a set of questions (and answers, if possible). Contrary to other similar systems, GEN is not built upon hand-crafted templates, and, instead of relying in patterns that only go to the lexical and syntactic level, it deeply explores the existence of semantic information in a flexible pattern matching process, allowing it to occur at different linguistic levels. In addition, instead of using a limited number of rules or models incorporated at design time, GEN is able to learn from questions corrected by the user, in order to perform better in future iterations.

In this work, we have also made a contribution to QG automatic evaluation. Many authors rely on automatic metrics, instead of relying on manual evaluations, as their computation is mostly free. However, corpora generally used as reference is very incomplete, containing just a couple of hypotheses per source sentence. With that in mind, we contribute with the MONSERRATE corpus, containing 26 times more questions per reference sentence, on average, than any other available dataset. The implications of such a large size for a reference are also studied, and we concluded that MONSERRATE is "exhaustive" enough for QG evaluation. We benchmark GEN against current state of the art QG systems, and show that our approach is able to generate quality questions and surpass a neural network approach, given as input just 8 seeds. We evaluate the systems both with automatic metrics, and through human annotators. Finally, we employ GEN in two different scenarios. First, we show that it can be used as an authoring tool to help professors create questions for their courses, by presenting the best questions in the top of a ranked list. In this experiment, GEN learns new patterns and ranks the generated questions due to a simulated teacher feedback. To the best of our knowledge, GEN is the only QG system that can be easily adapted to this scenario, and benefits from not requiring a linguistic expert as user. Secondly, we apply GEN in a Question Answering (QA) setup, where it is used to create questions that attempt to improve the performance of an external system.

Keywords: Question Generation, Evaluation, Corpora, Automatic Metrics, Natural Language Generation

Resumo

Geração de Perguntas (QG) é a tarefa de Processamento de Língua Natural (NLP) dedicada à geração automática de questões a partir de texto corrente. Isto pode ser útil em diferentes cenários, desde contextos educacionais, onde a geração de perguntas pode eliminar o esforço de professores e instrutores em criá-las para avaliar os seus alunos, até à população de uma base de conhecimento de um agente conversacional. Nesta tese apresentamos uma abordagem baseada em padrões, GEN, que automaticamente executa a tarefa de QG. Dada uma fonte de informação e um conjunto de seeds constituídas por triplos pergunta/resposta/frase, GEN devolve um conjunto de perguntas (e respostas, se possível). Ao contrário de outros sistemas semelhantes, GEN não foi construído com regras desenhadas manualmente e, ao invés de depender de padrões que apenas vão ao nível léxical e sintático, GEN explora a existência de informação semântica num processo flexível de correspondência de padrões, permitindo que ocorra a diferentes níveis linguísticos. Além disso, em vez de utilizar um número limitado de regras ou modelos incorporados desenhados apriori, GEN é capaz de aprender com perguntas corrigidas pelo utilizador, de forma a ter um melhor desempenho em futuras iterações.

Neste trabalho também contribuímos para a avaliação automática de QG. Muitos autores utilizam métricas automáticas, por oposição a avaliações manuais, dado que a sua computação é basicamente gratuita. No entanto, corpora tipicamente utilizados como referência são bastante incompletos, contendo apenas algumas hipóteses por frase fonte. Com isto em mente, contribuímos com o corpus MONSERRATE, contendo 26 vezes mais perguntas por frase, em média, que qualquer outro dataset disponível. As implicações de uma referência desta dimensão são também estudadas, e concluímos que MONSERRATE é 'exaustivo' o suficiente para a avaliação de QG. Fazemos benchmark do GEN contra sistemas de QG estado da arte, e mostramos que a nossa abordagem é capaz de gerar perguntas de qualidade e ultrapassa uma abordagem de redes neuronais, dadas apenas 8 *seeds*. Avaliamos os sistemas tanto com métricas automáticas como através de uma avaliação humana. Finalmente, utilizamos GEN em dois diferentes cenários. Primeiro mostramos que pode ser utilizado como uma ferramenta para ajudar professores a criar questões, apresentando as melhores perguntas no topo de uma lista ordenada. Nesta experiência, GEN aprende novos padrões e ordena as perguntas geradas devido a um feedback simulado do professor. Para o melhor do nosso conhecimento, GEN é o único sistema QG que pode ser facilmente adaptado a este cenário, e beneficia de não requerer um linguista como utilizador. Em segundo, aplicamos GEN a um cenário de Pergunta-Resposta (QA), onde é utilizado para criar perguntas que visam melhorar o desempenho de um sistema externo.

Palavras-Chave: Geração de Perguntas, Avaliação, Corpora, Métricas Automáticas, Geração de Língua Natural

Acknowledgements

This was quite the journey. When I started, I was far from imagining how much my life would change. I had my joys and sorrows, and it ends with a bittersweet taste. But if I got here, it is thanks to amazing people I have in my life.

First and foremost, I have to thank my advisors, Prof. Luísa Coheur and Prof. Eric Nyberg, for their patience and support, alongside theit insightful advises and motivation during these years, which made this work possible.

I also want to thank all my coworkers and colleagues, both at HLT (former L^{2}F) and LTI, specially Pedro Fialho, Vânia Mendonça, Francisco Raposo, and Eugénio Ribeiro, for their help and availability to discuss, share ideas, and collaborate.

A word for all friends that I made along the way, and also to those who were there from the beginning. People who, in one way or the other, made my journey more pleasant, either at lunch breaks, game nights, soccer matches, magic tournaments, or a simple drink: André C., Beatriz F., Bruno C., Inês M., Jesse S., Liliana M., Matt F., Melissa G., Michael S., Nuno D., Nuno M., Romeu M., Xavier V..

A special thanks to some folks not listed above, who were a constant in my life and who I know I can count on, no matter what: Ariana Santos, Diogo Godinho, and Serban Mogos. And finally, someone who I really do not have words to thank, and belongs to both groups before. Pedro Mota has been sharing my path for so long now, that it is hard to tell if he was a friend or colleague first. All I know is he ticks all those boxes and many more, he was a partner in all good moments I had, a shoulder in all bad, a brain when mine melted, a hand when I needed one. To you, my friend, my gratitude.

And last, but not least, I want to thank my family. They supported me this whole time, in all senses of the word, and I could not really ask for better people to have raised me. These years were not kind to us, but together we were able to overcome that. I really appreciate everything you have done for me, and I can only hope to be able to someday repay even half of that.

Lisboa, September 2020

Hugo Rodrigues

To my parents,

Ana Patinho, Luís Rodrigues

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Question Generation (QG) is the field of Natural Language Processing (NLP) that aims at automatically create questions (and its answers) from a given text. It has been studied for some time now [Du et al., 2017, Heilman, 2011, Indurthi et al., 2017, Mazidi and Nielsen, 2015, Rus et al., 2010, Serban et al., 2016] and had significant improvements in the past few years. Manually creating questions is a time-consuming task, and many different scenarios could benefit from automatically created questions. Educational settings, in which teachers have to create questions to assess their students, and the creation of a knowledge base for a chatbot that performs, for instance, customer support, are a few examples of how QG can be applied. Question Answering (QA), a related task that tries to find the answer to questions posed in natural language (in contrast to search engines that just retrieve a set of related documents given some keywords), can also benefit from automatically created questions, as these can provide, for instance, more training data for such systems. However, creating questions is a challenging task because natural language is flexible, and questions may be formulated in many forms.

QG systems are typically designed in one of two forms. In the first we have systems that rely on handcrafted linguistically motivated rules/templates that establish how to transform input sentences into questions. Heilman [2011], Mazidi and Nielsen [2015] and Mannem et al. [2010] are examples of such techniques. These systems' main disadvantage is that they need experts (possibly with a linguistic background) to build the used rules. In addition, it is not possible to change them after deployment. The second group of QG systems are based on neural networks. With the appearance of large datasets (like SQuAD [Rajpurkar et al., 2016]), it became possible to successfully train large neural networks with the goal of generating questions [Du et al., 2017, Indurthi et al., 2017, Serban et al., 2016, Subramanian et al., 2018, Wang et al., 2017, Zhou et al., 2018]. As is usual in Deep Learning, these systems require large collections of data to be trained, which are not always available.

In a real QG setting, users are likely required to correct at least some of the obtained questions. In this work we explore the concept of taking advantage of this implicit feedback to improve the performance of the proposed QG system. Linguistic-based approaches cannot directly take advantage of this, as their design is based in handcrafted rules, and neural QG systems will hardly benefit from a small set of manual corrections. However, a system could be designed to learn from question/answer/sentence seeds, so that corrected questions could be used to improve it. The-Mentor [Curto et al., 2012] is the only system, to the best of our knowledge, that employs a seed-based learning approach to QG, but it is no longer available. Although in other fields there are some systems using past interactions to improve their performance in future interactions [Mendes, 2013, Shima, 2015, Velardi et al., 2013], to the best of our knowledge no QG system does this.

In this work we present a example-based QG system using semantic features and the capability of learning from past interactions, something other systems, to date, are not able to. In addition, we create a dataset that makes automatic evaluation of QG possible, allowing for a faster development and evaluation of systems in this research area.

1.2 Thesis Statements

In this thesis, we propose GEN, a system that, as The-Mentor [Curto et al., 2012], is based in question/answer/sentence seeds, but, contrary to that work, takes advantage of several semantic features, both at token and sentence level. Thus, unlike other works that also use some sort of semantic patterns [Lindberg et al., 2013, Mannem et al., 2010, Mazidi and Nielsen, 2014], GEN automatically learns those patterns instead of relying in handcrafted rules. In addition, a flexible pattern matching is implemented – from a very general pattern matching at syntactic level, to a very constraint one, at lexical level – allowing to generate more but less precise questions or less but more accurate questions. Moreover, GEN is able to take advantage of the feedback provided by the end user. This feedback is used not only to collect more seeds, but also to learn to rank the generated questions, increasing the quality of the top generated questions. An important detail is that the user does not need to have any background in linguistics; just being able to correct the generated questions suffices.

Our research hypotheses are, thus, the following:

Hypothesis 1 (H1): *The proposed QG system outperforms current state of the art systems, measured by qualitative metrics as grammaticality, semantical correctness, relevance, among others, evaluated independently by the crowd through Amazon Mechanical Turk (AMT), or by quantitative metrics, like BLEU, ROUGE, or others automatic metrics.*

Another important aspect of the field of QG, shared by the QA field, is the lack of data to train new neural network systems. This has recently changed with the creation of datasets like SQuAD, but all questions are created by humans, which puts a limitation on the completeness of the dataset. Our system can be, thus, an important tool to generate a large dataset that complements other corpora like SQuAD. Specifically, QA systems can gain from being trained with more complete datasets. Note that the questions might not need to be perfectly generated in order to contribute positively to these QA systems:

Hypothesis 2 (H2): *The proposed QG system can be used to create a dataset of Question/Answer pairs that can be used by external QA systems as support data (i.e., training data). The impact can be evaluated by measuring the difference in performance those systems attain, as measured by quantitative metrics like accuracy and recall.*

QG systems can be a vital source of data, reducing the time spent by users in building their applications. GEN includes a component of learning to rank the generated questions by using the user's feedback, which can help filtering questions presented:

Hypothesis 3 (H3): *The proposed QG system will automatically generate questions that can be used or easily modified for usage. The implicit feedback obtained from the user (e.g,* *an instructor) can be used by our system to improve the quality of the generated questions. This can be assessed by measuring the quality of the questions on the top N questions, as if the user (instructor) were only presented those questions instead of all generated questions.*

One major difficulty in studying the first hypothesis is that, despite the growing interest in QG, evaluating these systems remains notably difficult. Evaluating if a question is good or not is a subjective matter and even employing automatic metrics, like BLEU, can be hard, as it requires an extensive dataset covering many acceptable possibilities, given that in QG many distinct good questions can be generated from the same input. In fact, many authors rely on automatic metrics, like BLEU or ROUGE, instead of relying on manual evaluations, as their computation is mostly free. However, corpora generally used as reference is very incomplete, containing just a couple of hypotheses per source sentence. Therefore, we also propose MONSERRATE, a dataset specifically built to automatically evaluate QG systems. With 26 questions, on average, associated to each source sentence, it attempts to be an "exhaustive" reference. With this corpus we also study the impact of a reference's size in evaluating QG systems. This leads us to our final hypothesis:

Hypothesis 4 (H4): *The proposed corpus,* MONSERRATE*, is more suited to perform automatic evaluation of QG systems using automatic metrics, like BLEU or ROUGE. It can be shown that the size of the reference impacts the perceived performances by measuring the metrics' scores at different reference sizes, and their statistical significance.*

We benchmark GEN and state of the art QG systems in MONSERRATE and SQuAD, and also employ GEN in two different scenarios: 1) a teaching setting, in which a "teacher" provides feedback to the generated questions, which lets GEN learn and rank new patterns; 2) a QA scenario, in which GEN (and other QG systems) is used to improve an external QA system performance. Besides quantitative metrics, like BLEU, ROUGE, or others automatic metrics, we also use qualitative metrics as grammaticality, semantical correctness, relevance, and utility, evaluated independently by the crowd through AMT.

1.3 Contributions

From this work result the following contributions:

- GEN, a complete QG system that:
  - automatically learns semantic patterns from seeds constituted of question/answer/sentence triplets, and generate questions using those patterns;
  - relies on a flexible pattern matching mechanism that allows to generate questions that are (more or less) constrained by the original seeds at different levels of linguistic information;
  - takes advantage from users' feedback to improve its results, by both learning new seeds that enlarge the pool of available patterns, and learning to weigh patterns, which increases the quality of the top generated questions.
- Monserrate, a corpus aimed at automatically evaluate QG systems containing an average number of questions per reference sentence 26 times larger than all available datasets.

In addition, this work resulted (or was used) in the following publications:

- Hugo Rodrigues, Luísa Coheur, and Eric Nyberg. Populating the knowledge base of a conversational agent: human vs. machine. *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, 2019
- Hugo Rodrigues, Luísa Coheur, and Eric Nyberg. Improving question generation with the teacher's implicit feedback. In *International Conference on Artificial Intelligence in Education*, pages 301–306. Springer, 2018
- Pedro Fialho, Hugo Rodrigues, Luísa Coheur, and Paulo Quaresma. L2F/INESC-ID at SemEval-2017 tasks 1 and 2: Lexical and semantic features in word and textual similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 213–219, Vancouver, Canada, August 2017. Association for Computational Linguistics

- Hugo Rodrigues, Luísa Coheur, and Eric Nyberg. QGASP: a framework for question generation based on different levels of linguistic information. In *Proceedings of the 9th International Natural Language Generation conference*, pages 242–243, Edinburgh, UK, September 5-8 2016. Association for Computational Linguistics. (demo)

1.4 Document Overview

This document is organized as follows:

Chapter 2 overviews the relevant related work, with special focus on Question Generation systems. The chapter is broadly divided by strategy, concluding with a section that covers the major points of interest.

Chapter 3 presents in detail our QG system, across all steps, illustrating from begin to end how a question is generated.

Chapter 4 describes one of our contributions, an extensive dataset created to make possible to automatically evaluate QG systems.

Chapter 5 benchmarks the current state of the art in QG, comparing our system with other available systems in two different settings.

Chapter 6 delves into another of our contributions, where our system learns from past interactions, in order to improve its performance over time.

Chapter 7 explores the use of automatically generated questions as data to improve other systems in a QA setting.

Chapter 8 concludes the document and sets possible paths of future work.

2 Related Work

In this chapter we overview the work done in Question Generation (QG) – Section 2.1 –, starting by giving special attention to approaches that use patterns and ending with the most recent approaches based on neural networks. We investigate different resources and tools that are useful to create a QG system or evaluate them (Section 2.2), before discussing in detail the evaluation process of QG systems (Section 2.3). Finally, we also look at works that use patterns in other domains and briefly discuss the case-based paradigm (Section 2.4).

2.1 Question Generation

The goal of QG is to automatically create questions from text. However, different types of questions may be asked. Usually, systems focus on Yes/No and factoid questions[1], as they tend to be easier to generate and can be extracted more frequently. However, it can be useful to create other types of questions, such as *Why* questions, or even more complex questions, like *Give an example of (...)*, *Enumerate (...)* or *Describe (...)* type of questions – Forăscu and Drăghici [2009] discuss a taxonomy for QG and other tasks.

Take into consideration the following sentence:

Ana, the first child of Carla, gave an apple
to Bob after class and he thanked her. (2.1)

[1]Who, what, where, when are the main examples.

Multiple questions can be created for the sentence presented:

Who gave Bob an apple?

What did Ana give to Bob? (2.2)

Who did Ana give an apple to?

When did Ana give an apple to Bob?

Of course, questions such as *Who gave Bob an apple after class?* are also valid; they only differ by having more information, but they target the same answer as the first example. Also, if anaphora resolution is not performed, from the second part of the sentence one can create the following question: *Who thanked her?*, which lacks enough information to be answered without context.

The limitlessness of possible questions goes even further when we take into consideration other verbs or semantic formulations. For instance, *From who did Bob receive an apple?* is a perfectly admissible question, but it requires another other sources of knowledge: the opposite of *give* is *receive*, and this action involves two subjects. These and other problems are discussed in greater detail by Heilman [2011].

QG has a few extra challenges. First, it contains a subtask of natural language generation: the questions created must be grammatical and semantically correct, besides, of course, being useful. Secondly, it is hard, if not impossible, to have a goldstandard of what are the *right* questions to create from a text. These two aspects are closely related with the evaluation process, later discussed in this chapter, but before addressing that matter we go over the related work on QG systems.

2.1.1 Rule-based Question Generation

QG has been studied for some time now and, from 2008 to 2011, there was a huge contribute to its research mostly due to a QG workshop [Rus and Lester, 2009, Rus et al., 2010, 2011, 2012][2] taking place, with the last two containing a Shared Task Evaluation Campaign

[2]`http://www.questiongeneration.org/`

[Rus et al., 2011, 2012]. Many papers were submitted during those 4 years, but only a few contain detailed descriptions of the systems and evaluation process. At the time, most systems employed a rule-based approach. In this section we will present the general approach that is followed, at least partially, by most systems. Kalady et al. [2010], Ali et al. [2010], Yao and Zhang [2010], Varga and Ha [2010], and Heilman [2011] are examples of such systems.

There are three main tasks in this type of approach shared among most QG systems: sentence decomposition, syntactic parsing, and Named Entity Recognition (NER). The first deals with sentences like Sentence 2.1, simplifying them in order to ease the process of question generation [Kalady et al., 2010, Yao and Zhang, 2010]. For instance, it could be separated in two different sentences: *Ana, the first child of Carla, gave an apple to Bob after class* and *He thanked her*, and it could also be stripped of additional information: *Ana is the first child of Carla.*

For the second task, syntactic parsers are widely used. They allow the mapping of sentences into trees, by grouping words and the associated tokens into nodes representing their syntactic tags. These tags, such as noun phrases (`NP`) and prepositional phrases (`PP`), identify different targets for the question generation process, with their textual value being the answer to those questions. Figure 2.1 shows an example of a syntactic tree for the sentence *Alexander Graham Bell is credited with inventing the telephone.*

Finally, for the third task, NER is used choose the correct Wh-word to use for the question. For example, the sentences *Bob hit Ana* and *The car hit Ana* are represented by an almost identical syntactic tree, but the chosen Wh-keyword must be different, depending on the subject of the sentence, if one wants to generate a question of the type *<Wh-word> hit Ana?*

Given the data acquired with the previous steps, these systems resort to a sort of rules or patterns to create the desired questions. For example, some systems design rules as $\texttt{NP}_1$ $\texttt{VB NP}_2 \rightarrow \texttt{Wh-word VB NP}_2\texttt{?}$, which covers the example before. Ali et al. [2010], Pal et al. [2010], Varga and Ha [2010] are examples of systems that use such strategy. Others, like Heilman and Smith [2009, 2010], Kalady et al. [2010] and Wyse and Piwek [2009], use tree operations directly on the parse trees to transform the sentences into the questions.

Many other minor tasks are performed as well, such as inversion of the subject and aux-

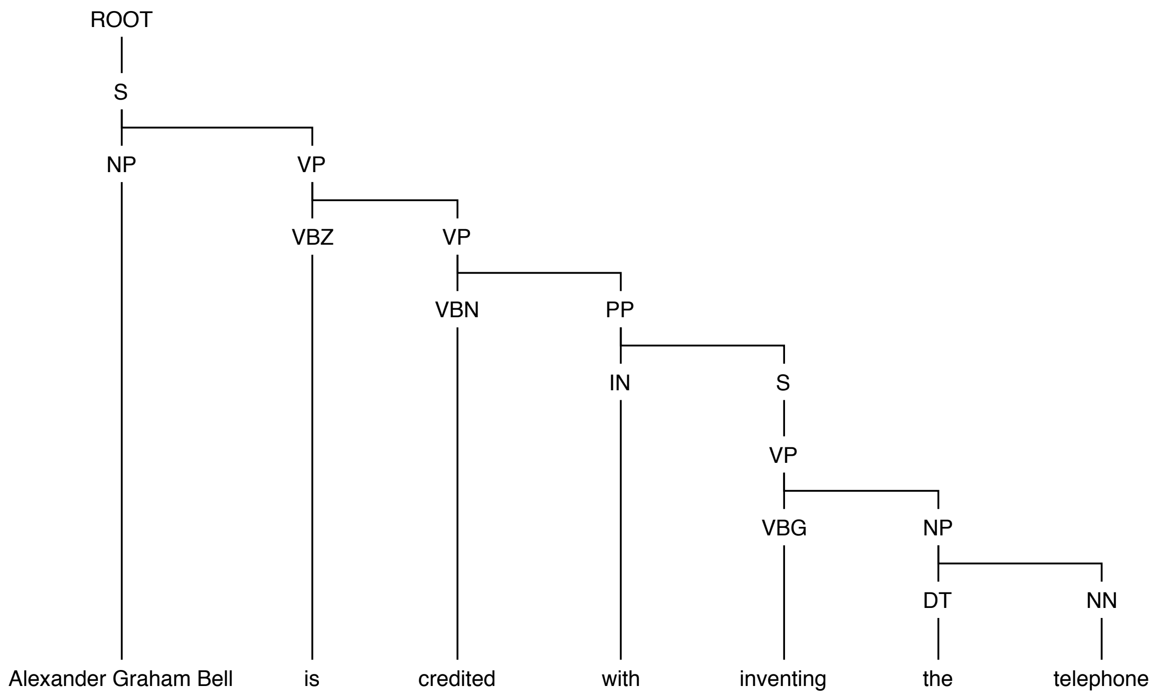


Figure 2.1: Parse tree for the sentence *Alexander Graham Bell is credited with inventing the telephone* with Stanford syntactic parser.

iliary verb, and decomposition of the main verb, (*i.e.*, the transformation *ate* to *did eat*). Other systems include coreference resolution [Kalady et al., 2010], as pointed to be needed by Heilman [2011]. Mazidi and Nielsen [2015] combines multiple techniques, like dependency parsers, syntactic parsers, and discourse cues (which the authors call of using multiple views of the text) to improve the quality of the questions generated. This approach has also been used in other domains (UIMA [Ferrucci and Lally, 2003], IBM Watson [Lally et al., 2012]), and will also be used in this thesis.

Some systems also include other semantic resources, besides NER and coreference resolution. The main example is the use of Semantic Role Labelers (SRLs) to help design the generation rules, as knowing the arguments of the verbs allow a finer detail on the transformational actions to be taken [Chen, 2009, Flor and Riordan, 2018, Keklik et al., 2019, Lindberg et al., 2013, Mannem et al., 2010, Mazidi and Nielsen, 2014, Pal et al., 2010]. Another example is the use of WordNet to elaborate on what kind of entity the *Which* questions are about (for instance, *Which country*), based on the hypernym of the target `NP` [Varga and Ha, 2010].

More recently, Araki et al. [2016] developed a QG system based on strong semantic features, extending the scope of the target from a sentence to a paragraph. It analyses events described in paragraphs and what triggers those events, and, through a set of manually created rules, it generates a question regarding that event. However, this system was possible due to the existence of a labelled corpus with such semantic relations (events, triggers, coreference, etc.), making it highly dependent of such data. The proof of concept is, nonetheless, an important step for this research topic.

Finally, some systems employ strategies to raise better questions to the top. Mannem et al. [2010] ranks the generated questions in the end, according to the depth of the verb in the parsed graph (it uses a dependency parser instead of a syntactic parser). Heilman [2011] applies an overgeneration method, which produces more questions (and potentially more errors) to apply, then, a statistically ranking strategy, by using a linear regression model, which pushes quality questions to the top, improving this way the top-N precision of the system. Lindberg et al. [2013] also build a classifier based on the judgement provided by the human rater, to classify the questions on the learning value (one of the parameters rated).

All these systems rely in rules which are mostly manually created, but some systems also tried to automatically create the patterns necessary to generate questions. TheMentor [Curto et al., 2011, 2012] is an example that uses a similar approach to Ravichandran and Hovy [2002], who used this approach for Question Answering (QA): Question/Answer pairs of a given type are submitted to a search engine and sentences that contain both the question and answer terms are turned into patterns, by replacing those terms by syntactic tags, from which it is possible to generate questions of the original type. We believe this type of approach is under-explored, and we drew inspiration in this strategy to build our system.

2.1.2 Question Generation with Neural Networks

Recently, with the technological advancement of GPUs, and the creation of large datasets to help them, neural networks gained popularity in different domains, putting state of the art results. Different architectures exist, each with different properties, useful for a multitude of tasks, from image recognition to natural language generation. The range of possible inputs and outputs is wide, but the core idea of neural networks is the same: a neuron. The neuron is inspired by the human neurons and synapses, in which information flows through it, being modified by the neuron. In a neuron, information (the input) is modified (the output) by a set of weights. These weights are to be learned through time, by seeing many examples. Neural networks are, of course, not a single neuron, but rather a composition of many neurons, assembled in such way that information flows through the network and is modified towards the desired output by the weights on each neuron. It is out of the scope of this document to thoroughly discuss all architectures and theoretical concepts behind them, but we will address works that use neural networks for QG.

The typical approach when it comes to designing QG systems with neural networks it to look at it as a sequence-to-sequence problem [Sutskever et al., 2014]. Tasks that are generically named sequence-to-sequence are those which both the input and the output are sequences (typically sequences of words or characters, which we will refer to as tokens). The most common example is Machine Translation, where the system tries to take an input sequence in the source language and translate it to the target language (the output sequence). Regarding QG, the approach is similar, as the output sequence will be a question that is related to the input sequence.

Sequence-to-sequence systems use an encoder-decoder architecture. The idea is that the encoder processes the input sequence, creating a latent representation of it, which is used by the decoder to generate the output sequence. Both the encoder and decoder are identical, in the sense that they are typically[3] composed of recurrent units (LSTM [Hochreiter and Schmidhuber, 1997] or GRU [Cho et al., 2014]). These units are designed to parse sequence of inputs, keeping *history*, as internal state, of what have been parsed so far. For each input, the unit transforms it to the output, modifying its internal state to keep information of what has been processed. Thus, the next input will be modified in function of what has been processed before.

On the decoder end, the internal state is set, at first, with the representation coming from the encoder. Each output of the decoder is also its own input for the next step, allowing

[3]Transformers are also sequence-to-sequence models that do not use recurrent units, as we explain later.

to keep state of what has been generated so far. Each of the outputs is connected to an activation function (for instance, softmax), which will tell what token in the lexicon should be generated. Sometimes, a copy mechanism is included (pointer softmax [Gulcehre et al., 2016] and CopyNet [Gu et al., 2016] are examples), as some rare words (names, for instance) from the input sequence need to be included in the output sequence, but are unlikely to belong to the output lexicon.

However, it is on the encoder side that some issues try to be solved. First, long sequences can pose a problem, because recurrent units do not capture long range dependencies, so the information fades away with the size of the sequence. Secondly, at generation time, different information can be useful, and a single latent representation might not be enough to capture it. To tackle the first issue, another recurrent unit is added to parse the sequence in the other direction (usually referred to as bi-; for instance two LSTMs are called bi-LSTM), thus minimizing the effect of its size, being the latent representation a composition of both directions' outputs. To address the second problem, a called attention mechanism is added to the architecture, where, for each input, it is calculated how important that input is when compared with the whole sequence. The more important, the more *attention* it gets, which will help the decoder side to decide what focus on. Many mechanisms exist, with the most employed being the ones presented by Luong et al. [2015] (Du et al. [2017], Liu et al. [2019], Tang et al. [2017]) and Bahdanau et al. [2014] (Serban et al. [2016], Wang et al. [2017], Yuan et al. [2017]). These attention mechanisms are also the idea behind Transformers [Vaswani et al., 2017]. This is a sequence-to-sequence model that discards the recurrent units discussed before, implementing a stack of attention mechanisms instead, overcoming some of their limitations, both conceptual (long input sequences) and technical (training resources needed). Transformers established a new standard on neural networks state of the art, being the base for BERT [Devlin et al., 2018], a model that can be successfully fine-tuned for a variety of tasks.

There are a few works that use this type of architecture for QG, but the input sequences vary among them. To the best of our knowledge, only two works take a single sentence or paragraph alone as input, while the others require the answer span as well. The former approach is more realistic and mimics better what previous QG systems did in the past, which is to take raw text and generate questions without extra data. Du et al. [2017] use an encoder-decoder architecture of bi-LSTMs, with an attention and copy mechanisms, while Kumar et al. [2018] use a similar approach with the difference being that the loss function is adapted to integrate typical evaluation metrics like ROUGE and BLEU. Other works take an answer span as additional input, which leads the generation towards specific questions, but they require a previous annotation step [Yuan et al., 2017, Zhou et al., 2018] or an attempt to identify it beforehand [Liu et al., 2019, Subramanian et al., 2018]. Other works use similar approaches to create questions focused on conversational agents [Wang et al., 2018].

The additional input of the answer is also behind dual strategies, that train a model aimed at solving two tasks jointly. In our case, typically QA is the joint problem chosen, given the similarity between the task and QG. For instance, Wang et al. [2017] uses an encoder-decoder model that is conditioned on a secondary input, either the answer (QG), or the question (QA), which is used to solve both tasks, iteratively. Another example, by Tang et al. [2017], looks at QA as a ranking problem, ie, it uses the answer selection task only, while the QG part of the model does use the answer as an input, while trying to minimize a common loss function for both tasks. Despite some technical differences approaching the task of QG, and while in theory the idea is promising, results showed that the gains are small.

More structured inputs have also been studied. Serban et al. [2016] and Indurthi et al. [2017] were one of the first to apply neural networks to QG, but using sets of triples (relation, entity A, entity B) as input of their network. The architecture is still similar, sharing the components of attention and copy mechanism on the encoder-decoder implementation, with the exception that the encoder is designed without having a recurrent unit (as the input is not a sequence), but by creating an encoding of the triple's content.

In this work we will not implement a neural based system, but we will be using using one of the available QG systems in our evaluations as a state of the art system: Du et al. [2017].

2.2 Resources for Question Generation

In a growing wide web world, more and more tools and information sources are available to researchers. Many of the systems described before use some of these to some extent, and so will our system. In this section we briefly mention a couple of examples.

2.2.1 Annotated Corpora, Lexical Resources and Datasets

WordNet [Miller, 1995], a lexical database for English, is one of the most used resources and is used to search for synonyms, hyponyms, and other relations between words. In QA, this can be useful not only to query expansion, creating different queries meaning the same, but also when it comes to Answer Extraction and Selection, as these relations may be important to find potential answers (for instance, to know that whale is a mammal, when answering the question *What is the largest mammal?*) [Ferrucci et al., 2010, Mendes, 2013, Prager et al., 2000]. In QG it can also be used, for example, to better select the Wh-word to use when generating new questions [Varga and Ha, 2010].

There are also many annotated corpora available, such as PennTreebank [Marcus et al., 1993], QuestionBank [Judge et al., 2006], PropBank [Palmer et al., 2005], and NomBank [Meyers et al., 2004]. The first two are a collection of corpora where text is annotated with Part of Speech (POS), creating a *bank of trees* representing the sentences' (or questions') parsing structures. PennTreebank consists of text from different sources such as the Wall Street Journal, while QuestionBank has questions from collections such as the QA tracks from Text REtrieval Conference (TREC)[4]. The remaining are an extension to the PennTreebank annotations focused in shallow semantics: PropBank focus in the verbs' roles, while NomBank focus on the nominalization of verbs and their semantic roles (*her* `claim` vs. *she* `claims`). These banks are typically used to train parsers that can be used by QG systems on their sentences' annotation.

There are two other important lexical resources to note as well: FrameNet [Baker et al., 2003] and VerbNet [Kipper et al., 2000]. In these, verbs' annotations are more directed to the roles themselves, while in the treebanks above these are generically represented with agent and

[4]http://trec.nist.gov/data/qamain.html

Table 2.1: Example of FrameNet, VerbNet and PropBank organization for verb *hit* in the sense of hitting a target.

Resource	**Frame/Class**	**Other Verbs**	**Arguments**		
FrameNet	Hit_target	pick off, shoot	Agent	Target	Instrument, Manner, ...
VerbNet	18.1	bang, bash, click, dash, squash, ...	Agent	Patient	Instrument
PropBank	hit.01	-	Arg0	Arg1	Arg2

object/theme in the form of `Arg0` and `Arg1` (instead of `agent` and `instrument`, for instance, for the verb *hit*). Also, PropBank and NomBank only cover the instances present in the original corpora, in opposition to these resources, which try to capture the whole semantics of existing verbs, only including a few examples as reference. This means that, in the banks, each verb belongs to its own class. Table 2.1 exemplifies how these resources are organized.

VerbNet is somehow similar to FrameNet, but more focused in grouping verbs according to their syntactic behavior, alike Levin's classes [Levin, 1993][5]. As an example, the `Apply_Heat` frame and `Cooking` class[6] share most of their verbs (like *broil, cook, fry, sear...*), but VerbNet includes *pickle* as a cooking verb, while FrameNet puts it under the `Preserving` frame, together with verbs like *cure, dry,* or *salt.* There is also a project that tries to map all these resources, called SemLink[7], that can be accessed at the Unified Verb Index[8]. The downloadable content contains text files with the mapping between PropBank, FrameNet and VerbNet as shown in Table 2.1. Besides their usefulness for SRLs, these resources can be used by QG systems that use rules at matching time. In our system we will be using some of these semantic resources to help create and apply the patterns used, as WordNet and SemLink.

Regarding corpora that could be used in the context of QG, either for training neural networks or evaluating systems, there has also been some developments in the last few years. The first QG competition, Question Generation Shared Task & Evaluation Challenge (QGSTEC),

[5]Baker and Ruppenhofer [2002] discusses the difference between FrameNet and Levin's classes in more detail.
[6]FrameNet categories are called frames, while VerbNet's are called classes.
[7]`http://verbs.colorado.edu/semlink`
[8]`http://verbs.colorado.edu/verb-index/index.php`

provided both a small corpus of development and test[9]. The development set contains 81 sentences associated with a few questions (typically from one to four questions), while the test set provides 90 sentences from which systems were supposed to generate questions of the indicated types.

A similar corpus is Engarte, from the Answer Validation Exercise (AVE)[10]. This corpus is composed of question-answer-snippet triples. Systems competing in AVE must label those triples as "true" or "false", depending if the answer can or cannot be derived from the given snippet. This corpus is closely related with the task of QA, as the idea behind it is to equip systems with the means to understand if a passage contains the correct answer to the given question. However, the answer slot is typically filled with an entire passage, instead of the actual answer.

Smith et al. [2008] published the results the students obtained in a QG course project – Question Answer Dataset (QAD). It includes three datasets, one for each year, partitioned in four sets of topics, each containing ten documents. Those documents are cleaned Wikipedia articles, and are indexed by a single document containing all questions generated by student's systems; each question has also answers and a difficulty metric, measured by the annotators. Many questions do not have answer attached, and many more are of the type `yes/no`, thus lacking variability.

More recently Stanford published a large dataset, SQuAD [Rajpurkar et al., 2016], also based on Wikipedia articles. The original version (1.1) contains over 100,000 crowdsourced questions across over 500 Wikipedia articles, divided in isolated paragraphs and Q/A pairs associated with those. It was recently extended (version 2.0) to contain 50,000 more unanswerable questions [Rajpurkar et al., 2018]. With the popularity of Amazon Mechanical Turk (AMT) and neural networks, other large corpora have been published. NewsQA [Trischler et al., 2016], is similar to SQuAD, but uses news texts and contains over 100,000 crowdsourced questions across over 10,000 news articles. Unlike these two, MS Marco [Nguyen et al., 2016] was not crowdsourced, but created from Bing's queries. It has over one million questions sampled from Bing's queries, associated with over eight million passages retrieved by the search

[9]`https://github.com/bjwyse/QGSTEC2010`
[10]`http://nlp.uned.es/clef-qa/repository/ave.php`

engine. These datasets can be used to train neural networks, but can also be employed for QG evaluation, as discussed later in Section 2.3.

2.2.2 Parsers and Tools

The resources described above are used to train parsers, like the Stanford parser [Klein and Manning, 2003] (Heilman [2011] uses it trained on PennTreebank), the Berkeley parser [Petrov and Klein, 2007] (TheMentor [Curto et al., 2011] use it trained on QuestionBank), OpenNLP[11,12], or Charniak [2000] parser (trained on PennTreebank). SRLs, like the Illinois Semantic Role Labeler [Punyakanok et al., 2008] (trained on PropBank), Senna [Collobert et al., 2011] (used by Mazidi and Nielsen [2015]), SwiRL[13] (trained on CoNLL – Computational Natural Language Learning) corpora[14]), used by Pal et al. [2010], SEMAFOR [Das et al., 2010] (trained on FrameNet), and ASSERT [Pradhan et al., 2004] (trained on PropBank) – used by Mannem et al. [2010] and Chen [2009].

Other useful tools include Tregex and Tsurgem [Levy and Andrew, 2006], that allow one to query and modify a parsing tree through regular expressions. Heilman [2011], Kalady et al. [2010], Wyse and Piwek [2009] use these in their work to transform the sentences' parsing trees into the desired questions.

In this thesis we will use some of the parsers to process text (Stanford parsers and Senna SRL), and Tregex to parse subtrees. These were selected based on preliminary experiments, reporting results of 86.7 F1 (Stanford constituent parser [Klein and Manning, 2003]), 91.0 accuracy (Stanford dependency parser [de Marneffe et al., 2006]), and 75.49 F1 (Senna SRL [Collobert et al., 2011]).

2.3 Evaluation of Question Generation Systems

Evaluation of QG systems is still a complex task, due to the nature of the problem. Given that it is a generation task, there is not a definitive *right* or *wrong* answer, but rather a multitude of viable acceptable outputs. Because it is virtually impossible to create a dataset

[11] `http://opennlp.apache.org/index.html`
[12] We found no evidence of what corpora is used to train the system.
[13] `http://www.surdeanu.info/mihai/swirl/index.php`
[14] We did not find any reference to the year of the corpus.

that contains all possible acceptable outputs, evaluation procedures often rely on manual evaluation according to some criteria. Evaluating if a question is good or not is subjective and, therefore, manual evaluation requires more than one annotator. This process is, thus, time consuming and expensive. Additionally, it is not replicable, and, thus, new studies cannot be directly compared with previous efforts.

Works employing a manual evaluation setting have different parameters under scrutiny. To enumerate a few examples, Du et al. [2017] ask human raters about naturalness and difficulty; Flor and Riordan [2018] and Kumar et al. [2018] ask about grammar, semantics, and relevance; Heilman and Smith [2009] use a multiple-choice error list, including among others formatting, vagueness, and grammar; Mazidi and Nielsen [2015] use a 3-point scale to assess the quality of the generated questions.

For this reason, authors often look at using automatic metrics, like BLEU or ROUGE, to evaluate their systems. While this type of evaluation has been an option for similar Natural Language Generation (NLG) tasks, in QG many distinct good questions can be generated from the same input, i.e., the expected target has a wider range of options. This makes automatic metrics less reliable as for judging a system's quality, unless the reference is extensive to cover many acceptable alternatives. Most datasets only have a single question per sentence, including the more recent large datasets like the previously mentioned SQuAD and MS Marco, but since neural network-based systems are trained on such corpora, researchers end up evaluating on the same datasets. Kumar et al. [2018], Yuan et al. [2017], Du et al. [2017] and Liu et al. [2019] are a few examples of works evaluated on SQuAD (which we will also be using), with the last one using MS Marco as well.

In this work we use both automatic metrics and human evaluation in our experiments, discussing in detail the conclusions drawn from each setting. Addtionally, we study the impact of a reference's size on using automatic metrics, leading to the contribution of a large dataset aimed at evaluation QG systems, and we also study how automatic metrics and human evaluation are correlated.

2.4 Other Related Work

2.4.1 Pattern-based Approaches in Other Domains

Using patterns dates back at least to Riloff [1996], who used them to create dictionaries. This can be extended to the creation of ontologies, as shown in OntoLearn Reloaded [Velardi et al., 2013], a system designed to create a taxonomy from scratch in any domain. To do so, the system identifies relevant terms from the domain, expands that lexicon by automatically extracting definition sentences referring those terms and builds a graph-like taxonomy connecting those. Learning from a dataset of manually annotated definitional sentences[15], the system identifies relations following the pattern <`Def, V, Hyp, R`> [Navigli and Velardi, 2010], corresponding, respectively, to the term to be defined (`Def` – *a chiaroscuro*), the verb phrase describing it (`V` – *is*), the phrase that represents the hypernym (`Hyp` – *a monochrome picture*) and the rest of the sentence that carries some differentiation meaning to the relation (`R` – *in arts*). These are applied to the corpora for each term in the initial terminology, extracting different definitions (that is, hypernym relations) for them. These are used to extend the ontology and the algorithm continues recursively, using the new extracted hypernyms as targets in each new iteration.

Snowball [Agichtein and Gravano, 2000] is another relation extraction system that also use patterns. Applied to a single domain (organization/location), it learns patterns from a small set of seeds (pairs of the relation to be acquired). The patterns consist of two tags (the entities to extract) and three flexible fields: on the left, right and middle of the tags (<`left, tag1, middle, tag2, right`>). Without going into much detail, when multiple instances are found, if they are similar enough the corresponding patterns are merged (that is, the vectors for the three fields (`left, middle, right`) are combined and reweighed). Finally, to find new pairs of relations, the generated patterns are used to match sentences containing both locations and organizations.

Shima [2015] developed a pattern acquisition system for paraphrases, accounting lexical diversity. The idea is that most systems are limited by different formulations of a given

[15]Available at `http://lcl.uniroma1.it/wcl`.

pattern, like `X died of Y` and `X has died of Y`, and are not able to expand to other semantically equivalent formulations, such as `X fell victim to Y` or `X succumbed to Y`. The system starts from a few seeds, like Ravichandran and Hovy [2002] does, but it adds a iterative approach, where patterns acquired in one iteration are used to extract new instances that work as new seeds in the following iteration. To introduce lexical diversity, the system, in each iteration, gives preference to new unseen words (or that do not share the same root). However, to avoid semantic drifts, the system also filters a few too generic patterns, defined by not containing content words (that is, patterns that only contain stop words and symbols – pattern slots). This work shares many ideas from Pantel and Pennacchiotti [2006], who also presents a similar work to harvest semantic relations, relying on multiple iterations of pattern application and instance extraction.

Both Pang et al. [2003] and, more recently, Narayan et al. [2016] use finite-state automata to generically represent multiple paraphrases, providing this way a tool to generate new paraphrases through unexplored paths of the automata. These are graphs created by merging individual sentence automata (through their shared nodes in the syntactic tree), where each transition corresponds to the realization of one of sentence's tokens. For example, *12 people died* and *twelve persons were killed* share a syntactic structure of `(S (NP (CD ..) (NN ..)) (VP ...))`, so the noun phrase would be represented as three nodes (one representing the beginning of the sentence), each containing two transitions containing the realization of *12/twelve* and *people/persons*. This way, when exploring the new paths, a new noun phrase could be generated, like *twelve people*.

2.4.2 Case-based Paradigm

Case-based reasoning (or analogical learning) is a paradigm that can be more easily understood as an example-based approach [Aamodt and Plaza, 1994]. The general idea is that a system employing such technique does not use general knowledge only, but also specific knowledge acquired with previous situations (cases/problems experienced). New problems are, thus, solved by reusing solutions used on similar past occasions.

The parallel can be drawn for our domain: previously seen question/sentence pairs are supposed to be examples for future questions or sentences. The *solution* to be reused is the set of operations applied to transform the sentence into the question, or vice-versa.

Although it is not our goal to do a in-depth analysis of the state of the art in this area, we find important to mention a few works and ideas that can be useful in our work.

According to Aamodt and Plaza [1994], case-based reasoning follows four steps: 1) `Retrieve`, where one is supposed to retrieve the most similar examples, 2) `Reuse`, in which the knowledge of the retrieved case is reused (by just copying or adapting it), 3) `Revise` the found solution, and 4) `Retain` the pieces of the new experience that will be useful in the future.

The first two steps are already used by most pattern-based systems we presented throughout this chapter: finding a pattern to apply and using it matches this part of the cycle. It is, however, on the two other steps we want to make a significant contribution, in comparison to the state of the art systems: being able to `Revise` patterns used and `Retain` knowledge on how the patterns perform.

One example that merges this kind of approach and pattern-based paradigms are example-based dialogue systems. Murao et al. [2003] developed a system aimed at retrieving shopping information, where the corpus was collected through a Wizard-of-Oz and the requests and replies are searched, found and modified through pattern matching and slot filling. A similar system and approach was also used by Jung et al. [2006] and Lee et al. [2009], while Nio et al. [2014] collects turn-based interactions from movie scripts and creates a chatbot type of system, where examples are searched through semantic and syntactic similarity, measured as an weighted function of the intersection of WordNet synsets and ration of POS tags.

Another representation of analogical reasoning are the class of problems that fit into the template $[A : B = C : D]$, which states a relation between the four entities. These relations can be in different combinations (for example, A is to B as C is to D is a as valid reasoning as A is to C as B is to D), and range many meanings, depending on the elements. For instance, morphological relations of the form *wife* is to *wives* as *wolf* is to *wolves* state how to pluralize words, while semantic relations of the type *wheel* is to *car* as *window* is to *house* represent meronymy. Langlais and Patry [2007] use this type of approach to represent words to be translated. The final goal is to be able to translated unknown words through the existing examples. Refer to Miclet et al. [2008] for a more in depth analysis on analogical similarity.

2.5 Discussion

In this chapter we looked at past work relevant for our thesis, with special focus on QG, but also covering other pattern-based approaches to other tasks. We started by analyzing traditional QG systems, followed by the recent developments thanks to neural networks.

We then discussed the available resources for the task, covering resources both for development and evaluation of QG systems, discussing afterwards the limitations of the evaluation setting in this field.

Finally, we briefly discussed other uses of pattern-based approaches and the closely related Case-based Paradigm.

While presenting many of the works, we mentioned some limitations they have or how their interesting ideas could be applied in our system. Here is a summary of the major inspirations for our work:

- Our system will be a pattern-based approach, like many others, but that uses semantic features to create more powerful patterns, using different representations of the input sentence ([Ferrucci and Lally, 2003, Mazidi and Nielsen, 2015]);
- Our patterns can be learned automatically, instead of being designed at implementation time ([Curto et al., 2011, Ravichandran and Hovy, 2002]);
- Our system can learn new patterns with time, by seeing new examples ([Shima, 2015, Velardi et al., 2013]);
- The interaction with the user will be used as implicit feedback to score the patterns and, therefore, future questions generated by them ([Aamodt and Plaza, 1994, Heilman, 2011, Mannem et al., 2010]).

3 The Pattern-based Question Generation System

In this chapter we present GEN, the Question Generation (QG) system we have developed. From a seed, GEN creates semantic patterns that are then used to generate questions from raw text. As previously stated, we follow the ideas presented by Ravichandran and Hovy [2002] and Curto et al. [2011], where seeds are used to generate patterns. However, our patterns also contain semantic information. In addition, implicit feedback from experts is used by GEN to create new seeds and also weigh patterns. These weights allow the system to score the generated questions and rank them in future iterations. The following sections describe each part of GEN.

3.1 GEN overview

GEN uses a pattern-based approach, and these patterns are created from seeds. Each seed is constituted by a Question/Answer (Q/A) pair, along with a sentence from which the question can be generated and where the answer can be found (from now on the "answer sentence"). For instance, the Q/A pair *Who created the telephone?/Alexander Graham Bell* plus the sentence *Alexander Graham Bell is credited with inventing the telephone* can be used as a seed; the latter is the answer sentence.

If GEN, being given a new sentence, is able to "match" it with the original answer sentence, a new Q/A pair is generated, according with a pattern. In order to specify how a new sentence is transformed into a new Q/A pair, we take advantage of the relation (from now on the "alignment") between the seed components – the Q/A pair and the answer sentence. Each pattern is, thus, composed of:

1. The (lexical, syntactic, semantic) information associated with each answer sentence. With this information GEN will decide if a match is possible between the answer sen-

tence and a new given sentence. The patterns that can be applied to a new given sentence are selected this way. As we will see, different types of matches are possible, leading to more constrained/flexible matches;

2. A Q/A pair and an alignment between it and the answer sentence. This dictates how to generate a new Q/A pair from a new sentence.

The information associated to the answer sentence will decide the matching possibilities between it and a new given sentence. If the information is purely lexical, patterns will be too restrictive and not many questions will be generated; if the information is just syntactic, patterns will be too loose. Thus, we need to find a balance so that we are able to generate questions, but without introducing much noise in the process. That is why we have embraced semantic information. However, even semantic information can be too restrictive. Many systems in the past resort to a linear view of the (answer) sentence, following in variations of the previous patterns, by replacing, for instance, the sentence tokens by their lexical lemmas or Named Entities (NEs) tags, such as `Date` [Ali et al., 2010, Curto et al., 2011, Varga and Ha, 2010]. In this work, we employ a multiple view/information approach used in various tasks, like Question Answering (QA) (IBM Watson, using UIMA [Ferrucci and Lally, 2003]) [Lally et al., 2012], or QG [Mazidi and Nielsen, 2014]. This way, in each answer sentence, tokens are annotated with different data, identifying them as stopwords, as NEs, as belonging to a WordNet synset, etc.; in addition, answer sentences are parsed by a constituent parser, a dependency parser and a Semantic Role Labeler (SRL).

Figure 3.1 depicts the parse trees for the answer sentence *Alexander Graham Bell is credited with inventing the telephone*[1]. Constituent trees are useful to grasp sequences of tokens that make sense together, like noun phrases, whereas dependency trees are useful to grasp long distance relationships, like how *inventing* is associated to *credited.* Semantic Role Labelers provide yet another representation of a sentence, where chunks that represent a role are seen together. As we will see, how these sentences are parsed and represented internally is a core piece of how GEN works.

[1]Example first introduced in Chapter 2.

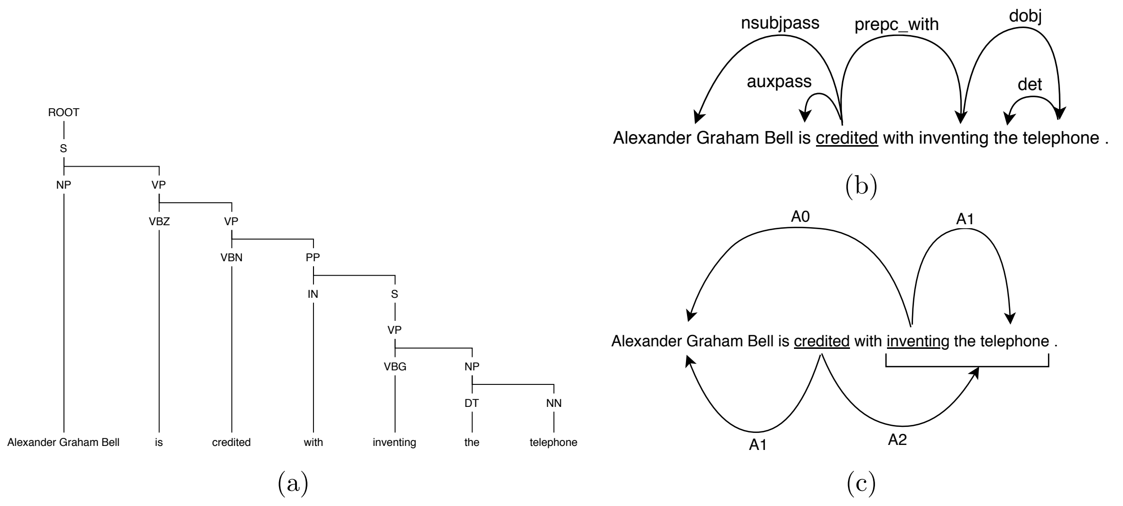


Figure 3.1: Parse trees for the answer sentence *Alexander Graham Bell is credited with inventing the telephone* obtained with (a) Stanford syntactic parser, (b) Stanford dependency parser, (c) Senna Semantic Role Labeler.

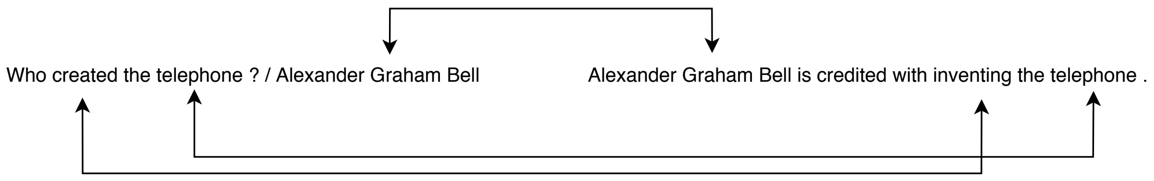

Who created the telephone ? / Alexander Graham Bell
Alexander Graham Bell is credited with inventing the telephone .

Figure 3.2: An alignment between the Q/A pair (*Who created the telephone?/Alexander Graham Bell*) and the answer sentence *Alexander Graham Bell is credited with inventing the telephone.*

In what concerns the alignment between the Q/A pair and the answer sentence, Figure 3.2 depicts this idea, by showing an alignment between the tokens of the Q/A pair with the ones of the answer sentence. Notice that the Wh-word is not aligned with the answer sentence. As we will see, this is why we need the Q/A pair in the pattern, so that the Q/A structure guides the generation of the new question.

Figure 3.3 shows a general overview of our proposal. In a first step, semantic patterns are created based on the seeds, in the Pattern Acquisition step (in green, in the bottom left of Figure 3.3). This step is detailed in Section 3.2. The top left part of the figure, in red, respects to the application of those patterns to new unseen sentences, resulting in new generated questions (Section 3.3). These two first steps were presented in a demo of

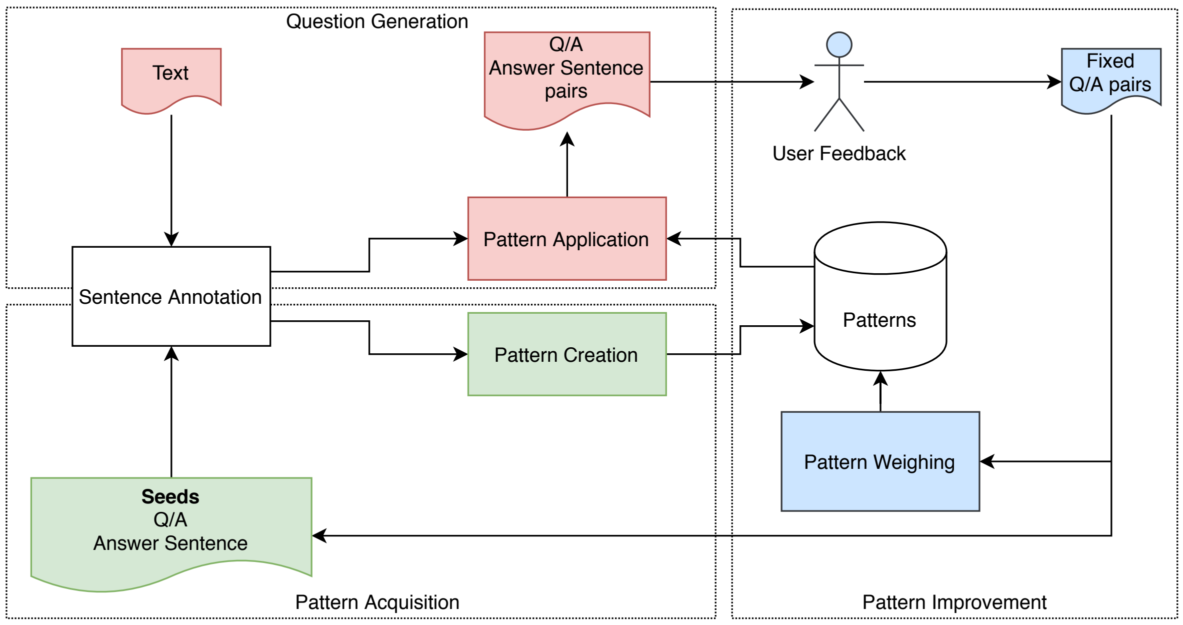


Figure 3.3: Proposed solution for GEN.

the system in its early stages [Rodrigues et al., 2016]. In Figure 3.3, on the right in blue, is also depicted the idea of using the implicit feedback of experts to create new seeds and weigh patterns across multiple iterations. This happens along the generation process, and is discussed in Section 3.4.

3.2 Pattern Acquisition

In this section we detail the Pattern Acquisition step (Figure 3.4). We start by formally defining the concept of a pattern (Section 3.2.1). Then, we describe the multiple view of the answer sentence (Section 3.2.2), and the alignment process between the Q/A pair and the answer sentence (Section 3.2.3).

3.2.1 GEN Patterns

We define a pattern $\mathcal{P}(Q, A, S)$ as a tuple $< Q/A, \mathcal{PA}(S), align(\widehat{Q/A}, S) >$, where:

- Q, A and S are the seed components, that is, they correspond to the Q/A pair and the answer sentence S, respectively;

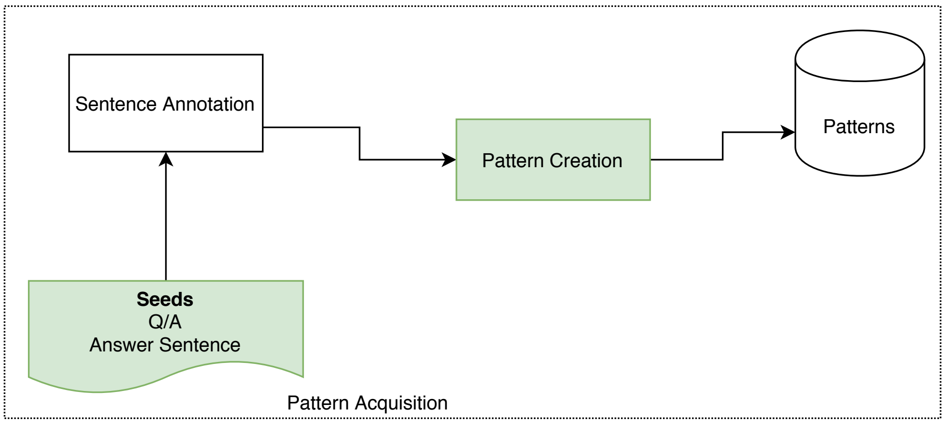


Figure 3.4: Creation of patterns from seeds (a Q/A pair and answer sentence).

- $\mathcal{PA}(S)$ is a predicate-argument structure that captures the different information about the answer sentence S;
- $\widehat{align(Q/A}, S)$ is the alignment between the tokens in Q/A and S.

This tuple contains all the needed information to create a new question Q' from a new unseen sentence S', as follows:

1. $\mathcal{PA}(S)$ is used to test how similar the new sentence S' and S are. If a match is found between $\mathcal{PA}(S)$ and a predicate-argument structure taken from S', $\mathcal{PA}(S')$, then S and S' are considered to be similar and the pattern can be applied to S';
2. If the pattern can be applied to S', then $\widehat{align(Q/A}, S)$, along with Q, establishes how to transform S' into the new Q-like question Q'.

3.2.2 Multiple View of the Answer Sentence

As previously mentioned, each sentence is enriched with information from multiple sources. At the token level, each token t is annotated with the following:

- Named Entities: if a word or multi-word expression is detected as a NE (we used regular expressions to extract dates in addition to the Stanford Named Entity Recognizer [Finkel et al., 2005]), it will be collapsed into a single token, and will be tagged with the NE type (for instance, `Person` or `Location`);

- WordNet: GEN identifies the synsets to which each token belongs, given by WordNet [Miller, 1995];
- Verb Sense: if the token is a verb, its frame and/or class is noted, according to FrameNet [Baker et al., 2003] and VerbNet [Kipper et al., 2000], respectively;
- Part of Speech (POS): the token is labeled with its POS tag (parsed by the Stanford parser);
- Word Embedding: the token is associated with its word embedding vector (by using Word2Vec [Mikolov et al., 2013]).

Then, at the sentence level, we use Stanford constituent and dependency parsers [de Marneffe et al., 2006, Klein and Manning, 2003] to create both constituent and dependency trees, and Senna SRL [Collobert et al., 2011] to obtain the semantic roles (as seen in Figure 3.1). The SRL provide us predicate-argument structures that we use to capture the semantic information of the answer sentence. That is, each predicate identified by the SRL in the answer sentence will generate a triple (the "predicate-argument structure") composed of:

- the root of the predicate (a verb);
- a set of arguments (associated to that verb);
- a set of subtrees, extracted from both the constituent and dependency trees, so that each subtree captures the arguments of the predicate.

Consider again Figure 3.1 and the answer sentence S, *Alexander Graham Bell is credited with inventing the telephone.* According to the SRL, the predicate *credited* will lead to the predicate-argument:

$$< credited, \{A_1, A_2\}, ST(A_1) \cup ST(A_2) >,$$

where $ST(A_1)$ contains all the subtrees that capture argument `A1` (*Alexander Graham Bell*) and $ST(A_2)$ contains the subtrees that capture argument `A2` (*with inventing the telephone*). For instance, examples of the latter are the following subtrees:

- `(PP (IN with) (S (VP (VBG inventing) (NP (DT the) (NN telephone)))))`,
- `(S (VP (VBG inventing) (NP (DT the) (NN telephone))))`,
- `inventing` $\xrightarrow{\texttt{dobj}}$ `telephone`, $\xrightarrow{\texttt{det}}$ `the`.

3.2.3 Alignment of Seeds' Components

As previously stated, GEN requires the seeds' components to be aligned, that is, it is necessary to align the Q/A pair with the answer sentence S. To perform such alignment, we require that:

R_1: the content of one sentence is included into the other. Without loss of generality, consider that the content of the Q/A seed is contained in the answer sentence S;

R_2: all tokens in Q/A are aligned with one and only one token in S;

R_3: no token in S is associated with more than one token in Q/A.

We find the best alignment between the Q/A pair and the answer sentence S, $\widehat{align(Q/A, S)}$, among all possible alignments, by satisfying Equation 3.1.

$$\widehat{align(Q/A, S)} = \arg\max_{a \in \mathcal{A}} \; score(a), \tag{3.1}$$

where $\mathcal{A}$ is the set of all possible alignments between Q/A and S that respect the previous requirements ($R_1 - R_3$), and $score(a)$ is the score given to alignment a. This score is given by the sum of the scores of each individual alignment, token-wise:

$$score(a) = \sum_{t^i \in T_{Q/A}, t^j \in T_S} score(align(t^i, t^j)), \tag{3.2}$$

where $align(t^i, t^j)$ is an alignment between tokens t^i and t^j, and $T_{Q/A}$ and T_S are the set of tokens in Q/A and S, respectively.

The alignment $\widehat{align(Q/A, S)}$ maximizes the alignment between the tokens from Q/A and S. As virtually any token in Q/A could align with a token in S, choosing the best set of token alignments is similar to the assignment problem, in which one is aiming at optimizing an utility function over a set of assignments – usually one tries to minimize the cost of doing X jobs by using X workers, each with a known hourly rate for each job. In our case we intend to pick the best pairwise token alignments that maximize the overall quality of the alignment $\widehat{align(Q/A, S)}$, given that we only use each token once. Therefore, instead of jobs and workers, we have tokens belonging to Q/A and S, each with a $score(align(t^i, t^j))$. So, being M a matrix of dimension $|t_{Q/A}| \times |t_S|$, each position M_{ij} contains the alignment score for $align(t^i, t^j)$.

The Hungarian algorithm [Kuhn, 1955] is a combinatorial optimization algorithm designed to solve the assignment problem. We adapted the Hungarian algorithm to our problem. The assignment problem usually takes an equal number of jobs and workers, but an adaptation is possible by adding the necessary dummy lines/columns if the matrix is not square. The original problem tries to minimize the utility function, while we are trying to maximize the value of the overall alignment. To make for this adaptation, we convert the values to have a minimization problem instead, replacing each cell by $max - M_{ij}$, where max is the maximum value present in the whole matrix.

Notice that for both Q/A and S we only consider for alignment tokens that are not stopwords, as the exact stopwords are unlikely to appear in the a new given sentence and are, thus, irrelevant to establish a relationship between them.

The missing piece for the alignment process is how tokens themselves are aligned and how those are scored. We designed five functions to this end, described next. Each description focus on the function's definition, and not exactly on how they are applied in conjunction. We opted to apply them in cascade (that is, if one fails, the next one listed is used), but these functions could be combined differently if we would like. All values were empirically set.

Lexical The first one, $equiv_L$, performs a lexical comparison of the two tokens:

$$equiv_L(t^i, t^j) = \begin{cases} 1 & \text{if } t^i = t^j \\ 0.75 & \text{if } t^i \neq t^j \wedge lemma(t^i) = lemma(t^j) \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

Verb sense $equiv_{VB}$ compares the sense of two tokens if they are both verbs and are related according to SemLink[2]. This resource is a mapping between PropBank [Palmer et al., 2005], VerbNet and FrameNet. If the two tokens belong to the same set in any of the resources, they are considered to match.

$$equiv_{VB}(t^i, t^j) = \begin{cases} 0.75 & \text{if } sense(t^i) = sense(t^j) \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

For example, *make* and *build* both belong to the frame `Building` of FrameNet, which would make the function return 0.75 for those tokens.

Named Entity $equiv_{NE}$ compares the tokens' content, if both tokens are NEs of the same type.

$$equiv_{NE}(t^i, t^j) = \begin{cases} 1 & \text{if } t^i = t^j \\ 0.9 & includes(t^i, t^j) \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

The function $includes(t^i, t^j)$ uses a set of rules (based on regular expressions) to determine if two tokens are referring to the same entity, or if two tokens represent the same date. For instance, both *Obama* and `D01 M01 Y2014` are included in *Barack Obama* and `M01 Y2014`, respectively.

WordNet $equiv_{WN}$ matches two tokens if their path distance traversing WordNet synsets is below a manually defined threshold. We compute the path distance by traversing the synsets upwards until finding the least common subsumer [Resnik, 1995]. For each node up,

[2]`http://verbs.colorado.edu/semlink`

a decrement of 0.1 is awarded, starting at 1.0.

$$equiv_{WN}(t^i, t^j) = \begin{cases} 1 & \text{if } syn(t^i) = syn(t^j) \\ x & \text{if } syn\big(hyp(t^i)\big) \supset hyp(t^j) \\ x & \text{if } hyp(t^i) \subset syn\big(hyp(t^j)\big) \\ 0 & \text{otherwise,} \end{cases} \tag{3.6}$$

where $syn(t)$ gives the synset of the token t, $hyp(t)$ gives the hypernyms of t, and $x = 1 - max(n \times 0.1, m \times 0.1)$, with n and m being the number of nodes traversed in the synsets of t^i and t^j respectively. If no concrete common subsumer is found, then 0 is the result returned. For example, *feline* and *cat* have the common synset `feline`, one node above where *cat* belongs, thus returning $1 - 0.1 = 0.9$. *Dog* and *cat* result in $1 - 0.2$, as one needs to go up two nodes for both tokens to find the common synset `carnivore`. We do not go up to generic synsets, like `artifact` or `item`.

Word2Vec $equiv_{W2V}$ computes the cosine similarity between the vector embeddings representing the two tokens t^i and t^j:

$$equiv_{W2V}(t^i, t^j) = \cos\big(emb(t^i), emb(t^j)\big), \tag{3.7}$$

where $emb(t)$ is the vector representing the word embedding for the token t. We use the Google News word2vec models available [Mikolov et al., 2013][3]. If the token is composed by more than one word (in the case of a NE for example), their vectors are added before computing the cosine similarity. For example, *car* and *vehicle* obtain a cosine similarity of 0.78, while *car* and *New York* result in a score of 0.07.

Notice that in our cascade approach, instead of returning a 0, each one of the previous functions call the next function in the list. For instance, our implementation of $equiv_L$ is:

[3]`https://code.google.com/archive/p/word2vec/`

Table 3.1: Scores obtained during the alignment process between the seed components: Q/A (column) and answer sentence (row).

	Alex. ... Bell	credited	inventing	telephone
created	..	..	0.75	..
telephone	..	..	..	1.0
Alex. ... Bell	1.0	..	..	..

$$equiv_L(t^i, t^j) = \begin{cases} 1 & \text{if } t^i = t^j \\ 0.75 & \text{if } t^i \neq t^j \wedge lemma(t^i) = lemma(t^j) \\ equiv_{VB}(t^i, t^j) & \text{otherwise} \end{cases}$$

Two of these functions ($equiv_{WN}$ and $equiv_{W2V}$) were used in 2017's SemEval Task 2 [Fialho et al., 2017].

In order to illustrate the whole alignment process, consider again the seed composed of the Q/A pair *Who created the telephone?/Alexander Graham Bell*, and the answer sentence *Alexander Graham Bell is credited with inventing the telephone.* Matrix M, in Table 3.1, contains:

- as rows, the tokens (that are not stopwords) from Q/A;
- in the columns the non-stopword tokens from S.

Each cell contains the similarity score obtained by the *equiv* functions for that pair of tokens. For simplicity, all `word2vec` values were ignored, as they are much lower (close to zero). In the end, the preferred alignment is the one seen in the previously shown Figure 3.2, chosen by the Hungarian algorithm, consisting on the cells seen in the matrix. A detailed analysis to the alignment method proposed can be found in Appendix A, where we compare its effectiveness against two state of the art aligners.

Finally, before creating the pattern, the predicate-arguments of S are checked for two conditions considering the alignment found. First, all tokens in $\mathcal{PA}(S)$ must belong to the alignment found, or the pattern is not created. Then, arguments can only contain tokens from

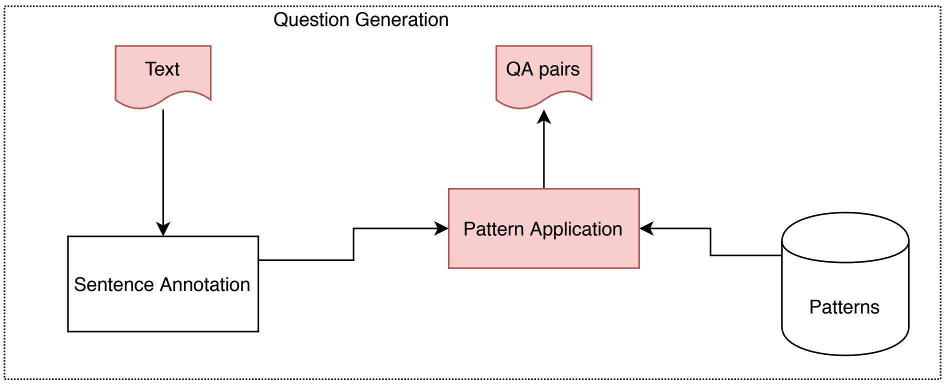


Figure 3.5: Question Generation process, from a new sentence S' into a new question Q'.

either the question or the answer, but never both. These two are enforced to make sure the arguments only have information about the Q/A pair and to be possible to distinguish the answer chunk from the rest of the question in the answer sentence. If the predicate-argument respects those conditions, then a pattern is created: $< Q/A, \mathcal{PA}(S), \widehat{align(Q/A}, S) >$.

Taking our running example, there are two predicate-arguments: one for *credited* and another for *inventing*. The first one contains the token *credited* itself, which is not part of the chosen alignment, so it is discarded. For the other, all tokens in the $\mathcal{PA}$ belong to the alignment, and each argument contains tokens from either only the question or answer. Therefore, a pattern is created.

3.3 Question Generation

In this section we detail how GEN takes the previously learned patterns and applies them to new unseen sentences, generating new questions. As discussed before, in order to apply a pattern to a new sentence, the latter needs to be "similar" to the answer sentence originating the pattern, so that a match occurs and a question is generated. In this section, we describe the Question Generation process (depicted in Figure 3.5), starting with the general conditions that need to be met so that a match occurs between sentences (Section 3.3.1) and moving to the match at the token level (Section 3.3.2) and at the tree level (Section 3.3.3). We conclude by presenting the final generation step (Section 3.3.4).

3.3.1 Sentence Matching

Let $\mathcal{P}(Q/A, S) = < Q/A, \mathcal{PA}(S), \widehat{align(Q/A}, S) >$ be a pattern, and S' a new unseen sentence[4]. GEN "matches" the two sentences if there is a predicate-argument (from now on $\mathcal{PA}(S')$) resulting from S' that "matches" $\mathcal{PA}(S)$. Following the previous definition of predicate-arguments (predicate's root, set of arguments, and set of subtrees), let these these be defined as follows:

$$\mathcal{PA}(S) = \ < p_S, \{A_S^1, ..., A_S^n\}, ST(A_S^1) \cup ... \cup ST(A_S^n) >,$$

and

$$\mathcal{PA}(S') = \ < p_{S'}, \{A_{S'}^1, ..., A_{S'}^m\}, ST(A_{S'}^1) \cup ... \cup ST(A_{S'}^m) >$$

The pattern $\mathcal{P}(Q/A, S)$ can only be applied to S' if the following conditions are verified:

$C_1 \ : \ equiv(p_S, p_{S'}) \neq 0;$

$C_2 \ : \ \{A_S^1, ..., A_S^n\} \subseteq \{A_{S'}^1, ..., A_{S'}^m\};$

$C_3 \ : \ \forall st_s \in ST(A_S^i) \ \exists st_{s'} \in ST(A_{S'}^j) : match(st_s, st_{s'}) \neq false.$

In other words, the predicate roots must be equivalent (Condition C_1), all arguments of $\mathcal{PA}(S)$ must be present in $\mathcal{PA}(S')$ (Condition C_2), and for each one of those arguments, the corresponding subtrees must *match* (Condition C_3). If these condition are met, then S' is transformed into a new question Q' by replacing the tokens in Q with the new tokens from S', as we will see in Section 3.3.4. However, before that, in Section 3.3.2 and Secton 3.3.3, we define the *equiv* and *match* functions, respectively. As we will see, these functions can be defined in different ways, leading to more constrained or flexible matches.

3.3.2 Token Matching

The *equiv* function could be defined by the functions presented in Section 3.2 (Equations 3.3 to 3.7), but can also be defined by a couple more functions. For instance, if we

[4]The same processing applied to S in the Pattern Acquisition step is applied here to new sentences S'.

define $equiv = equiv_L$, this function will be very restrictive, and, given a new sentence S', the pattern $\mathcal{P}(Q/A, S)$ can be applied to it only if S' has the same predicate as S. Therefore, we implemented more flexible functions that can be used as the *equiv* function.

Named Entity This function is a modification of Equation 3.5, $equiv_{NE}$. The previous version required that the entities were equivalent, which makes sense for an alignment task. However, when trying to generate new items from unseen text, we cannot put such a constraint on the process. Actually, any NE should be able to fill that slot on the pattern, as long as it shares the same type. The new version reflects this approach:

$$equiv_{NE}(t^i, t^j) = \begin{cases} 1 & \text{if } type(t^i) = type(t^j) \\ 0 & \text{otherwise} \end{cases} \tag{3.8}$$

Syntactic Noun This function tries to relax the matching process, making nouns to match independently of their semantic meaning. This can introduce much noise to the process, but it can also widen the generation task by putting less restrictions into the matching process.

$$equiv_{SynN}(t^i, t^j) = \begin{cases} 1 & \text{if } Pos(t^i) = Pos(t^j) = Noun \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$

Syntactic Verb Identical to the previous one, but tailored for verbs only.

$$equiv_{SynV}(t^i, t^j) = \begin{cases} 1 & \text{if } Pos(t^i) = Pos(t^j) = Verb \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

Once again, these functions can be applied in different ways. Notice also that we assume that there is alignment between two tokens every time the function *equiv* does not return 0.

Algorithm 1 Algorithm for tree matching.

```
1:  $match(T_1, T_2)$
2:  $align \leftarrow []$
3:  $n_1 = T_1.root$
4:  $n_2 = T_2.root$
5:  if $equiv(n_1, n_2) \leq 0 \vee |n_1.c| \neq |n_2.c|$ then
6:      return []
7:  else
8:      $align \leftarrow align(n_1, n_2)$
9:  end if
10: for all $i \in n_1.c$ do
11:     $a \leftarrow match(n_1^i, n_2^i)$
12:     if $a = []$ then
13:         return []
14:     else
15:         $align \leftarrow a$
16:     end if
17: end for
18: return $align$
```

3.3.3 Tree Matching

Function *match* captures the equivalence between two subtrees, so that the system decides if a pattern should be applied or not. Two (sub)trees match if they are structurally similar and their tokens match (according to the previous *equiv* function, in whatever way it is defined). We created several versions of the *match* function, some more flexible than others. This flexibility is not only associated with the *equiv* function, as seen before, but also with the match performed over the subtrees representing parts of the sentences.

Strict Tree Matching Algorithm 1 details the tree matching process, where $n.c$ represents the children of node n (a subtree or a token). It starts by comparing the roots of the two (sub)trees (Line 5). If the roots are equivalent – using the *equiv* function – and the number of children is the same, the algorithm is recursively applied to their children (Line 10). If the two trees are successfully matched recursively through their entire structure, an alignment between the two trees is returned, collected during its execution (Lines 8 and 15).

Subtree Matching This version is more relaxed than the previous one, as it accepts that arguments of $\mathcal{PA}(S')$ can have more elements in their subtrees when compared to the ones from S. In other words, instead of looking for a match in the subtree $ST(A^m_{S'})$, for a given argument A^m, we look for a match in all subtrees of that subtree. For example, if a noun phrase is accompanied by an adjective, and the pattern only expects a noun phrase alone, GEN will be able to ignore the adjective and match the noun phrase subtree.

Subtree Flex Matching Here we are relaxing *match* a step further. While in the above scenario we are still looking for structurally equivalent subtrees, with the `Subtree Flex Matching` we are looking to find subtrees that are just "similar". To do so, we use Tregex [Levy and Andrew, 2006] to create a flexible regular expression for tree nodes (for instance, *N** matches *NN*). Each subtree belonging to a pattern is transformed into a template that is used in the matching process. For instance, in the last section we created a pattern where the predicate-argument in the pattern had its argument `A1` represented by two subtrees. Taking `NP (DT (the)) (NN (telephone))` as an example, the following expression would be generated: `/N* << ( /D* $ /N* )`. This expression finds a subtree that starts with a noun (`N*`) that contains, as children at any level, a determinant and a noun (`D*` and `N*`, respectively). Considering again the recurring example, and a new given sentence *Vasco Da Gama discovered the sea route to India*, the chunk *the sea route* would be matched against *the telephone*.

Argument Matching Finally, we designed a more extreme solution in which the *match* function is true for any subtree. The idea here is that, for each argument in a pattern, the system should try to create a question by replacing the whole argument with the new one, ignoring the structure of the subtrees of either the pattern or the new sentence. Using *Vasco da Gama* example, the whole argument *the sea route to India* would match *the telephone* and, thus, replace it in the original question Q.

Finally, if the argument being tested corresponds to the answer A in the pattern, GEN checks for NEs on both subtrees being matched. If they exist, they need to be of the same type, or the generation process stops. If no NE is found in either, then the generation process proceeds as normal. The idea is to make sure that the new question is appropriately targeting the same kind of data, while not limiting GEN if the Named Entity Recognition (NER) fails to find a NE (or it does not exist).

3.3.4 Generation

If the predicate-arguments $\mathcal{PA}(S)$ and $\mathcal{PA}(S')$ match, then the new sentence S' can be used to create a question, by following the alignment between S and Q/A. However, the transformation of S' into a new question Q' (of the type Q) is done with the help of the alignments *align* returned by the previous defined *match* function. Each token aligned from S' to S can then be mapped to Q by following the alignment between S and Q/A, and they will replace the corresponding tokens in Q.

In other words, each token $t^k \in S'$ that was aligned with a token $t^j \in S$ that is mapped to a token t^i in the original question Q will take its place in the generated question. This means, thus, that non mapped tokens in Q will remain. For example, Wh-words in questions will not be mapped to tokens in S, which will be kept in the final new question, providing the same type of question.

This replacement is straightforward for both `Strict Tree Matching` and `Subtree Matching`. For the other two, which are more flexible, there might not be a direct alignment between tokens in S and S' (the system can match longer chunks in the new sentence). Therefore, for these two approaches, for each argument matched in the predicate-argument, we replace all tokens from Q which are aligned to tokens in S belonging to that argument. For example, *the telephone* is aligned with tokens in S belonging to the argument `A1`, so they will be replaced by all new tokens from S' that belong to `A1`.

Finally, we make an exception for the predicate-argument verb, which is the main verb of the question Q as well. Here, we conjugate the auxiliary verb in the question in the same mode of the new matched token (from Condition 1 of Section 3.3.1), adjusting the main verb accordingly. This is an attempt to adjust the question formulation to the current sentence S' conjugation used. For instance, if the pattern contains a question regarding a past event but the new sentence regards something yet to happen, it makes sense that the new question does not use the past sentence.

Considering again the running example and the new given sentence *Vasco Da Gama discovered the sea route to India*, because *discovered* and *inventing* are related (*equiv* function), and both have two arguments in their sentences, `A0` and `A1`, the sentences will match. Then, what remains to be tested are the subtrees that cover both arguments of each predicate. The tokens themselves will also match through semantic function *equiv*. For instance, *Vasco Da Gama* and *Alexander Graham Bell* are both NEs of the same type, so they are considered to match. This will result in generating different questions, depending on the tree matching strategy employed. It will not produce results using the `Strict Tree Matching`, as the subtrees representing each argument `A1` are not structurally identical, but will generate questions for the other strategies. For instance, the question *Who discovered the sea route to India?/Vasco Da Gama* is generated with the `Argument Matching` strategy, by replacing the tokens in the original question with the ones from the new sentence's corresponding arguments.

3.4 Improving Generation with Expert's Feedback

If QG systems are used as an authoring tool for professors when creating content for a educational system, there is the need for the professor to manually curate those questions, not only selecting the most appropriate, but also correcting them of any mistake they might contain. This implicit feedback is never used by systems as a source of reliable data, being this way wasted. We think any correction made by a professor can play an important role in the development of a QG system. As a consequence of the professor's work, every pair constituted by a sentence in the learning material plus a generated question can be used by the system, after the teacher's corrections, as a new seed. This allows the system to enlarge its pool of available patterns, increasing its generation power. Nevertheless, this might also lead to a possible problem of over-generation. If a teacher needs to parse dozens of questions to find a good one (whatever her evaluation criteria is), then the system's usefulness might not be that interesting. Some QG systems, such as the one described by Heilman [2011], already rank the generated questions, pushing the better ones to the top, although not taking advantage of human feedback. In this section we show how we take advantage of the teachers' feedback, not only to create new seeds, but also to indirectly evaluate how well the patterns are behaving. The main idea is to use the corrections made by humans as a mean to evaluate the quality of the pattern that generated the edited question. Questions needing major fixes are probably from worse patterns, while questions not requiring much editing are likely to

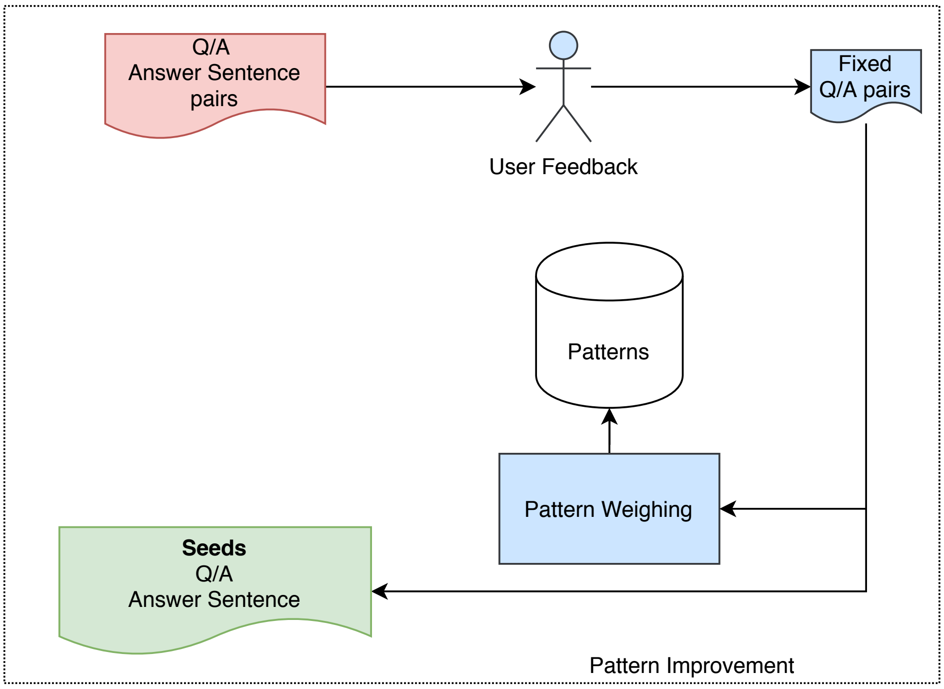


Figure 3.6: Creation of new Q/A pairs to be used as seeds in future iterations, along with the pattern weighing step.

come from well behaved patterns. All generated questions with previous patterns can be used to augment the pool of available seeds. Figure 3.6 depicts this idea.

This idea is closely related to the concepts of case-based reasoning introduced in the last chapter, in Section 2.4.2. In the last sections we described how GEN uses seeds to learn new patterns and then applies them, which corresponds to the first two steps of case-based reasoning [Aamodt and Plaza, 1994]: `Retrieve` and `Reuse`. Now this section approaches the last two steps, `Revise` and `Retain`. The former is done by the user, when correcting the questions generated, while the latter is done indirectly, by scoring the patterns.

This section is divided into two parts: the first corresponds to the validation of the generated Q/A pairs to be used by the system in a new pattern acquisition step. The second discusses the evaluation of the patterns used by the system. This part of the work borrows ideas from works like Mendes [2013], Pantel and Pennacchiotti [2006] and Shima [2015], discussed in the Related Work (Chapter 2), and preliminary results were published [Rodrigues et al., 2018].

3.4.1 Learning New Seeds

Although the task of learning new seeds can be done with no quality control at all, it might be useful to guarantee the correctness of the generated Q/A pairs (as well as the answer sentence) before adding them to the new set of seeds to be used. Having a human rating the system's output is costly, but necessary because there is no *right* question to ask about a given text or sentence, but rather multiple questions can be valid. Therefore, the questions and answer sentence are presented to the user, who assesses them. Given this feedback, the system gets to know what questions were correctly generated and, thus, are a good source to be a new seed.

Each pair of new Q/A and answer sentence S can then be used as a new seed pair, feeding the system into a new Pattern Acquisition step. As typically the new sentences are different from the original, and the questions themselves can be edited in such a way they become different from the patterns' questions, this will lead to the creation of new patterns, enlarging this way the pool of available patterns.

3.4.2 Pattern Scoring

Given an expert in the loop we can take their feedback to score the patterns. Let x be the score of a pattern $\mathcal{P}$. This score starts at 1.0 and is adjusted along the way, in function of the generation task:

$$x^{t+1} = update(x^t, Q', Q''),$$

where *update* is a function that takes a score x^t, a new generated question Q' and its edited version Q'' to compute a new score x^{t+1}. This *update* function requires two things:

- a method that evaluates the difference between the generated question, Q', and its edited version, Q'',
- a way to incorporate that score into x^t.

We use lexical metrics like to measure how similar both questions are. Then, to update the score, we apply the Weighted Majority Algorithm [Littlestone and Warmuth, 1994], or

Exponentially Weighed Average Forecast [Cesa-Bianchi and Lugosi, 2006]. The original concept for these strategies cannot be replicated, but we adapted them for our scenario. We treat each pattern as an expert, and the generated questions as guesses from the experts. The better the guesses (that is, the more successful the generation of questions is), the better rating the expert (the pattern) will have. The successfulness of a pattern is determined by the similarity between the questions it generates and their corrected versions:

$$successful(Q', Q'') = \begin{cases} 1 & \text{if } sim(Q', Q'') > th \\ 0 & \text{otherwise} \end{cases}, \tag{3.11}$$

where th is a threshold and sim a similarity function. For sim, we considered Overlap and a normalized version of Levenshtein [1966]. The latter gives an intuitive way to evaluate the editing effort of the human annotator in correcting a question. In specific, it gives us the edit cost considering the following operations at word level: (a) adding a new word; (b) eliminating a word; (c) transforming/replacing a word. We set the same cost for the three operations. We opted to use Levenshtein's normalized version, that takes into consideration the size of the longest question (the Levenshtein value is divided by the size of the longest question), which normalizes the scores into the range [0-1].

The score of a pattern is updated at each step based on its previous score and the technique used, as described below.

Weighted Majority Algorithm This technique penalizes a pattern if it is not successful and rewards it otherwise:

$$w_t(\mathcal{P}) = \begin{cases} 1 & \text{if t = 0} \\ w_{t-1}(1-b)^{-1} & \text{if successful} \\ w_{t-1}(1-l) & \text{otherwise} \end{cases} \tag{3.12}$$

where l is the loss penalty for a non successful generation and b the bonus reward for a successful generation. If b is set to 0, then no bonus is awarded, becoming the score a decaying factor only.

Exponentially Weighed Average Forecast Here the score of a pattern is updated by a decaying factor relative to its successfulness, i.e., the better it performs, the less its score is penalized:

$$w_t(\mathcal{P}) = \begin{cases} 1 & \text{if t} = 0 \\ w_{t-1} * e^{-l * \frac{1}{similarity}} & \text{otherwise} \end{cases} \tag{3.13}$$

where l is the loss penalty and *similarity* is the value of similarity given by the function *sim*.

3.5 Discussion

In this chapter we introduced GEN, our QG system. It employs a pattern-based approach using semantic features and different matching strategies to generate new questions from unseen input sentences. It draws some ideas from past works in different areas, namely the concept of using multiple views of text [Lally et al., 2012, Mazidi and Nielsen, 2014], and learning patterns from seeds [Curto et al., 2011, Ravichandran and Hovy, 2002]. Additionally, GEN has a learning component new to QG systems, which uses implicit feedback from the user to improve its performance over time. This is done in two ways: by scoring the patterns that generated those questions to which GEN is getting feedback, and also by using the corrected versions as new seeds to learn new patterns. Previous works did question reranking [Heilman, 2011] and automatic pattern learning [Curto et al., 2011, Ravichandran and Hovy, 2002, Shima, 2015], but never as a whole cohesive strategy using real user interactions.

We must note that our implementation has some empirically set parameters and uses some heuristics. However, GEN was created modularly and all choices showed to be reasonable during its design. Also, despite some fixed settings, GEN is prepared to be easily extended. For example, more annotators (like NERs) or parsers can be added, thus allowing more detailed representations of the sentences parsed. This also means GEN could be used in other languages, given the resources exist, as the overall concept is language agnostic, depending solely on having access to the required resources for the target language. Additionally, the *equiv* function could use more specialized forms of token matching, or different approaches to the used features could be employed, and the tree matching process is prepared to propagate their scores from the leaves to the top.

Computationally, the pattern acquisition step mainly depends on the Hungarian algorithm (parsers aside), which solves the assignment problem in polynomial time of the number of tokens. At generation time, GEN matching functions rely in tree matching, which, depending on the strategy, can go from linear to quadratic time complexity, in the worst case scenario. GEN also relies on external parsers (Stanford's constituent and dependency parsers, and Senna SRL [Collobert et al., 2011, de Marneffe et al., 2006, Klein and Manning, 2003]), which have an associated error rate themselves. It is possible that such errors can limit GEN's generality. However, as GEN learns patterns and generates questions based on the same parser outputs, it is expected that GEN, at least partially, overcomes such limitations, given those errors are consistent. Additionally, new parsers can be incorporated, which means more specific topics or data can always be addressed.

4 The Monserrate Corpus

In this chapter we introduce Monserrate, a new exhaustive (gold) reference to allow researchers to perform automatic evaluation of Question Generation (QG) systems. We first draw attention to the need of such corpus (Sections 4.1 and 4.2), then describe our thorough process of data collection (Section 4.3), and finally study the impact and utility of such corpus in Section 4.4, addressing Hypothesis 4.

4.1 Motivation

Many metrics used in evaluation processes, like BLEU, assume the availability of a corpus in which its instances should cover the set of possible targets. Namely, in QG, the reference should have many question-hypotheses associated with each sentence, covering many formulations. However, to the best of our knowledge, there is no corpus like that and, more often than not, corpora only contain a single reference per sentence, which can introduce unfair factors into automatic evaluation, by rewarding a system for hitting the reference hypothesis, despite many different *good* questions might be generated and deemed acceptable from a single sentence.

With such limitation in mind, researchers often end up using human annotators to evaluate the quality of the generated questions. While perfectly accepted as a fair process of judging the quality of a system, this kind of evaluation process is expensive, time-consuming, and non-replicable. These factors are even more limiting at developing time, when one is looking for fast, but still precise, indicators of how the system is improving over time.

Thus, we started investigating how impactful the size of a reference can be, and how the different existing metrics behave for different reference sizes. Our hypothesis is that the size of a reference has implications on the perception of the results obtained, with focus on smaller

Table 4.1: Examples from FatSQuAD, with only the first question belonging to LittleSQuAD.

Sentence	**Questions**
For instance, a snare drum sounds higher pitched than a bass drum though both have indefinite pitch, because its sound contains higher frequencies.	*Which drum has a higher perceived pitch even though they both have indefinite pitch?* Which drum has a higher frequency sound? How does the pitch of snare drum relate to bass drum? Why does a snare drum sound higher pitched than a bass drum?

references.

4.2 Preliminary Experiment

In a preliminary experiment, we took a single article from the well know SQuAD dataset [Rajpurkar et al., 2018], presented in Section 2.2. The article was extracted from the test set, and contains 54 unique sentences. We evaluated how two state of the art systems performed when using the reference as it is (from now on LittleSQuAD) and using an extended version of it (FatSQuAD). A collaborative process was employed to create the new extended reference: 10 individuals extended the original 35 questions associated with the 54 sentences with brand new questions, enriching LittleSQuAD not only by adding new questions, but also by correcting each others' contributions. This resulted in 138 questions pairs being added to the original corpus, totalizing 173 questions for the 54-sentence article, which corresponds to almost five times more questions than in the original version (no questions were added to two of the sentences). Table 4.1 shows an example of the final corpus (the first question was already present in the original corpus).

4.2.1 Experimental Setup

We used two state of the art QG systems in this experiment, covering two different approaches to the problem, both presented in Chapter 2. The first one is the work of Heilman and Smith [2009], from now on H&S, and employs a linguistic driven rule based approach. The second one is the work of Du et al. [2017], from now on D&A, and is a neural network system.

We set up the former to prefer Wh-questions and limit questions to 30 tokens (*–prefer-wh –max-length 30*). We retrieved the top questions with a score above 2.5. This setting is similar to some studies performed with H&S. Considering D&A, we trained the network with the same configuration as the authors. The model generates a question for each input sentence, in both the sentence- and paragraph-level models. We chose the sentence-level model to keep all systems equal, and, also, to mimic the annotators who only had access to the individual sentences. The model was trained on a partition of SQuAD's training set (more details in Section 5.1).

We run the two systems on the set of 54 sentences and then use both LittleSQuAD and FatSQuAD as references to evaluate the generated questions.

As for the evaluation process, we used a publicly available library containing different lexical and semantic metrics. The Maluuba project [Sharma et al., 2017][1] contains lexical metrics typically used, like BLEU, METEOR, and ROUGE, and other metrics based on word embeddings: Embedding Average Cosine Similarity (EACS), SkipThought Cosine Similarity (STCS), Vector Extrema Cosine Similarity (VECS), and Greedy Matching Score (GMS). These are briefly detailed below.

4.2.1.1 BLEU

BLEU [Papineni et al., 2002] is typically used to evaluate machine translation systems and has also been used for QG, as it compares a candidate sentence with a reference of acceptable hypotheses. The BLEU score is computed by calculating a modified precision on the shared n-grams between the candidate and the reference. The values employed for n are typically between 1 and 4.

4.2.1.2 METEOR

METEOR [Banerjee and Lavie, 2005] is a metric that is supposed to correlate better with human evaluations. It aligns the candidate sentence with a reference sentence by performing

[1]`https://github.com/Maluuba/nlg-eval`

token alignment. The precision and recall of those alignments are used to compute a F-score that will lead to the final score.

4.2.1.3 ROUGE

ROUGE [Lin, 2004] is another lexical metric that also computes a F-score between the candidate sentence and the reference hypothesis. In the Maluuba's library, ROUGE_L is used, which is based on the Longest common subsequence between the two sentences.

4.2.1.4 Embedding Average Cosine Similarity

EACS computes the cosine similarity of two sentence embeddings. The sentence embedding is formed by averaging the word embeddings of each of the sentences' tokens.

4.2.1.5 SkipThought Cosine Similarity

STCS also computes the cosine similarity between two sentence embeddings. However, it is based on the Skip-Thought model [Kiros et al., 2015], a recurrent network trained to predict the next and previous sentence of the input sentence, which is encoded into a sentence embedding. These embeddings showed to have good performance in semantic relatedness, and are used here as an alternative to averaging the sentence's token embeddings.

4.2.1.6 Vector Extrema Cosine Similarity

VECS [Forgues et al., 2014] also computes the cosine similarity between two sentence embeddings, but in this case each embedding is created by taking the most extreme (maximum) value among all token embeddings, for each dimension.

4.2.1.7 Greedy Matching Score

This is the only embedding-based score that does not use sentence embeddings. Instead, it takes each token embedding in the candidate and maximizes its similarity with a token on the reference, summing all those scores for all tokens. Then it performs the same task inverting the candidate and reference hypothesis roles, and averages both scores [Rus and Lintean, 2012].

Table 4.2: Average scores obtained on LittleSQuAD, for H&S and D&A, measured by automatic metrics.

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
H&S	**36.30**	**21.74**	**33.72**	**9.29**	**83.97**	**72.75**	**47.78**	**54.53**
D&A	29.25	15.94	33.48	3.95	82.52	69.10	42.83	51.98

Table 4.3: Average scores obtained on FatSQuAD, for H&S and D&A, measured by automatic metrics.

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
H&S	**57.53**	**37.95**	**65.54**	**33.27**	**91.45**	**83.50**	**64.57**	**72.31**
D&A	44.97	23.04	51.54	9.72	89.34	77.72	56.44	61.77

4.2.2 Results

H&S and D&A generated 191 and 54 questions, respectively. Results are shown in Tables 4.2 and 4.3. Table 4.2 contains results for LittleSQuAD and it shows how close the two system appear to be, especially when considering metrics like BLEU1, so typically used to evaluate QG systems with SQuAD, while Table 4.3 shows a clear change on how these two systems compare: H&S is clearly above D&A in all metrics.

These results show two things: first, the size of the reference seems to translate into different perceived performances, even for such small experiments like this. Secondly, the metric used can dictate different analysis to the results. Even if what is perceived to be the better system does not change in this case, how close they are might be misleading depending on the metric used to assess that.

Therefore, we decided to fully study this phenomena, by creating a new reference aimed at helping conduct automatic QG evaluation. Hypothetically, if a reference would contain all possible formulations of all questions possible to ask about a given sentence, then an automatic metric could be used to compute a score that would distinguish two different systems. Although the goal of building a dataset that covers all possible natural questions is unattainable, by using the appropriate methodology it is possible to build a corpus that plausibly contains most of these questions, covering a large percentage of the cases. This led us to Hypothesis 4, which we address in this chapter.

Table 4.4: Statistics of MONSERRATE corpus.

	$\#Instances$	$\#Words$	$len(Instances)\ {}^{\text{min}}_{\text{max}}$	Avg $Words/Instance$
Sentences	73	1657	[6 - 62]	22.70 ± 9.97
Questions	1916	14608	[3 - 24]	7.62 ± 2.70

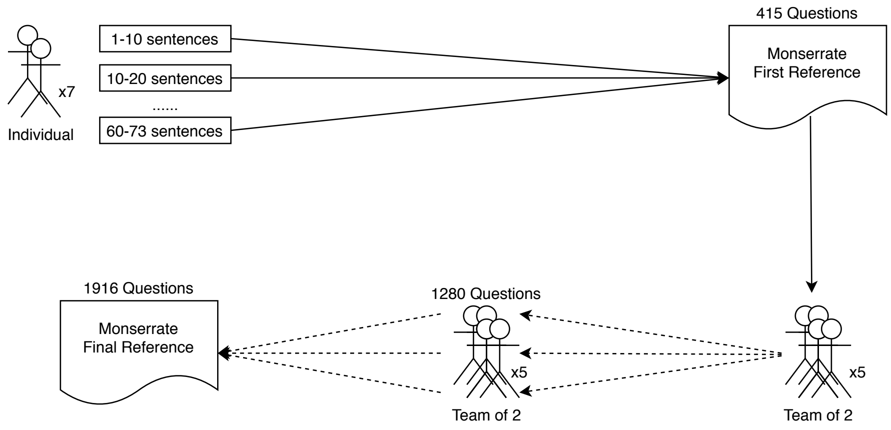


Figure 4.1: Overall process used for acquisition of MONSERRATE. At first, 7 individuals created 415 questions. Then, five teams of 2 individuals were asked to create more questions and inter-correct all questions, leading to the final 1916 questions.

4.3 Monserrate Corpus

In this section we detail the process of building a reference corpus for QG evaluation that we have named MONSERRATE. We collected two online texts, in English, about Monserrate palace, containing 73 sentences[2]. Table 4.4 shows some statistics about these texts.

Overall, a panel of 17 individuals has been involved in the process of creating MONSERRATE, although, purposely, different sets of individuals have participated in different steps of the corpus generation process. All individuals were non-native English speakers, although with high proficiency. The corpus was built using a three step process, where questions have been produced, filtered, revised, and combined. Figure 4.1 depicts the overall process.

In the first step we asked 7 individuals to create all the questions they considered that could be extracted from the sentences in the corpus. Each individual was assigned between 10 and 11 sentences, selected at random. This resulted in an initial corpus with 415 questions (reported in Rodrigues et al. [2018]). A simple assessment of its coverage confirmed that

[2] `https://github.com/hprodrig/MONSERRATE_Corpus`

many natural questions were left out.

The task of extending the initial corpus was then assigned to the remaining members of the panel. This group consisted of 10 different individuals. For the second and third steps (see below), the subjects were grouped in teams of two, in order to encourage discussion among the pairs, allowing each pair to both spend more time in new less obvious questions and introduce less errors.

In the second step, the groups were asked to correct the first 415 questions. From the original corpus, 42 questions were corrected by at least one group. Then, in another round, all the groups assessed the corrections and decided if they should be accepted, rejected, or if both versions were acceptable. 37 of the 42 corrections were voted to replace their previous version, while 5 were considered as good as the original question.

Finally, in a third step, the different groups were also asked to extend the corpus with new questions. 1280 questions were created across the five groups. Afterwards, all questions created by each group were validated by two other groups. Table 4.5 shows the agreement for this last step. Agreement exists if both groups either mark a question as valid or fix/edit the question in the exact same way. For example, the first row of Table 4.5 shows that the first group created 205 questions, and the two groups validating those questions made the same decision for 78% of them. Kappa Cohen metric [Cohen, 1960] is usually reported to measure inter-annotator agreement. However, the task performed by the 5 groups, despite having a big percentage of agreement, has one type of agreement which occurs much more often: agreement on not modifying the question (accepting it as is). This means there is a problem not in disagreement quantity, but in agreement allocation. Figure 4.2 tries to depict this phenomena. Pontius and Millones [2011] argue that the traditional kappa Cohen metric is not suitable in many cases, as it may not be informative, and defend the use of disagreement and allocation quantities as the numbers to consider. Here we report one modified kappa metric more suitable for our task: k-quantity [Pontius and Millones, 2011]. The values obtained (Table 4.5) are considered to indicate moderate to good agreement between the annotators [Landis and Koch, 1977].

After the groups' inter-validation, and without counting repetitions, our reference cor-

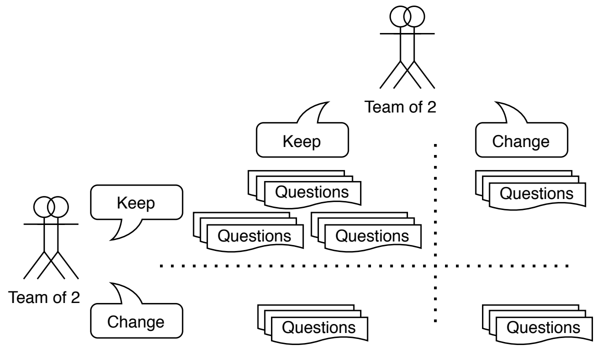


Figure 4.2: Illustration on how agreement between teams breaks down. A large portion of the questions are labeled as to keep by both teams, which leads to a high agreement. However, traditional kappa Cohen is not sensitive to this.

Table 4.5: Agreement on reference cross-validation.

	Questions	Agreement %	k-quantity	k-standard
g0	205	78	0.84	0.24
g1	250	78	0.86	0.12
g2	393	91	0.97	0.01
g3	387	87	0.95	0.16
g4	106	76	0.84	0.41

Table 4.6: Example questions associated with a sentence in Monserrate.

Sentence	**Questions**
When you buy the ticket, you will receive a map which allows you to go around easily by yourself.	How can I get a map? How can I get a map of the palace? What does one receive upon buying the ticket? What will you receive when you buy a ticket? Why is a map useful?

pus was extended to 1916 questions. Table 4.4 reports relevant statistics regarding the final content of the corpus and Table 4.6 shows a small sample from it. To the best of our knowledge this is the first corpus of this type. Although we cannot claim that it contains all the questions that can be possibly extracted from the given text, Monserrate is certainly more "exhaustive" than the existing data that was created with similar or other purposes, and it is publicly available[3].

[3]`https://github.com/hprodrig/MONSERRATE_Corpus`

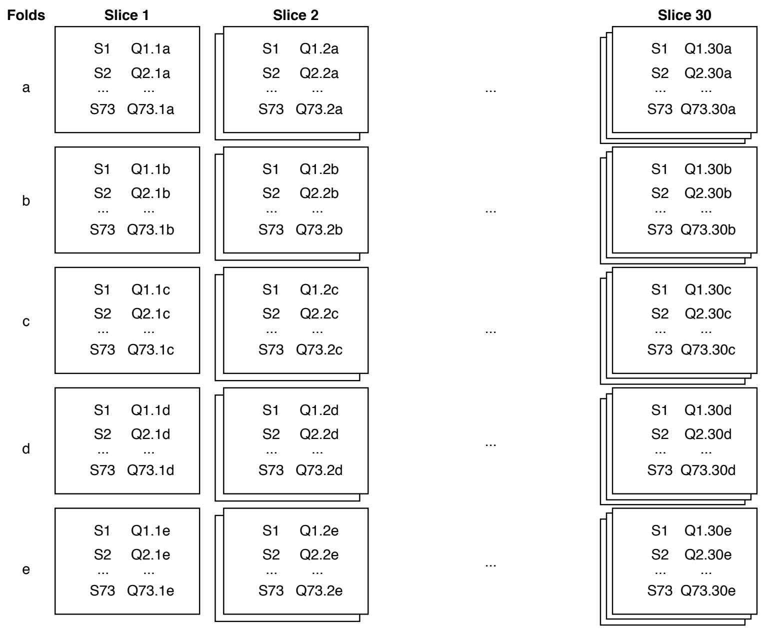


Figure 4.3: Each slice is composed of at most N questions associated with each sentence, with N going from 1 to 30. Due to randomness, we consider 5 different folds per slice, which are augmented in parallel (represented from `a` to `e`).

4.4 Is Monserrate a Fair Reference for QG Evaluation?

Our hypothesis is that current datasets are not suitable for automatic evaluation, because despite being large, they are small at the reference level. MONSERRATE is a much larger reference, but is it enough to get reliable results? Is it necessary to have such an exhaustive corpus? In this study, we try to understand the impact of the size of the reference in the automatic evaluation of QG systems.

This experiment looks at MONSERRATE in what we call "slices", starting at one question per sentence. In other words, we cut the reference in such way that each sentence has only one question as reference. Then, sequentially, each slice is augmented, one question (per sentence) at a time, up to 30, making it a larger reference. Results are, obviously, expected to improve over time, but what we are looking for is how long they take to stabilize – if that happens at all. Additionally, as a slice is randomly picked among the whole reference, we perform five different slices (working as five folds) and show their average results. At each step, each slice is extended separately and randomly. Figure 4.3 shows the concept of slices and folds.

We use the same experimental setting as before (Section 4.2.1), with the addition of our own system, GEN. The bootstrap of our system was done using 8 seeds (which can be consulted in Section 5.1).

In Figure 4.4 we can see the average metrics' score for the 5 folds at each slice, with sizes from 1 to 30, and also considering the whole reference. The trend is similar for all metrics (remaining BLEU scores not shown, but behaving similarly): results are less distinguishable at smaller slices, with larger standard deviations, which means that the reference used can change the interpretation of the results significantly. However, as we enrich the reference, it becomes slightly easier to tell the systems apart for the lexical metrics (ROUGE, METEOR, BLEU), as the error bars no longer intersect. Additionally, as the standard deviation of the five folds approaches zero, the collection is close to the point where the selection is exhaustive and has no impact on the overall results. Finally, the lines of all graphics behave similarly, approaching a plateau at some point, typically around slice 17, which means that adding more data to the reference has a small impact on the overall results from that point on. Results also seem to indicate that the metric chosen is not as important as the quality of the reference, if we are considering the overall average results only.

Embedding-based metrics seem to behave more identically for all systems, and reach the plateau earlier, which is worth discussing. Since embeddings are related to the *meaning* of the words they represent, it makes sense that embedding-based metrics are less sensitive to changes in syntax, and more focused on the overall meaning. Given that many generated questions will be, in some form or shape, related to the hypothetical questions, these metrics will show less prowess to distinguish them, which means they are likely not interesting metrics to consider for this task. Take, for example, VECS, which seems to report constant scores across systems, being impossible to distinguish the systems.

As we observed, the figures suggest that around slice 17 metrics approach a plateau. One way to confirm this is by taking the change rate from one slice to the next. In other words, the derivative of the curve will tell how the curve is evolving. Reaching a plateau means the derivative is approaching zero, which we can use to establish a threshold to which we would say the results are not improving enough. Figure 4.5 shows the derivative curves for all metrics

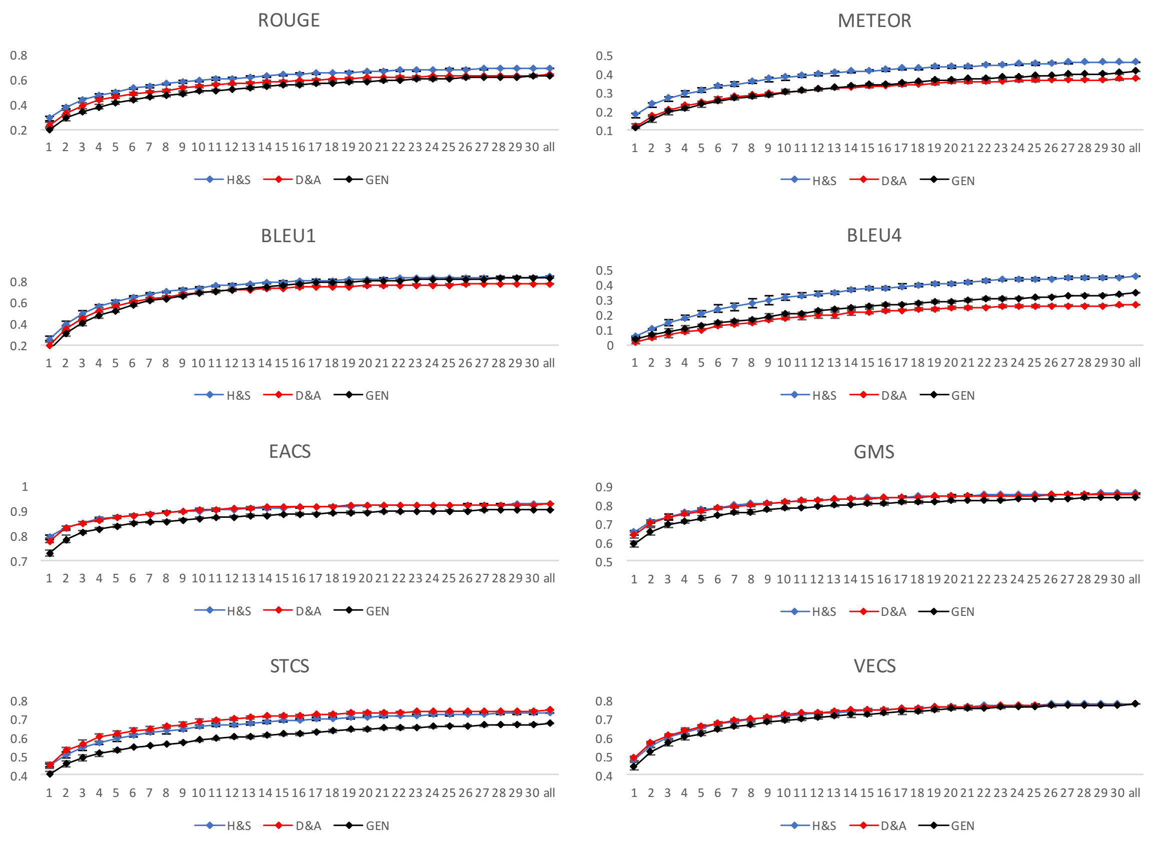


Figure 4.4: Scores with automatic metrics, averaged on 5 folds, using as reference the slices of size 1-30, and the whole reference (all).

and systems, and a threshold set to 0.01. The graphs were cut on the left to make clear where the threshold is crossed. Results show that, with a few exceptions, it is from around slice 17 that results start to slow down their improvement, meaning that, system-wise, performances are hardly impacted with larger references.

4.4.1 Statistical Significance of Slice's Increments

For a given system it is of course expected that results for each slice are worse than the next ones, even if by a small margin, but the significance of those improvements has still to be tested. The point in which differences stop being significant shows the point where the reference can be considered exhaustive, as adding more data will not change significantly the performance obtained. This analysis should be done at the reference level, that is, comparing the scores obtained by each sliced reference instead of by each system. In other words, we

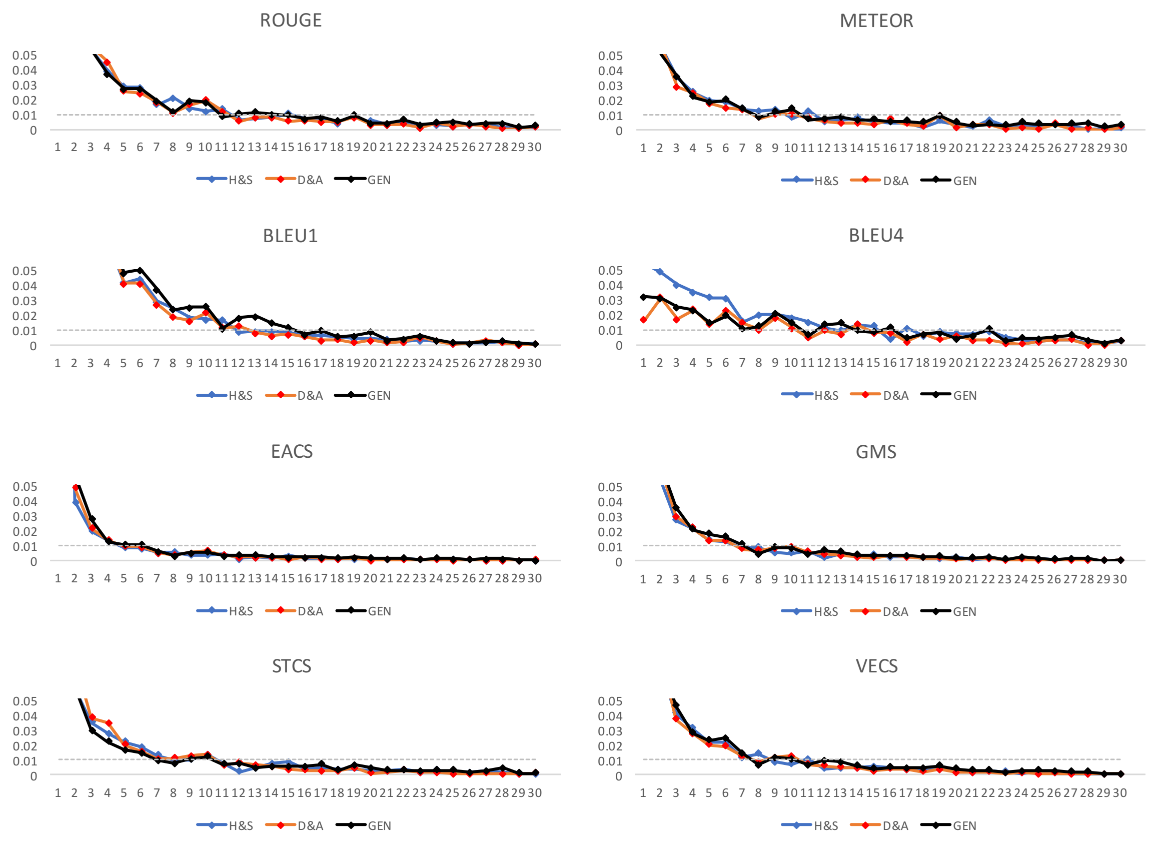


Figure 4.5: Derivate graphs for all metrics and systems presented in Figure 4.4, with a threshold of 0.1 represented by the dashed line. The graphs were cut on the left for readability purposes.

evaluate each sliced reference coverage against the different systems. Our hypothesis is that the size of the reference impacts the results. In other words, the null hypothesis can be stated as:

$\boldsymbol{H_0}$: *The scores obtained by a given metric do not change significantly by adding more questions to the reference.*

Figure 4.6 depicts a confusion matrix-style image where the t-test between each slice i and j is computed, considering the average of the five folds. Each t-test compares two sets of 73 instances (size of the corpus at sentence-level), where instances are the scores each system obtains against the slice's reference for one of the 73 sentences. Each cell contains the result of the t-test between slice i and j, being colored in green if $p < 0.01$, meaning that a jump from slice i to j is significant. We can observe that all jumps are significant for

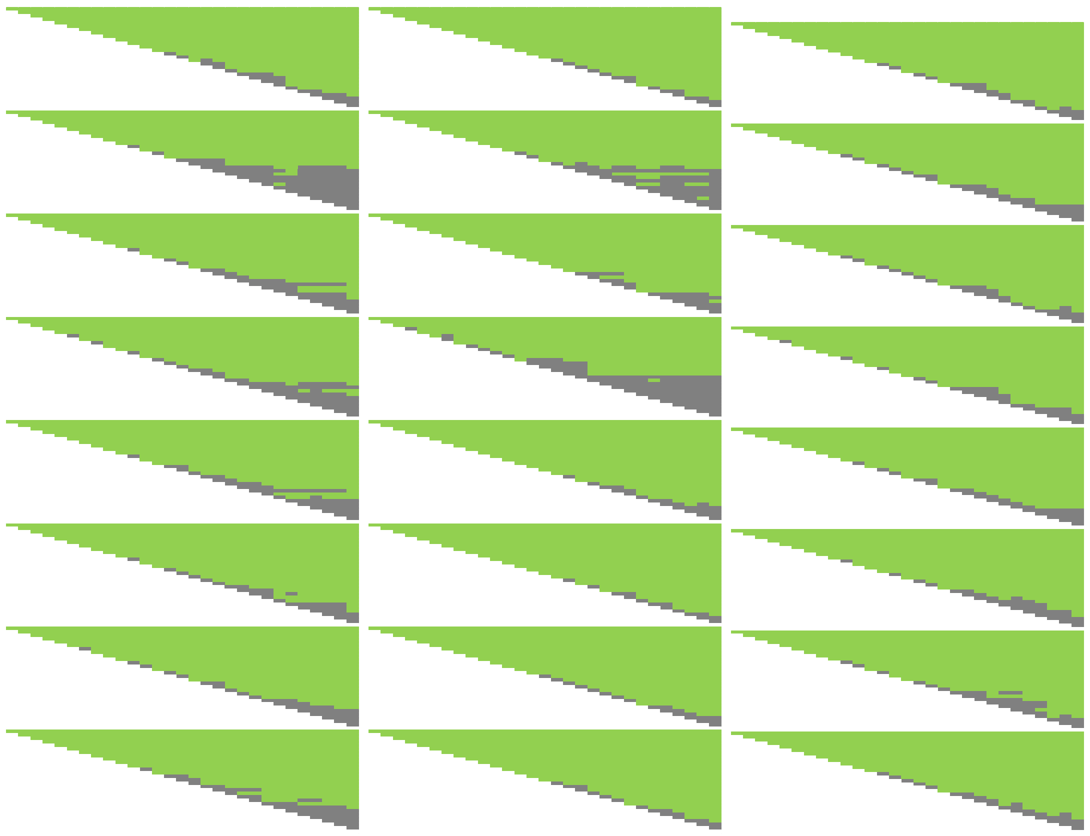


Figure 4.6: Confusion matrices for the significance tests of all slices, for each metric and system (metrics in order, top to bottom: ROUGE, METEOR, BLEU1, BLEU4, EACS, GMS, STCS, VECS; systems in order, left to right: H&S, D&A, GEN).

shorter references (top-left of all matrices), and that they stop being significant around slice 17 (varying slightly depending on the metric and system). Therefore, only for references with more than 17 questions per reference the null hypothesis can be rejected, meaning that the size of the reference impacts the results obtained. The pattern is similar across all matrices, despite some shifts. Interestingly, sometimes the values are close to the threshold cut, showing some colored points in the middle of a gray area. However, above those points, p values are typically much lower than the threshold of 0.01, which makes us believe those are just outliers. In Figure 4.7 we highlight one of the matrices (D&A, BLEU1), where it is more clear that the differences between slices from size 18 up (in this case) stop being significant.

To summarize, these results seem to indicate that MONSERRATE is exhaustive enough to perform automatic QG evaluation,

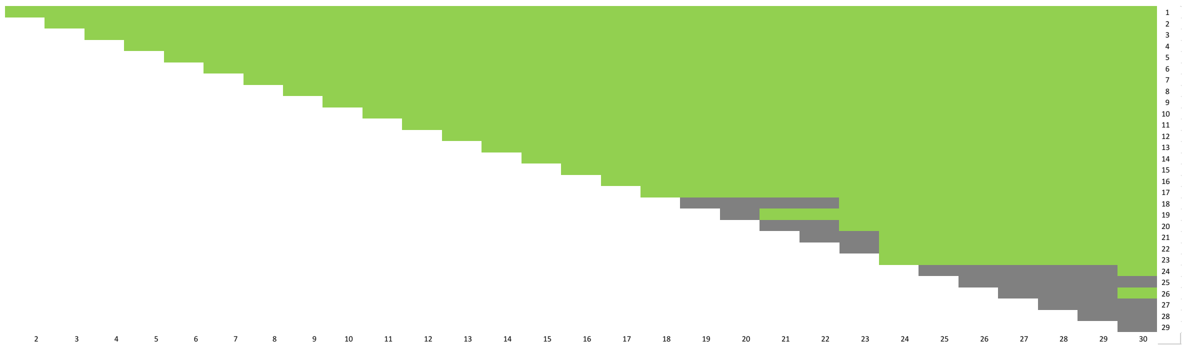


Figure 4.7: A closer look to the confusion matrix for D&A - BLEU1, corresponding to the fourth row/second column in Figure 4.6.

4.4.2 Systems' Precision

In the past sections we only have looked at the average scores obtained by each system for all metrics. This gives no reward for hitting exact questions as they appear on the reference, but rather being consistently as close as possible to the reference. Therefore, we also analyzed what is the precision of each system, considering a hit when a system scores perfectly with a given metric, that is, when the reference contains the generated question, *ipsis verbis*. Again, it is expected that with smaller references results are poorer, and specially hard to distinguish.

Figure 4.8 shows the precision obtained by each system, at each slice, by how each metric dictates an exact match. As expected, precision increases with the reference and systems become distinguishable, whereas at smaller slices precision is almost identical and standard deviation is much larger (seen by the large error bars), implying the chosen reference can bias the results tremendously. BLEU, as a metric that uses all associated questions to be computed, performs in a different way, showing greater values of precision and increasing much faster with the size of the reference, which does not correspond to a truthful exact match. Without accounting for BLEU, we see that systems are only able to get up to 6%, revealing that systems can be consistent at generating close questions, but not at getting them exactly accurate.

A slightly different experiment is to allow a small threshold in each metric when considering a hit in the reference, instead of just exact matches. The reasoning is that QG systems may introduce small errors that are easily corrected, or that minor variations of a reference are

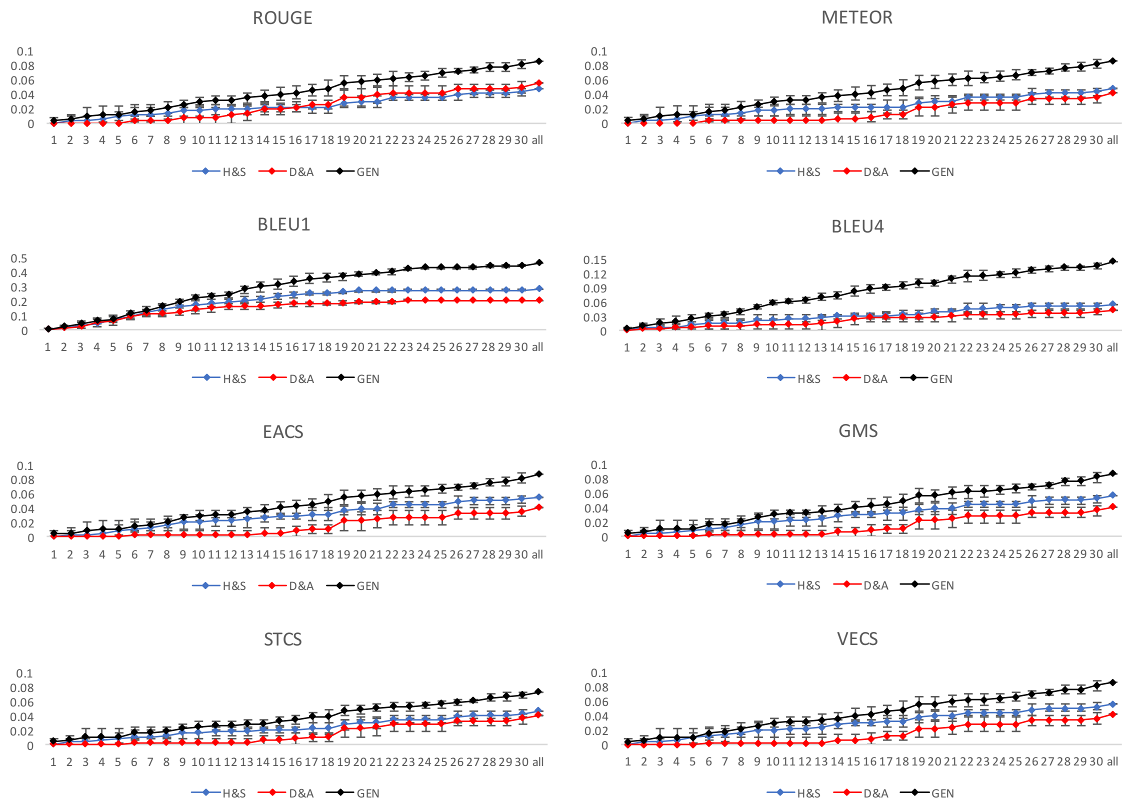


Figure 4.8: Precision of each system when using an exact match, as dictated by each metric, using as reference the slices of size 1-30, and the whole reference (all).

not included, penalizing systems more harshly than needed. In other words, if the score given by a metric is above the threshold set, we consider the generated question to have a match in the reference. Figure 4.9 shows precision for all metrics with a small threshold (scores above 0.9 are considered a match). The overall precision is slightly larger, as expected, and, as before, for smaller slices results are indistinguishable, but they start being more relevant for larger slices. However, it is interesting to note that each metric rewards a different system, which was not the case previously. This is explained by the manually threshold of 0.9, which is not suitable for all metrics, as the precision values increase in different rates for each metric. Therefore, this type of approach must be conducted with caution, as results might not be easily analyzed.

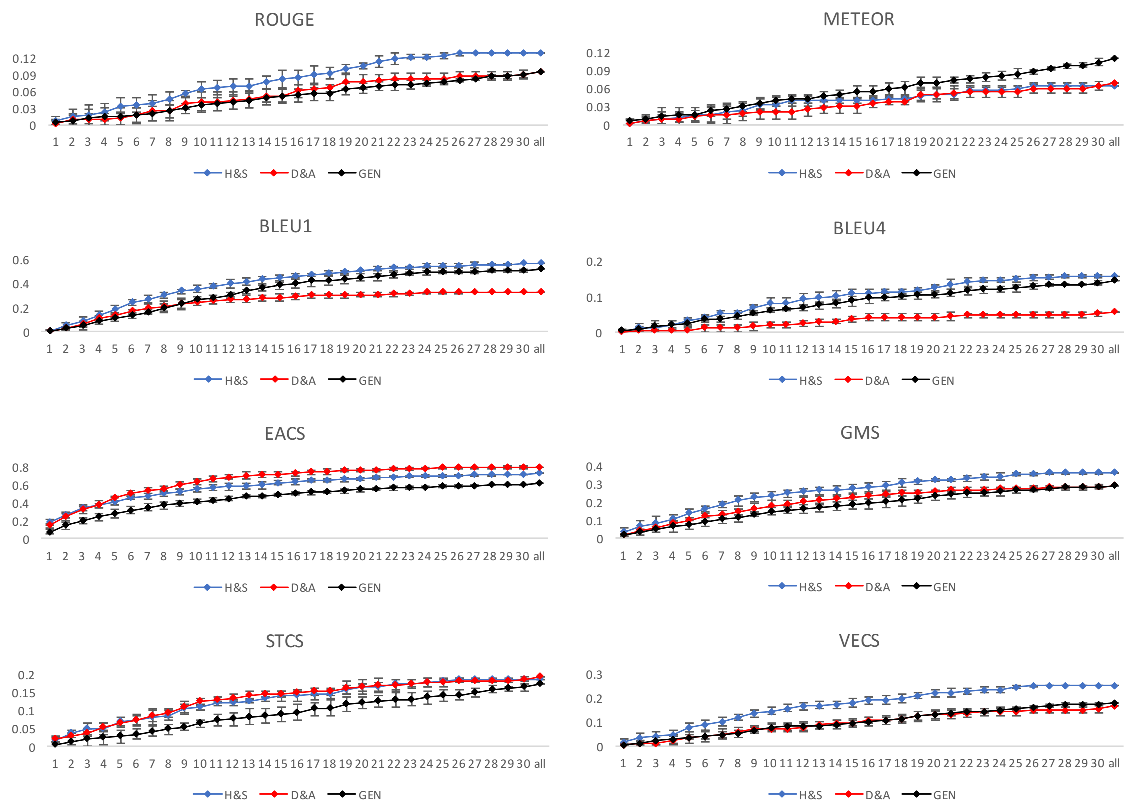


Figure 4.9: Precision of each system when using an exact match with a small threshold, as dictated by each metric, using as reference the slices of size 1-30, and the whole reference (all).

4.5 Discussion

In this chapter we introduced MONSERRATE, a new corpus created through multiple steps in a thoroughly fashion among 17 individuals. We aimed at creating an as exhaustive as possible dataset such that it becomes possible to perform automatic evaluation for QG systems.

We showed that current datasets are not suitable for such evaluation settings, because corpora does not include extensive reference hypotheses, and that MONSERRATE is able to fix that issue, being, on average, 26 times larger, on the reference side, than typical available datasets. Results showed that, for any system and metric, the reference is large enough to get consistent results (average scores), and, most importantly, that small references can be misleading when using automatic metrics to draw conclusions about the performance of QG systems. Thus, the size of the reference seems to be of more importance than the metrics used, which behave differently and can lead to disparate conclusions.

We studied how the reference behaved by increasing its size and concluded that, from a certain point, enlarging the reference would not lead to significant improvements in any system, for all metrics. We found this point to be a size of 17, but it is not guaranteed this value works for all domains.

Finally, if one is looking to evaluate systems by hitting the exact questions in a reference, results showed that it is still difficult to use precision as a metric (up to 6% of precision obtained), which means that systems introduce small errors in their generation process, missing this way an exact match with the reference.

5 Benchmarking Question Generation

In this chapter we take different Question Generation (QG) systems and compare them using two different datasets: our own MONSERRATE and the widely used SQuAD. We first go in depth regarding the experimental setup in Section 5.1, and then we present and discuss the results obtained in Sections 5.2 (automatic evaluation) and 5.3 (human evaluation).

5.1 Experimental Setup

We use two different corpora in our QG evaluation: SQuAD and MONSERRATE. The former is a crowd-sourced dataset, created with the goal of assessing Question Answering (QA) systems in the task of Reading Comprehension, but also widely used in the task of QG, as it is composed of a set of questions associated with a few sentences. The latter is an extensive reference of questions aimed at helping to automatically evaluate QG systems. SQuAD was described in Section 2.2 in greater detail, while MONSERRATE was presented in the previous chapter.

MONSERRATE was used in the same way as before, that is, the full corpus of 73 sentences is used as the input of each system, and all questions associated are used as reference.

For SQuAD, we use its first version (1.1), which contains no adversarial questions. Its test set is hidden, so we took a portion of the training to use as test set. We split it in roughly the same size of the development set provided, which contains over 10k questions. This corresponds to approximately 11% of the training set, which went down from 87k entries to 77k questions. The test set portion is used in our evaluation and contains 6689 unique sentences, and a total of 10354 questions as reference. Both the train and development sets were used to train D&A's system.

To benchmark current QG systems, we use two state-of-the-art systems using two different

Table 5.1: Seeds used in the Pattern Acquisition phase of GEN.

Support Sentence	Question
Leonardo da Vinci was born on April 15, 1452.	When was Leonardo da Vinci born?
Lee Harvey Oswald was assassinated by Jack Ruby.	Who killed Lee Harvey Oswald?
Paris is located in France.	Where is Paris located?
Porto is located 313 km from Lisbon.	How far is Lisbon from Porto?
Yesterday, Bob took butter from the fridge.	Where did Bob take butter from?
John baked cookies in the oven.	What did John bake in the oven?
Cooking is the art, technology, science and craft of preparing food for consumption.	What is cooking?
Science is a systematic enterprise that builds and organizes knowledge in the form of testable explanations and predictions about the universe.	What is a systematic enterprise that builds and organizes knowledge in the form of testable explanations and predictions about the universe?

approaches, and our own GEN. All were presented in the experimental setup from previous chapter, in Section 4.2.1, and are used here with the same parameterizations. To summarize, we are using H&S [Heilman and Smith, 2009] and D&A [Du et al., 2017], and our own system, GEN, described in Chapter 3, using all different matching parameterizations: `Strict`, `Subtree`, `Subtree Flex`, and `Argument`. Table 5.1 shows the used seeds to bootstrap the system. We chose only 8 seeds that cover different question types for two reasons: first, we want to test how GEN performs with such a limited number of initial seeds; secondly, it gives room for GEN to improve when using the implicit feedback, which will be addressed in Chapter 6.

Regarding evaluation, we also use the same automatic metrics presented in last chapter, from Maluba project [Sharma et al., 2017][1]: ROUGE, METEOR, BLEU (BLEU1 and BLEU4), Embedding Average Cosine Similarity (EACS), Greedy Matching Score (GMS), SkipThought Cosine Similarity (STCS), and Vector Extrema Cosine Similarity (VECS). Automatic metrics compute a score for each system based on the generated questions and the reference. The closer the questions are to the reference, the higher that score will be. Therefore, what automatic metrics capture is not necessarily the quality of the generated questions, but rather the system's recall, given a reference, as discussed in the previous chapter.

[1]`https://github.com/Maluuba/nlg-eval`

Table 5.2: Number of questions per type obtained on Monserrate by H&S, D&A and GEN. For the latter, the parameterizations follow the same order as before, but are presented with shorter names. We omit GEN's `Strict` parametrization, as it generated a single question.

	H&S	D&A	GEN SubT	GEN SubT Flex	GEN Args	GEN All
Questions	108	73	6	127	122	209
Who	29 (26.85%)	20 (27.40%)	2 (33.33%)	33 (25.98%)	47 (38.52%)	68 (32.54%)
What	60 (55.56%)	34 (46.58%)	2 (33.33%)	74 (58.27%)	49 (40.16%)	100 (47.85%)
Where	3 (2.78%)	3 (4.11%)	2 (33.33%)	1 (0.79%)	8 (6.56%)	10 (4.78%)
When	12 (11.11%)	4 (5.48%)	-	16 (12.60%)	17 (13.93%)	27 (12.92%)
Which	-	1 (1.37%)	-	-	-	-
How many	4 (3.70%)	5 (6.85%)	-	-	-	-
How long	-	3 (4.11%)	-	-	-	-
Other	-	3 (4.11%)	-	3 (2.36%)	1(0.82%)	4 (1.91%)

5.2 Results with Automatic Metrics

This section reports the results obtained by the three systems in the two datasets, Monserrate and SQuAD.

Being given the 73 sentences from the Monserrate corpus, H&S system generated 108 questions (after discarding more than 200 questions below the 2.5 score threshold), and GEN generated 209 questions. As D&A is limited to one question per sentence, 73 questions were generated.

Regarding SQuAD, given its 6689 unique sentences, H&S obtained 12370 questions with score above the threshold set, while D&A generated one question per sentence as discussed. GEN generated a total of 19946 questions, split across the different parameterizations (some overlap exists).

In Table 5.2 we present the total number of questions generated by each system on Monserrate, along with the discrimination of their types. One column missing is GEN's `Strict` parameterization, as it only generated a single question, of the type `Where`. The first thing to note is how D&A is capable of generating more types of questions when compared with the other systems, with more focus on `What` questions, which is the most common type across all systems. GEN, on the other hand, fails to create some types of questions, but that is consequence of the chosen initial seeds. However, it is more capable of generating `When` and `Where` questions. The `Other` category, for D&A, includes variations of the `How` type (for instance,

Table 5.3: Number of questions per type obtained on SQuAD by H&S, D&A and GEN. For the latter, the parameterizations follow the same order as before, but are presented with shorter names. We omit GEN's `Strict` parametrization, which generated 152 questions. (*includes *In what year* questions)

	H&S	D&A	GEN SubT	GEN SubT Flex	GEN Args	GEN All
Questions	12370	6889	1868	10846	11384	19446
Who	2470 (19.97%)	534 (7.75%)	497 (26.61%)	2072 (19.10%)	3957 (34.76%)	5963 (30.66%)
What	7837 (63.35%)	4298 (62.39%)	926 (49.57%)	6790 (62.60%)	4353 (38.24%)	8780 (45.15%)
Where	482 (3.90%)	231 (3.35%)	217 (11.62%)	465 (4.29%)	1194 (10.49%)	1634 (8.40%)
When	1164 (9.41%)	841* (12.20%)	228 (12.21%)	1515 (13.97%)	1871 (16.44%)	3058 (15.73%)
Which	-	22 (0.32%)	-	-	-	-
How many	191 (1.54%)	587 (8.52%)	-	-	-	-
How long	-	89 (1.29%)	-	-	-	-
Whose	211 (1.71%)	-	-	-	-	-
If	11 (0.09%)	4 (0.06%)	-	-	-	-
Why	-	16 (0.23%)	-	-	-	-
Other	4 (0.03%)	356 (5.17%)	-	4 (0.04%)	9 (0.08%)	11 (0.06%)

how long), whereas for GEN they correspond to ill-formed questions.

Table 5.3 reports the same statistics for SQuAD dataset, and it shows how differently the three systems behave. D&A only generates a question per sentence, but it is much more diversified on the types of question it outputs. GEN, as before, due to the limited number of patterns used, only generates four types of questions. Due to the nature of the corpora, `What` type questions seem more prevalent than others, with both D&A and H&S generating over 60% questions of that type. In that regard, GEN is more balanced, generating less than 50%. However, looking into each parameterization individually, we can see that `Subtree Flex` behaves similarly to those two systems, while the other matching modes get a more even distribution of question types. Depending on the results, it might be possible that certain parameterizations are more suitable for certain question types, which could be a good indicator to boost GEN's performance.

The number of questions is less relevant than the quality of the questions themselves. How well the systems are able to recover the reference is one way to perceive this. The results obtained with automatic metrics are reported in Tables 5.4 and 5.5, respectively for MONSERRATE and SQuAD. We only show the most common types (`who`, `what`, `when` for MONSERRATE, adding `where` and `how many` for SQuAD). On MONSERRATE, GEN's `Subtree` parameterization only has 2 questions for each of the first two types, so it is marked with an

Table 5.4: Scores obtained with automatic metrics, per question type (`who`, `what`, `when`), obtained on MONSERRATE by H&S, D&A and GEN. (*`Subtree` matching strategy only generated 2 questions per type.)

`Who`	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
H&S	68.89	45.96	84.20	46.99	91.68	85.50	75.76	77.27
D&A	67.32	43.67	75.24	32.49	**92.43**	**86.34**	78.00	**82.63**
GEN Subtree*	67.98	34.06	**100.00**	5.52e-05	83.50	85.32	76.18	76.52
GEN Subtree Flex	**70.15**	41.87	**89.65**	42.53	89.70	85.59	74.43	81.88
GEN Argument	67.21	**48.90**	78.59	**47.00**	91.66	85.36	**78.90**	80.99
GEN All	64.78	42.57	80.46	37.31	89.88	83.84	75.05	79.46
`What`	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
H&S	**67.61**	**44.22**	82.91	**40.72**	92.13	85.55	69.63	77.55
D&A	62.88	35.07	77.06	22.89	**92.63**	85.30	**73.42**	75.83
GEN Subtree*	**73.11**	37.67	**100.00**	**85.36**	92.36	**88.30**	57.20	**81.97**
GEN Subtree Flex	61.89	39.76	**84.46**	28.07	89.83	82.95	64.02	75.63
GEN Argument	64.09	41.16	82.67	27.23	92.09	**85.78**	64.81	**80.04**
GEN All	61.84	38.55	83.60	27.23	90.32	83.57	63.20	75.99
`When`	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
H&S	**77.10**	51.76	86.78	60.61	94.98	89.56	**78.18**	82.71
D&A	62.56	43.17	82.25	30.60	90.75	84.41	63.83	71.71
GEN Subtree	-	-	-	-	-	-	-	-
GEN Subtree Flex	67.76	42.45	**92.87**	54.18	91.66	86.44	67.48	80.66
GEN Argument	73.42	**56.42**	88.95	**64.59**	**95.19**	**89.83**	72.36	**86.74**
GEN All	67.81	47.22	89.23	52.58	92.75	87.16	68.19	82.40

asterisk on the table, noting that highlighted results in such rows come from a small sample size. The parameterization may lead to the best results, but poses a problem of precision versus recall: a total of 6 questions for 73 sentences is, indeed, low for a system of this nature.

In MONSERRATE, results point to some parameterizations of GEN being better at certain types of question. For example, for `when` type questions, `Argument` surpasses both H&S and D&A (except for ROUGE metric), while for `what` questions GEN seems to perform slightly under H&S. Finally, for `who` questions it is not clear which version is better, as, depending on the metric chosen, it appears to surpass the other state of the art systems, which once again shows how a given metric is not a clear indication of the best performing system.

On SQuAD, however, GEN shows to be less capable of recovering the reference, as measured by the automatic metrics. `Strict` seems to perform much better on `When` questions, while `Subtree` is the worst performing parameterization. `Argument` and `Subtree Flex` be-

Table 5.5: Scores obtained with automatic metrics, per question type (`who`, `what`, `where`, `when`, `how many`), obtained on SQuAD by H&S, D&A and GEN.

`Who`	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
H&S	31.99	**19.57**	31.85	9.39	82.29	67.32	50.23	53.43
D&A	**36.60**	18.09	**35.94**	**10.34**	**82.71**	**69.12**	**52.15**	**54.22**
GEN Strict	16.02	8.10	9.74	1.65	62.32	53.33	39.89	43.50
GEN Subtree	15.18	6.84	8.62	0.39	65.33	54.63	38.33	43.68
GEN Subtree Flex	22.34	10.93	17.12	2.67	72.51	60.29	44.15	47.16
GEN Argument	21.25	12.88	18.73	4.48	76.67	61.16	46.09	48.75
GEN All	20.68	11.61	17.11	3.44	74.67	60.25	44.69	47.62
`What`	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
H&S	**34.47**	**20.22**	**35.42**	**8.56**	84.91	70.33	**50.60**	**54.59**
D&A	33.66	16.36	34.35	5.75	**85.00**	**70.46**	49.71	52.60
GEN Strict	20.94	9.56	15.79	2.15e-06	70.44	61.15	37.00	46.16
GEN Subtree	18.78	8.44	13.22	1.02	71.17	60.95	35.46	43.41
GEN Subtree Flex	24.73	12.38	21.68	2.90	77.82	65.04	41.07	47.44
GEN Argument	27.02	14.05	23.63	4.22	79.48	66.63	42.33	49.67
GEN All	24.72	12.43	21.56	3.04	77.77	65.01	44.69	47.62
`Where`	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
H&S	**33.82**	**22.33**	**33.32**	**9.62**	**84.94**	**70.34**	**51.64**	**56.97**
D&A	33.61	18.64	32.46	7.98	83.67	69.58	50.47	55.56
GEN Strict	22.70	19.42	16.49	6.23e-06	76.21	62.86	44.91	53.53
GEN Subtree	18.27	12.32	13.60	0.23	76.21	61.17	41.63	49.79
GEN Subtree Flex	26.57	16.62	24.06	4.41	81.29	65.94	45.06	53.54
GEN Argument	24.67	15.75	22.94	5.07	81.19	65.40	44.20	52.97
GEN All	23.68	15.04	21.47	4.13	80.38	64.60	43.92	52.13
`When`	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
H&S	36.57	**23.82**	36.89	11.87	**86.50**	71.97	53.08	58.51
D&A	**41.38**	22.79	**41.12**	**13.02**	86.32	**73.62**	**53.30**	**59.80**
GEN Strict	31.23	18.76	26.81	10.04	78.84	68.52	43.86	60.25
GEN Subtree	18.70	9.45	14.15	1.61	74.45	61.48	37.85	49.36
GEN Subtree Flex	25.90	13.62	23.80	3.95	80.61	66.04	41.82	53.12
GEN Argument	25.55	14.70	24.04	5.04	81.17	66.01	42.05	53.31
GEN All	24.17	13.20	22.15	3.73	80.23	65.20	41.16	52.19
`How many`	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
H&S	**40.26**	**28.53**	**40.38**	**14.76**	**87.65**	**74.00**	**54.92**	**60.37**
D&A	36.27	20.74	37.32	8.03	86.81	72.12	53.17	56.84

have similarly, with preference for `What` questions for the former, and `Where` for the latter. Comparing the two state of the art systems, H&S performs much better for `How many` type questions, while D&A performs better for `When` questions. For the remaining (`Who`, `What`, and `Where`) results are close. As the larger percentage of questions come from these types, overall

Table 5.6: Overall scores obtained with automatic metrics on MONSERRATE, for H&S, D&A and GEN.

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
H&S	**69.00**	46.38	83.71	**45.56**	92.51	**86.11**	73.26	77.92
D&A	63.71	37.58	77.40	26.63	**92.52**	85.51	**74.47**	77.54
GEN Strict	50.00	23.02	75.00	1.8e-8	83.39	70.11	51.87	67.58
GEN Subtree	60.60	30.67	**95.93**	28.45	86.40	80.38	60.28	74.07
GEN Subtree Flex	64.66	40.63	86.91	35.25	90.10	84.14	67.09	77.96
GEN Argument	65.81	**46.44**	81.80	40.61	92.25	85.86	71.17	**80.89**
GEN All	63.13	41.09	77.90	34.33	90.52	83.97	67.60	77.90

Table 5.7: Overall scores obtained with automatic metrics on SQuAD, for H&S, D&A, and GEN.

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
H&S	34.17	**20.66**	34.79	**9.21**	84.55	69.90	**50.89**	**54.93**
D&A	**34.81**	17.71	**35.48**	7.12	**85.21**	**70.91**	50.64	54.20
GEN Strict	22.03	12.33	16.47	2.32	70.86	60.92	40.14	49.13
GEN Subtree	17.71	8.57	12.04	0.80	70.62	59.39	37.18	44.93
GEN Subtree Flex	24.56	12.47	21.23	3.07	77.36	64.34	41.92	48.46
GEN Argument	24.57	13.99	21.95	4.54	78.98	64.52	47.77	50.30
GEN All	23.30	12.52	20.28	3.36	77.42	63.55	42.48	48.69

the differences are not noticeable.

Finally, we report the overall results obtained by each system when using automatic metrics. Tables 5.6 and 5.7 present the obtained results, respectively for MONSERRATE and SQuAD. For GEN, we present the results separated by matching technique, from `Strict` to `Argument`.

On MONSERRATE, results show that H&S perform better than D&A according to most metrics, while GEN performs overall better than D&A but slightly lower than H&S. As expected, due to the characteristics of the reference, the pattern of the results are much different for SQuAD. Namely, GEN is not capable to recover from the reference as well as the counterparts, and it is not clear which one from H&S and D&A is better.

5.3 Human Evaluation

In last section we delved into automatic evaluation, and how it measures the capability of covering a reference, which might not necessarily translate to question quality. In this section we explore human evaluation as a mean to evaluate the quality of the generated questions.

5.3.1 Task Description

We used Amazon Mechanical Turk (AMT) to obtain manual evaluations for the generated questions. We used AMT to ask people (turkers) their opinion about the generated questions with the three systems, according to different metrics: grammaticality, semantics, plausibility, and utility. Each attribute was scored from 1 to 3. Quoting directly from the instructions submitted, each was defined as:

- **Grammaticality** refers to common errors one finds in sentences, from simple errors such as typos or repeated words, to more complex errors like disagreement in number or phrase structure. Here we will be looking at the grammar of the question.
- **Meaningfulness** is about the semantics of a sentence, i.e., how a sentence makes sense in the real world. In other words, it asks if there is a representation of the meaning the sentence tries do convey. Again, we will be looking at the semantics of the question.
- **Plausibility** addresses the possibility of a question being answered by the answer sentence. A question may be grammatically correct and meaningful, but still be not plausible given the **context** of the answer sentence.
- **Utility** of a question is related to its interest for real world applications, such as quizzes, tests, and FAQs. Questions that would be interesting just for linguistic purposes, for example, would be considered of less interest.

More details can be found in Appendix B, including an example of a complete HIT submitted. The HITs were composed of a section containing instructions, followed by a couple of complete examples of the task, before presenting the actual task to perform. We collected answers of 3 workers per HIT, at $0.15 per task. We batched 100 questions at a time, and published each batch in a different day. The idea was that, in one hand, workers could find a large enough batch that they could learn from the instructions and perform the task repeatedly for a set of questions, but, on the other hand, they would not see many similar questions or source sentences when performing the task, avoiding both tiredness and bias when evaluating the questions.

5.3.2 AMT Results

We asked turkers on AMT to follow the guidelines presented before. For Monserrate, due to its small size on the corpus side, we collected all questions from the three systems and split them in four batches, randomizing the order in which the questions appeared. For SQuAD, due to the large number of questions, we randomly selected a subset of the corpus and, then, collected all questions coming from sentences to which each system had generated at least a question for, so it is easier to directly compare all systems. We selected 98 questions per system and used slightly larger batches, in two separate days, totalizing 196 questions per batch[2].

Before discussing the results obtained, we look at the average of the standard deviations obtained for all questions. This gives us an insight on how the humans responses were aligned. With three responses per attribute, and scores taking values from 1 to 3, there are some number of combinations of scores possible: for instance, if all scores are equal, standard deviation for that instance will be 0; if they are all different, it will be 1. If just one human gives a score either above or below the others, it will be 0.58, and if one human gives a score two points above of below the others, it will be 1.15. Each HIT submitted has its own standard deviation, so we averaged all those. Tables 5.8 and 5.9 show these results for all four attributes (grammar, semantic, plausibility, and utility), for the three systems, on both Monserrate and SQuAD, respectively. The tables also summarize the baselines described, where $\odot$ representes a generic score, $\oplus$ and $\ominus$ represent a score either above or below that, and $\bigoplus$ and $\bigominus$ represent a score two points above or below that.

Results are consistent across both datasets and all metrics and systems, ranging from 0.58 to 0.71, which is closer to the baseline for a single score being either above or below the other two. This means that, on average, the three human evaluators did not disagree much more than a single point among them.

[2]The extra 98 questions will be reported in Chapter 6.

Table 5.8: Average standard deviation of all HITs, per attribute and system, for human evaluation scores obtained on AMT for questions generated on MONSERRATE. The bottom summarizes the possible standard deviations of a single HIT.

	Questions	Grammar	Semantic	Plausibility	Utility
H&S	108	0.71	0.63	0.65	0.67
D&A	73	0.70	0.70	0.67	0.64
GEN All	209	0.65	0.64	0.60	0.65
Standard Deviation for all possible combinations of scores in a HIT					
⊙ ⊙ ⊙	0	⊙ ⊙ ⊕	0.58	⊙ ⊙ ⨁	1.15
⊖ ⊙ ⊕	1	⊙ ⊙ ⊖	0.58	⊙ ⊙ ⊖	1.15

Table 5.9: Average standard deviation of all HITs, per attribute and system, for human evaluation scores obtained on AMT for questions generated on SQuAD. The bottom summarizes the possible standard deviations of a single HIT.

	Questions	Grammar	Semantic	Plausibility	Utility
H&S	98	0.61	0.67	0.63	0.66
D&A	98	0.66	0.59	0.67	0.65
GEN All	98	0.66	0.58	0.65	0.67
Standard Deviation for all possible combinations of scores in a HIT					
⊙ ⊙ ⊙	0	⊙ ⊙ ⊕	0.58	⊙ ⊙ ⨁	1.15
⊖ ⊙ ⊕	1	⊙ ⊙ ⊖	0.58	⊙ ⊙ ⊖	1.15

Finally, we study the actual scores given by the evaluators. To calculate the scores from human evaluation, we look at the data in two ways: taking the average score from the three workers, and their median. Each punishes disagreement in a different way, if we consider two out of three answers to be identical: average, compared to median, pushes down the overall score if the outlier is lower, while median pushes down the score when the outlier has a higher score.

Tables 5.10 and 5.11 show the scores for all systems according to the different attributes, alongside their average, taking respectively the humans' average and median scores per question, for MONSERRATE.

The first thing to note is H&S obtains the best scores overall, with GEN coming close behind, while D&A performs below them. However, it is interesting to see D&A obtaining its best results in `grammaticality` (its only score above 2.0), losing on the other metrics. This

Table 5.10: Human evaluation scores obtained on AMT, for all atributes and average, on MONSERRATE, for H&S, D&A, and GEN, taking the average of the scores per question.

	Questions	Grammar	Semantic	Plausibility	Utility	Avg
H&S	108	**2.18** ±0.48	**2.13** ±0.46	**2.20** ±0.50	**2.03** ±0.48	**2.14**
D&A	73	2.14 ±0.46	1.99 ±0.45	1.99 ±0.44	1.75 ±0.44	1.97
GEN Strict*	1	1.67 ±0.00	1.67 ±0.00	**2.67** ±0.00	**2.33** ±0.00	2.08
GEN Subtree*	6	**2.33** ±0.42	1.83 ±0.35	2.11 ±0.62	2.00 ±0.52	2.07
GEN Subtree Flex	127	2.15 ±0.50	2.04 ±0.52	2.15 ±0.49	2.01 ±0.48	2.09
GEN Argument	122	2.15 ±0.56	2.06 ±0.47	2.16 ±0.49	1.99 ±0.42	2.09
GEN All	209	2.14 ±0.56	2.05 ±0.49	2.14 ±0.50	1.98 ±0.46	2.08

Table 5.11: Human evaluation scores obtained on AMT, for all atributes and average, on MONSERRATE, for H&S, D&A, and GEN, taking the median of the scores per question.

	Questions	Grammar	Semantic	Plausibility	Utility	Avg
H&S	108	2.19 ±0.76	**2.17** ±0.65	**2.26** ±0.70	**2.04** ±0.68	**2.16**
D&A	73	**2.22** ±0.67	1.99 ±0.70	1.99 ±0.61	1.70 ±0.64	1.97
GEN Strict*	1	1.00 ±0.00	1.00 ±0.00	**3.00** ±0.00	**3.00** ±0.00	2.00
GEN Subtree*	6	**2.50** ±0.84	1.83 ±0.75	2.17 ±0.98	2.00 ±0.98	2.13
GEN Subtree Flex	127	2.19 ±0.72	2.04 ±0.70	2.24 ±0.66	1.98 ±0.66	2.11
GEN Argument	122	2.15 ±0.75	1.99 ±0.65	2.24 ±0.62	2.00 ±0.62	2.09
GEN All	209	2.15 ±0.73	2.02 ±0.67	2.22 0.65±	1.97 ±0.64	2.09

highlights the power of neural networks, as they can learn concrete linguistic phenomena from seeing many examples, but then fails at capturing more latent phenomena, as `semantics` and `utility`. However, it should be noted that D&A was trained on SQuAD, optimizing only towards automatic metrics, which do not value these subjective metrics, so there might be hope for neural networks if these concerns are incorporated into their architectures.

Regarding GEN, results coming from parameterizations `Strict` and `Subtree` can be misleading, as they have a low count of questions (1 and 6 in total, respectively), so we will not address them. However, results show that GEN is able to generate questions rated closed to H&S, only losing more significantly in `semantics`.

Regarding SQuAD, Tables 5.12 and 5.13 report the obtained scores, following the same procedure as before, taking the average and median scores per question, respectively. These results are really interesting, for different reasons. First, as seen with MONSERRATE, D&A continues to show more prowess in generating grammatically correct questions, obtaining

Table 5.12: Human evaluation scores obtained on AMT, for all atributes and average, on SQuAD, for H&S, D&A, and GEN, taking the median of the scores per question.

	Questions	Grammar	Semantic	Plausibility	Utility	Avg
H&S	98	2.14 ±0.53	2.10 ±0.41	**2.19** ±0.44	2.01 ±0.44	2.10
D&A	98	**2.22** ±0.47	**2.13** ±0.47	2.14 ±0.47	2.04 ±0.48	**2.13**
GEN Strict *	3	1.89 ±0.51	2.22 ±0.69	**2.33** ±0.33	**2.33** ±0.33	2.19
GEN Subtree	13	2.15 ±0.50	2.08 ±0.45	2.10 ±0.53	2.10 ±0.34	2.11
GEN Subtree Flex	57	2.20 ±0.48	**2.19** ±0.48	**2.22** ±0.42	**2.12** ±0.41	**2.18**
GEN Argument	78	2.13 ±0.52	2.09 ±0.45	2.17 ±0.40	2.05 ±0.41	2.11
GEN All	98	2.13 ±0.52	2.10 ±0.47	2.17 ±0.44	2.07 ±0.42	2.12

Table 5.13: Human evaluation scores obtained on AMT, for all atributes and average, on SQuAD, for H&S, D&A, and GEN, taking the median of the scores per question.

	Questions	Grammar	Semantic	Plausibility	Utility	Avg
H&S	98	2.16 ±0.74	2.09 ±0.59	2.20 ±0.63	1.99 ±0.62	2.11
D&A	98	**2.32** ±0.67	**2.13** ±0.64	2.20 ±0.70	2.01 ±0.68	2.17
GEN Strict *	3	1.67 ±0.58	**2.33** ±0.58	**2.33** ±0.58	**2.67** ±0.58	2.25
GEN Subtree	13	2.15 ±0.80	2.08 ±0.64	2.15 ±0.69	2.23 ±0.60	2.15
GEN Subtree Flex	57	2.28 ±0.70	**2.21** ±0.59	**2.26** ±0.61	**2.21** ±0.65	**2.24**
GEN Argument	78	2.23 ±0.68	2.13 ±0.52	2.18 ±0.57	2.10 ±0.62	2.17
GEN All	98	2.22 ±0.73	2.12 ±0.56	2.23 ±0.61	2.12 ±0.60	2.18

higher scores in that metric than the others. In addition, maybe due to being trained in the same type of data, it can also beat the other systems on `semantics`, according to human evaluation. However, and still continuing the pattern observed on MONSERRATE, it is GEN which is able to generate more useful questions as measured by `utility`. GEN as a whole also performs slightly lower than D&A, but better than H&S. Additionally, `Subtree Flex` shows the best results overall, although for a smaller number of questions. Secondly, automatic metrics had put GEN performing below than the other two systems on SQuAD, which human evaluation does not support. Selecting a subset of all questions may be a reason for that discrepancy, reason why we decided to compare the systems using the automatic metrics once more, but only for the human evaluated questions. Table 5.14 shows that the results are slightly better overall when compared with the whole dataset (Table 5.7), but that they are improved proportionally, i.e., GEN is shown to still be much worse than the counterparts. This shows, once again, how important an exhaustive reference is to perform automatic evaluation

Table 5.14: Scores obtained with automatic metrics on SQuAD subset used for human evaluation, for H&S, D&A, and GEN.

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
H&S	39.46	25.99	39.56	12.79	87.08	73.76	56.29	60.72
D&A	38.19	21.22	36.52	9.97	85.75	73.53	53.86	57.90
GEN All	28.65	17.14	25.38	6.61	80.69	67.30	47.48	55.47

in QG, and that GEN is, indeed, capable of generating as good questions as their counterparts on SQuAD.

5.3.3 Evaluation Examples

Given the obtained scores from both automatic metrics and human evaluators, we can analyze some examples on how the systems perform. Table 5.15 shows different examples of questions generated by each system on MONSERRATE, alongside the scores obtained with both automatic metrics and human evaluation.

We can see examples of either high scores in both evaluations (double up arrows), low scores in both (double down arrows), or mixed scores. This shows how complex is the evaluation of QG systems, and the pros and cons of each process. Depending on how one wants to analyze the data, different conclusions can be reached. For instance, one can say that high scored questions with automatic metrics are misevaluated and human evaluation is able to penalize those (row 4). However, the opposite reasoning can be applied to row 7, where a good question, with perfect score in all automatic metrics (meaning it exists in the reference and was, thus, created by a human), obtained a low average score on human evaluation.

Table 5.15 also shows some good and bad examples of questions generated by each system, highlighting some of their limitations and qualities. For example, we can see some errors that D&A introduces in the generation process, namely repetition of words, but also the introduction of new vocabulary (*good* instead of *fine* in its the first question). GEN shows its prowess in short questions, but how the matching techniques can introduce noise: the last row shows a question generated by `Subtree Flex`, where most of the important content of the noun phrase was discarded.

Table 5.15: Examples of generated questions by the three systems (H&S, D&A, and GEN) on MONSERRATE, along the scores obtained with both automatic metrics (AM) and the human evaluation (HE) – average score – with a visual indication of a high (↑) or low score (↓).

System	Answer Sentence	Question	AM	HE
H&S GEN	*This palace, the Cook family's summer residence, was constructed on the ruins of the neo-Gothic mansion built by the English merchant, Gerard de Visme, who built the first palace of Monserrate.*	*Who built the first palace of Monserrate?*	1.0 ↑ (all)	2.25 ↑
H&S	*The Park and Palace of Monserrate were classified as a Property of Public Interest in 1993, and were included in the Cultural Landscape of Sintra, which has been classified by UNESCO as World Heritage since 1995.*	*What has Sintra been classified by as world heritage since 1995?*	0.82 ↑ (BLEU1)	1.75 ↓
D&A	*The eclecticism is a fine example of the Sintra Romanticism, along with other nearby palácios, such as the Pena Palace and the Quinta do Relógio.*	*what is a good example of the sintra romanticism?*	0.88 ↑ (METEOR)	2.83 ↑
D&A	*For about fifty years the building was virtually abandoned and it deteriorated a lot.*	*how long was the building of the building abandoned?*	0.89 ↑ (BLEU1)	1.92 ↓
D&A	*The terrace leads out into the large park.*	*what is the main park that leads out into the large park?*	0.35 ↓ (METEOR)	1.58 ↓
GEN	*Monserrate Palace was visited by Lord Byron, the Anglo-Scottish poet and leading figure in the Romantic Movement.*	*What did Lord Byron visit?*	0.77 ↑ (ROUGE)	2.42 ↑
GEN	*This palace, the Cook family's summer residence, was constructed on the ruins of the neo-Gothic mansion built by the English merchant, Gerard de Visme, who built the first palace of Monserrate.*	*What did the English merchant build?*	1.0 ↑ (all)	1.83 ↓
GEN	*The palace was designed by the architects Thomas James Knowles (father and son) and built in 1858, having been commissioned by Sir Francis Cook, Viscount of Monserrate.*	*What did Monserrate commission?*	0.17 ↓ (METEOR) 1.0 ↑ (BLEU1)	1.5 ↓
		Who commissioned son?	0.17 ↓ (METEOR)	1.6 ↓

5.3.4 Correlation to Automatic Metrics

Last section highlighted the discrepancies in both automatic and human evaluation. Given that we collected data from human evaluators and also have computed automatic metrics for multiple questions, we were interested in checking if there was any correlation between them. Automatic metrics are reported as correlating with human judgment [Banerjee and Lavie, 2005, Lin, 2004, Papineni et al., 2002], but some recent works have shown that for some tasks or datasets the metrics may not show strong correlation to human evaluation [Liu et al., 2016, Novikova et al., 2017]. Given the available data and the evidence from last section, this would be a good opportunity to corroborate either of these hypotheses.

We did two types of correlation studies: we checked for correlation between each metric and the AMT scores, and calculated the rank correlation between the orderings given by each metric as if they would be used to rank the questions. The former is done by computing Spearman correlation [Spearman, 1987], and the latter using Kendall's rank correlation [Kendall, 1938].

Surprisingly (or not), we found no strong evidence of correlation. The values obtained do not surpass 0.30, which implies a weak correlation. Based on the previous experiments, this could be expected on SQuAD dataset, but it seemed that Monserrate would show a stronger correlation.

Tables 5.16 and 5.17 shows the Spearman correlation values obtained for all metrics, for all questions evaluated, on Monserrate and SQuAD, respectively. It appears there are hints of slightly stronger correlation on Monserrate, probably an indication of the importance of having an exhaustive corpus when using automatic metrics.

We also calculated these values for each system individually, and those results can be found in Appendix C. The results are all similar to the ones presented here, and show just a slight evidence of correlation between each automatic metric and each subjective metric. Semantic metrics also show smaller correlation values than the more typical lexical metrics, which supports the conclusion from Chapter 4, where we showed how these metrics are not suitable for QG evaluation.

Using the automatic metrics as a ranking strategy also did not prove effective to establish

Table 5.16: Correlation between all automatic metrics and AMT evaluations, for all questions evaluated on MONSERRATE, using AMT average results (top) and median results (bottom).

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Grammar	0.14	0.14	0.15	0.16	0.08	0.11	0.09	0.13
Semantics	0.14	0.17	0.10	0.17	0.09	0.11	0.10	0.13
Plausibility	0.17	0.18	0.16	0.15	0.11	0.15	0.14	0.17
Utility	0.12	0.17	0.15	0.12	0.08	0.12	0.08	0.13
Average	0.19	0.22	0.18	0.20	0.12	0.16	0.13	0.18
Grammar	0.12	0.08	0.17	0.14	0.06	0.10	0.06	0.11
Semantic	0.12	0.17	0.10	0.17	0.07	0.10	0.10	0.10
Plausibility	0.13	0.11	0.18	0.09	0.06	0.11	0.10	0.13
Utility	0.12	0.16	0.13	0.11	0.07	0.11	0.10	0.15
Average	0.17	0.19	0.21	0.18	0.10	0.15	0.12	0.17

Table 5.17: Correlation between all automatic metrics and AMT evaluations, for all questions evaluated on SQuAD, using AMT average results (top) and median results (bottom).

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Grammar	0.15	0.08	0.13	0.09	0.08	0.12	0.14	0.10
Semantics	0.15	0.10	0.10	0.08	0.08	0.10	0.18	0.09
Plausibility	0.19	0.15	0.14	0.12	0.10	0.20	0.24	0.17
Utility	0.14	0.12	0.12	0.11	0.10	0.14	0.17	0.13
Average	0.20	0.15	0.16	0.13	0.12	0.18	0.24	0.15
Grammar	0.11	0.04	0.10	0.08	0.06	0.09	0.08	0.07
Semantics	0.14	0.08	0.10	0.07	0.08	0.10	0.17	0.09
Plausibility	0.12	0.08	0.09	0.07	0.05	0.13	0.17	0.09
Utility	0.09	0.08	0.06	0.07	0.04	0.10	0.09	0.10
Average	0.16	0.10	0.12	0.10	0.08	0.15	0.18	0.12

a correlation between the automatic metrics and the AMT evaluation, as seen in Tables 5.18 and 5.19, for all evaluated questions on MONSERRATE and SQuAD, respectively.

In the end, it is safe to say that, for the task of QG, automatic metrics do not correlate well with question quality, as they are measuring the capability of recovering the reference instead of assessing the question directly. For example, if a similar question is not in the reference, automatic metrics will not score highly, while human evaluation will be able to reward it. This is related to one of the topics discussed throughout this thesis, which is the quality of the references used. Comparing MONSERRATE and SQuAD, although all correlation values are small, we can see that MONSERRATE obtains higher values of correlation, in both Spearman

Table 5.18: Kendall's rank correlation between all automatic metrics and AMT evaluations, for all questions evaluated on MONSERRATE, using AMT average results (top) and median results (bottom).

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Grammar	-0.03	0.08	0.13	0.08	0.06	0.06	0.03	0.06
Semantics	0.00	0.11	0.09	0.09	0.06	0.07	0.04	0.06
Plausibility	-0.03	0.14	0.13	0.10	0.08	0.08	0.06	0.09
Utility	0.01	0.11	0.10	0.07	0.05	0.06	0.02	0.05
Average	0.01	0.16	0.13	0.13	0.09	0.10	0.08	0.11
Grammar	-0.05	0.04	0.14	0.05	0.04	0.04	-0.01	0.03
Semantics	-0.02	0.07	0.10	0.06	0.04	0.05	0.00	0.03
Plausibility	-0.06	0.08	0.13	0.03	0.04	0.05	0.01	0.05
Utility	-0.01	0.07	0.10	0.03	0.04	0.04	0.00	0.04
Average	-0.03	0.13	0.15	0.11	0.08	0.09	0.06	0.09

Table 5.19: Kendall's rank correlation between all automatic metrics and AMT evaluations, for all questions evaluated on SQuAD, using AMT average results (top) and median results (bottom).

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Grammar	0.09	0.04	0.08	0.06	0.07	0.07	0.06	-0.04
Semantics	0.09	0.05	0.07	0.08	0.05	0.05	0.10	-0.04
Plausibility	0.12	0.09	0.10	0.11	0.10	0.14	0.12	-0.11
Utility	0.09	0.07	0.08	0.08	0.09	0.10	0.09	-0.08
Average	0.13	0.08	0.10	0.11	0.09	0.11	0.12	-0.09
Grammar	0.07	0.01	0.07	0.05	0.06	0.05	0.03	-0.02
Semantics	0.08	0.03	0.07	0.06	0.04	0.05	0.09	-0.01
Plausibility	0.07	0.05	0.06	0.06	0.07	0.09	0.08	-0.04
Utility	0.05	0.03	0.04	0.02	0.04	0.07	0.05	-0.04
Average	0.11	0.06	0.08	0.09	0.07	0.10	0.09	-0.07

and Kendall's, with highlight for the average score, which we believe to be consequence of MONSERRATE being a larger reference. This was also empirically supported when we discussed the results obtained on SQuAD using both evaluation procedures.

One could say these results voided our contribution of MONSERRATE. However, as pointed by Novikova et al. [2017], there is advantages in using automatic metrics. Although there is week correlation overall with human evaluation, it was shown that there is moderate correlation in automatic metrics for low rated instances. In other words, automatic metrics can be useful to detect where a system is failing. This fact, in conjunction to the evidence of better correlation for a larger dataset, prove that MONSERRATE is still a valuable resource to the community, both in development and test settings, and that the missing piece is a better automatic metric for the task of QG.

5.4 Discussion

In this chapter we benchmarked two state of the art systems and our own GEN, on two different datasets: the widely used SQuAD, and our new MONSERRATE, introduced in the previous chapter. We performed a detailed analysis on both automatic metrics and human evaluation, and showed that GEN is able to compete with H&S and surpass D&A on MONSERRATE. SQuAD led to an apparent lower performance, which was not corroborated by human evaluation, where GEN surpassed the other systems, which supports our claims that the quality of the reference has a major impact on QG evaluation.

We also did a correlation study between the automatic metrics and the human evaluation, and showed that there is no strong correlation between the two. However, correlation values were overall lower on SQuAD, which suggests that with a shallower reference it is harder to establish a correspondence between automatic metrics and human evaluation. A larger reference like MONSERRATE showed better correlation, but a better automatic metric for the task of QG might be the missing piece.

Concerning GEN intrinsic performance, the different tree matching strategies suggest that they are the core piece of GEN as of now, and potential improvements would imply modifying them. We addressed the trade off between the more rigid approach of `Strict` matching, leading to better results at cost of fewer generated questions, or the `Argument` matching, that is more flexible but less accurate. The latter does not use the *equiv* function, for example, while the former depends on it, reason why we believe the empirically set values for *equiv* function are not a major concern of the system's performance.

6 Using Implicit Feedback

Even if the automatic question generation process can help in many different tasks, there is usually a validation step performed by the end user, who parses the questions and fixes errors or discard bad questions. Typically, these corrections are wasted, although being informative. In other words, they can be seen as implicit feedback on the system's performance.

In this chapter we focus on this implicit feedback given by an expert when correcting the system's output. This feedback is useful for GEN in two dimensions: first, it tells the system how accurate the generation process is, which can be used to score the patterns generating the questions. Scoring can be used to weigh the patterns, which indirectly translates into being able to rank the generated questions. Secondly, each corrected question can be used as a trustful new seed that can be used to learn new patterns, ideally covering new templates of questions.

In this chapter we explore these ideas, by incorporating them into GEN in a iterative process to simulate future interactions.

6.1 Experimental setup

In order to capture the effects of the implicit feedback over time, we simulate the iteration process of system-user interaction by batching the input source. Therefore, instead of using a single target corpus as a one-time input, we batch it in multiple smaller inputs, allowing the system to evolve over time by using feedback obtained in previous batches.

Algorithm 2 shows the pseudo-code for the iterative process. Being given an original set of seeds, patterns are generated as previously described in Chapter 3 (line 5). The learning materials are divided in batches of sentences to which patterns are applied, resulting in new questions (line 8). Every generated question is presented to the teacher to be corrected or

Algorithm 2 Iterative loop for online learning with GEN.

```
1: $seeds \leftarrow \{(s_1, q_1), ..., (s_n, q_n)\}$
2: $Batches \leftarrow \{Batch_1, ..., Batch_m\}$
3: $P \leftarrow \{\}$
4: for all $b \in Batches$ do
5:     $P \leftarrow P \cup generatePatterns(seeds)$
6:     $seeds \leftarrow \{\}$
7:     for all $s \in b$ do // $s$ is a single sentence in the batch $b$
8:         $Q \leftarrow generateQuestions(s, P)$
9:         for all $q \in Q$ do
10:            $q' \leftarrow correctQuestion(q)$
11:            if $q' \neq null$ then
12:                $seeds \leftarrow seeds \cup \{(s, q')\}$
13:            end if
14:            $P \leftarrow weighPattern(q, q', P)$
15:        end for
16:        $P \leftarrow discardPatterns(P)$
17:    end for
18: end for
```

discarded (line 10). After being corrected by the teacher, each question is associated with the sentence that originated it and added to the set of seeds, allowing the creation of new patterns (line 12), and also used to score the pattern that generated it (line 14). Finally, the patterns that generated only discarded questions are removed from the pool of patterns, reducing the number of ill questions to be generated in future batches (line 16). The process repeats for all batches (lines 4-18).

From Section 3.4, we mentioned two techniques to update the score of a given pattern: Weighed Majority Algorithm (WMA) and Exponentially Weighed Average Forecast (EWAF). Each has specific values that need to be set, like the loss penalty or the similarity function *sim*.

To evaluate the generated questions against the reference we use the same metrics used in previous chapters, but at top N, i.e., N is a cut to the top questions in the list of all questions. In these experiments we take $N = 5$, $N = 10$ and $N = 20$, as if the teacher would be only presented those top N questions.

We used MONSERRATE in these experiments, as it is an extensive reference that allows a easier automatic validation of the results. The batches for Algorithm 2 were created by

Table 6.1: Parametrization of the different variables in the weighing strategies, WMA and EWAF: function *sim*, its threshold *th*, penalties and bonus values.

	sim	***th***	***penalty***	***bonus***
WMA	Overlap, Lev	0.9, 0.8	0.1, 0.2	0.1, 0.3, 0.5
EWAF	Overlap, Lev	-	0.1, 0.2	-

splitting the corpus in equal parts. We tried three different sizes: 7 (leading to roughly 10 batches), 10 (7 batches), and 12 (6 batches)[1].

We run different parameterizations for the weighing techniques described. Besides the different *sim* functions described before in Section 3.4 (Overlap and normalized Levenshtein), we also set the different parameters for both. As both strategies were adapted to our problem, we empirically chose these values. For WMA we set the following weights for *penalty* and *bonus* parameters, respectively, to: 0.1, 0.2, and 0.1, 0.3, 0.5. The threshold *th* (Equation 3.11) for *sim* function was set to 0.9 and 0.8. For EWAF we set the *penalty* to values of 0.1 and 0.2 – the threshold *th* and *bonus* parameters are not applicable. Table 6.1 summarizes this information.

We also set two baselines. The first corresponds to the `original patterns`, used in Chapter 5, applied to all batches (i.e., there is no learning of patterns with new batches). Because there is no ranking of the generated questions, and to get a more accurate baseline, we average three different random orderings for this baseline. The second corresponds to the algorithm of learning patterns, but with no weighing strategy in place (`baseline`). Again, the generated questions in each batch are not ordered, so all the reported results correspond to the average of three different random orderings as well.

6.2 Overall Results

In this section we present the results obtained with the batching strategy for pattern learning and scoring. The combination of all different parameters from Table 6.1 lead to a large number of possible combinations. As we will discuss throughout this section, some configurations tend to perform similarly, if not equally. Therefore, as we draw these conclusions

[1]Note that MONSERRATE has 73 sentences.

Table 6.2: Comparison of the weighing strategies against the baselines, measured by automatic metrics on MONSERRATE, at top 5. Scores normalized by the best score obtained in each metric. Overall results for batches of size 10 (7 batches) – averaged on all but first batch.

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Original Patterns	0.74	0.62	0.83	0.48	0.94	0.84	0.76	0.82
Baseline	0.92	0.81	**1.00**	0.67	0.99	0.91	0.86	**1.00**
EWAF-Lev-01	0.97	*0.78*	*0.99*	0.80	0.99	0.91	0.89	*0.99*
EWAF-Lev-02	0.99	0.93	*0.96*	0.97	**1.00**	**1.00**	0.95	*0.99*
EWAF-Overlap-01	**1.00**	**1.00**	*0.96*	**1.00**	**1.00**	**1.00**	**0.96**	*0.99*
EWAF-Overlap-02	0.97	0.82	*0.99*	0.87	0.99	0.91	0.88	*0.99*
WMA-Lev-08-01	0.97	*0.79*	*0.98*	0.85	0.99	0.91	**0.93**	*0.98*
WMA-Lev-08-02	0.97	**0.82**	*0.99*	0.88	0.99	0.91	**0.93**	*0.99*
WMA-Lev-09-01	**0.98**	0.81	*0.99*	**0.89**	0.99	**0.92**	**0.93**	**1.00**
WMA-Lev-09-02	0.97	*0.79*	*0.98*	0.85	0.99	0.91	**0.93**	*0.99*
WMA-Overlap-08-01-B03	*0.90*	**0.85**	*0.92*	*0.45*	0.99	**0.92**	**1.00**	*0.97*

along the text, we will omit some configurations from the tables and discussion, for sake of readability. Complete data can be found in Appendix D, where all results are listed.

The first iteration was run for batches of size 10 (7 batches). Tables 6.2 to 6.4 show the results for the top N questions ranked, with N equal to 5, 10, and 20, respectively, and for the different configurations, averaging the results obtained in all but the first batch, as the learning phase only starts after acquiring data from the first batch. The names indicate the configuration following the same order of Table 6.1. For instance, `WMA-Lev-08-01-B03` corresponds to using WMA as weighing strategy, with *sim* function being Levenshtein, the threshold *th* 0.8, the *penalty* 0.1 and the *bonus* 0.3.

The results are normalized across each column by the best score obtained. We opted to present results this way so it is easier to understand to what degree strategies can improve over others. A score of 1.00 thus represents the best obtained score for that metric among all strategies.

The tables are organized by strategy (EWAF and WMA), and then by other parameters. Highlighted results correspond to improvements against the baseline for that strategy, but not necessarily the best result attained overall. Italicized values correspond to results worse than the baseline. Finally, non-highlighted results are better than the baseline, but not

Table 6.3: Comparison of the weighing strategies against the baselines, measured by automatic metrics on MONSERRATE, at top 10. Scores normalized by the best score obtained in each metric. Overall results for batches of size 10 (7 batches) – averaged on all but first batch.

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Original Patterns	0.74	0.68	0.82	0.53	0.95	0.84	0.78	0.83
Baseline	0.94	0.93	**1.00**	0.83	0.99	0.92	0.90	**1.00**
EWAF-Lev-01	0.97	0.94	*0.98*	0.85	0.99	0.92	0.92	*0.98*
EWAF-Lev-02	0.98	0.94	*0.99*	0.96	**1.00**	**1.00**	0.95	*0.98*
EWAF-Overlap-01	**1.00**	**0.99**	*0.99*	**1.00**	**1.00**	**1.00**	**0.96**	*0.99*
EWAF-Overlap-02	0.97	0.96	*0.98*	0.88	**1.00**	0.92	0.92	*0.98*
WMA-Lev-08-01	0.97	0.98	*0.98*	0.90	0.99	0.92	0.96	*0.98*
WMA-Lev-08-02	**0.98**	**1.00**	*0.98*	0.92	0.99	0.92	**0.97**	**1.00**
WMA-Lev-09-01	0.97	0.97	*0.98*	**0.94**	0.99	0.92	0.96	*0.99*
WMA-Lev-09-02	0.97	0.98	*0.98*	0.90	0.99	0.92	0.96	*0.99*
WMA-Overlap-08-01-B03	*0.88*	*0.82*	*0.93*	*0.44*	0.99	*0.91*	**1.00**	*0.94*

the best results for that strategy. For instance, in Table 6.2, the forth row corresponds to `EWAF-Lev-01`, and surpasses the baseline for all metrics except METEOR, BLEU1 and Vector Extrema Cosine Similarity (VECS), although it does not surpass `EWAF-Overlap-01` in any metric (which are therefore highlighted in that row).

Analyzing the results per strategy for all cuts of N, the first thing to note is that the baseline improves over the `original patterns` (from 17% up for lexical measures), which means that learning new seeds already improves the original GEN. Then, we can see that for EWAF the results are typically similar, with slightly tendency for better performance for `EWAF-Overlap-01`, across all top N. For `WMA`, without *bonus*, results are also similar across all parameters (similarity measure *sim*, threshold *th*, and *penalty*), with lightly preference for smaller penalties and tighter thresholds (`WMA-Lev-09-01` for example). Tables only show `Lev` parameterizations, as `Overlap` leads to similar performances[2]. Finally, for `WMA` configurations with *bonus*, results are typically identically among them, independently of the values chosen (only one configuration shown)[3]. If at top 5 there are some improvements in some ends, it rapidly vanishes when increasing N to 10 and 20, meaning it is more prejudicial than beneficial.

[2]Consult Appendix D for the complete tables.

[3]See footnote 2.

Table 6.4: Comparison of the weighing strategies against the baselines, measured by automatic metrics on MONSERRATE, at top 20. Scores normalized by the best score obtained in each metric. Overall results for batches of size 10 (7 batches) – averaged on all but first batch.

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Original Patterns	0.76	0.75	0.81	0.65	0.96	0.85	0.85	0.85
Baseline	0.89	0.93	0.96	0.87	0.98	0.91	0.97	0.95
EWAF-Lev-01	0.92	0.97	*0.95*	0.90	0.98	0.92	0.99	0.98
EWAF-Lev-02	0.99	0.97	**1.00**	0.96	**1.00**	**1.00**	0.99	**1.00**
EWAF-Overlap-01	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
EWAF-Overlap-02	0.92	0.97	*0.95*	0.90	0.98	0.92	0.99	0.98
WMA-Lev-08-01	0.90	0.94	*0.94*	0.83	0.98	0.91	**1.00**	0.97
WMA-Lev-08-02	0.90	0.94	*0.94*	0.83	0.98	0.91	**1.00**	0.97
WMA-Lev-09-01	**0.94**	**0.99**	**0.97**	**0.95**	**0.99**	**0.92**	0.98	**0.99**
WMA-Lev-09-02	0.92	0.96	*0.95*	0.90	0.98	0.91	0.98	0.98
WMA-Overlap-08-01-B03	*0.88*	*0.85*	0.96	*0.72*	*0.97*	*0.90*	*0.93*	*0.94*

Table 6.5: Comparison of the weighing strategies against the baselines, measured by automatic metrics on MONSERRATE, at top 5. Scores normalized by the best score obtained in each metric. Overall results for batches of size 7 (10 batches) – averaged on all but first batch.

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Original Patterns	0.85	0.73	0.85	0.52	0.95	0.93	0.87	0.85
Baseline	0.95	0.91	**1.00**	0.68	**1.00**	**1.00**	0.97	0.99
EWAF-Lev-01	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**0.98**	**1.00**
EWAF-Lev-02	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**0.98**	**1.00**
EWAF-Overlap-01	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**0.98**	**1.00**
EWAF-Overlap-02	0.99	**1.00**	*0.93*	0.99	**1.00**	**1.00**	**0.98**	**1.00**
WMA-Lev-08-01	0.95	0.94	*0.97*	0.91	*0.99*	*0.98*	**1.00**	0.99
WMA-Lev-08-02	0.95	**0.97**	*0.97*	**0.92**	*0.99*	*0.98*	**1.00**	*0.96*
WMA-Lev-09-01	**0.96**	0.94	*0.98*	0.90	*0.99*	*0.98*	0.99	*0.96*
WMA-Lev-09-02	0.95	0.94	*0.97*	0.91	*0.99*	*0.98*	**1.00**	*0.96*
WMA-Overlap-08-01-B03	*0.90*	*0.82*	*0.99*	*0.59*	**1.00**	*0.99*	*0.84*	*0.96*

Tables 6.5 to 6.7 show the same evaluation procedure for the top 5, 10 and top 20 ranked questions, respectively, for the same experiment with batches of size 7 (10 batches). For $N = 5$ results overall are typically worse than the baseline, except for the `EWAF` configurations. For greater values of N, the pattern witnessed from the previous experiment (7 batches) occurs as well: `EWAF` strategy shows the best improvements, but `WMA` also shows improvements across all metrics. The major difference is the *bonus* configurations, that show improvements as

Table 6.6: Comparison of the weighing strategies against the baselines, measured by automatic metrics on MONSERRATE, at top 10. Scores normalized by the best score obtained in each metric. Overall results for batches of size 7 (10 batches) – averaged on all but first batch.

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Original Patterns	0.99	0.87	0.89	**1.00**	0.97	0.97	0.89	0.91
Baseline	*0.91*	0.88	0.95	*0.71*	0.98	0.98	0.95	0.97
EWAF-Lev-01	**1.00**	**1.00**	**1.00**	*0.92*	**1.00**	**1.00**	0.98	**1.00**
EWAF-Lev-02	**1.00**	**1.00**	**1.00**	*0.94*	**1.00**	**1.00**	0.98	**1.00**
EWAF-Overlap-01	**1.00**	**1.00**	**1.00**	*0.94*	**1.00**	**1.00**	0.98	**1.00**
EWAF-Overlap-02	**1.00**	**1.00**	0.98	*0.94*	**1.00**	**1.00**	0.98	**1.00**
WMA-Lev-08-01	0.98	**0.96**	0.99	*0.88*	**1.00**	**1.00**	0.98	**1.00**
WMA-Lev-08-02	0.98	**0.96**	0.99	*0.88*	**1.00**	**1.00**	0.98	0.97
WMA-Lev-09-01	**0.99**	**0.96**	**1.00**	*0.89*	**1.00**	**1.00**	**1.00**	0.97
WMA-Lev-09-02	**0.99**	**0.96**	0.99	*0.87*	0.99	**1.00**	0.99	0.97
WMA-Overlap-08-01-B03	*0.89*	*0.81*	**0.96**	*0.59*	0.98	0.98	*0.87*	*0.94*

Table 6.7: Comparison of the weighing strategies against the baselines, measured by automatic metrics on MONSERRATE, at top 20. Scores normalized by the best score obtained in each metric. Overall results for batches of size 7 (10 batches) – averaged on all but first batch.

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Original Patterns	0.91	0.85	0.86	0.88	0.94	0.91	0.85	0.87
Baseline	0.93	0.92	0.95	*0.82*	0.99	0.98	0.97	0.97
EWAF-Lev-01	0.97	0.98	0.97	0.88	**1.00**	0.99	0.99	0.99
EWAF-Lev-02	**1.00**	**1.00**	0.99	0.99	**1.00**	**1.00**	**1.00**	**1.00**
EWAF-Overlap-01	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.99	**1.00**
EWAF-Overlap-02	0.97	0.97	0.98	0.89	**1.00**	0.99	0.99	0.99
WMA-Lev-08-01	**0.99**	**0.98**	**0.98**	**0.91**	**1.00**	**1.00**	**1.00**	**1.00**
WMA-Lev-08-02	**0.99**	**0.98**	**0.98**	**0.91**	**1.00**	**1.00**	**1.00**	**1.00**
WMA-Lev-09-01	**0.99**	**0.98**	0.97	0.89	**1.00**	0.99	**1.00**	0.99
WMA-Lev-09-02	**0.99**	**0.98**	0.97	0.90	**1.00**	0.99	0.99	0.99
WMA-Overlap-08-01-B03	**0.95**	**0.93**	**1.00**	**0.83**	0.99	0.98	*0.92*	0.97

well, although in much less degree than the counterpart configurations.

Finally, we did the same experiment for larger batches of 12 sentences each, leading to 6 batches. Tables 6.8 to 6.10 show the same configurations as before. The same trend as before apply here, with the most improvements being witnessed at top 5 and top 10 across all metrics, specially for `EWAF` configurations. For top 20, improvements are less pronounced, and even non existent for the bonus configurations, while the baseline still improves over the

Table 6.8: Comparison of the weighing strategies against the baselines, measured by automatic metrics on MONSERRATE, at top 5. Scores normalized by the best score obtained in each metric. Overall results for batches of size 12 (6 batches) – averaged on all but first batch.

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Original Patterns	0.84	0.79	0.90	0.66	0.94	0.91	0.80	0.90
Baseline	0.93	0.89	0.93	0.88	0.97	0.95	0.90	0.94
EWAF-Lev-01	**1.00**	**1.00**	**0.94**	**1.00**	0.97	0.95	**1.00**	0.95
EWAF-Lev-02	**1.00**	**1.00**	**0.94**	**1.00**	0.97	0.95	**1.00**	**0.99**
EWAF-Overlap-01	**1.00**	**1.00**	**0.94**	**1.00**	0.97	0.95	**1.00**	**0.99**
EWAF-Overlap-02	**1.00**	**1.00**	**0.94**	**1.00**	0.97	0.95	**1.00**	0.95
WMA-Lev-08-01	**0.96**	**0.90**	*0.91*	*0.85*	*0.96*	*0.94*	**0.98**	*0.91*
WMA-Lev-08-02	**0.96**	**0.90**	*0.91*	*0.85*	*0.96*	*0.94*	**0.98**	**0.96**
WMA-Lev-09-01	0.95	0.89	*0.89*	*0.78*	*0.96*	*0.94*	**0.98**	**0.96**
WMA-Lev-09-02	0.95	0.89	*0.89*	*0.78*	*0.96*	*0.94*	**0.98**	**0.96**
WMA-Overlap-08-01-B03	**0.99**	**0.98**	**1.00**	*0.77*	**1.00**	**1.00**	*0.86*	**1.00**

Table 6.9: Comparison of the weighing strategies against the baselines, measured by automatic metrics on MONSERRATE, at top 10. Scores normalized by the best score obtained in each metric. Overall results for batches of size 12 (6 batches) – averaged on all but first batch.

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Original Patterns	0.90	0.86	0.94	0.75	0.96	0.94	0.87	0.92
Baseline	0.94	0.92	0.97	0.81	0.98	0.97	0.95	0.98
EWAF-Lev-01	**0.99**	**1.00**	*0.95*	**1.00**	0.98	0.97	**1.00**	0.96
EWAF-Lev-02	**0.99**	**1.00**	*0.95*	**1.00**	0.98	0.97	**1.00**	**1.00**
EWAF-Overlap-01	**0.99**	**1.00**	*0.95*	**1.00**	0.98	0.97	**1.00**	**1.00**
EWAF-Overlap-02	**0.99**	**1.00**	*0.95*	**1.00**	0.98	0.97	**1.00**	0.96
WMA-Lev-08-01	**1.00**	**0.99**	*0.95*	**0.95**	0.98	**0.98**	**1.00**	0.96
WMA-Lev-08-02	**1.00**	**0.99**	*0.95*	**0.95**	0.98	**0.98**	**1.00**	**1.00**
WMA-Lev-09-01	0.97	0.93	*0.93*	0.91	*0.97*	*0.97*	0.98	0.98
WMA-Lev-09-02	0.97	0.93	*0.93*	0.91	*0.97*	*0.97*	0.98	0.98
WMA-Overlap-08-01-B03	**0.97**	0.92	**1.00**	*0.64*	**1.00**	**1.00**	*0.85*	**1.00**

original patterns.

To summarize, we can see that our approach is successful in two ends. First, learning new patterns leads to improvements compared to using the original patterns, even if not using the implicit feedback of the user to weigh the patterns. Then, by using the implicit feedback, we are able to score the patterns to improve the results obtained even further, by ranking the questions.

Table 6.10: Comparison of the weighing strategies against the baselines, measured by automatic metrics on MONSERRATE, at top 20. Scores normalized by the best score obtained in each metric. Overall results for batches of size 12 (6 batches) – averaged on all but first batch.

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Original Patterns	0.88	0.81	0.90	0.74	0.98	0.94	0.88	0.89
Baseline	0.97	0.93	**1.00**	0.87	0.99	0.99	**1.00**	0.98
EWAF-Lev-01	0.99	**0.99**	*0.97*	**0.97**	**1.00**	**1.00**	*0.97*	**1.00**
EWAF-Lev-02	0.99	**0.99**	*0.98*	**0.97**	**1.00**	**1.00**	*0.97*	**1.00**
EWAF-Overlap-01	**1.00**	**0.99**	*0.98*	**0.97**	**1.00**	**1.00**	*0.97*	**1.00**
EWAF-Overlap-02	**1.00**	**0.99**	*0.98*	**0.97**	**1.00**	**1.00**	*0.97*	**1.00**
WMA-Lev-08-01	**1.00**	0.99	*0.98*	0.97	**1.00**	**1.00**	*0.97*	**1.00**
WMA-Lev-08-02	**1.00**	0.99	*0.98*	0.97	**1.00**	**1.00**	*0.97*	**1.00**
WMA-Lev-09-01	0.99	0.99	*0.97*	0.95	**1.00**	**1.00**	*0.97*	**1.00**
WMA-Lev-09-02	**1.00**	**1.00**	*0.98*	**1.00**	**1.00**	**1.00**	*0.97*	**1.00**
WMA-Overlap-08-01-B03	*0.93*	*0.86*	*0.96*	*0.61*	0.99	*0.98*	*0.91*	*0.96*

The results obtained improved across all metrics for different cuts of N (5, 10, 20), for some configurations. Even with semantic metrics, that we deemed to be not as discriminative, the strategy showed some improvements. Both ROUGE and METEOR show gains when compared to the baselines, although at different extents, indicating the effectiveness of this approach.

BLEU, on the other hand, is the metric with more erratic behavior (mostly BLEU1 but also applies to BLEU4). It shows less improvements overall for the weighing strategies when compared to both baselines, sometimes even scoring below the baseline. Additionally, the *bonus* configurations[4] show some improvements in some runs, contradicting their trend for all other metrics. This could be explained by the nature of the BLEU metric, as it uses the whole reference instead of each of the hypotheses alone. For example, the question *What designs the palace?*, generated from *The palace was designed by the architects Thomas James Knowles (father and son) and built in 1858, having been commissioned by Sir Francis Cook, Viscount of Monserrate* obtains a high score of 0.75, when the reference is composed of: 1) *Who designed the palace?* 2) *When was the palace built?* 3) *Who commissioned the construction of the palace?* 4) *What was Sir Francis Cook noble tittle?*. This means that the baseline is not punished for having ill-formulated questions on the top as much as it should when using

[4]Those with `B` in their name.

Table 6.11: Number of patterns and questions per batch, along with the number of questions discarded by the expert, and the average question editing, for batches of size 12 (6 batches), on MONSERRATE.

Size 12	**1**	**2**	**3**	**4**	**5**	**6**
Patterns	11	24	33	39	47	42
Questions	132	176	266	267	191	503
Unique	52	41	54	72	77	111
Discarded	23	74	55	71	26	31
%	17.42	42.05	20.68	26.59	13.61	6.16
Edit Avg	0.36	0.40	0.40	0.45	0.43	0.44

BLEU as a metric, contributing to the contradictory results.

6.3 Evolution over Batches

As seen before, results show gains in all metrics against the baselines, although to different extents, proving that the technique is effective, even for different batching sizes. However, we must note that there is a clear trade-of between the batches' sizes: if they are too small, there is not enough data to learn new patterns and update the weights; if they are too large, there is more data to learn the weights, but less time to see the impact of that training.

The results presented in the previous section respect to the average results on across all but the first batch, but do not show how GEN performs along time. Tables 6.11 to 6.13 show the number of patterns and questions per batch for the different runs, along with the number of questions discarded by the expert and the average normalized Levenshtein distance between the generated question and its correction. When looking at Table 6.13 we can see that the sentences in the batches influence the number of questions generated: it appears that some sentences are better sources of questions than others, which will lead to different outcomes, including not generating any questions or less than 20 questions (highlighted), which means top 20 results are meaningless for those batches. Nevertheless, results point to a overall increase of patterns and generated questions, with the number of questions discarded increasing at first but then diminishing with time. The edit cost, measured by normalized Levenshtein, seems to be more or less constant across all batches but the first, with tendency to increase over time.

Table 6.12: Number of patterns and questions per batch, along with the number of questions discarded by the expert, and the average question editing, for batches of size 10 (7 batches), on MONSERRATE.

Size 10	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Patterns	11	24	22	28	30	36	33
Questions	121	175	67	215	155	108	379
Unique	50	35	25	54	56	52	95
Discarded	23	74	1	56	27	11	28
%	19.01	42.29	1.49	26.05	17.42	10.19	7.39
Edit Avg	0.35	0.45	0.42	0.37	0.43	0.45	0.44

Table 6.13: Number of patterns and questions per batch, along with the number of questions discarded by the expert, and the average question editing, for batches of size 7 (10 batches), on MONSERRATE.

Size 7	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Patterns	11	26	21	20	28	31	30	32	32	43
Questions	210	237	37	46	213	85	120	0	236	369
Unique	50	42	**10**	**14**	51	24	37	-	70	90
Discarded	36	108	3	1	62	24	34	-	7	27
%	17.14	45.57	8.11	2.17	29.11	28.24	28.33	-	2.97	7.32
Edit Avg	0.36	0.40	0.36	0.37	0.36	0.44	0.43	-	0.48	0.40

Given that each batch seems to lead to different performances based on its content, it is also interesting to analyze their results individually, instead of looking at the overall average score. We picked `EWAF-Overalp-01` and `WMA-Lev-08-02` for this analysis, as they are the most consistent and successful strategies across all experiments. As noted before, with semantic metrics it is hard to understand how much the performance changes, so we look at the lexical metrics only (ROUGE, METEOR, BLEU1, BLEU4).

In Figures 6.1 to 6.3 is shown the evolution for `EWAF-Overalp-01` and `WMA-Lev-08-02` for all 3 runs – batches of size 10, 7, and 12 (7, 10 and 6 batches, respectively). The figures presented depict the difference in score between the baseline scores and the ones obtained with the given strategy, for each of the batches individually, that is, the graph is not a cumulative evolution of the scores. The figures are sequentially in pairs; for instance, for batches of size 10, Figure 6.1 shows the evolution for `EWAF-Overalp-01` and `WMA-Lev-08-02` at top and bottom, respectively, for top N 5, 10, 20 from left to right.

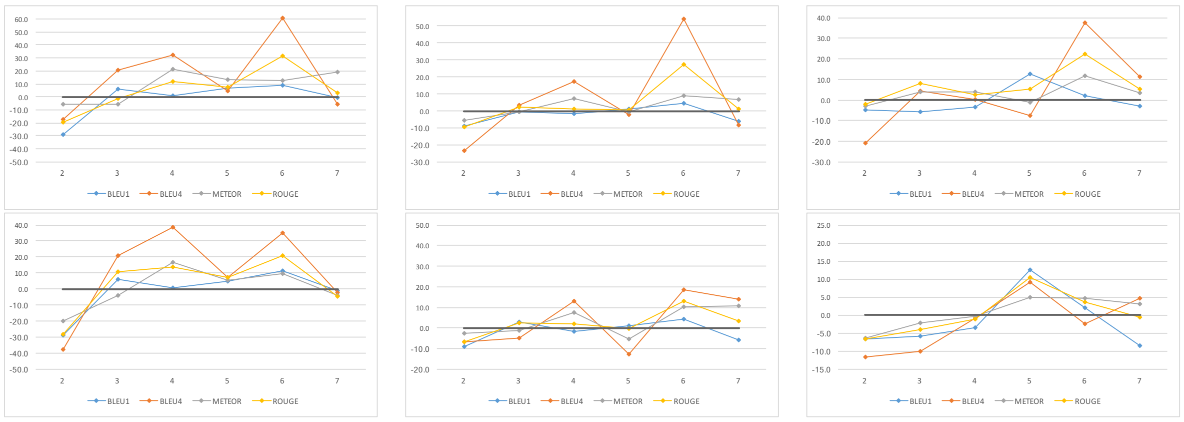


Figure 6.1: Difference between the baseline score and the score obtained by `EWAF-Overalp-01` (top) and `WMA-Lev-08-02` (bottom), over all but the first batch. Analysis for batches of size 10 (7 batches), at top 5 (left), 10 (middle), and 20 (right).

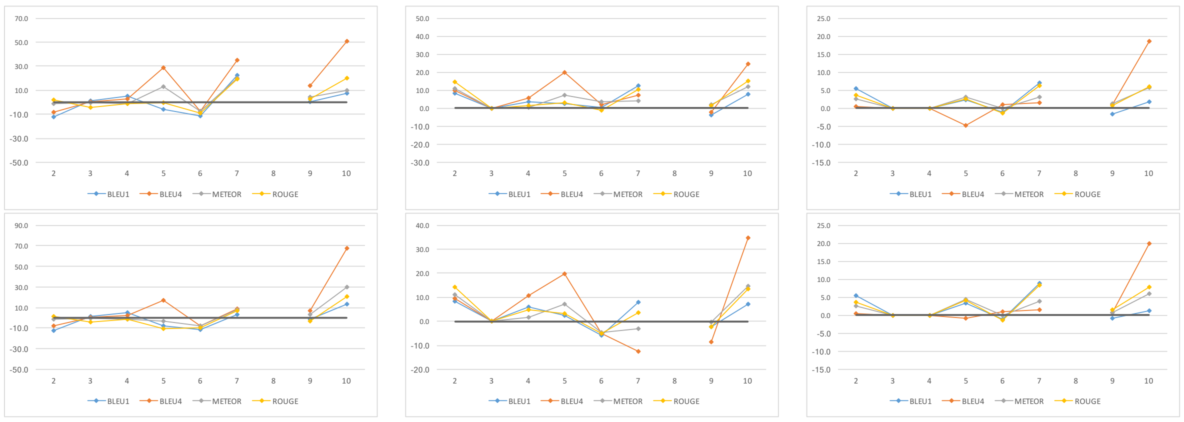


Figure 6.2: Difference between the baseline score and the score obtained by `EWAF-Overalp-01` (top) and `WMA-Lev-08-02` (bottom), over all but the first batch. Analysis for batches of size 7 (10 batches), at top 5 (left), 10 (middle), and 20 (right).

In all cases we can see batches that surpass the baseline and others that do not. However, we can see that the gains surpass the losses, seen by how often and by how much the lines stand over the zero axis.

We take a closer look at Figure 6.2, which depicts the evolution for batches of size 7 (10 batches). We can see, at top 20, that almost no loss is happening (almost all points are positive, i.e., they are above the axis). In one hand, because the batches generate less questions (sometimes close or even less than 20), there is less room to lose points to the baseline. On the other hand, that room still exists and it is being avoided, that is, the system is able to consistently improve the ordering of the questions without committing mistakes.

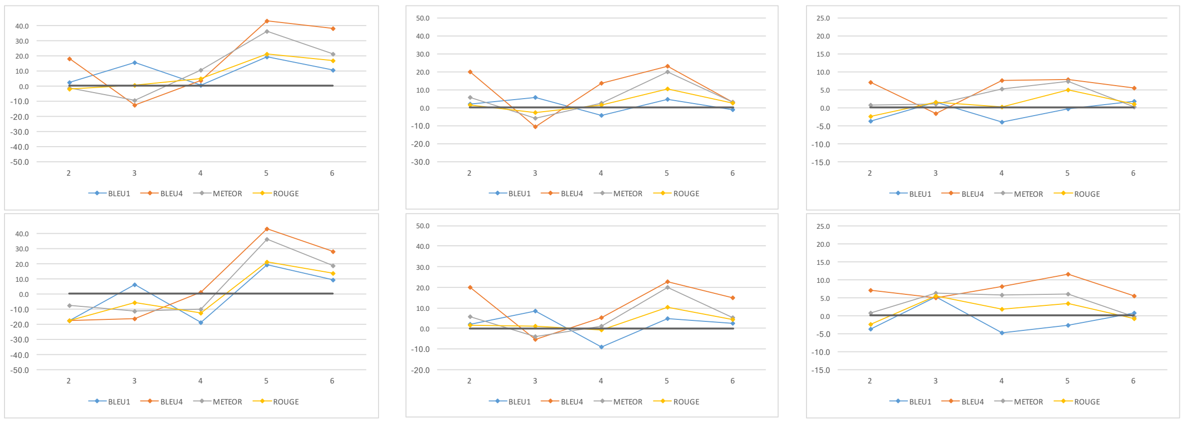


Figure 6.3: Difference between the baseline score and the score obtained by `EWAF-Overalp-01` (top) and `WMA-Lev-08-02` (bottom), over all but the first batch. Analysis for batches of size 12 (6 batches), at top 5 (left), 10 (middle), and 20 (right).

However, this does not happen for smaller values of N. For instance, top 5 shows more losses except for a few batches.

To conclude, we believe larger batches have greater impact on the performance of these strategies. The sweet spot might not be easy to find but, if the corpus is large (Monserrate is large on the reference side, but not in number of sentences), batches of 10 sentences should be good enough to make a difference. Additionally, the contents of the batches can impact the perceived performance of these strategies but, in the end, the gains outweigh the losses, as seen by the overall scores. Therefore, one cannot make conclusions based on a single iteration but should, rather, look to apply these strategies on the long term.

6.4 Ordering of Input

The experiments presented split the corpus in the same order, which may introduce bias into the equation. It is feasible that a lucky ordering of inputs lead to learning better weights, showing an apparent improvement. In this section we repeat the experiment for batches created from a shuffled version of the corpus. We chose 7 batches of 10 sentences for this second version of the experiment.

Results are presented in Tables 6.14 to 6.16. Although with different gains, we can see that there are gains in all metrics for the same strategies and configurations, for all values of

Table 6.14: Comparison of the weighing strategies against the baselines, measured by automatic metrics on MONSERRATE (shuffled order), at top 5. Scores normalized by the best score obtained in each metric. Overall results for batches of size 10 (7 batches) – averaged on all but first batch.

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Original Patterns	0.91	0.87	0.91	0.61	**1.00**	0.98	0.89	0.95
Baseline	*0.89*	*0.79*	*0.90*	0.65	*0.99*	0.98	0.92	*0.94*
EWAF-Lev-01	0.98	0.99	0.98	0.99	**1.00**	0.99	**0.98**	0.99
EWAF-Lev-02	0.98	0.99	0.98	0.99	**1.00**	0.99	**0.98**	0.99
EWAF-Overlap-01	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.97	**1.00**
EWAF-Overlap-02	0.98	0.99	0.98	0.99	**1.00**	0.99	**0.98**	0.99
WMA-Lev-08-01	*0.87*	0.91	*0.86*	0.76	*0.98*	*0.96*	0.96	0.96
WMA-Lev-08-02	*0.89*	0.90	*0.90*	0.86	*0.99*	*0.97*	0.95	0.97
WMA-Lev-09-01	**0.99**	**0.94**	**0.95**	0.85	**1.00**	**1.00**	**1.00**	**1.00**
WMA-Lev-09-02	0.92	0.91	*0.88*	**0.97**	*0.98*	0.98	0.97	0.97
WMA-Overlap-08-01-B03	*0.85*	*0.78*	*0.87*	*0.39*	*0.98*	*0.96*	**0.93**	*0.94*

Table 6.15: Comparison of the weighing strategies against the baselines, measured by automatic metrics on MONSERRATE (shuffled order), at top 10. Scores normalized by the best score obtained in each metric. Overall results for batches of size 10 (7 batches) – averaged on all but first batch.

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Original Patterns	0.96	0.94	0.99	0.79	0.99	0.98	0.89	0.96
Baseline	*0.94*	*0.90*	*0.97*	*0.77*	0.99	0.99	0.92	0.96
EWAF-Lev-01	**1.00**	**1.00**	**0.99**	**1.00**	**1.00**	**1.00**	**0.99**	**1.00**
EWAF-Lev-02	**1.00**	**1.00**	**0.99**	**1.00**	**1.00**	**1.00**	**0.99**	**1.00**
EWAF-Overlap-01	0.97	0.96	*0.97*	0.91	0.99	0.99	0.97	0.99
EWAF-Overlap-02	0.99	0.99	*0.98*	0.97	0.99	0.99	0.98	**1.00**
WMA-Lev-08-01	*0.95*	0.96	*0.95*	0.95	0.99	0.98	0.99	0.99
WMA-Lev-08-02	*0.95*	**0.97**	*0.95*	**0.98**	0.99	0.99	0.99	0.99
WMA-Lev-09-01	**0.99**	0.96	*0.97*	0.92	0.99	**1.00**	**1.00**	**1.00**
WMA-Lev-09-02	0.96	**0.97**	*0.93*	0.93	0.99	0.99	0.99	0.99
WMA-Overlap-08-01-B03	*0.94*	*0.88*	**1.00**	*0.59*	*0.97*	*0.98*	**0.93**	0.96

N (with less effectiveness on top 20).

The evolution of the learning process, however, is completely different, as seen in Table 6.17 and Figure 6.4. Comparing both runs of 7 batches we can see that the trend is completely different, which means that the performance of the system evolves as well with the content seen. As suggested before, the ordering impacts the content of the batches and, directly, their

Table 6.16: Comparison of the weighing strategies against the baselines, measured by automatic metrics on Monserrate (shuffled order), at top 20. Scores normalized by the best score obtained in each metric. Overall results for batches of size 10 (7 batches) – averaged on all but first batch.

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Original Patterns	**1.00**	0.97	**1.00**	0.89	0.99	0.98	0.93	0.97
Baseline	*0.97*	*0.94*	*0.96*	*0.83*	0.99	0.99	0.96	0.98
EWAF-Lev-01	*0.97*	**1.00**	*0.94*	**0.96**	0.99	**1.00**	**0.99**	**0.99**
EWAF-Lev-02	*0.97*	**1.00**	*0.94*	**0.96**	0.99	**1.00**	**0.99**	**0.99**
EWAF-Overlap-01	*0.97*	0.98	*0.94*	0.94	0.99	0.99	**0.99**	**0.99**
EWAF-Overlap-02	*0.97*	0.98	*0.94*	0.94	0.99	0.99	**0.99**	**0.99**
WMA-Lev-08-01	*0.97*	0.98	*0.93*	0.99	**1.00**	**1.00**	**1.00**	0.99
WMA-Lev-08-02	*0.98*	**0.99**	*0.93*	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
WMA-Lev-09-01	*0.99*	0.97	*0.97*	**1.00**	0.99	**1.00**	0.98	0.99
WMA-Lev-09-02	*0.97*	0.97	*0.94*	0.95	0.99	0.99	0.98	0.99
WMA-Overlap-08-01-B03	*0.97*	*0.90*	*0.98*	*0.77*	*0.98*	0.99	*0.94*	*0.97*

Table 6.17: Number of patterns and questions per batch, along with the number of questions discarded by the expert, and the average question editing, for batches of size 10 (7 batches), on Monserrate (shuffled order).

Size 10 (v2)	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Patterns	11	23	31	35	38	53	41
Questions	51	130	268	290	432	123	519
Unique	32	52	98	87	111	26	115
Discarded	4	25	74	25	50	41	84
%	7.84	19.23	27.61	8.62	11.57	33.33	16.18
Edit Avg	0.40	0.42	0.46	0.37	0.44	0.49	0.42

outcomes. Thus, these differences are expectable. However, in the end, the average results improve when compared with the baseline. Therefore, it is hard to reach a conclusion on how the system evolves over time, and if later batches perform better, as they depend on the previous ones, but we can at least say the overall performance is improved.

6.5 Learning New Seeds on SQuAD

As we discussed before, SQuAD as a reference has a few limitations, reason why we performed the whole experiment in this chapter using Monserrate, as it is much easier to use automatic metrics to evaluate with it. However, to understand how GEN would perform

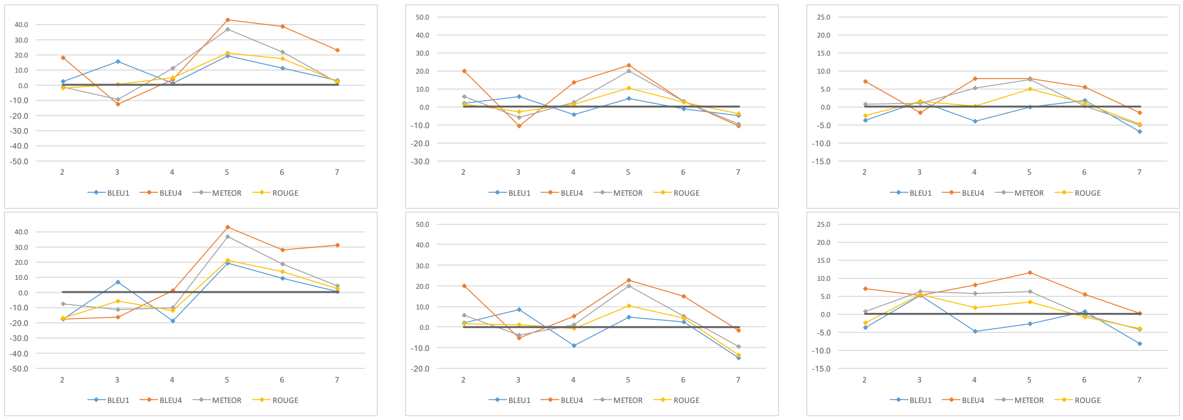


Figure 6.4: Difference between the baseline score and the score obtained by `EWAF-Overalp-01` (top) and `WMA-Lev-08-02` (bottom), over all batches but the first. Analysis for 7 batches (shuffled), at top 5 (left), 10 (middle), and 20 (right).

in a different corpus, we decided to repeat the experiment on SQuAD.

Due to the high cost of correcting questions, this time we executed a single attempt using bathes of size 10, showing the results to the two best weighing techniques also used in the previous section (`EWAF-Overalp-01` and `WMA-Lev-08-02`). The whole process is identical to the one presented at the beginning of the chapter, in Section 6.1. Because SQuAD is a larger corpus, it will lead to many batches of size 10. On Monserrate it is possible to do 7 batches, so here we extended to 10 batches in order to explore what happens when more batches are possible.

Tables 6.18 to 6.20 show the results obtained for both both weighing strategies (`EWAF-Overalp-01` and `WMA-Lev-08-02`), using the same thresholds of top 5, 10, and 20. Table 6.21 contains the statistics on the evolution of the batches regarding the number of patterns questions generated and discarded, and the average edit cost measured by normalized Levenshtein. Figure 6.5 depicts the gains obtained by each of the strategies against the baseline, for each batch but the first.

Results show a similar trend to the previous experiment, both in overall results and per batch, with gains in most batches that overcome those with losses.

Table 6.18: Overall results at top 5 on SQuAD, measured by automatic metrics, for batches of size 10 (10 batches) – average on all but first batch.

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Baseline	0.87	0.81	0.86	0.26	0.99	0.97	0.88	**1.00**
EWAF-Overlap-01	*0.80*	*0.79*	*0.82*	0.41	*0.98*	*0.95*	*0.87*	**1.00**
WMA-Lev-08-02	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.97

Table 6.19: Overall results at top 10 on SQuAD, measured by automatic metrics, for batches of size 10 (10 batches) – average on all but first batch.

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Baseline	0.97	0.87	0.93	0.66	0.98	**1.00**	0.91	0.94
EWAF-Overlap-01	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.99	**1.00**
WMA-Lev-08-02	0.98	0.95	0.95	0.89	0.99	**1.00**	**1.00**	**1.00**

Table 6.20: Overall results at top 20 on SQuAD, measured by automatic metrics, for batches of size 10 (10 batches) – average on all but first batch.

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Baseline	0.97	0.87	0.91	0.67	0.99	**1.00**	0.94	0.96
EWAF-Overlap-01	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
WMA-Lev-08-02	0.99	0.92	0.95	0.66	**1.00**	**1.00**	**1.00**	0.99

Table 6.21: Number of patterns and questions per batch, along with the number of questions discarded by the expert, and the average question editing, for batches of size 10 (10 batches), on SQuAD.

Size 7	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Patterns	11	21	38	44	66	81	108	116	121	126
Questions	41	142	125	267	394	435	541	265	457	411
Unique	25	82	40	108	107	103	144	49	95	92
Discarded	6	27	10	33	63	40	204	45	51	101
%	14.63	19.01	8.00	12.36	15.99	9.20	37.71	16.98	11.16	24.57
Edit Avg.	0.43	0.49	0.40	0.39	0.39	0.37	0.36	0.47	0.42	0.49

6.6 Transfer Learning

Given that the weighing strategies proved to be useful, we decided to select a few patterns from the initial experiment, that learned patterns on Monserrate, and use them on SQuAD. The idea with this experiment is to test how well the learning in one domain can

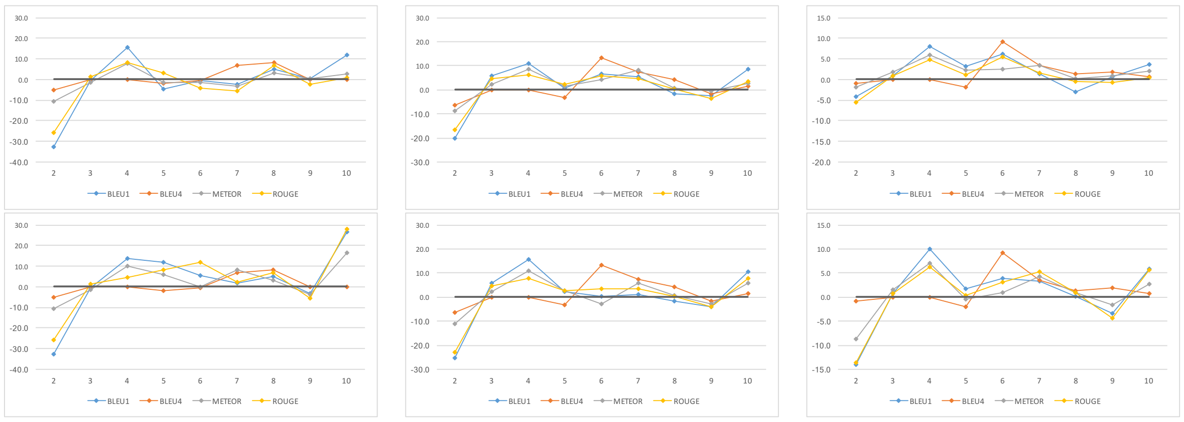


Figure 6.5: Difference between the baseline score and the score obtained by `EWAF-Overalp-01` (top) and `WMA-Lev-08-02` (bottom), over all batches but the first. Analysis for 10 batches of size 10 on SQuAD, at top 5 (left), 10 (middle), and 20 (right).

Table 6.22: Overall results obtained on SQuAD subset used for human evaluation, for GEN using the original and selected patterns, measured by automatic metrics.

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
GEN All	28.65	17.14	25.38	6.61	80.69	67.30	47.48	55.47
GEN Selected	23.14	13.95	21.20	3.58	79.87	64.24	45.00	49.73

Table 6.23: Human evaluation scores obtained on AMT for SQuAD, for GEN using the original and selected patterns, averaging the scores per question.

	Questions	grammar	semantic	plausibility	utility	Avg
GEN All	98	2.13 ±0.52	2.10 ±0.47	2.17 ±0.44	2.07 ±0.42	2.12
GEN Selected	98	2.02 ±0.50	2.05 ±0.43	2.14 ±0.50	2.02 ±0.47	2.06

be applied to another domain. We selected the best scored patterns obtained from applying `EWAF-Overlap-0.1` to bathes of size 10, picking those with a score over 0.6, in a total of 25 patterns.

We used the same subset of SQuAD used for the Amazon Mechanical Turk (AMT) evaluations from Section 5.3, resulting in 8890 unique questions from the 1889 sentences. Results with both automatic metrics and from AMT are reported in Table 6.22, and Tables 6.23 and 6.24, respectively. The questions submitted for AMT were mixed with the other's systems questions, in the procedure presented in Section 5.3.

Somewhat surprisingly, the original patterns behave better than the selected patterns,

Table 6.24: Human evaluation scores obtained on AMT for SQuAD, for GEN using the original and selected patterns, taking the median of the scores per question.

	Questions	grammar	semantic	plausibility	utility	Avg
GEN All	98	2.22 ±0.73	2.12 ±0.56	2.23 ±0.61	2.12 ±0.60	2.18
GEN Selected	98	2.03 ±0.70	2.01 ±0.60	2.11 ±0.69	2.02 ±0.59	2.04

which goes against our expectations, as results obtained in the previous sections pointed towards weighted patterns being more effective.

Further analysis revealed that some phenomena of pattern drifting may be the cause of this outcome. For example, the question *What does the terrace lead?* was generated from the input sentence *The terrace leads out into the large park.*, and was corrected to *Where does the terrace lead?*. This correction was used as a new seed, creating a new pattern of the type `Where`. This is indeed the expected behavior, and we can see that GEN was able to learn a new type of pattern by using the user's feedback. However, there is a caveat with this new pattern: the expected answer is *the large park*, which is not labeled as a location, either by a `Location` Named Entity (NE) or a `AM-LOC` semantic role. It belongs, instead, to a generic `A1` semantic role, and, losing such constraint, makes this new pattern looser than desired, leading to questions as: *Where did the movement suffer?*, from *By 1973, the PAIGC was in control of many parts of Guinea, although the movement suffered a setback in January 1973 when Cabral was assassinated.*

6.7 Discussion

In this chapter we delved into using implicit feedback as means to improve GEN. This feedback comes directly from the user who has to correct the generated questions to be used. These corrections are used in two ways: first, they are used as trustful sources of new seeds to learn new patterns from; secondly, the correction of the question itself can be used as a function of how well the generation process performed. This, indirectly, establishes how good the pattern that generated such question is, and that evaluation is used to score the pattern and, therefore, future questions coming from that same pattern.

We studied how GEN evolved over time, by batching MONSERRATE and evaluating the performance of the system as new patterns were learned and past ones were weighted. Results with automatic metrics showed that GEN was able to obtain higher overall scores across all metrics, in different parameterizations for the weighing strategies applied, when compared with the established baselines. Using the same experiment setting on SQuAD showed similar improvements, validating the strategy of using implicit feedback as means to improve GEN over time.

However, it should be noted that, although overall scores were higher, results for single batches may vary. We showed that sequencing can influence the results obtained on a given batch, and, for that reason, we cannot claim that there is a best way to apply these weighing strategies. Rather, results point to overall gains in the long term, independently of the size of the batches and corpora used, with improvements going up from 10%, depending on the metric and strategy used. Moreover, results suggest that batches of at least size 10 are large enough to promote improvements on the long-term.

Finally, we applied the best scored patterns obtained on MONSERRATE to SQuAD, attempting to show that learning patterns in one domain can be useful in another domain. However, contrary to our expectations, results with both automatic metrics and humans showed that it was not the case for this concrete experiment. Further analysis revealed some phenomena of pattern drifting, which could have contributed to the outcome.

7 Using Question Generation for Question Answering

In this chapter we study how Question Generation (QG) systems can be used in another domain closely related: Question Answering (QA). This goes back to our Hypothesis 2 detailed at the beginning of this work. In resume, our hypothesis is that it is possible to use QG to create a dataset of Question/Answer pairs that can be used by external systems as support data.

QA is the task of automatically answering questions posed in natural language. It is not our goal to discuss the field's related work, but we must highlight the recent works based on neural networks. As it happened in many other fields, Deep Learning also had impact in the QA task. SQuAD [Rajpurkar et al., 2018], that we use for our QG evaluations, was designed for QA, and is used to both train and evaluate QA systems on the task of extracting the correct answer from answer sentences associated with the test questions.

Many approaches were used from the beginning to tackle SQuAD [Rajpurkar et al., 2018], but transformers, in form of BERT [Devlin et al., 2018], and other posterior variants [Lan et al., 2019, Zhang et al., 2019], set the bar incredibly high in comparison to previous approaches [Bahdanau et al., 2017, Wang and Jiang, 2016]. SQuAD task is about extracting the correct answer to the questions, and both H&S and GEN are able to generate questions and their answers. However, we did not evaluate them on that task in this work, while D&A is not even able to generate answers. Therefore, considering these concerns, it felt that a traditional QA setting might not be a fair scenario to test our hypothesis, reason why we extended our experiments to a slightly different scenario.

In the second scenario we decided to drop the answer from the equation, and tackle QA as a retrieval problem. In other words, instead of trying to extract an answer to the question, we are focused on retrieving the appropriate answer sentence to the input question. This is particularly important for questions that require a more elaborate answer, like FAQs, or scenarios where an entire sentence is an appropriate response, such as conversational agents.

We used MONSERRATE for this scenario. However, given that BERT models have also shown strong results in sentence similarity [Reimers and Gurevych, 2019], we looked for a second dataset that would be more complex than straight factoid QA like MONSERRATE would provide. Khan Academy[1] is a online site for educational purposes, containing online courses and providing both the content and means for personalized learning. Guanliang Chen and Houben [2018] report a dataset based on Khan Academy, showing that it contains more complex questions, going byond the first level of Bloom's Taxonomy [Bloom et al., 1956], while SQuAD, according to the same authors, consists mostly of questions of the first level.

This chapter is organized as follows: in Section 7.1 we detail the experimental setup for both scenarios, in Section 7.2 we explore the SQuAD task, in Section 7.3 we delve into the second scenario (QA as a retrieval task) and finally, in Section 7.4, we draw the conclusions of this set of experiments.

7.1 Experimental Setup

For the first scenario, the SQuAD task, we take the training set and generate questions with both H&S and GEN. The questions require the answer as well, reason why we are not using D&A in this experiment. The pool of new questions is appended to the original training set, which is then used to fine tune a BERT model [Devlin et al., 2018][2]. The fine tuned model is then applied to SQuAD's dev set, and both F1 and exact match are used to evaluate the different model's performances in answer extraction. The model is fine tuned using the same parameters suggested by the authors, except for a larger batch size (14 instead of 12), and smaller maximum sequence length (256 instead of 384), to obtain a model size we could use.

For the second scenario, which sees QA as a retrieval problem, a similar workflow is followed. For MONSERRATE, we used all generated questions by H&S, D&A, and GEN, reported in Chapter 5, for dataset augmentation (this scenario does not require answers, reason why

[1]`https://www.khanacademy.org`

[2]Code can be found in `https://github.com/google-research/bert`.

Table 7.1: Exact match and F1 score obtained on SQuAD dev set by using a BERT model fine tuned on SQuAD train set (`base`), and augmented with generated questions with either H&S and GEN, using sentences (`S`) and paragraphs (`P`).

	Exact Match	F1
base	**81.14**	**88.61**
+ H&S (`S`)	80.75	88.12
+ GEN (`S`)	80.66	88.27
+ H&S (`P`)	80.55	88.19
+ GEN (`P`)	80.12	87.86

D&A was included in the experiments). For the Khan Academy dataset, we randomly selected 1000 instances from the test set, which were given to the same three systems. To perform the retrieval task, we used Sentence Transformers [Reimers and Gurevych, 2019], based on BERT, to perform sentence similarity by applying cosine similarity using the sentence embeddings generated by the model. We set two baselines, where new input test questions are compared directly with either all available sentences in the corpora or the available questions in the corpora. Then, we augment the dataset with the questions generated by the systems, and compare the test input questions with the extended version of the corpus. We measure the performance in both scenarios by calculating the accuracy in retrieving the correct answer sentence.

7.2 SQuAD Task

As explained before, we applied both H&S and GEN on SQuAD train set, generating 12664 and 34735 unique triples sentence/question/answer, respectively. These were added to the train set, and used to fine-tune a BERT model, as described by Devlin et al. [2018]. As SQuAD is actually pairing full paragraphs to question/answer pairs, we augment the train set with versions of triplets consisting of either sentences or paragraphs .

Table 7.1 reports the results for the baseline (`base`), and for the augmented versions with H&S and GEN, both using sentences (`S`) or paragraphs (`P`). Results show no gains in any configuration, which shows how strong of a baseline BERT is on its own. Other works have suggested improvements using similar strategies for this task [Duan et al., 2017, Tang et al., 2017], but gains were marginal and the architecture used was not based on BERT. We studied other different parameterizations of the fine tuning process and dataset augmentation, but we were not able to surpass the baseline. One possible explanation for this behavior is that the generated questions are focused in specific linguistic phenomena that both systems are able to capture, hurting this way the fine-tuned model generality.

7.3 Retrieval Question Answering

This section covers the scenario where we look at QA as a retrieval task. We first will look at results obtained on MONSERRATE, and then at the Khan Academy dataset.

7.3.1 Monserrate Dataset

Imagining someone would like to build a conversational agent about Monserrate Palace, they would need a knowledge based to answer questions about it. Instead of having someone populate the knowledge base of the agent, the developer could use QG to automate that task. With that scenario in mind, we used MONSERRATE as a both a source for the initial knowledge base and test set. In Chapter 4 we split MONSERRATE in slices of different sizes to study the impact of a reference size in QG evaluation. Here we take the five folds of the slice of size 1[3] to be used as both test sets and the initial knowledge base of the agent. For instance, taking fold `1.a` as a test set, we use folds `1.b` to `1.e` as the initial knowledge base of the agent. Each question in `1.a` is used as the new input test question the agent would receive, and it is compared either with the sentences in the corpus (baseline 1), the questions in the knowledge base (baseline 2), or the augmented knowledge base.

We used two types of sentence embeddings from Sentence Transformers: `nli` and `stsb`. They differ on the datasets used for fine tuning the original BERT-large model. Details can be found on the repository[4] and in Reimers and Gurevych [2019].

Tables 7.2 and 7.3 report the results obtained in this experiment for both models, `nli` and `stsb` respectively. The tables show all combinations of with slices `1.a` to `1.e` in any configuration possible of using them either as test fold or knowledge base fold. For example, the first rows report the experiment using `1.a` as test fold and `1.b` as the initial knowledge base.

[3]Each fold is a subset of the dataset where each sentence is associated with a single question.
[4]`https://github.com/UKPLab/sentence-transformers/tree/master/docs/pretrained-models`

The results presented are the accuracy obtained for retrieving the correct answer sentences when comparing the test fold questions to either the sentences of the corpus, the knowledge base questions, or the knowledge base augmented with questions generated by H&S, D&A, or GEN. Summarizing, the tables contain:

- `Test Fold` – the fold used as test set, that is, the input questions being tested;
- `KB Fold` – the fold used as the initial knowledge base;
- `Sentences` – the accuracy obtained by comparing the input questions to the sentences on the knowledge base;
- `KB Qs` – the accuracy obtained by comparing the input questions to the questions on the knowledge base;
- `+ sys` – the accuracy obtained by comparing the input questions to the questions on the augmented knowledge base with questions generated by system `sys` (or all of them together).

A first look to the results highlights two important takeaways: first, the embeddings used have tremendous impact on the accuracies obtained (approximately 7% difference on the average result). Secondly, comparing the input questions to those in a knowledge base performs poorly opposed to just comparing them directly to the target sentences of the corpus (about 10% difference, on average, for both models). One possible explanation is that the models are not trained on questions, which might tamper with the sentence embeddings obtained for the questions, thus hurting the cosine similarity scores used to select the final answer sentence.

Regarding the knowledge base augmentation with generated questions by the systems, we can see that there are some fold configurations where this strategy shows gains, with more impact using `nli` model. D&A reports the lowest impact, but generating a single question per sentence might contribute to that. Overall, it is GEN which shows more consistent gains across both experiments, but without consistently outperforming the baseline. Using all

Table 7.2: Accuracy obtained for QA as a retrieval task, using MONSERRATE as test set and original knowledge base, and augmented with questions generated with H&S, D&A, GEN, and all systems together. Sentence embeddings used from `nli` model.

Test Fold	KB Fold	Sentences	KB Qs	+ H&S	+ D&A	+ GEN	+ All
	b		30.14	38.36	32.88	43.84	47.95
	c		36.99	39.73	39.73	45.21	49.32
a	d	49.32	31.51	38.36	36.97	42.47	45.21
	e		30.14	36.99	36.99	42.47	47.95
	avg		32.19 ±2.82	38.36 ±0.97	36.64 ±2.45	43.49 ±1.14	47.60 ±1.49
	a		30.14	39.73	32.88	36.97	**42.47**
	c		35.62	38.36	**42.47**	**45.21**	**46.58**
b	d	41.10	34.25	**43.84**	38.36	38.36	**45.21**
	e		38.36	**42.47**	41.10	38.36	41.10
	avg		34.59 ±2.97	41.10 ±2.17	38.70 ±3.67	39.73 ±3.21	**43.84** ±2.17
	a		35.62	**47.95**	38.36	**50.68**	**52.05**
	b		35.62	**45.21**	**43.84**	**45.21**	**52.05**
c	d	39.73	27.40	35.62	34.25	34.25	**42.47**
	e		31.51	38.36	38.36	35.62	**42.47**
	avg		32.53 ±3.41	**41.78** ±4.99	38.70 ±3.41	**41.44** ±6.81	**47.26** ±2.17
	a		28.77	**42.47**	34.25	**45.21**	**52.05**
	b		32.88	**42.47**	**39.73**	**50.68**	**53.43**
d	c	36.99	34.25	**39.73**	34.25	**45.21**	**50.69**
	e		38.36	**42.47**	**42.47**	**49.32**	**54.79**
	avg		33.56 ±3.42	**41.78** ±1.19	**37.67** ±3.56	**47.60** ±2.45	**52.74** ±1.53
	a		31.51	**45.21**	41.10	42.47	**53.42**
	b		32.88	43.84	41.10	42.47	**50.68**
e	c	43.84	36.99	41.10	43.84	42.47	**50.68**
	d		39.73	**45.21**	**47.95**	**47.95**	**56.16**
	avg		35.27 ±3.27	43.84 ±1.68	43.49 ±2.80	**43.84** ±2.37	**52.74** ±2.27
avg Test Fold		42.19 ±4.19	33.63 ±1.17	41.37 ±1.76	39.04 ±2.35	**43.22** ±2.65	**48.84** ±2.45
	a	40.41 ±2.47	31.51 ±2.56	**43.84** ±3.06	36.64 ±3.27	**43.84** ±4.94	**50.00** ±4.39
avg	b	42.47 ±4.65	32.88 ±1.94	42.47 ±2.56	39.38 ±4.04	**45.55** ±3.12	**51.03** ±2.03
KB	c	42.81 ±4.48	35.96 ±1.14	39.73 ±0.97	40.07 ±3.67	**44.52** ±1.19	**49.32** ±1.68
Fold	d	43.49 ±3.67	33.22 ±4.48	40.75 ±3.92	39.38 ±5.16	40.75 ±5.07	**47.26** ±5.26
	e	41.78 ±4.59	34.59 ±3.80	40.07 ±2.45	39.73 ±1.17	41.44 ±5.16	**46.58** ±5.39

systems together shows the best results overall, consistently surpassing the baseline except when fold `1.a` is used as test fold.

On the bottom of both tables we also report the average accuracies for each fold when used as a knowledge base (instead of averaging with focus on the test folds). This is useful to understand that it is when the knowledge base is weaker (that is, it performs poorly by

Table 7.3: Accuracy obtained for QA as a retrieval task, using Monserrate as test set and original knowledge base, and augmented with questions generated with H&S, D&A, GEN, and all systems together. Sentence embeddings used from `stsb` model.

Test Fold	KB Fold	Sentences	KB Qs	+ H&S	+ D&A	+ GEN	+ All
a	b	52.05	38.36	50.68	45.21	52.05	**57.53**
	c		35.62	41.10	42.47	43.84	45.21
	d		38.36	46.58	43.84	41.10	47.95
	e		28.77	39.73	38.36	41.10	47.95
	avg		35.27 ±3.92	44.52 ±4.39	42.47 ±2.56	44.52 ±4.49	49.66 ±4.68
b	a	49.32	34.25	42.47	41.10	46.58	**50.68**
	c		41.10	45.21	47.95	47.95	**52.05**
	d		34.25	46.58	47.95	45.21	**52.05**
	e		41.10	46.58	43.84	47.95	**50.68**
	avg		37.67 ±3.42	45.21 ±1.68	45.21 ±2.91	46.92 ±1.14	**51.37** ±0.68
c	a	54.79	36.99	47.95	42.47	**56.16**	**58.90**
	b		41.10	52.05	53.42	**60.27**	**63.01**
	d		36.99	43.84	47.95	47.95	53.42
	e		42.47	50.68	38.36	49.32	52.05
	avg		39.38 ±2.45	48.63 ±3.14	45.55 ±5.68	53.42 ±5.03	**56.85** ±4.39
d	a	41.10	32.88	**47.95**	38.36	**53.42**	**58.90**
	b		31.51	**43.84**	39.73	**50.68**	**58.90**
	c		30.14	41.10	32.88	**42.47**	**54.79**
	e		**43.84**	**52.05**	**47.95**	**58.90**	**60.27**
	avg		34.59 ±5.43	**46.23** ±4.15	39.73 ±5.39	**51.37** ±5.93	**58.22** ±2.05
e	a	47.95	32.88	**49.32**	41.10	47.95	**63.01**
	b		39.73	46.58	43.84	49.32	**57.53**
	c		38.36	46.58	43.84	42.47	**54.79**
	d		39.73	47.95	47.95	49.32	**60.27**
	avg		37.67 ±2.28	47.60 ±1.14	44.18 ±2.45	47.26 ±2.28	**58.90** ±3.06
avg Test Fold		49.04 ±4.62	36.92 ±1.75	46.44 ±1.51	43.42 ±2.14	48.70 ±3.23	**55.00** ±2.97
avg KB Fold	a	48.29 ±4.88	34.25 ±1.68	46.92 ±2.63	40.75 ±1.49	**51.03** ±3.92	**60.27** ±1.94
	b	48.97 ±5.16	27.67 ±3.69	48.29 ±3.27	45.55 ±4.97	**53.08** ±4.26	**55.71** ±3.60
	c	47.60 ±4.04	36.30 ±4.05	43.49 ±2.45	41.78 ±5.52	**44.18** ±2.25	**56.62** ±4.66
	d	51.03 ±2.63	37.33 ±2.03	46.23 ±1.49	46.92 ±1.78	45.89 ±3.14	**53.42** ±1.12
	e	49.32 ±5.13	39.04 ±6.01	47.26 ±4.79	42.21 ±4.04	49.32 ±6.35	**55.82** ±4.48

itself) that systems are able to be more useful, highlighted by GEN's performance on helping when folds `a` to `c` are used as the initial knowledge base, and all systems together, surpassing the baseline across all folds.

Table 7.4: Examples from the Khan Academy dataset.

Answer Paragraph	Question
they 're not going to see earth as it is now. they 're going to see the region of space where earth is at a super primitive stage, shortly after the big bang. and when i use words like " shortly , " i use that also loosely.	what was there before big bang theory ?
and that 's why you see two things here. one here, and you see one here. and those are basically tubes.	what would happen to you if one of your veins popped ?

Table 7.5: Number of instances to which systems were able to generate questions, along with the number of questions generated on the 1000 instances of Khan Academy test set. The last row contains to intersection (at the instance level) of all systems.

	Instances	Questions
H&S	582	1669
D&A	923	4104
GEN	564	3907
GEN Selected	540	12927
All	345	14126

7.3.2 Khan Academy Dataset

We performed a similar experiment on a portion of Khan Academy dataset, as it is supposed to be composed of much harder questions [Guanliang Chen and Houben, 2018]. Table 7.4 shows a couple of examples from the 1000 instances selected for the test set. As it can be seen, there are some limitations with this dataset, due to sometimes not even a single clue of relationship between the question and the paragraph exists (second example), which anticipates this scenario to be a hard task to perform.

We applied the usual three systems (H&S, D&A, and GEN), and also GEN using the selected patterns for the experiment in Section 6.6. Table 7.5 reports the number of questions generated, along with the number of instances to which each system was able to generate questions. The bottom row shows the statistics for the intersection of all systems.

We used Sentence Transformers again, but this time only `stsb` model was employed, as it showed to perform better in the previous experiment. In this scenario there is no initial knowledge base, so the only baseline is to compare the input questions to the paragraphs in the dataset (`Paragraph`), and to compare them with each sentence of those paragraphs

Table 7.6: Accuracy obtained for QA as a retrieval task, using 1000 instances from Khan Academy dataset as test set. Sentence embeddings used from `stsb` model. Input questions are compared with the paragraphs, sentences, and augmented versions of them with questions generated with H&S, D&A, and GEN (both with original patterns and the ones selected for the experiment in Section 6.6).

	Paragraphs	Sentences	Generated Qs	P + Q	S + Q	Max
H&S	19.20	18.10	10.20	13.20	10.50	11.50
D&A			10.80	**19.30**		10.60
GEN			9.70	13.90	12.10	12.80
GEN Selected			9.70	13.90	11.10	12.80
All	19.20	18.10	15.00	**20.80**	**20.00**	17.70

individually (`Sentence`). Then we compare the input test questions to the generated questions by each system (`Generated Qs`), to an augmented version of the paragraphs (or sentences), which consists of appending the generated questions to the paragraph (or sentences) – `P + Q` (or `S + Q`) –, and finally also considering the maximum similarity score given by all of the aforementioned (`Max`).

Table 7.6 shows the accuracy obtained in the 1000 instances of the test set for each of the configurations described. As seen, only D&A can barely surpass the baseline, showing once again that sentence embeddings might not be suitable for questions. Using all systems together it is possible to obtain slightly better improvements, but no more than 2% (`P+Q` and `S+Q`).

Note that the impact of each system is limited to those instances to which they were able to generate questions. This means that, except for D&A, systems can only impact the accuracy of about half the test dataset. Table 7.7 reports the impact of each system looking only to those instances where a difference can be made by all systems, that is, the 345 instances to which all systems were able to generate questions. We can see the results improve overall, but it is still hard to surpass any of the baselines. The improved accuracy is likely related to the instances in analysis here. Given that all systems were able to generate a question, it might mean the paragraphs are closer to the first example of Table 7.4 than the second one. Unlike the other systems, D&A showed the least improvements now, and that can be explained by the number of instances it was impacting overall (over 900), compared to others

Table 7.7: Accuracy obtained for QA as a retrieval task, using all instances from the 1000 in the Khan Academy dataset test set to which all systems generated a question. Sentence embeddings used from `stsb` model. Input questions are compared with the paragraphs, sentences, and augmented versions of them with questions generated with H&S, D&A, and GEN (both with original patterns and the ones selected for the experiment in Section 6.6).

	Paragraphs	Sentences	Generated Qs	P + Q	S + Q	Max
H&S	25.51	28.70	22.32	26.38	24.64	21.45
D&A			15.07	24.06		13.91
GEN			18.84	26.38	22.61	23.19
GEN Selected			19.42	25.80	23.19	24.38
All	25.51	28.70	26.09	28.41	**30.43**	27.54

systems (about 500 instances). All systems together were able to improve the accuracy result by approximately 5%, when using `S+Q`.

7.4 Discussion

In this chapter we investigated how QG systems could be useful in another domain closely related: Question Answering. Our hypothesis was that automatically generated questions could improve the performance of an external system on its task of QA. We approached this by looking at QA as typical task of answer extraction, by using SQuAD task, and by looking at QA as a retrieval task.

Results did not met our expectations, with no gains on the SQuAD task. BERT models have shown to be powerful in multiple tasks, and this was confirmed here, where data augmentation was not useful for the SQuAD task.

On QA as a retrieval task, interesting results were obtained, with sporadic small gains being shown on both Monserrate and Khan Academy scenarios. Once again, by using BERT-based sentence embeddings, results revealed difficult to be surpassed, with the baseline of using question to sentence comparison establishing itself as a decent approach on its own. One possible reason is that these embeddings are not suited for questions, which can hurt the application of cosine similarity as a way to identify the best target.

Despite the results obtained, we believe there is room for QG play a role by creating many questions without human intervention. However, the solution might rely on using an ensemble of systems instead, or by training full models instead of fine-tuning already powerful BERT models. For example, question embeddings can be an interesting tool for many applications, and training a model with such goal will require a large quantity of questions.

8 Conclusions and Future Work

This thesis approaches the problem of Question Generation (QG) by revisiting the pattern-based approaches studied in the past, integrating semantic features to what are typically lexico-syntactic patterns. Different matching techniques allowed for different levels of quality control during the generation process, and we hypothesized that integrating such features in a QG system could improve its performance overall. Additionally, knowing how often, in real QG applications, the user is required to correct at least some portion of the generated questions, we hypothesized that it would be possible to build a QG system that could take advantage of this implicit feedback. Other linguistic-based approaches cannot use this directly, due to their designed based on handcrafted rules, and neural QG can hardly benefit from such small increments as well.

Given this research context, in the next sections we highlight our contributions and point to possible future work.

8.1 Contributions

This work resulted in the following contributions:

- We proposed GEN, a QG system that automatically learns semantic patterns from seeds constituted of question/answer/sentence triplets, and generate questions using those patterns (in Rodrigues et al. [2016] we describe a first version of GEN). We designed mechanisms that allows GEN to use a flexible pattern matching strategy, from more restrict lexical-based matching (`Strict Tree Matching`) to a more loose uncontrolled matching strategy (`Argument Matching`), aiming at creating less but better questions, or more and less precise questions, respectively. These semantic features at matching level were used on SemEval 2017 [Fialho et al., 2017], for the word similarity task.

We showed how the developed system competes with state of the art systems, given a small set of initial seeds, measured both by automatic metrics and humans. Introducing semantic features proved to be crucial, with questions generated by the more flexible matching strategies performing overall better than the more strict matching strategies, that are not able to generate many questions. Although GEN was not able to consistently surpass the other state of the art systems, it showed to be competitive when using as few as 8 seeds. Depending on the metric used to evaluate the results, and the parameterizations employed, GEN outperformed other systems, thus partially confirming our Hypothesis 1;

- GEN is designed to incorporate the user feedback, an idea applied in other domains but not to QG yet. The goal is to be able to enlarge the pool of available patterns by learning new seeds, and at the same time to learn weighing those patterns, so that it is possible to rank the generated questions. The ability to learn from the data seen and adapt to the user's feedback might be a valuable tool in a field that shows to be hard to properly evaluate and even fully automatize. If that is the case, then being able to become closer to the desired output is crucial, and using the implicit feedback is a step in the right direction. This strategy proved to be particularly effective if we look at the top 10-20 questions, clearly surpassing the unranked baseline version of the system, obtaining improvements going up from 10%, depending on the metric and strategy used. Preliminary results were presented in Rodrigues et al. [2018]. This respects to Hypothesis 3, completely validating our claim;

- We also contribute with an extensive corpus, MONSERRATE, that allows for automatic QG evaluation. This dataset has, on average, over 26 questions per reference sentence, making it the largest corpus available, reference-wise (a first version of MONSERRATE was reported in Rodrigues et al. [2019]). We studied how the size of a reference impacts QG evaluation, and how automatic metrics are dependent of the corpora dimension. We showed that MONSERRATE is exhaustive enough for automatic QG evaluation and that a good reference is more important than the choice of automatic metric. This corroborates our Hypothesis 4;

- Finally, we studied how QG could help external systems as support for other tasks, like Question Answering (QA). This concerns Hypothesis 2, and results were not conclusive. While in some scenarios of QA as a retrieval task, we were able to show slight improvements by augmenting an existing knowledge base (although results were not surpassing the baselines obtained consistently), in the SQuAD task, a classic QA scenario, we were not able to show improvements by enlarging the training set of a BERT model.

8.2 Future Work

For future work we believe it could be possible to improve GEN in some ends. For instance, the scoring and matching technique could use more sophisticated update strategies or other features to determine the successfulness of a pattern (for instance, Tree Kernels [Moschitti, 2006], neural networks, or even at a lower level how to use semantic features like WordNet). The current system supports the propagation of the matching scores (both at tree- and token-level), which could be also used to rate the questions and patterns during the generation step without need of human intervention. Additionally, everyday new state of the art results are attained in different Natural Language Processing (NLP) tasks accessory to QG, which implies some models used will be, eventually, outdated. GEN also relies in some empirically set values that would be better selected automatically, although the impact is likely minimum. Another aspect that could be improved is to use anaphora resolution to solve pronouns. This can be helpful in specific domains, but 1) it can easily be added *a priori*, without interfering with GEN; and 2) the use of pronouns was not a phenomena that was particularly prevalent in the scenarios to which we applied GEN. That said, we feel that the overall design and concepts are more important than these particular implementation decisions, and the system's success was not be significantly damaged by such choices. However, GEN works under the assumption the information is contained in a single sentence, not being able to generate questions that require a full paragraph to be answered. Something worth exploring in the future, but that requires changing how GEN creates patterns from the sentence level to a more coarse level.

On GEN's performance, we observed that different parameterizations perform differently across both datasets, and it was not clear which one is better. In fact, results suggest that some work better for certain question types, which is something worth investigating in the future, because it can allow GEN to be more selective during question generation if it is able to learn when it is worth to apply a specific matching strategy. Additionally, a more fine-grained pattern scoring strategy could be used, so that each of the tree matching approaches GEN uses (from `Strict Tree Matching` to `Argument Matching`) could have its own score contribution towards the patterns' scores.

Using QG as a mean to help on other tasks, like QA, revealed to be unsuccessful on SQuAD by fine-tuning a BERT model with data augmentation, but showed interesting results in QA as a retrieval task. Future work may include using automatically generated questions to train full models instead, for example with the goal of creating question embeddings. Additionally, an ensemble of QG systems might prove to be a better approach than using them individually.

Regarding Monserrate, and the size of corpora used for QG evaluation, it would be interesting to perform the same study on other datasets and domains, in order to understand if the conclusions reached in this work can be broadly applied. Another interesting direction to explore is to use Amazon Mechanical Turk (AMT) (and the same evaluation procedure) to establish how humans would evaluate the questions on Monserrate. In other words, we could draw a better comparison between the scores obtained by the QG systems and the questions on Monserrate corpus itself. It is also not clear that a suitable evaluation metric exists for the task of QG. The perceived best system depends on the metric used, which can introduce unfairness to the evaluation procedures. We suggested that an ensemble of systems could be a better approach to QG, and maybe the same goes for how evaluation should be conducted in this field: an ensemble of metrics could be a suitable answer, or even going a step further, by creating a new metric or exploring trainable metrics, like COMET [Rei et al., 2020].

Bibliography

Agnar Aamodt and Enric Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Commun.*, 7(1):39–59, March 1994.

Eugene Agichtein and Luis Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, DL ’00, pages 85–94, New York, NY, USA, 2000. ACM.

Eneko Agirre, Aitor Gonzalez-Agirre, Iñigo Lopez-Gazpio, Montse Maritxalar, German Rigau, and Larraitz Uria. Semeval-2016 task 2: Interpretable semantic textual similarity. In Steven Bethard, Daniel M. Cer, Marine Carpuat, David Jurgens, Preslav Nakov, and Torsten Zesch, editors, *SemEval@NAACL-HLT*, pages 512–524. The Association for Computer Linguistics, 2016.

Husam Ali, Yllias Chali, and Sadid A. Hasan. Automation of question generation from sentences. In *Proceedings of QG2010: The Third Workshop on Question Generation*, June 2010.

Jun Araki, Dheeraj Rajagopal, Sreecharan Sankaranarayanan, Susan Holm, Yukari Yamakawa, and Teruko Mitamura. Generating questions and multiple-choice answers using semantic analysis of texts. In *COLING*, 2016.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014.

Dzmitry Bahdanau, Tom Bosc, Stanislaw Jastrzebski, Edward Grefenstette, Pascal Vincent, and Yoshua Bengio. Learning to compute word embeddings on the fly. *CoRR*, abs/1706.00286, 2017.

C. F. Baker and J. Ruppenhofer. Framenet’s frames vs. Levin’s verb classes. In *Proceedings of the 28th Annual Meeting of the Berkeley Linguistics Society*, 2002.

Collin F. Baker, Charles J. Fillmore, and Beau Cronin. The structure of the FrameNet database. *International Journal of Lexicography*, 16(3):281–296, 2003.

Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

Benjamin S. Bloom, M. D. Engelhart, E. J. Furst, W. H. Hill, and David R. Krathwohl. The Classification of Educational Goals. In *Taxonomy of educational objectives*, page 207. Longmans, Green, 1956.

Chris Brockett. Aligning the rte 2006 corpus. Technical report, June 2007.

Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, Learning, and Games.* 01 2006. ISBN 978-0-521-84108-5.

Eugene Charniak. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*, NAACL 2000, pages 132–139, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

W Chen. Aist, g., mostow, j.: Generating questions automatically from informational text. In *Proceedings of the 2nd Workshop on Question Generation (AIED 2009)*, pages 17–24, 2009.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

J. Cohen. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1):37, 1960.

Trevor Cohn, Chris Callison-Burch, and Mirella Lapata. Constructing corpora for the development and evaluation of paraphrase systems. *Computational Linguistics*, 34(4):597–614, 2008.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12: 2493–2537, November 2011.

Sérgio Curto, Ana Cristina Mendes, and Luísa Coheur. Exploring linguistically-rich patterns for question generation. In *Proceedings of the UCNLG+Eval: Language Generation and Evaluation Workshop*, UCNLG+EVAL '11, pages 33–38, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

Sérgio Curto, Ana Cristina Mendes, and Luísa Coheur. Question generation based on lexico-syntactic patterns learned from the web. *Dialogue & Discourse*, 3(2):147–175, March 2012.

Dipanjan Das, Nathan Schneider, Desai Chen, and Noah A. Smith. Probabilistic frame-semantic parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 948–956, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. Generating typed dependency parses from phrase structure parses. In *Proceedings of the International Conference on Language, Resources and Evaluation (LREC)*, pages 449–454, 2006.

Michael Denkowski and Alon Lavie. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*, 2014.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

Bill Dolan, Chris Quirk, and Chris Brockett. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

Xinya Du, Junru Shao, and Claire Cardie. Learning to ask: Neural question generation for reading comprehension. In *Association for Computational Linguistics (ACL)*, 2017.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. Question generation for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 866–874, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

David Ferrucci and Adam Lally. Accelerating corporate research in the development, application and deployment of human language technologies. In *Proceedings of the HLT-NAACL 2003 Workshop on Software Engineering and Architecture of Language Technology Systems - Volume 8*, SEALTS '03, page 67–74, USA, 2003. Association for Computational Linguistics.

David A. Ferrucci, Eric W. Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John M. Prager, Nico Schlaefer, and Christopher A. Welty. Building watson: An overview of the deepqa project. *AI Magazine*, 31(3):59–79, 2010.

Pedro Fialho, Hugo Rodrigues, Luísa Coheur, and Paulo Quaresma. L2F/INESC-ID at SemEval-2017 tasks 1 and 2: Lexical and semantic features in word and textual similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 213–219, Vancouver, Canada, August 2017. Association for Computational Linguistics.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, page 363–370, USA, 2005. Association for Computational Linguistics. doi: 10.3115/1219840.1219885.

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*, pages 254–263, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

Gabriel Forgues, Joelle Pineau, Jean-Marie Larchevêque, and Réal Tremblay. Bootstrapping dialog systems with word embeddings. In *Nips, modern machine learning and natural language processing workshop*, volume 2, 2014.

Corina Forăscu and Iuliana Drăghici. Question generation: Taxonomies and data. In *Proceedings of the Second Workshop on Question Generation*, 2009.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. Incorporating copying mechanism in sequence-to-sequence learning. *CoRR*, abs/1603.06393, 2016.

Claudia Hauff Guanliang Chen, Jie Yang and Geert-Jan Houben. Learningq: A large-scale dataset for educational question generation, 2018.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 140–149, Berlin, Germany, August 2016. Association for Computational Linguistics.

Michael Heilman. *Automatic Factual Question Generation from Text.* PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 2011.

Michael Heilman and Noah A. Smith. Question generation via overgenerating transformations and ranking. Technical report, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 2009.

Michael Heilman and Noah A. Smith. Good question! statistical ranking for question generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 609–617, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

Sathish Indurthi, Dinesh Raghu, Mitesh M. Khapra, and Sachindra Joshi. Generating natural language question-answer pairs from a knowledge graph using a RNN based question generation model. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 376–385, Valencia, Spain, April 2017. Association for Computational Linguistics.

John Judge, Aoife Cahill, and Josef van Genabith. Questionbank: Creating a corpus of parse-annotated questions. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 497–504, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

S. Jung, C. Lee, and G.G. Lee. Dialog Studio: An Example Based Spoken Dialog System Development Workbench. In *Proceedings of the Dialogs on dialog: Multidisciplinary Evaluation of Advanced Speech-based Interactive Systems. Interspeech2006-ICSLP satellite workshop, Sept 2006, Pittsburgh*, 2006.

Saidalavi Kalady, Ajeesh Illikottil, and Rajarshi das. Natural language question generation using syntax and keywords. In *Proceedings of QG2010: The Third Workshop on Question Generation*, June 2010.

Onur Keklik, Tugkan Tuglular, and Selma Tekir. Rule-based automatic question generation using semantic role labeling. *IEICE TRANSACTIONS on Information and Systems*, 102 (7):1362–1373, 2019.

Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

Karin Kipper, Hoa Trang Dang, and Martha Palmer. Class-based construction of a verb lexicon. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 691–696. AAAI Press, 2000.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Skip-thought vectors. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, page 3294–3302, Cambridge, MA, USA, 2015. MIT Press.

Dan Klein and Christopher D. Manning. Fast exact inference with a factored model for natural language parsing. In *In Advances in Neural Information Processing Systems 15 (NIPS*, pages 3–10. MIT Press, 2003.

H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

Vishwajeet Kumar, Ganesh Ramakrishnan, and Yuan-Fang Li. A framework for automatic question generation from text using deep reinforcement learning, 08 2018.

A. Lally, J. M. Prager, M. C. McCord, B. K. Boguraev, S. Patwardhan, J. Fan, P. Fodor, and J. Chu-Carroll. Question analysis: How watson reads a clue. *IBM Journal of Research and Development*, 56(3.4):2:1–2:14, 2012.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations, 2019.

J. Richard Landis and Gary G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):pp. 159–174, 1977.

Philippe Langlais and Alexandre Patry. Translating unknown words by analogical learning. In Jason Eisner, editor, *EMNLP-CoNLL*, pages 877–886. ACL, 2007.

Cheongjae Lee, Sangkeun Jung, Seokhwan Kim, and Gary Geunbae Lee. Example-based dialog modeling for practical multi-domain dialog system. *Speech Communication*, 51(5): 466–484, 2009.

V. I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707–710, February 1966.

B. Levin. *English Verb Classes and Alternations: A Preliminary Investigation.* University of Chicago Press, 1993.

Roger Levy and Galen Andrew. Tregex and tsurgeon: tools for querying and manipulating tree data structures. In *In 5th International Conference on Language Resources and Evaluation*, 2006.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

David Lindberg, Fred Popowich, John Nesbit, and Phil Winne. Generating natural language questions to support learning on-line. In *Proceedings of the 14th European Workshop on Natural Language Generation*, pages 105–114, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

N. Littlestone and M.K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212 – 261, 1994. ISSN 0890-5401.

Bang Liu, Mingjun Zhao, Di Niu, Kunfeng Lai, Yancheng He, Haojie Wei, and Yu Xu. Learning to generate questions by learning what not to generate. In *The World Wide Web Conference*, WWW '19, page 1106–1118, New York, NY, USA, 2019. Association for Computing Machinery.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016.

Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

Prashanth Mannem, Rashmi Prasad, and Aravind Joshi. Question generation from paragraphs at upenn: Qgstec system description. In *Proceedings of QG2010: The Third Workshop on Question Generation*, pages 84–91, 2010.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330, 1993.

Karen Mazidi and Rodney D. Nielsen. Linguistic considerations in automatic question generation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 321–326, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

Karen Mazidi and Rodney D. Nielsen. Leveraging multiple views of text for automatic question generation. In *Artificial Intelligence in Education - 17th International Conference, AIED 2015, Madrid, Spain, June 22-26, 2015. Proceedings*, pages 257–266, 2015.

Ana Cristina Mendes. *When the Answer comes into Question in Question-Answering*. PhD thesis, Instituto Superior Técnico, Universidade de Lisboa, Lisbon, Portugal, 2013.

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. The nombank project: An interim report. In *In Proceedings of the NAACL/HLT Workshop on Frontiers in Corpus Annotation*, 2004.

Laurent Miclet, Sabri Bayoudh, and Arnaud Delhay. Analogical dissimilarity: Definition, algorithms and two experiments in machine learning. *J. Artif. Intell. Res. (JAIR)*, 32: 793–824, 2008.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

George A. Miller. Wordnet: a lexical database for english. *Commun. ACM*, 38:39–41, November 1995.

Alessandro Moschitti. Making tree kernels practical for natural language learning. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy, April 2006. Association for Computational Linguistics.

Hiroya Murao, Nobuo Kawaguchi, Shigeki Matsubara, Yukiko Yamaguchi, and Yasuyoshi Inagaki. Example-based spoken dialogue system using woz system log. In *In: Proc. 4th SIGDIAL Workshop on Discourse and Dialogue*, pages 140–148, 2003.

Shashi Narayan, Siva Reddy, and Shay B Cohen. Paraphrase generation from latent-variable pcfgs for semantic parsing. In *The 9th International Natural Language Generation conference*, page 153, 2016.

Roberto Navigli and Paola Velardi. Learning word-class lattices for definition and hypernym extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1318–1327, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. Ms marco: A human generated machine reading comprehension dataset. November 2016.

Lasguido Nio, Sakriani Sakti, Graham Neubig, Tomoki Toda, Mirna Adriani, and Satoshi Nakamura. Developing non-goal dialog system based on examples of drama television. In *Natural Interaction with Robots, Knowbots and Smartphones*, pages 355–361. Springer, 2014.

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. Why we need new evaluation metrics for nlg. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017.

Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.

Santanu Pal, Tapabrata Mondal, Partha Pakray, Dipankar Das, and Sivaji Bandyopadhyay. Qgstec system description–juqgg: A rule based approach. *Boyer & Piwek (2010)*, pages 76–79, 2010.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. The proposition bank: An annotated corpus of semantic roles. *Comput. Linguist.*, 31(1):71–106, March 2005.

Bo Pang, Kevin Knight, and Daniel Marcu. Syntax-based alignment of multiple translations: Extracting paraphrases and generating new sentences. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 102–109, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

Patrick Pantel and Marco Pennacchiotti. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 113–120, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135.

Slav Petrov and Dan Klein. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics, Proceedings of the Main Conference*, pages 404–411, 2007.

Robert G. Pontius and Marco Millones. Death to Kappa: birth of quantity disagreement and allocation disagreement for accuracy assessment. *International Journal of Remote Sensing*, 32(15):4407–4429, August 2011.

Sameer S. Pradhan, Wayne Ward, Kadri Hacioglu, James H. Martin, and Daniel Jurafsky. Shallow semantic parsing using support vector machines. In *HLT-NAACL*, pages 233–240, 2004.

John Prager, Eric Brown, Dragomir R. Radev, and Krzysztof Czuba. One search engine or two for question-answering. In *In Nineth Text REtrieval Conference, Gaithersburg,USA*, 2000.

V. Punyakanok, D. Roth, and W. Yih. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2), 2008.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad, 2018.

D. Ravichandran and Eduard Hovy. Learning surface text patterns for a question answering system. In *Proceedings of the 40th ACL conference. Philadelphia, PA.*, 2002.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. Comet: A neural framework for mt evaluation. *arXiv preprint arXiv:2009.09025*, 2020.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'95, pages 448–453, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

Ellen Riloff. Automatically generating extraction patterns from untagged text. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, AAAI'96, page 1044–1049. AAAI Press, 1996.

Hugo Rodrigues, Luísa Coheur, and Eric Nyberg. QGASP: a framework for question generation based on different levels of linguistic information. In *Proceedings of the 9th International Natural Language Generation conference*, pages 242–243, Edinburgh, UK, September 5-8 2016. Association for Computational Linguistics. (demo).

Hugo Rodrigues, Luísa Coheur, and Eric Nyberg. Improving question generation with the teacher's implicit feedback. In *International Conference on Artificial Intelligence in Education*, pages 301–306. Springer, 2018.

Hugo Rodrigues, Luísa Coheur, and Eric Nyberg. Populating the knowledge base of a conversational agent: human vs. machine. *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, 2019.

Vasile Rus and James Lester. The 2nd workshop on question generation. In Vania Dimitrova, Riichiro Mizoguchi, Benedict du Boulay, and Arthur C. Graesser, editors, *AIED*, volume 200 of *Frontiers in Artificial Intelligence and Applications*, page 808. IOS Press, 2009.

Vasile Rus and Mihai Lintean. A comparison of greedy and optimal assessment of natural language student input using word-to-word similarity metrics. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 157–162, Montréal, Canada, June 2012. Association for Computational Linguistics.

Vasile Rus, Brendan Wyse, Paul Piwek, Mihai Lintean, Svetlana Stoyanchev, and Cristian Moldovan. Overview of the first question generation shared task evaluation challenge. In *Proceedings of the Sixth International Natural Language Generation Conference (INLG 2010)*, July 2010.

Vasile Rus, Paul Piwek, Svetlana Stoyanchev, Brendan Wyse, Mihai Lintean, and Cristian Moldovan. Question generation shared task and evaluation challenge: status report. In *Proceedings of the 13th European Workshop on Natural Language Generation*, ENLG '11, pages 318–320, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

Vasile Rus, Brendan Wyse, Paul Piwek, Mihai Lintean, Svetlana Stoyanchev, and Cristian Moldovan. A detailed account of the first question generation shared task evaluation challenge. *Dialogue & Discourse*, 3(2):177–204, March 2012.

Iulian Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Y. Bengio. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. pages 588–598, 03 2016.

Shikhar Sharma, Layla El Asri, Hannes Schulz, and Jeremie Zumer. Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation. *CoRR*, abs/1706.09799, 2017. URL `http://arxiv.org/abs/1706.09799`.

Hideki Shima. *Paraphrase Pattern Acquisition by Diversifiable Bootstrapping.* PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 2015.

Noah A. Smith, Michael Heilman, and Rebecca Hwa. Question generation as a competitive undergraduate course project. In *Proceedings of the NSF Workshop on the Question Generation Shared Task and Evaluation Challenge*, September 2008.

C. Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 100(3/4):441–471, 1987. ISSN 00029556. URL `http://www.jstor.org/stable/1422689`.

Sandeep Subramanian, Tong Wang, Xingdi Yuan, Saizheng Zhang, Adam Trischler, and Yoshua Bengio. Neural models for key phrase extraction and question generation. In *Proceedings of the Workshop on Machine Reading for Question Answering*, pages 78–88, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-2609.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc., 2014.

Duyu Tang, Nan Duan, Tao Qin, and Ming Zhou. Question answering and question generation as dual tasks. *CoRR*, abs/1706.02027, 2017.

Adam Trischler, Tong Wang, Xingdi (Eric) Yuan, Justin D. Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. Newsqa: A machine comprehension dataset, November 2016.

Vincent Van Asch. Macro-and micro-averaged evaluation measures [[basic draft]], 2013.

Andrea Varga and Le An Ha. Wlv: A question generation system for the qgstec 2010 task b. In *Proceedings of QG2010: The Third Workshop on Question Generation*, June 2010.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

Paola Velardi, Stefano Faralli, and Roberto Navigli. Ontolearn reloaded: A graph-based algorithm for taxonomy induction. *Computational Linguistics*, 39(3):665–707, 2013.

Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer. *CoRR*, abs/1608.07905, 2016.

Tong Wang, Xingdi Yuan, and Adam Trischler. A joint model for question answering and question generation. *CoRR*, abs/1706.01450, 2017.

Yansen Wang, Chenyi Liu, Minlie Huang, and Liqiang Nie. Learning to ask questions in open-domain conversational systems with typed decoders. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2193–2203, Melbourne, Australia, July 2018. Association for Computational Linguistics.

Brendan Wyse and Paul Piwek. Generating questions from openlearn study units. In *AIED 2009 Workshop Proceedings Volume 1: The 2nd Workshop on Question Generation*, 2009.

Xuchen Yao and Yi Zhang. Question generation with minimal recursion semantics. In *Proceedings of QG2010: The Third Workshop on Question Generation*, June 2010.

Xingdi Yuan, Tong Wang, Caglar Gulcehre, Alessandro Sordoni, Philip Bachman, Saizheng Zhang, Sandeep Subramanian, and Adam Trischler. Machine comprehension by text-to-text neural question generation. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 15–25, Vancouver, Canada, August 2017. Association for Computational Linguistics.

Zhuosheng Zhang, Yuwei Wu, Hai Zhao, Zuchao Li, Shuailiang Zhang, Xi Zhou, and Xiang Zhou. Semantics-aware bert for language understanding, 2019.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. Neural question generation from text: A preliminary study. In Xuanjing Huang, Jing Jiang, Dongyan Zhao, Yansong Feng, and Yu Hong, editors, *Natural Language Processing and Chinese Computing*, pages 662–671, Cham, 2018. Springer International Publishing.

A Alignment Evaluation

The evaluation of the Pattern Acquisition step is closely related with the alignment task we perform, as described in the Section 3.2. The similarity lies on the process we take to create the patterns: the mapping of the seeds' components, between the answer sentence and the Q/A pair, is automatically extracted by aligning them. We work with an answer sentence and a Q/A pair, but the task of aligning tokens is independent of the type of sentences; thus, the alignment task can be used as a proxy to evaluate our mapping process. The following sections describe the evaluation of this step as follows: we first discuss the available datasets and which one was used in our study, then we describe the metrics used in our evaluation, and finally we present the results.

A.1 Datasets

There are many datasets closely related with the alignment and/or paraphrase detection task, but just a few are monolingual (namely in English, which is the language we are working on).

Microsoft Research Paraphrase Corpus (MSRP)[1] is specific for paraphrases [Dolan et al., 2004], and contains a training set of 4076 sentence pairs (67.5% of which are positive examples) and a test set of 1725 sentence pairs (66.5% positive). However, the corpus does not contain any indication of how the sentences align and, thus, is not suitable for our experiments.

SemEval 2016 [Agirre et al., 2016] provided a dataset for the Interpretable Semantic Textual Similarity (iSTS) task, containing two corpora[2]. One is a set of 756 sentence pairs of news headlines, and the other is a set of 750 sentence pairs of image captions. However, it does not include paraphrases only, but also sentence pairs that capture contradictions and other linguistic phenomena, which we are not interested in, and since they are not annotated as such, it becomes impracticable to use.

Another available corpus is the manually aligned RTE 2006 corpus, provided by Microsoft Research [Brockett, 2007][3]. It contains 2400 sentence pairs, both positive and negative, where positive examples imply an entailment relationship between the pair. However, similarly to the previous dataset, the negative examples were not annotated as such, making this dataset hard to use as well.

Finally, the Edinburgh dataset [Cohn et al., 2008][4] is uniquely a set of paraphrases pairs, manually aligned. It contains a training set of 714 sentence pairs and a test set of 306 pairs, with all pairs being positive examples. This fulfills our needs, being our choice for our study.

However, we detected some lack of consistency across the dataset, namely on how mul-

[1]https://www.microsoft.com/en-us/download/details.aspx?id=52398

[2]http://alt.qcri.org/semeval2016/task2/

[3]https://www.microsoft.com/en-us/research/publication/aligning-the-rte-2006-corpus/

[4]http://www.ling.ohio-state.edu/~mwhite/data/coling12/edinburgh-json-20130322.tgz

tiword Named Entities (NEs) were aligned. For instance, there are examples of *East Timor* being aligned as a unique token and as two separate alignments. In addition, the corpus captures long-distance alignments of token sequences that we are not aiming at capturing in our specific task. For example, in the paraphrase pair *Fujian 's gross national product.../the gross national product of...*, there is an alignment between *'s* and *the .. of*. While technically correct (*'s* is indeed referring to a property *of* something), this is the kind of alignment we are not interested in and are not performing in GEN. To cope with these limitations we modified and extended (automatically at first, and manually afterwards) the corpus to contain multiple possible alignments for NEs, and also removed uninformative alignments such as the one showed. We performed these alterations on the training set only, which we use in our experiments. An example for the second entry of the corpus is in Figure A.1; note how it is acceptable to align *Doyle* in the first sentence with either the full name or just the strict identical name. The modified corpus is available online[5].

In Figure A.2 is shown the number of instances that require a certain number of alignments (Figures A.2a and A.2b – top), and the number of alignments each instance of the corpus requires, ordered increasingly (Figures A.2c and A.2d – bottom). The graphs on the left report numbers after removing stopwords. As one can see in the top graphs, the corpus resembles a normal distribution, excepting a few outliers. This phenomena is more evident when keeping stopwords, which can vary significantly in number from one instance to another. The graphs also have a longer tail to the right side, showing this variability as well. When removing stopwords, the instance with highest number of alignments drops from 46 alignments to 24, whereas the number of instances with the most common number of alignments almost duplicates. This clearly shows that, without considering stopwords, the content of the sentences is much more similar regarding the number of alignments required. This is also supported on the bottom half of the figure, where the ladder effect changes its shape: it is less taller and each step is longer. Finally, it is important to note that, after removing stopwords, a few instances lack alignments, as they are short and lack content words; for instance *"well , why should there be any more ?"* is a sentence including only tokens present in the stopword list, resulting in a empty sentence, content-wise.

A.2 Evaluation Metrics

To evaluate the alignments we use precision, recall and F1 measure, defined as follows:

$$precision = \frac{tp}{tp + fp},$$

$$recall = \frac{tp}{tp + fn},$$

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall},$$

where tp, fp and fn are true positives, false positives and false negatives, respectively.

Given a generic evaluation measure $B(tp, fp, fn)$ calculated in function of true positives, false positives and false negatives, there are two ways of calculating B across multiple datasets

[5] `http://hlt.inesc-id.pt/~hpr/edinburghModified/gold.train.alignments.fixed.xml`

```
<instance id="news-common:2143">
  (...)
    <align>
      <hypothesis>
        <a>
          <token1>to</token1>
          <token2>to</token2>
        </a>
      </hypothesis>
    </align>
    <align>
      <hypothesis>
        <a>
          <token1>Colby</token1>
          <token2>college</token2>
        </a>
      </hypothesis>
    </align>
    <align>
      <hypothesis>
        <a>
          <token1>Doyle</token1>
          <token2>Doyle</token2>
        </a>
      </hypothesis>
      <hypothesis>
        <a>
          <token1>Doyle</token1>
          <token2>Timothy Doyle</token2>
        </a>
      </hypothesis>
    </align>
    <align>
      <hypothesis>
        <a>
          <token1>said</token1>
          <token2>said</token2>
        </a>
      </hypothesis>
    </align>
    <align>
      <hypothesis>
        <a>
          <token1>.</token1>
          <token2>.</token2>
        </a>
      </hypothesis>
    </align>
  </instance>
```

Figure A.1: Part of the xml specification for the instance "news-common:2143" from the Edinburgh corpus training set – *Hackett and Rossignol did not know each other and Hackett had no connection to Colby , Doyle said ./State police Lt. Timothy Doyle said Hackett and Rossignol did not know each other , and that Hackett had no connection to the college .*

$d \in D$: B_{micro} and B_{macro} [Van Asch, 2013]:

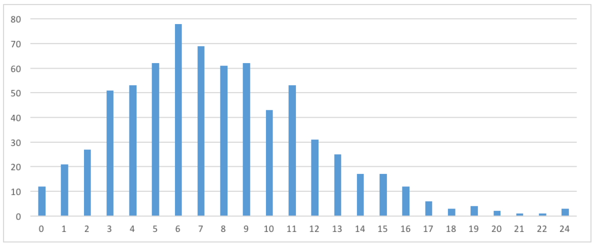
$$B_{micro} = B(\sum_d tp_d, \sum_d fp_d, \sum_d fn_d)$$

$$B_{macro} = \frac{1}{|D|} \sum_d B(tp_d, fp_d, fn_d),$$

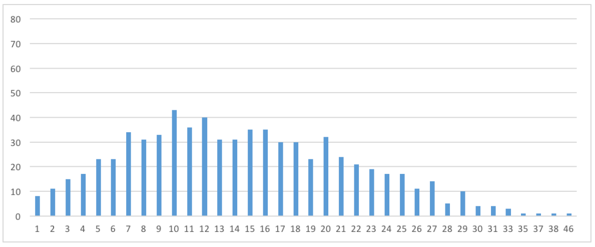
where tp_d is the number of true positives for dataset d (likewise for fp_d and fn_d).

B_{macro} averages each individual B_d, meaning each dataset has an equal weight on the final score, whereas B_{micro} averages the overall counts, which makes larger datasets d have more weight, dominating smaller datasets.

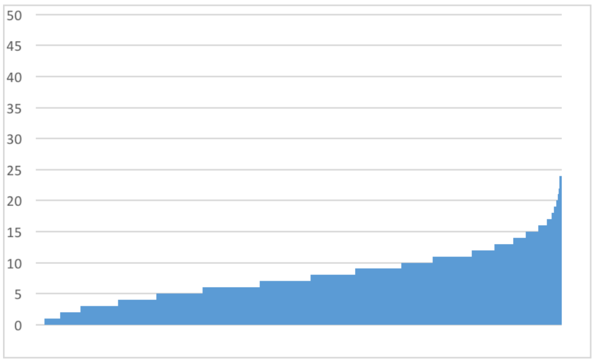
In our concrete case, we have a single dataset, but each instance in the dataset contains multiple alignments, which means B can be calculated for each instance individually. In other words, each instance on our dataset can be treated as a dataset d for purposes of the equation above.



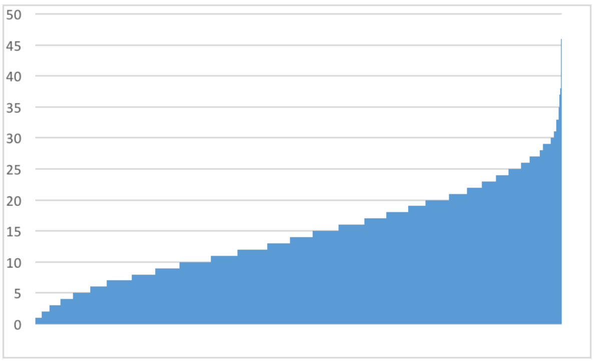
(a) Number of instances in the corpus with n alignments required, no stopwords considered.



(b) Number of instances in the corpus with n alignments required, stopwords considered.



(c) Number of alignments per instance, ordered increasingly, no stopwords considered.



(d) Number of alignments per instance, ordered increasingly, no stopwords considered.

Figure A.2: Graphical analysis of the Edinburgh corpus training set. Above, the figures show the number of instances in the corpus with n number of alignments required (A.2a and A.2b). Below it is depicted the number of alignments per each instance in the corpus requires, ordered increasingly (A.2c and A.2d). The left side of the figure shows the analysis with no stopwords (A.2a and A.2c), while the right side keeps them.

Each instance has a number of required alignments that is function of the sentences' length. However, the difficulty of aligning two sentences, although growing with their size, is not dictated by that factor alone. Thus, it would not make sense to give more weight to Bs calculated for those larger instances. For this reason, we chose to macro-average the results in this experiment [Van Asch, 2013]:

$$precision_{macro} = \frac{1}{|D|}\sum_{d} precision(d),$$

$$recall_{macro} = \frac{1}{|D|}\sum_{d} recall(d),$$

where $|D|$ is the size of our dataset and d is an instance on D. tp is the number of alignments found which are present in the goldstandard, fp is the number of alignments found which do not belong to the goldstandard, and fn is the number of alignments not found which are present in the goldstandard. Precision and recall can be thus rewritten as:

$$precision = \frac{|gold \cap alignments|}{|alignments|},$$

$$recall = \frac{|gold \cap alignments|}{|gold|},$$

	Stopwords Removed			**With Stopwords**		
	Precision	Recall	F1	Precision	Recall	F1
Lexical	$0.90_{\pm 0.17}$	$0.58_{\pm 0.30}$	0.70	$0.76_{\pm 0.21}$	$0.65_{\pm 0.26}$	0.70
WN	$0.58_{\pm 0.28}$	$0.57_{\pm 0.29}$	0.58	$0.50_{\pm 0.25}$	$0.50_{\pm 0.22}$	0.50
W2V	$0.17_{\pm 0.25}$	$0.72_{\pm 0.30}$	0.27	$0.06_{\pm 0.10}$	$0.79_{\pm 0.26}$	0.12
All	$0.17_{\pm 0.25}$	$0.72_{\pm 0.30}$	0.27	$0.06_{\pm 0.10}$	$0.79_{\pm 0.26}$	0.12

Table A.1: Average macro-precision, recall and F-measure for all runs, considering all matches found.

where *gold* is the set of alignments from the goldstandard and *alignments* is the set of alignments found by the system being evaluated.

A.3 Experimental Setup

We use the modified Edinburgh training set in our experiments. Our alignment system, built as specified in Section 3.2, is used and compared against GIZA++ [Och and Ney, 2003][6], which implements the IBM-4 alignment model, and Meteor aligner module [Denkowski and Lavie, 2014][7]. We calculate macro precision, recall and F1 as previously mentioned for all systems.

A.4 Experimental Results

We created 6 different configurations: one for each of the *equiv* functions in Section 3.2.3 (Equations 3.3 to 3.7), and a last one using them all combined. We also divided these in two runs: one where we remove stopwords, and another were no stopwords were removed at all. As Figure A.1 shows, the goldstandard includes stopwords, which were not accounted when calculating the system's performance for the first runs, and punctuation, which were discarded overall.

The statistics of the alignments before the final selection performed by the Hungarian algorithm are not very informative as a whole, but they still provide some insight. For instance, if an alignment in the goldstandard has $|S_1|$ alignments, two of which are verbs, the function $equiv_{VB}$ will at most find 2 alignments, which results in a really low recall. On the other hand, $equiv_{W2V}$ is able to generate an alignment for almost every pair of words, which means the precision will be really low and recall really high. Therefore, the pre-Hungarian runs are only useful to assess the recall of $equiv_{W2V}$ and the precision of specific functions, as $equiv_L$ and $equiv_{WN}$. Results over all 714 pairs are presented in Table A.1 – instances with no alignments found have precision 1 and recall 0.

As expected, $equiv_L$ does not obtain perfect precision because some alignments are required between non-identical tokens. On the other hand, recall is not perfect as well for $equiv_{W2V}$ because sometimes multiple-token to one token alignments are required, and our system is not able to capture them (for instance, *said in an interview* should be aligned to

[6]http://www.statmt.org/moses/giza/GIZA++.html
[7]http://www.cs.cmu.edu/~alavie/METEOR/

	Stopwords Removed				**With Stopwords**			
	Precision	Recall	F1	# Pairs	Precision	Recall	F1	# Pairs
Lexical	$0.99_{\pm 0.08}$	$0.09_{\pm 0.27}$	0.17	90	$0.99_{\pm 0.07}$	$0.07_{\pm 0.24}$	0.12	61
WN	$0.97_{\pm 0.14}$	$0.10_{\pm 0.27}$	0.18	111	$1_{\pm 0.00}$	$0.02_{\pm 0.12}$	0.03	11
W2V	$0.85_{\pm 0.24}$	$0.41_{\pm 0.41}$	0.55	419	$0.87_{\pm 0.22}$	$0.23_{\pm 0.35}$	0.37	248
All	$0.82_{\pm 0.25}$	$0.44_{\pm 0.40}$	0.58	458	$0.82_{\pm 0.23}$	$0.43_{\pm 0.41}$	0.56	416

Table A.2: Average macro-precision, recall and F-measure post Hungarian Algorithm application, along with the number of pairs with a solution found.

	Stopwords Removed			**With Stopwords**		
	Precision	Recall	F1	Precision	Recall	F1
Lexical	$0.93_{\pm 0.21}$	$0.76_{\pm 0.31}$	0.84	$0.86_{\pm 0.19}$	$0.78_{\pm 0.33}$	0.82
WN	$0.83_{\pm 0.31}$	$0.70_{\pm 0.31}$	0.76	$1_{\pm 0.00}$	$1_{\pm 0.00}$	1
W2V	$0.74_{\pm 0.26}$	$0.72_{\pm 0.25}$	0.73	$0.64_{\pm 0.23}$	$0.68_{\pm 0.24}$	0.66
All	$0.72_{\pm 0.26}$	$0.69_{\pm 0.28}$	0.71	$0.69_{\pm 0.22}$	$0.75_{\pm 0.24}$	0.72

Table A.3: Average macro-precision, recall and F-measure for the instances with solutions found with Hungarian Algorithm

told – one could argue that *said* could be an alignment hypothesis for *told*, but we have not extended the corpus for this cases).

However, the real important results are the ones obtained after the final selection, that is, post-Hungarian algorithm application. These are shown in Table A.2. For *equiv* functions with small coverage, like $equiv_{VB}$, no final solution is found, as it would be expected. For other functions with more applicability, the set of alignments will be much smaller than before, impacting this way the precision score (which should go up), and recall (which should go down). For instance, for a given S_1 sentence, $equiv_{W2V}$ will have at most $|S_1|$ alignments, instead of $|S_1|^{|S_2|}$ as pre-application of the Hungarian algorithm; therefore, it might not cover all gold alignments, but precision will improve dramatically.

As one can see, $equiv_L$, which had really good scores (namely recall and F1), dropped significantly because it is not able to find a solution by itself for the alignments, obtaining only 90 and 61 solutions for both runs (although with perfect precision for 73 and 32 of them, respectively). Results are also overall worse for runs that do not remove stopwords, because they require tokens to be matched that are not necessarily present on the paired sentence more frequently.

Table A.3 shows the results considering only the instances to which a solution was found. F1 values go up significantly, meaning the obtained alignments are of good quality. Regarding the *equiv* function as a whole, we can see that adding semantic functions provides a good tool to be able to align sentences that would not be aligned just through lexical alignment strategies. When looking only to the pairs to which a solution was found, one can see the differences are substantial, and removing stopwords increases the number of solutions found at a low cost of precision and recall. Given the conclusions from the previous paragraph, we imagine, however, that more errors are being introduced in the run with stopwors, but

	Precision	Recall	F1
GIZA++	$0.55_{\pm 0.19}$	$0.65_{\pm 0.21}$	0.60
Meteor	$0.69_{\pm 0.17}$	$0.78_{\pm 0.17}$	0.73
GEN pre-Hungarian -	$0.82_{\pm 0.25}$	$0.44_{\pm 0.40}$	0.58
GEN post-Hungarian -	$0.72_{\pm 0.26}$	$0.69_{\pm 0.28}$	0.71
GEN no-restriction -	$0.68_{\pm 0.21}$	$0.72_{\pm 0.26}$	0.70
GEN pre-Hungarian +	$0.82_{\pm 0.23}$	$0.43_{\pm 0.41}$	0.56
GEN post-Hungarian +	$0.69_{\pm 0.22}$	$0.75_{\pm 0.24}$	0.72
GEN no-restriction +	$0.73_{\pm 0.26}$	$0.67_{\pm 0.30}$	0.70

Table A.4: Average macro-precision, recall and F-measure on the modified Edinburgh corpus training set, for GIZA++ and Meteor (top), and GEN pre- and post-Hungarian application, and no-restriction used, when using stopwords (+) or removing them (-).

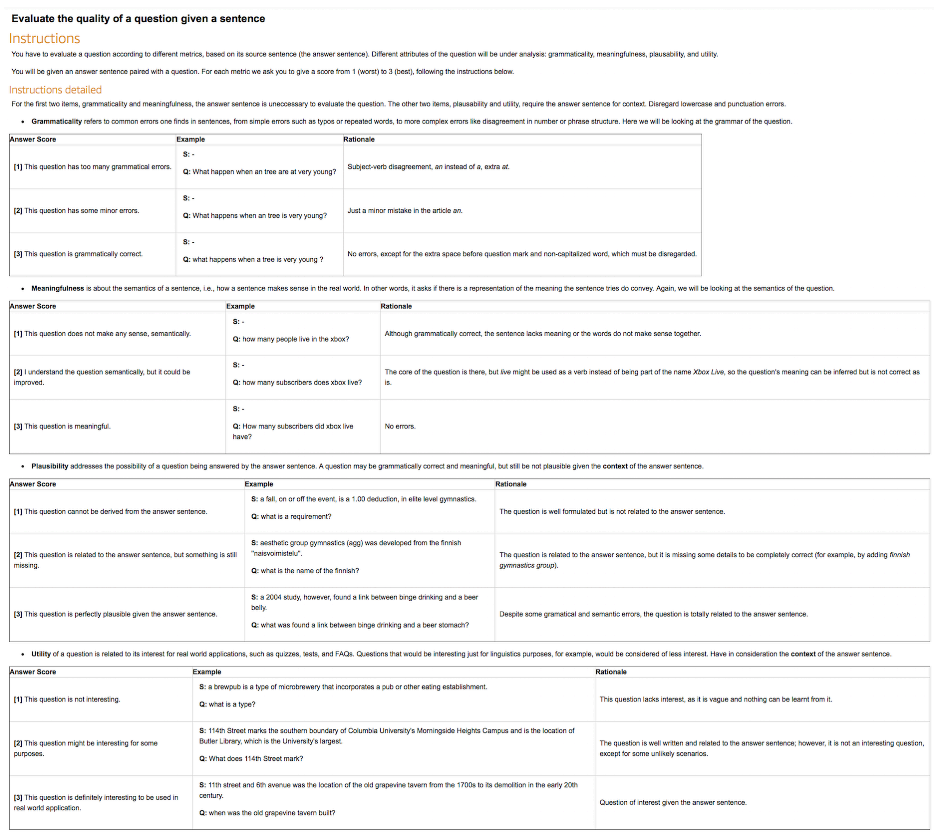
these are masked by the large number of correct but unimportant alignments of stopwords. Regarding the drop for $equiv_{W2V}$, we believe the reason is that this function is able to pair any two tokens, even if with a low score, meaning a match for missing tokens is still produced, despite being incorrect, which lowers its performance.

Finally, Table A.4 reports on the top rows the results obtained with GIZA++ and Meteor aligner. As one can see, the results are significantly better than ours if we consider the whole picture (pre-Hungarian), but are on par with the results for pairs we find a solution to (post-Hungarian). However, that is a unfair comparison – whichever we use. Therefore, the table shows too the results obtained by our system if no restriction was imposed to the Hungarian Algorithm, that is, if we do not require all terms on the shortest sentence to have a match (Requirment R_2 from Section 3.2.3). As one can see, the results also improve significantly, being on par with the ones reported for post-Hungarian application.

Upon further analysis, we found that many pairs, extracted from novels, contained a couple of dialogues. This introduces noise for the segmenter, which ends up treating instances on some pairs separately. What this means, in the end, is that our system ends up with less content for some pairs, trying its best to align the corrupted sentences, damaging this way both precision and recall. However, this is something we will find with real text, which means these errors will always exist. Because it is not our goal to compete in alignment tasks but, rather, to prove that our alignment module is suitable for our task, we decided to not tamper with the evaluation done. We believe, though, that our results could be even better than presented, if those instances were dealt with.

B Amazon Mechanical Turk HIT

In Section 5.3 of Chapter 5 we described the Amazon Mechanical Turk (AMT) evaluation procedure, along with the guidelines presented to the evaluators. Here we show a complete HIT, split in two figures for readability. Figure B.1 contains the detailed instructions included on each HIT, with examples for each score and metric. Figure B.2 shows the remaining of the HIT, containing complete examples of the task, followed by the question to be evaluated.



Evaluate the quality of a question given a sentence

Instructions

You have to evaluate a question according to different metrics, based on its source sentence (the answer sentence). Different attributes of the question will be under analysis: grammaticality, meaningfulness, plausability, and utility.

You will be given an answer sentence paired with a question. For each metric we ask you to give a score from 1 (worst) to 3 (best), following the instructions below.

Instructions detailed

For the first two items, grammaticality and meaningfulness, the answer sentence is uneccessary to evaluate the question. The other two items, plausability and utility, require the answer sentence for context. Disregard lowercase and punctuation errors.

- **Grammaticality** refers to common errors one finds in sentences, from simple errors such as typos or repeated words, to more complex errors like disagreement in number or phrase structure. Here we will be looking at the grammar of the question.

Answer Score	Example	Rationale
[1] This question has too many grammatical errors.	**S:** - **Q:** What happen when an tree are at very young?	Subject-verb disagreement, *an* instead of *a*, extra *at*.
[2] This question has some minor errors.	**S:** - **Q:** What happens when an tree is very young?	Just a minor mistake in the article *an*.
[3] This question is grammatically correct.	**S:** - **Q:** what happens when a tree is very young ?	No errors, except for the extra space before question mark and non-capitalized word, which must be disregarded.

- **Meaningfulness** is about the semantics of a sentence, i.e., how a sentence makes sense in the real world. In other words, it asks if there is a representation of the meaning the sentence tries do convey. Again, we will be looking at the semantics of the question.

Answer Score	Example	Rationale
[1] This question does not make any sense, semantically.	**S:** - **Q:** how many people live in the xbox?	Although grammatically correct, the sentence lacks meaning or the words do not make sense together.
[2] I understand the question semantically, but it could be improved.	**S:** - **Q:** how many subscribers does xbox live?	The core of the question is there, but *live* might be used as a verb instead of being part of the name *Xbox Live*, so the question's meaning can be inferred but is not correct as is.
[3] This question is meaningful.	**S:** - **Q:** How many subscribers did xbox live have?	No errors.

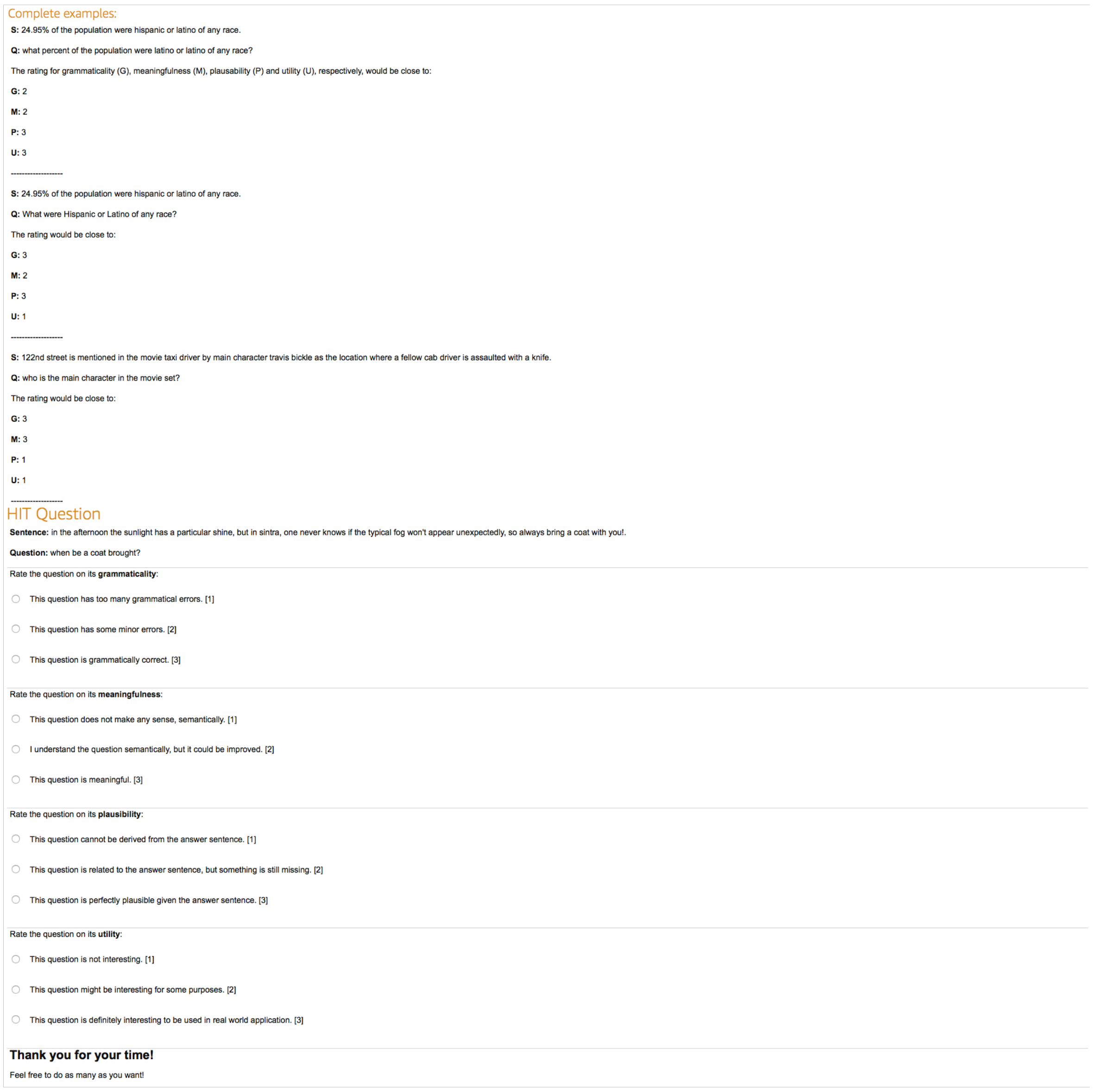
- **Plausibility** addresses the possibility of a question being answered by the answer sentence. A question may be grammatically correct and meaningful, but still be not plausible given the **context** of the answer sentence.

Answer Score	Example	Rationale
[1] This question cannot be derived from the answer sentence.	**S:** a fall, on or off the event, is a 1.00 deduction, in elite level gymnastics. **Q:** what is a requirement?	The question is well formulated but is not related to the answer sentence.
[2] This question is related to the answer sentence, but something is still missing.	**S:** aesthetic group gymnastics (agg) was developed from the finnish "naisvoimistelu". **Q:** what is the name of the finnish?	The question is related to the answer sentence, but it is missing some details to be completely correct (for example, by adding *finnish gymnastics group*).
[3] This question is perfectly plausible given the answer sentence.	**S:** a 2004 study, however, found a link between binge drinking and a beer belly. **Q:** what was found a link between binge drinking and a beer stomach?	Despite some gramatical and semantic errors, the question is totally related to the answer sentence.

- **Utility** of a question is related to its interest for real world applications, such as quizzes, tests, and FAQs. Questions that would be interesting just for linguistics purposes, for example, would be considered of less interest. Have in consideration the **context** of the answer sentence.

Answer Score	Example	Rationale
[1] This question is not interesting.	**S:** a brewpub is a type of microbrewery that incorporates a pub or other eating establishment. **Q:** what is a type?	This question lacks interest, as it is vague and nothing can be learnt from it.
[2] This question might be interesting for some purposes.	**S:** 114th Street marks the southern boundary of Columbia University's Morningside Heights Campus and is the location of Butler Library, which is the University's largest. **Q:** What does 114th Street mark?	The question is well written and related to the answer sentence; however, it is not an interesting question, except for some unlikely scenarios.
[3] This question is definitely interesting to be used in real world application.	**S:** 11th street and 6th avenue was the location of the old grapevine tavern from the 1700s to its demolition in the early 20th century. **Q:** when was the old grapevine tavern built?	Question of interest given the answer sentence.

Figure B.1: First part of a submitted HIT, containing the instructions for the evaluators, with examples for each score.



Complete examples:

S: 24.95% of the population were hispanic or latino of any race.

Q: what percent of the population were latino or latino of any race?

The rating for grammaticality (G), meaningfulness (M), plausability (P) and utility (U), respectively, would be close to:

G: 2

M: 2

P: 3

U: 3

S: 24.95% of the population were hispanic or latino of any race.

Q: What were Hispanic or Latino of any race?

The rating would be close to:

G: 3

M: 2

P: 3

U: 1

S: 122nd street is mentioned in the movie taxi driver by main character travis bickle as the location where a fellow cab driver is assaulted with a knife.

Q: who is the main character in the movie set?

The rating would be close to:

G: 3

M: 3

P: 1

U: 1

HIT Question

Sentence: in the afternoon the sunlight has a particular shine, but in sintra, one never knows if the typical fog won't appear unexpectedly, so always bring a coat with you!.

Question: when be a coat brought?

Rate the question on its **grammaticality**:

○ This question has too many grammatical errors. [1]

○ This question has some minor errors. [2]

○ This question is grammatically correct. [3]

Rate the question on its **meaningfulness**:

○ This question does not make any sense, semantically. [1]

○ I understand the question semantically, but it could be improved. [2]

○ This question is meaningful. [3]

Rate the question on its **plausibility**:

○ This question cannot be derived from the answer sentence. [1]

○ This question is related to the answer sentence, but something is still missing. [2]

○ This question is perfectly plausible given the answer sentence. [3]

Rate the question on its **utility**:

○ This question is not interesting. [1]

○ This question might be interesting for some purposes. [2]

○ This question is definitely interesting to be used in real world application. [3]

Thank you for your time!

Feel free to do as many as you want!

Figure B.2: Second part of a submitted HIT, containing complete examples of the task and the question to be evaluated.

C Correlation Between AMT Evaluation and Automatic Metrics

In Section 5.3.4 of Chapter 5 we studied the correlation between Amazon Mechanical Turk (AMT) evaluation and automatic metrics. We presented results for all questions evaluated, and omitted that analysis per system, as results are similar. The following tables present those results for H&S (Table C.1), D&A (Table C.2), and GEN (Table C.3), for questions evaluated from MONSERRATE, and Tables C.4 to C.6 show the results for questions from SQuAD, respectively for H&S, D&A, and GEN.

Table C.1: Correlation between all automatic metrics and AMT evaluations, for H&S questions evaluated on MONSERRATE, using AMT average results (top) and median results (bottom).

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Grammar	0.09	0.04	0.14	0.13	0.21	0.13	0.12	0.13
Semantic	0.15	0.15	0.13	0.20	0.14	0.17	0.10	0.13
Plausibility	0.23	0.16	0.28	0.24	0.25	0.27	0.16	0.23
Utility	0.13	0.10	0.15	0.13	0.22	0.20	0.17	0.15
Average	0.20	0.15	0.23	0.23	0.28	0.26	0.18	0.22
Grammar	0.17	0.08	0.23	0.23	0.28	0.20	0.09	0.22
Semantic	0.13	0.18	0.13	0.20	0.08	0.12	0.12	0.12
Plausibility	0.15	0.05	0.24	0.16	0.16	0.17	0.09	0.17
Utility	0.11	0.05	0.19	0.12	0.19	0.16	0.12	0.16
Average	0.20	0.13	0.28	0.25	0.26	0.24	0.15	0.24

Table C.2: Correlation between all automatic metrics and AMT evaluations, for D&A questions evaluated on MONSERRATE, using AMT average results (top) and median results (bottom).

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Grammar	0.09	0.12	0.08	0.15	0.10	0.10	0.07	0.09
Semantic	-0.08	0.03	-0.05	-0.01	-0.06	-0.12	-0.02	0.07
Plausibility	0.14	0.23	0.14	0.10	0.20	0.14	0.26	0.13
Utility	0.05	0.14	0.08	0.09	-0.04	0.07	0.09	0.15
Average	0.07	0.17	0.09	0.11	0.07	0.06	0.13	0.15
Grammar	0.01	0.01	0.06	0.11	0.05	0.04	0.05	0.01
Semantic	-0.11	0.05	-0.02	-0.01	-0.01	-0.09	0.01	0.05
Plausibility	0.22	0.23	0.21	0.10	0.25	0.21	0.31	0.14
Utility	0.02	0.12	0.09	0.09	-0.09	0.04	0.00	0.12
Average	0.04	0.14	0.11	0.11	0.07	0.07	0.13	0.11

Table C.3: Correlation between all automatic metrics and AMT evaluations, for GEN questions evaluated on MONSERRATE, using AMT average results (top) and median results (bottom).

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Grammar	0.18	0.18	0.17	0.18	0.01	0.11	0.08	0.14
Semantic	0.18	0.20	0.12	0.19	0.10	0.15	0.13	0.14
Plausibility	0.13	0.15	0.10	0.09	0.02	0.09	0.11	0.14
Utility	0.12	0.18	0.13	0.08	0.06	0.10	0.07	0.11
Average	0.20	0.23	0.17	0.18	0.06	0.14	0.12	0.17
Grammar	0.13	0.11	0.19	0.11	-0.05	0.06	0.04	0.07
Semantic	0.16	0.18	0.11	0.19	0.08	0.13	0.10	0.10
Plausibility	0.09	0.07	0.12	0.02	-0.03	0.06	0.06	0.10
Utility	0.14	0.19	0.09	0.08	0.10	0.12	0.14	0.15
Average	0.19	0.20	0.18	0.15	0.04	0.13	0.12	0.15

Table C.4: Correlation between all automatic metrics and AMT evaluations, for H&S questions evaluated on SQuAD, using AMT average results (top) and median results (bottom).

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Grammar	0.10	0.06	0.08	0.11	0.08	0.08	0.12	0.02
Semantic	0.04	0.05	-0.01	0.05	0.01	-0.01	0.07	-0.05
Plausibility	0.13	0.07	0.04	0.06	0.06	0.14	0.18	0.09
Utility	0.05	0.09	0.06	0.06	0.05	0.11	0.10	0.03
Average	0.10	0.08	0.06	0.09	0.07	0.10	0.15	0.03
Grammar	0.09	0.04	0.09	0.09	0.09	0.10	0.07	0.04
Semantic	0.01	0.02	0.00	0.02	-0.03	-0.05	0.03	-0.04
Plausibility	0.06	0.03	-0.02	-0.02	0.05	0.11	0.16	0.05
Utility	0.14	0.14	0.13	0.06	0.11	0.19	0.09	0.13
Average	0.11	0.08	0.07	0.06	0.08	0.13	0.12	0.06

Table C.5: Correlation between all automatic metrics and AMT evaluations, for D&A questions evaluated on SQuAD, using AMT average results (top) and median results (bottom).

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Grammar	0.20	0.10	0.18	0.13	0.07	0.15	0.21	0.14
Semantic	0.22	0.17	0.20	0.13	0.15	0.16	0.33	0.17
Plausibility	0.30	0.24	0.30	0.20	0.23	0.30	0.43	0.27
Utility	0.31	0.23	0.30	0.23	0.25	0.28	0.36	0.29
Average	0.34	0.24	0.32	0.23	0.23	0.29	0.43	0.28
Grammar	0.16	0.07	0.12	0.12	0.02	0.11	0.13	0.07
Semantic	0.21	0.13	0.17	0.10	0.16	0.18	0.31	0.17
Plausibility	0.29	0.23	0.29	0.22	0.28	0.32	0.35	0.26
Utility	0.11	0.07	0.11	0.10	0.10	0.11	0.17	0.16
Average	0.27	0.17	0.24	0.19	0.20	0.26	0.33	0.23

Table C.6: Correlation between all automatic metrics and AMT evaluations, for GEN questions evaluated on SQuAD, using AMT average results (top) and median results (bottom).

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Grammar	0.15	0.09	0.13	0.01	0.08	0.11	0.10	0.13
Semantic	0.18	0.12	0.13	0.07	0.08	0.12	0.14	0.13
Plausibility	0.16	0.15	0.12	0.10	0.06	0.16	0.12	0.16
Utility	0.08	0.11	0.07	0.06	0.07	0.10	0.10	0.12
Average	0.19	0.15	0.15	0.08	0.09	0.16	0.15	0.18
Grammar	0.09	0.07	0.10	0.02	0.06	0.06	0.06	0.10
Semantic	0.22	0.15	0.17	0.12	0.11	0.16	0.23	0.17
Plausibility	0.02	0.01	0.01	0.00	-0.14	0.00	0.00	-0.02
Utility	0.08	0.11	0.05	0.10	0.03	0.08	0.11	0.08
Average	0.15	0.12	0.12	0.09	0.03	0.11	0.14	0.12

Table C.7: Rank correlation between all automatic metrics and AMT evaluations, for GEN questions evaluated on MONSERRATE, using AMT average results (top) and median results (bottom).

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Grammar	0.10	0.10	0.14	0.09	0.03	0.07	0.01	0.07
Semantic	0.12	0.13	0.10	0.10	0.07	0.09	0.03	0.08
Plausibility	0.08	0.11	0.10	0.06	0.03	0.05	0.01	0.06
Utility	0.09	0.12	0.10	0.07	0.06	0.06	0.03	0.07
Average	0.13	0.15	0.12	0.11	0.06	0.10	0.05	0.10
Grammar	0.07	0.06	0.16	0.05	-0.01	0.02	-0.04	0.02
Semantic	0.10	0.10	0.13	0.07	0.05	0.08	-0.02	0.05
Plausibility	0.05	0.07	0.12	-0.01	0.01	0.03	-0.04	0.04
Utility	0.06	0.07	0.08	0.01	0.05	0.04	-0.01	0.04
Average	0.11	0.12	0.13	0.08	0.04	0.08	0.03	0.08

D Implicit Feedback Results Detailed

In this chapter are presented all results obtained for all configurations for the experiment reported in Chapter 6. We run the experiment for batches of sizes 7, 10, and 12 (10, 7, and 6 batches, respectively).

We also run different parameterizations for the weighing techniques (see Section 3.4). Besides the different *sim* functions, we also set the *penalties* and *bonus* for both. For Weighed Majority Algorithm (WMA) we set the following weights for *penalty* and *bonus* parameters, respectively, to: 0.1, 0.2, and 0.1, 0.3, 0.5. The threshold *th* (Equation 3.11) for *sim* function was set to 0.9 and 0.8. For Exponentially Weighed Average Forecast (EWAF) we set the *penalty* to values of 0.1 and 0.2. The threshold *th* and *bonus* parameters are not applicable. Table D.1 summarizes this information.

Tables D.2 to D.13 show all results and complete the data first presented in Chapter 6.

Table D.1: Parametrization of the different variables: function *sim*, its threshold *th*, penalties and bonus values.

	sim	***th***	***penalty***	***bonus***
WMA	Overlap, Lev	0.9, 0.8	0.1, 0.2	0.1, 0.3, 0.5
EWAF	Overlap, Lev	-	0.1, 0.2	-

Table D.2: Comparison of the weighing strategies against the baselines, measured by automatic metrics on Monserrate, at top 5. Scores normalized by the best score obtained in each metric. Overall results for batches of size 10 (7 batches) – averaged on all but first batch.

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Original Patterns	0.74	0.62	0.83	0.48	0.94	0.84	0.76	0.82
Baseline	0.92	0.81	**1.00**	0.67	0.99	0.91	0.86	1.00
EWAF-Lev-01	0.97	*0.78*	*0.99*	0.80	*0.99*	0.91	0.89	
EWAF-Lev-02	0.99	0.93	*0.96*	0.97	1.00	1.00	0.95	*0.99*
EWAF-Overlap-01	**1.00**	**1.00**	*0.96*	**1.00**	**1.00**	**1.00**	**0.96**	*0.99*
EWAF-Overlap-02	0.97	0.82	*0.99*	0.87	*0.99*	0.91	0.88	*0.99*
WMA-Lev-08-01	0.97	*0.79*	*0.98*	0.85	0.99	0.91	0.93	*0.98*
WMA-Lev-08-02	0.97	**0.82**	*0.99*	0.88	0.99	0.91	0.93	*0.99*
WMA-Lev-09-01	**0.98**	0.81	*0.99*	**0.89**	0.99	**0.92**	**0.93**	*1.00*
WMA-Lev-09-02	0.97	*0.79*	*0.98*	0.85	0.99	0.91	0.93	*0.99*
WMA-Overlap-08-01	0.97	*0.79*	*0.98*	0.85	0.99	0.91	0.93	*0.98*
WMA-Overlap-08-02	0.97	**0.82**	*0.99*	**0.88**	0.99	0.91	0.93	*0.99*
WMA-Overlap-09-01	0.97	0.81	*0.99*	0.88	**0.99**	0.92	0.93	*0.99*
WMA-Overlap-09-02	0.97	*0.79*	*0.98*	0.85	0.99	0.91	0.93	*0.99*
WMA-Lev-08-01-B01	*0.90*	**0.85**	*0.92*	*0.45*	**0.99**	**0.92**	**1.00**	*0.97*
WMA-Overlap-08-01-B01	*0.90*	**0.85**	*0.92*	*0.45*	**0.99**	**0.92**	**1.00**	*0.97*
WMA-Overlap-08-01-B03	*0.90*	**0.85**	*0.92*	*0.45*	**0.99**	**0.92**	**1.00**	*0.97*
WMA-Overlap-08-01-B05	*0.90*	**0.85**	*0.92*	*0.45*	**0.99**	**0.92**	*0.83*	*0.97*
WMA-Overlap-08-02-B03	*0.90*	**0.85**	*0.92*	*0.45*	**0.99**	**0.92**	*0.83*	*0.97*
WMA-Overlap-08-02-B05	*0.90*	**0.85**	*0.92*	*0.45*	**0.99**	**0.92**	*0.83*	*0.97*

Table D.3: Comparison of the weighing strategies against the baselines, measured by automatic metrics on Monserrate, at top 10. Scores normalized by the best score obtained in each metric. Overall results for batches of size 10 (7 batches) – averaged on all but first batch.

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Original Patterns	0.74	0.68	0.82	0.53	0.95	0.84	0.78	0.83
Baseline	0.94	0.93	**1.00**	0.83	0.99	0.92	0.90	1.00
EWAF-Lev-01	0.97	0.94	*0.98*	0.85	*0.99*	*0.92*	0.92	*0.98*
EWAF-Lev-02	0.98	0.94	*0.99*	0.96	1.00	1.00	0.95	*0.98*
EWAF-Overlap-01	**1.00**	**0.99**	*0.99*	**1.00**	**1.00**	**1.00**	**0.96**	*0.99*
EWAF-Overlap-02	0.97	0.96	*0.98*	0.88	1.00	*0.92*	0.92	*0.98*
WMA-Lev-08-01	0.97	0.98	*0.98*	0.90	*0.99*	*0.92*	0.96	*0.98*
WMA-Lev-08-02	**0.98**	**1.00**	*0.98*	0.92	*0.99*	**0.92**	**0.97**	1.00
WMA-Lev-09-01	0.97	0.97	*0.98*	**0.94**	*0.99*	*0.92*	0.96	*0.99*
WMA-Lev-09-02	0.97	0.98	*0.98*	0.90	*0.99*	*0.92*	0.96	*0.99*
WMA-Overlap-08-01	0.97	0.98	*0.98*	0.90	*0.99*	*0.92*	0.96	*0.98*
WMA-Overlap-08-02	**0.98**	**1.00**	*0.98*	0.92	*0.99*	**0.92**	**0.97**	*1.00*
WMA-Overlap-09-01	0.97	0.97	*0.98*	**0.94**	*0.99*	*0.92*	0.96	*0.99*
WMA-Overlap-09-02	0.97	0.98	*0.98*	0.90	*0.99*	*0.92*	0.96	*0.99*
WMA-Lev-08-01-B01	*0.88*	*0.82*	*0.93*	*0.44*	*0.99*	*0.91*	*0.86*	*0.94*
WMA-Overlap-08-01-B01	*0.88*	*0.82*	*0.93*	*0.44*	*0.99*	*0.91*	*0.86*	*0.94*
WMA-Overlap-08-01-B03	*0.88*	*0.82*	*0.93*	*0.44*	*0.99*	*0.91*	**1.00**	*0.94*
WMA-Overlap-08-01-B05	*0.88*	*0.82*	*0.93*	*0.44*	*0.99*	*0.91*	*0.86*	*0.94*
WMA-Overlap-08-02-B03	*0.88*	*0.82*	*0.93*	*0.44*	*0.99*	*0.91*	*0.86*	*0.94*
WMA-Overlap-08-02-B05	*0.88*	*0.82*	*0.93*	*0.44*	*0.99*	*0.91*	*0.86*	*0.94*

Table D.4: Comparison of the weighing strategies against the baselines, measured by automatic metrics on MONSERRATE, at top 20. Scores normalized by the best score obtained in each metric. Overall results for batches of size 10 (7 batches) – averaged on all but first batch.

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Original Patterns	0.76	0.75	0.81	0.65	0.96	0.85	0.85	0.85
Baseline	0.89	0.93	0.96	0.87	0.98	0.91	0.97	0.95
EWAF-Lev-01	0.92	0.97	*0.95*	0.90	0.98	0.92	0.99	0.98
EWAF-Lev-02	0.99	0.97	**1.00**	0.96	1.00	1.00	0.99	1.00
EWAF-Overlap-01	**1.00**	**1.00**	1.00	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
EWAF-Overlap-02	0.92	0.97	*0.95*	0.90	0.98	0.92	0.99	0.98
WMA-Lev-08-01	0.90	0.94	*0.94*	*0.83*	*0.98*	0.91	**1.00**	0.97
WMA-Lev-08-02	0.90	0.94	*0.94*	*0.83*	*0.98*	0.91	**1.00**	0.97
WMA-Lev-09-01	**0.94**	**0.99**	0.97	**0.95**	**0.99**	**0.92**	0.98	**0.99**
WMA-Lev-09-02	0.92	0.96	*0.95*	0.90	0.98	0.91	0.98	0.98
WMA-Overlap-08-01	0.92	0.96	*0.95*	0.90	0.98	0.91	0.98	0.98
WMA-Overlap-08-02	0.92	0.96	*0.95*	0.90	0.98	0.91	0.98	0.98
WMA-Overlap-09-01	**0.94**	0.97	**0.97**	0.92	0.99	0.92	0.98	0.99
WMA-Overlap-09-02	0.92	0.96	*0.95*	0.90	0.98	0.91	0.98	0.98
WMA-Overlap-08-01-B05	*0.88*	*0.87*	**0.96**	*0.83*	*0.98*	**0.91**	*0.95*	*0.95*
WMA-Overlap-08-01-B03	*0.88*	*0.85*	**0.96**	*0.72*	*0.97*	*0.90*	*0.93*	*0.94*
WMA-Overlap-08-02-B05	*0.88*	*0.87*	**0.96**	*0.83*	*0.98*	**0.91**	*0.95*	*0.95*
WMA-Overlap-08-02-B03	*0.88*	*0.85*	**0.96**	*0.72*	*0.97*	*0.90*	*0.93*	*0.94*

Table D.5: Comparison of the weighing strategies against the baselines, measured by automatic metrics on MONSERRATE, at top 5. Scores normalized by the best score obtained in each metric. Overall results for batches of size 7 (10 batches) – averaged on all but first batch.

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Original Patterns	0.85	0.73	0.85	0.52	0.95	0.93	0.87	0.85
Baseline	0.95	0.91	1.00	0.68	1.00	1.00	0.97	0.99
EWAF-Lev-01	**1.00**	**1.00**	**1.00**	**1.00**	*1.00*	*1.00*	**0.98**	**1.00**
EWAF-Lev-02	**1.00**	**1.00**	**1.00**	**1.00**	*1.00*	*1.00*	**0.98**	**1.00**
EWAF-Overlap-01	**1.00**	**1.00**	**1.00**	**1.00**	*1.00*	*1.00*	**0.98**	**1.00**
EWAF-Overlap-02	0.99	1.00	*0.93*	0.99	*1.00*	*1.00*	0.98	1.00
WMA-Lev-08-01	*0.95*	0.94	*0.97*	0.91	*0.99*	*0.98*	1.00	*0.99*
WMA-Lev-08-02	*0.95*	**0.97**	*0.97*	**0.92**	*0.99*	*0.98*	**1.00**	*0.96*
WMA-Lev-09-01	**0.96**	0.94	*0.98*	0.90	*0.99*	*0.98*	0.99	*0.96*
WMA-Lev-09-02	*0.95*	0.94	*0.97*	0.91	*0.99*	*0.98*	1.00	*0.96*
WMA-Overlap-08-01	*0.95*	0.94	1.00	0.91	*0.99*	*0.98*	1.00	*0.99*
WMA-Overlap-08-02	*0.95*	0.92	*0.96*	0.87	*0.99*	*0.98*	*0.95*	*0.96*
WMA-Overlap-09-01	*0.95*	0.93	**1.00**	0.86	*0.99*	*0.98*	0.98	*0.96*
WMA-Overlap-09-02	*0.95*	0.94	*0.97*	0.91	*0.99*	*0.98*	1.00	*0.96*
WMA-Overlap-08-02-B05	*0.90*	*0.82*	*0.99*	*0.59*	*1.00*	*0.99*	*0.84*	*0.96*
WMA-Overlap-08-02-B03	*0.90*	*0.82*	*0.99*	*0.59*	*1.00*	*0.99*	*0.84*	*0.96*
WMA-Overlap-08-01-B05	*0.90*	*0.82*	*0.99*	*0.59*	*1.00*	*0.99*	*0.84*	*0.96*
WMA-Overlap-08-01-B03	*0.90*	*0.82*	*0.99*	*0.59*	*1.00*	*0.99*	*0.84*	*0.96*

Table D.6: Comparison of the weighing strategies against the baselines, measured by automatic metrics on MONSERRATE, at top 10. Scores normalized by the best score obtained in each metric. Overall results for batches of size 7 (10 batches) – averaged on all but first batch.

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Original Patterns	0.99	0.87	0.89	**1.00**	0.97	0.97	0.89	0.91
Baseline	*0.91*	0.88	0.95	*0.71*	0.98	0.98	0.95	0.97
EWAF-Lev-01	**1.00**	1.00	1.00	*0.92*	1.00	1.00	**0.98**	1.00
EWAF-Lev-02	1.00	**1.00**	**1.00**	*0.94*	**1.00**	**1.00**	0.98	**1.00**
EWAF-Overlap-01	1.00	**1.00**	**1.00**	*0.94*	**1.00**	**1.00**	0.98	**1.00**
EWAF-Overlap-02	1.00	**1.00**	0.98	*0.94*	**1.00**	**1.00**	0.98	**1.00**
WMA-Lev-08-01	*0.98*	0.96	0.99	*0.88*	**1.00**	1.00	0.98	**1.00**
WMA-Lev-08-02	*0.98*	0.96	0.99	*0.88*	**1.00**	1.00	0.98	*0.97*
WMA-Lev-09-01	*0.99*	0.96	1.00	*0.89*	1.00	1.00	**1.00**	*0.97*
WMA-Lev-09-02	*0.99*	0.96	0.99	*0.87*	0.99	1.00	0.99	*0.97*
WMA-Overlap-08-01	0.99	0.99	**1.00**	*0.94*	1.00	1.00	0.99	1.00
WMA-Overlap-08-02	*0.98*	0.96	0.99	*0.91*	1.00	1.00	0.96	*0.97*
WMA-Overlap-09-01	0.99	0.97	0.99	*0.93*	1.00	1.00	0.99	*0.97*
WMA-Overlap-09-02	**1.00**	**0.99**	1.00	*0.94*	1.00	**1.00**	1.00	*0.97*
WMA-Overlap-08-02-B05	*0.89*	*0.81*	**0.96**	*0.59*	*0.98*	*0.98*	*0.87*	*0.94*
WMA-Overlap-08-02-B03	*0.89*	*0.81*	**0.96**	*0.59*	*0.98*	*0.98*	*0.87*	*0.94*
WMA-Overlap-08-01-B05	*0.89*	*0.81*	**0.96**	*0.59*	*0.98*	*0.98*	*0.87*	*0.94*
WMA-Overlap-08-01-B03	*0.89*	*0.81*	**0.96**	*0.59*	*0.98*	*0.98*	*0.87*	*0.94*

Table D.7: Comparison of the weighing strategies against the baselines, measured by automatic metrics on MONSERRATE, at top 20. Scores normalized by the best score obtained in each metric. Overall results for batches of size 7 (10 batches) – averaged on all but first batch.

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Original Patterns	0.91	0.85	0.86	0.88	0.94	0.91	0.85	0.87
Baseline	0.93	0.92	0.95	*0.82*	0.99	0.98	0.97	0.97
EWAF-Lev-01	0.97	0.98	0.97	0.88	1.00	0.99	0.99	0.99
EWAF-Lev-02	**1.00**	**1.00**	0.99	0.99	1.00	**1.00**	**1.00**	1.00
EWAF-Overlap-01	1.00	1.00	**1.00**	**1.00**	**1.00**	**1.00**	0.99	**1.00**
EWAF-Overlap-02	0.97	0.97	0.98	0.89	1.00	0.99	0.99	0.99
WMA-Lev-08-01	0.99	0.98	0.98	0.91	1.00	1.00	1.00	1.00
WMA-Lev-08-02	0.99	0.98	0.98	0.91	1.00	1.00	1.00	1.00
WMA-Lev-09-01	0.99	0.98	0.97	0.89	1.00	0.99	**1.00**	0.99
WMA-Lev-09-02	**0.99**	0.98	0.97	0.90	1.00	0.99	0.99	0.99
WMA-Overlap-08-01	0.99	**0.98**	0.98	**0.93**	1.00	1.00	0.99	1.00
WMA-Overlap-08-02	0.99	0.98	**0.99**	**0.93**	**1.00**	**1.00**	0.99	**1.00**
WMA-Overlap-09-01	0.98	0.97	0.98	0.91	1.00	1.00	0.99	1.00
WMA-Overlap-09-02	0.98	0.98	**0.99**	**0.93**	**1.00**	**1.00**	0.99	**1.00**
WMA-Overlap-08-01-B05	**0.95**	**0.95**	**1.00**	**0.95**	**0.99**	**0.99**	*0.95*	**0.98**
WMA-Overlap-08-01-B03	**0.95**	0.93	**1.00**	0.83	0.99	0.98	*0.92*	*0.97*
WMA-Overlap-08-02-B05	**0.95**	**0.95**	**1.00**	**0.95**	**0.99**	**0.99**	*0.95*	**0.98**
WMA-Overlap-08-02-B03	**0.95**	0.93	**1.00**	0.83	0.99	0.98	*0.92*	*0.97*

Table D.8: Comparison of the weighing strategies against the baselines, measured by automatic metrics on MONSERRATE, at top 5. Scores normalized by the best score obtained in each metric. Overall results for batches of size 12 (6 batches) – averaged on all but first batch.

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Original Patterns	0.84	0.79	0.90	0.60	0.94	0.91	0.80	0.90
Baseline	0.93	0.89	0.93	0.81	0.97	0.95	0.90	0.94
EWAF-Lev-01	**1.00**	**1.00**	**0.94**	**0.91**	**0.97**	*0.95*	**1.00**	0.95
EWAF-Lev-02	**1.00**	**1.00**	**0.94**	**0.91**	**0.97**	*0.95*	**1.00**	**0.99**
EWAF-Overlap-01	**1.00**	**1.00**	**0.94**	**0.91**	**0.97**	*0.95*	**1.00**	**0.99**
EWAF-Overlap-02	**1.00**	**1.00**	**0.94**	**0.91**	**0.97**	*0.95*	**1.00**	0.95
WMA-Lev-08-01	**0.96**	**0.90**	*0.91*	*0.77*	**0.96**	*0.94*	**0.98**	0.91
WMA-Lev-08-02	**0.96**	**0.90**	*0.91*	*0.77*	**0.96**	*0.94*	**0.98**	**0.96**
WMA-Lev-09-01	0.95	*0.89*	*0.89*	*0.71*	*0.96*	*0.94*	**0.98**	0.96
WMA-Lev-09-02	0.95	*0.89*	*0.89*	*0.71*	*0.96*	*0.94*	**0.98**	0.96
WMA-Overlap-08-01	**0.96**	**0.90**	*0.91*	*0.77*	**0.96**	*0.94*	**0.98**	*0.91*
WMA-Overlap-08-02	**0.96**	**0.90**	*0.91*	*0.77*	**0.96**	*0.94*	**0.98**	**0.96**
WMA-Overlap-09-01	**0.96**	**0.90**	*0.91*	*0.77*	**0.96**	*0.94*	**0.98**	**0.96**
WMA-Overlap-09-02	**0.96**	**0.90**	*0.91*	*0.77*	**0.96**	*0.94*	**0.98**	**0.96**
WMA-Overlap-08-02-B03	**0.99**	**0.98**	**1.00**	*0.70*	**1.00**	**1.00**	*0.86*	**1.00**
WMA-Overlap-08-02-B05	**0.99**	**0.98**	**1.00**	**1.00**	**1.00**	**1.00**	*0.86*	**1.00**
WMA-Overlap-08-01-B05	**0.99**	**0.98**	**1.00**	*0.70*	**1.00**	**1.00**	*0.86*	**1.00**
WMA-Overlap-08-01-B03	**0.99**	**0.98**	**1.00**	*0.70*	**1.00**	**1.00**	*0.86*	**1.00**

Table D.9: Comparison of the weighing strategies against the baselines, measured by automatic metrics on MONSERRATE, at top 10. Scores normalized by the best score obtained in each metric. Overall results for batches of size 12 (6 batches) – averaged on all but first batch.

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Original Patterns	0.90	0.86	0.94	0.75	0.96	0.94	0.87	0.92
Baseline	0.94	0.92	0.97	0.81	0.98	0.97	0.95	0.98
EWAF-Lev-01	**0.99**	**1.00**	*0.95*	**1.00**	**0.98**	*0.97*	**1.00**	*0.96*
EWAF-Lev-02	**0.99**	**1.00**	*0.95*	**1.00**	**0.98**	*0.97*	**1.00**	**1.00**
EWAF-Overlap-01	**0.99**	**1.00**	*0.95*	**1.00**	**0.98**	*0.97*	**1.00**	**1.00**
EWAF-Overlap-02	**0.99**	**1.00**	*0.95*	**1.00**	**0.98**	*0.97*	**1.00**	*0.96*
WMA-Lev-08-01	**1.00**	**0.99**	*0.95*	**0.95**	0.98	**0.98**	**1.00**	0.96
WMA-Lev-08-02	**1.00**	**0.99**	*0.95*	**0.95**	**0.98**	**0.98**	**1.00**	**1.00**
WMA-Lev-09-01	0.97	0.93	*0.93*	0.91	*0.97*	*0.97*	0.98	0.98
WMA-Lev-09-02	0.97	0.93	*0.93*	0.91	*0.97*	*0.97*	0.98	0.98
WMA-Overlap-08-01	**1.00**	**0.99**	*0.95*	**0.95**	**0.98**	**0.98**	**1.00**	0.96
WMA-Overlap-08-02	**1.00**	**0.99**	*0.95*	**0.95**	**0.98**	**0.98**	**1.00**	**1.00**
WMA-Overlap-09-01	**1.00**	**0.99**	*0.95*	**0.95**	**0.98**	**0.98**	**1.00**	**1.00**
WMA-Overlap-09-02	**1.00**	**0.99**	*0.95*	**0.95**	**0.98**	**0.98**	**1.00**	**1.00**
WMA-Overlap-08-02-B03	**0.97**	*0.92*	**1.00**	*0.64*	**1.00**	**1.00**	*0.85*	**1.00**
WMA-Overlap-08-02-B05	**0.97**	*0.92*	**1.00**	**0.93**	**1.00**	**1.00**	*0.85*	**1.00**
WMA-Overlap-08-01-B05	**0.97**	*0.92*	**1.00**	*0.64*	**1.00**	**1.00**	*0.85*	**1.00**
WMA-Overlap-08-01-B03	**0.97**	*0.92*	**1.00**	*0.64*	**1.00**	**1.00**	*0.85*	**1.00**

Table D.10: Comparison of the weighing strategies against the baselines, measured by automatic metrics on Monserrate, at top 20. Scores normalized by the best score obtained in each metric. Overall results for batches of size 12 (6 batches) – averaged on all but first batch.

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Original Patterns	0.88	0.81	0.90	0.74	0.98	0.94	0.88	0.89
Baseline	0.97	0.93	1.00	0.87	0.99	0.99	1.00	0.98
EWAF-Lev-01	0.99	0.99	*0.97*	0.97	1.00	1.00	*0.97*	1.00
EWAF-Lev-02	0.99	0.99	*0.98*	0.97	1.00	1.00	*0.97*	1.00
EWAF-Overlap-01	**1.00**	**0.99**	*0.98*	**0.97**	**1.00**	**1.00**	*0.97*	**1.00**
EWAF-Overlap-02	**1.00**	**0.99**	*0.98*	**0.97**	**1.00**	**1.00**	*0.97*	**1.00**
WMA-Lev-08-01	1.00	0.99	*0.98*	0.97	**1.00**	**1.00**	*0.97*	**1.00**
WMA-Lev-08-02	1.00	0.99	*0.98*	0.97	**1.00**	**1.00**	*0.97*	**1.00**
WMA-Lev-09-01	0.99	0.99	*0.97*	0.95	1.00	1.00	*0.97*	1.00
WMA-Lev-09-02	**1.00**	**1.00**	*0.98*	**1.00**	1.00	1.00	*0.97*	1.00
WMA-Overlap-08-01	1.00	0.99	*0.98*	0.97	**1.00**	**1.00**	*0.97*	**1.00**
WMA-Overlap-08-02	1.00	0.99	*0.98*	0.97	**1.00**	**1.00**	*0.97*	**1.00**
WMA-Overlap-09-01	0.99	0.99	*0.97*	0.95	1.00	1.00	*0.97*	1.00
WMA-Overlap-09-02	**1.00**	**1.00**	*0.98*	**1.00**	1.00	1.00	*0.97*	1.00
WMA-Overlap-08-02-B03	*0.93*	*0.86*	*0.96*	*0.63*	*0.99*	*0.98*	*0.91*	*0.96*
WMA-Overlap-08-02-B05	*0.93*	*0.86*	*0.96*	*0.61*	*0.99*	*0.98*	*0.91*	*0.96*
WMA-Overlap-08-01-B05	*0.93*	*0.86*	*0.96*	*0.61*	*0.99*	*0.98*	*0.91*	*0.96*
WMA-Overlap-08-01-B03	*0.93*	*0.86*	*0.96*	*0.61*	*0.99*	*0.98*	*0.91*	*0.96*

Table D.11: Comparison of the weighing strategies against the baselines, measured by automatic metrics on Monserrate (shuffled order), at top 5. Scores normalized by the best score obtained in each metric. Overall results for batches of size 10 (7 batches) – averaged on all but first batch.

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Original Patterns	0.91	0.87	0.91	0.61	**1.00**	0.98	0.89	0.95
Baseline	0.89	0.79	0.90	0.65	0.99	0.98	0.92	0.94
EWAF-Lev-01	0.98	0.99	0.98	0.99	1.00	0.99	**0.98**	0.99
EWAF-Lev-02	0.98	0.99	0.98	0.99	1.00	0.99	**0.98**	0.99
EWAF-Overlap-01	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.97	**1.00**
EWAF-Overlap-02	0.98	0.99	0.98	0.99	1.00	0.99	**0.98**	0.99
WMA-Lev-08-01	*0.87*	0.91	*0.86*	0.76	0.98	*0.96*	0.96	0.96
WMA-Lev-08-02	0.89	0.90	*0.90*	0.86	0.99	*0.97*	0.95	0.97
WMA-Lev-09-01	**0.99**	**0.94**	**0.95**	0.85	**1.00**	**1.00**	**1.00**	**1.00**
WMA-Lev-09-02	0.92	0.91	*0.88*	**0.97**	0.98	*0.98*	0.97	0.97
WMA-Overlap-08-01	0.90	0.94	0.90	0.87	0.99	*0.97*	0.95	0.96
WMA-Overlap-08-02	0.93	0.92	0.94	0.89	0.99	0.99	0.95	0.98
WMA-Overlap-09-01	**0.99**	**0.94**	**0.95**	0.89	**1.00**	**1.00**	**1.00 1.00**	
WMA-Overlap-09-02	*0.88*	0.89	*0.89*	0.84	0.98	*0.96*	0.94	0.96
WMA-Overlap-08-02-B03	*0.85*	*0.78*	*0.87*	*0.39*	*0.98*	*0.96*	**0.93**	**0.94**
WMA-Overlap-08-02-B05	*0.85*	*0.78*	*0.87*	*0.39*	*0.98*	*0.96*	**0.93**	**0.94**
WMA-Overlap-08-01-B05	*0.85*	*0.78*	*0.87*	*0.39*	*0.98*	*0.96*	**0.93**	**0.94**
WMA-Overlap-08-01-B03	*0.85*	*0.78*	*0.87*	*0.39*	*0.98*	*0.96*	**0.93**	**0.94**

Table D.12: Comparison of the weighing strategies against the baselines, measured by automatic metrics on MONSERRATE (shuffled order), at top 10. Scores normalized by the best score obtained in each metric. Overall results for batches of size 10 (7 batches) – averaged on all but first batch.

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Original Patterns	0.96	0.94	0.99	0.79	0.99	0.98	0.89	0.96
Baseline	0.94	0.90	0.97	0.77	0.99	0.99	0.92	0.96
EWAF-Lev-01	**1.00**	**1.00**	**0.99**	**1.00**	**1.00**	**1.00**	**0.99**	**1.00**
EWAF-Lev-02	**1.00**	**1.00**	**0.99**	**1.00**	**1.00**	**1.00**	**0.99**	**1.00**
EWAF-Overlap-01	0.97	0.96	0.97	0.91	*0.99*	0.99	0.97	0.99
EWAF-Overlap-02	0.99	0.99	0.98	0.97	0.99	0.99	0.98	1.00
WMA-Lev-08-01	0.95	0.96	*0.95*	0.95	**0.99**	*0.98*	0.99	0.99
WMA-Lev-08-02	0.95	0.97	*0.95*	**0.98**	**0.99**	0.99	0.99	0.99
WMA-Lev-09-01	**0.99**	0.96	**0.97**	0.92	*0.99*	**1.00**	**1.00**	**1.00**
WMA-Lev-09-02	0.96	**0.97**	*0.93*	0.93	*0.99*	0.99	0.99	0.99
WMA-Overlap-08-01	0.95	0.95	*0.95*	0.91	*0.99*	*0.98*	0.97	0.99
WMA-Overlap-08-02	0.95	0.96	*0.95*	0.88	*0.99*	0.99	0.97	0.99
WMA-Overlap-09-01	**0.99**	0.96	**0.97**	0.88	*0.99*	**1.00**	**1.00**	**1.00**
WMA-Overlap-09-02	0.95	0.95	*0.95*	0.91	*0.99*	*0.98*	0.97	0.99
WMA-Overlap-08-02-B03	*0.94*	*0.88*	**1.00**	*0.59*	*0.97*	*0.98*	**0.93**	*0.96*
WMA-Overlap-08-02-B05	*0.94*	*0.88*	**1.00**	*0.59*	*0.97*	*0.98*	**0.93**	*0.96*
WMA-Overlap-08-01-B05	*0.94*	*0.88*	**1.00**	*0.59*	*0.97*	*0.98*	**0.93**	*0.96*
WMA-Overlap-08-01-B03	*0.94*	*0.88*	**1.00**	*0.59*	*0.97*	*0.98*	**0.93**	*0.96*

Table D.13: Comparison of the weighing strategies against the baselines, measured by automatic metrics on MONSERRATE (shuffled order), at top 20. Scores normalized by the best score obtained in each metric. Overall results for batches of size 10 (7 batches) – averaged on all but first batch.

	ROUGE	METEOR	BLEU1	BLEU4	EACS	GMS	STCS	VECS
Original Patterns	**1.00**	0.97	**1.00**	0.89	0.99	0.98	0.93	0.97
Baseline	0.97	0.94	0.96	0.83	0.99	0.99	0.95	0.98
EWAF-Lev-01	**0.97**	**1.00**	*0.94*	**0.96**	**0.99**	**1.00**	**0.99**	**0.99**
EWAF-Lev-02	**0.97**	**1.00**	*0.94*	**0.96**	**0.99**	**1.00**	**0.99**	**0.99**
EWAF-Overlap-01	0.97	0.98	*0.94*	0.94	0.99	*0.99*	0.98	0.99
EWAF-Overlap-02	0.97	0.98	*0.94*	0.94	0.99	*0.99*	0.98	0.99
WMA-Lev-08-01	0.97	0.98	*0.93*	0.99	1.00	1.00	1.00	0.99
WMA-Lev-08-02	0.98	**0.99**	*0.93*	**1.00**	**1.00**	**1.00**	0.99	**1.00**
WMA-Lev-09-01	**0.99**	0.97	**0.97**	1.00	0.99	1.00	0.98	0.99
WMA-Lev-09-02	0.97	0.97	*0.94*	0.95	0.99	*0.99*	0.97	0.99
WMA-Overlap-08-01	0.98	0.99	*0.94*	0.97	1.00	1.00	**1.00**	1.00
WMA-Overlap-08-02	0.98	0.99	*0.94*	0.93	1.00	1.00	1.00	1.00
WMA-Overlap-09-01	0.98	0.94	0.96	0.89	1.00	1.00	0.96	0.99
WMA-Overlap-09-02	0.99	0.98	*0.95*	0.98	1.00	1.00	0.98	1.00
WMA-Overlap-08-02-B03	*0.96*	*0.88*	0.96	*0.73*	*0.98*	*0.98*	*0.93*	*0.97*
WMA-Overlap-08-02-B05	**0.97**	*0.90*	**0.98**	*0.77*	*0.98*	*0.99*	*0.94*	*0.97*
WMA-Overlap-08-01-B05	**0.97**	*0.90*	**0.98**	*0.77*	*0.98*	*0.99*	*0.94*	*0.97*
WMA-Overlap-08-01-B03	**0.97**	*0.90*	**0.98**	*0.77*	*0.98*	*0.99*	*0.94*	*0.97*