Universiteit van Amsterdam

MSc Artificial Intelligence
Master Thesis

Complex Question Answering
by Pairwise Passage Ranking and Answer Style Transfer

by

Ioannis Tsiamas
12032239

9th July 2020

48 ECTS
1 November 2019 - 30th June 2020

Supervisor:
Dr. Pengjie Ren

Examiner:
Dr. Mohammad Aliannejadi

Acknowledgements

Working on this thesis for the past eight months was a great challenge. I am delightful for its completion and the knowledge I acquired during this time. This thesis would not have been possible without my supervisor, Pengjie Ren, and the valuable guidance that he provided me.

This thesis was also part of a research internship at Zeta Alpha. It has been wonderful to be part of such an awesome team, and I wish them the best of luck in building a great product. Many thanks to my supervisor at Zeta Alpha, Jakub Zavrel, and my colleague, Marzieh Fadaee, for their recommendations throughout the project and their always helpful feedback.

Special thanks to my roommate, classmate, and good friend, Tijs Maas, for his comments and suggestions, as well as the countless research discussions that we had. I wish him good luck with his thesis as well.

Finally, I would like to thank my family for their constant and unconditional support.

Abstract

Complex question answering (QA) refers to the task where a non-factoid query is answered using natural language generation (NLG), given a set of textual passages. A vital component of a high-quality QA system is the ability to rank the available passages, and thus focus on the most relevant information during answer generation. Another critical component of a QA system is learning from multiple answering styles, extractive, or abstractive through a process called multi-style learning, which helps to increase the performance of the individual styles by developing style-agnostic question answering abilities. This research tackles complex question answering, by focusing on these two essential features of a system, aiming to provide an improved framework for the task. Firstly, ranker modules are usually pointwise, ranking the passages in an absolute way by considering each of them individually from the rest, which is a potential bottleneck to their performance. Thus, in this research, an attention-based pairwise ranker is proposed that ranks passages in a relative way, by identifying the comparable relevance of each passage to the rest. Secondly, it is questionable whether multi-style learning is sufficient to combat the common data shortage in the abstractive-styled answers, which possibly leads to the style being under-trained. To mitigate this, a Style-transfer model is introduced, that first learns a mapping from the extractive to the abstractive style and is subsequently used to generate synthetic abstractive answers that can be utilized during multi-style training. The recently proposed Masque model (Nishida et al., 2019), which is a multi-style QA model that uses a pointwise ranker, serves as a baseline for this thesis's experiments. Inspired by the Masque architecture, PREAST-QA is proposed by combining both pairwise ranking and style transfer. PREAST-QA achieves competitive results in the MS-MARCO v2.1 NLG task and an improvement of 0.87 ROUGE-L points in abstractive answer generation over the Masque baseline. The success of the proposed model can be attributed to its increased ranking abilities and its use of high-quality synthetic data generated from the Style-transfer model, which further signifies the positive effects of multi-style learning, especially for low-resource query types.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction to Question Answering

Question-Answering (QA) is a sub-field of Information Retrieval (IR) and Natural Language Processing (NLP), which is concerned with providing an answer to a query, given some textual passage. Based on the nature of the query, QA can be distinguished into two categories: factoid and complex. Factoid QA refers to the answering of queries that deal with facts, which are usually “where”, “who”, and “when” type of queries. The challenge of factoid QA is to locate the answer in the passage and derive it from there. The second category, complex QA, refers to “how”, “what” and “why” type of queries, which usually require a higher level of reasoning over the passage, and a complete answer is not necessarily located in a single textual span. Moreover, QA can be further categorized on the answer type: extractive and abstractive. In extractive QA, the answer is purely extracted from the passage, wherein abstractive QA, the answer is synthesized in a generative way, and novel text is produced, that is not necessarily part of the passage. Factoid queries can be answered in both extractive and abstractive ways, while for complex queries, it is usually only possible to be answered in abstraction. Examples of factoid and complex queries, along with the different answering forms, can be found in Table 1.1. From these examples, it easy to identify the increased difficulty of answering complex queries, where information from multiple parts of the passage has to be combined to produce a complete answer. The current research is focused on these complex queries and on methods that generate abstractive, well-formed answers, even when the query is of factoid type.

Query: weather in amsterdam november
Passage: ... Averages for Amsterdam in November. November can be wet and chilly in Amsterdam but it can also still be a very good time of the year to visit. However, visitors will need to bring layers and a waterproof jacket because November is the wettest month of the year, averaging 90mm of rain. ...
Answer(extractive): wet and chilly
Answer(abstractive): In Amsterdam, the weather is wet and chilly in November.
Query: what is agriculture and why is important
Answer: Agriculture is the source of supply of food, clothing, medicine and employment all over the world. It is important to human beings because it forms the basis for food security. It helps human beings grow the most ideal food crops and raise the right animals with accordance to environmental factors.

Table 1.1: Examples of factoid and complex queries.

First example: factoid query, part of the passage, along with extractive and abstractive answers. Underlined is the answer in the passage. Second example: a complex query along with its answer, which is by default only abstractive.

This research was also conducted as part of an internship at Zeta Alpha[1], a company that focuses on understanding and navigating scientific literature, especially in the field of Artificial Intelligence (AI). Thus, another goal of this thesis is to develop systems that can answer queries within the domain of AI research, which are primarily complex (Table 1.2).

How does overfitting happen in a neural network?
Why is ReLU the most common activation function used in neural networks?
What is a convolutional neural network?

Table 1.2: Complex queries in the domain of AI research.

1.2 Neural Models

Modern approaches to the task of QA are neural encoder-decoder models, which work in a sequence-to-sequence manner, and have been particularly successful in not only question-answering but also in translation (Sutskever, Vinyals and Le, 2014)

[1]https://www.zeta-alpha.com/

and summarization Nallapati et al., 2016, among others tasks. The encoder reads and understands the passage and the query, models their interactions, and produces meaningful representations. Given these representations, the decoder generates the answer, token-by-token, in an auto-regressive way, by using the previously generated tokens (Figure 1.1).

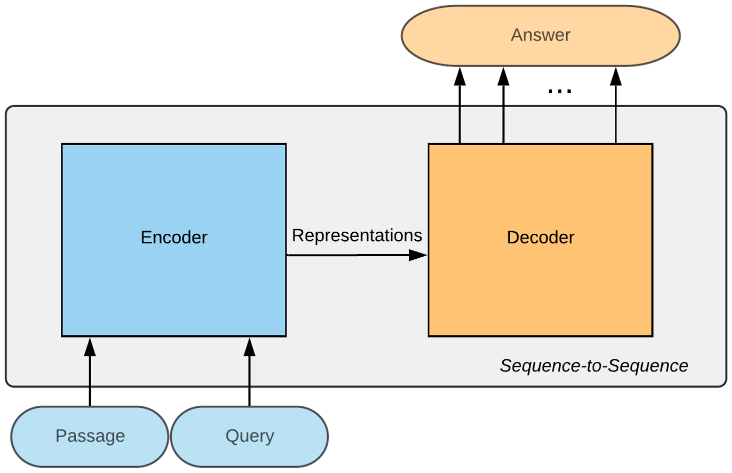


Figure 1.1: A sequence-to-sequence model.

The encoder receives as input the passage and the query, and models their interactions to produce meaningful representations. The representations are fed into the decoder, which generates the answer, one token at a time.

Traditionally, Recurrent Neural Networks (RNN), and more specifically, their Long Short-term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) counterparts, have been used in the encoder and decoder, for understanding the query and the passage, and generating an answer. These are usually coupled with an attention mechanism (Bahdanau, Cho and Bengio, 2014) to enhance the decoder's information from the source sequence. The interactions between the query and the passage contribute to significantly more informative representations and can be modeled effectively using attention variants like Bidirectional Attention Flow (BiDAF) (Seo et al., 2016) or Dynamic Coattention (DCN) (Xiong, Zhong and Socher, 2016). For extractive QA, Pointer Networks (Vinyals, Fortunato and Jaitly, 2015) can be used to find the most probable span in the passage that contains the answer, while for abstractive approaches, tokens are generated from a learned vocabulary. Furthermore, Pointer-Generator Networks (See, P. J. Liu and Manning, 2017) provide the flexibility to either copy a token from the passage or generate it from the fixed vocabulary, thus combining both extractive and abstractive methods. Finally, in recent years, the Transformer (Vaswani et al., 2017), which is a model relying heavily on attention mechanisms, has been a key to the success of many approaches that solve sequence-to-sequence problems and thus has also replaced LSTMs in QA tasks.

1.3 Complex Scenarios

The methods described above work relatively well for simple QA frameworks, but in more realistic scenarios, there is not a single, but multiple available passages for a query, where one may imagine the results of a search engine as an example. Additionally, a query might not even be answerable with the information of the provided passages. The multiple passages and the non-answerable queries, add another layer of complexity to the QA task, potentially making the encoder-decoder model incapable of solving it, since generating an answer from the combined passages increases in difficulty with their number, as the decoder is not able to focus on the critical information. Thus, a crucial part of QA becomes the ability to rank the available passages based on their relevance to the query, subsequently guiding the decoder's focus. This procedure can be carried out by a ranker module that uses the representations of the query and each passage to assign them a relevance score. Since the ranker and the decoder rely on the same representations, it can be beneficial to share the encoder in a multi-task framework (Caruana, 1997). Finally, a module that discriminates answerable from non-answerable queries can optionally be added to the multi-task framework (Figure 1.2), along with the ranking and generation tasks.

Similarly, as the main task of answer generation can benefit from learning to rank and classify, it can additionally benefit from learning different answering styles. Given the query of the first example (Table 1.1), one can identify the two styles being the short-extractive and the longer-abstractive. An answer of either structure relies on the same processes to be generated, and thus a model can share the two answering tasks. The effect of multi-style learning is even more significant in scenarios where there is a shortage of available examples for a given style (Nishida et al., 2019). Thus, a multi-style QA model can learn apart from style-specific, also style-agnostic answering, increasing the performance of the under-represented style.

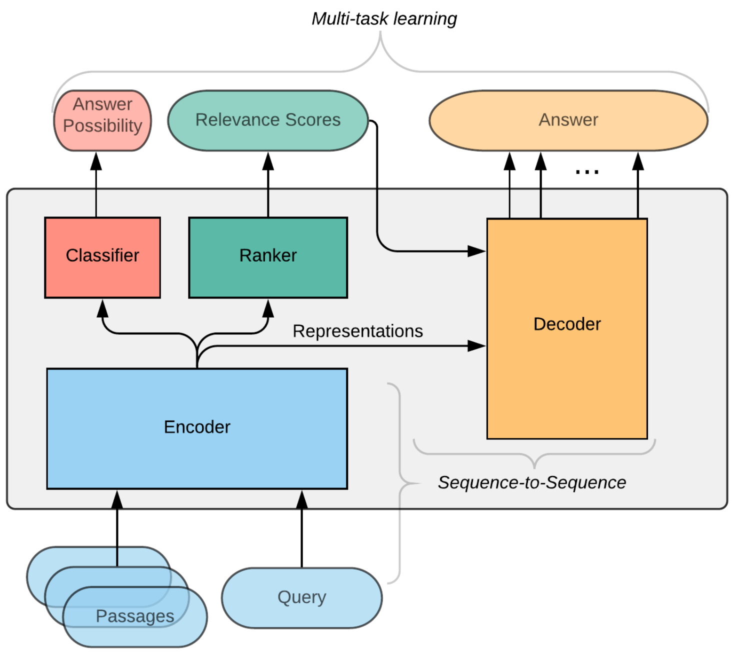


Figure 1.2: A multi-task sequence-to-sequence model.

The encoder receives the available passages and the query and models their interactions to produce meaningful representations. The representations are fed into the classifier, ranker, and decoder, to discriminate the answerability of the query, obtain relevance scores for each of the passages and if possible, generate an answer. The decoder additionally receives the output of the ranker, providing extra guidance during answer generation.

The recent advancements in Question Answering are not only attributed to new architectures like the Transformer, but also to large, labeled datasets that enable models to learn more effectively. One example of such a dataset is the MS-MARCO v2.1 (Nguyen et al., 2016), which has a size of 1 million queries issued by users of the Bing search engine. A collection of relevant and non-relevant passages is included for every query, and if this collection contains at least one relevant passage, one or more answers are provided for it. Finally, for a portion of the answerable queries, additional well-formed answers that are more abstractive than the standard ones, are also given. The multiple passages, non-answerable queries, and multiple answers establish the MS-MARCO, as a complex dataset, that is very similar to real-world scenarios, and requires systems to perform various tasks at once, in order to model it effectively.

The recently proposed Masque (Nishida et al., 2019) is a QA model that combines both the features described above, being multi-task and multi-style. It is based on a Transformer encoder-decoder architecture that additionally uses a Multi-Source Pointer-generator network, thus having the ability to either copy a token from the passages and the query or generate an original one from a fixed vocabulary. The passage-query interactions are modeled using a Dual Attention module that updates their respective representations simultaneously and bidirectionally, based on a common similarity matrix. Relevance ranking and answerability discrimination are carried out by linear layers that classify each passage as relevant or not to the query and the query as answerable or not. The decoder is conditioned on either of the two answering styles, extractive or abstractive, and can thus generate an answer in different styles. Masque proved to be particularly effective, achieving state-of-the-art results on the Natural Language Generation (NLG) task of MS Marco v2.1, with its success being attributed mainly to multi-style learning and effective passage ranking.

1.4 Motivation and Contributions

As argued in Nishida et al., 2019, passage ranking will be a key in developing QA systems that outperform humans in the task. Although the ranker module of Masque achieved high performance in discriminating between relevant and non-relevant passages, it does not use the full amount of available information provided for each query. More specifically, it operates in a pointwise manner, by assigning a relevance score to each passage independently from the rest. Consequently, the pointwise ranker (*PointRnk*) aims to rank the passages for each query in an absolute way, and thus, it is debatable whether it can effectively rank challenging cases that require a relative point of view. To mitigate this issue, and considering the potential benefits of increasing the efficacy of the ranker, a Pairwise Ranker (*PairRnk*) is proposed, that approaches ranking in a relative way. The PairRnk method uses a transformer layer that enables passage-to-passage attention, allowing the passages to exchange information and obtain globally updated representations. Subsequently, it models the comparative relevance of each passage to the rest, by a series of pairwise comparisons, where it breaks down the task from classifying n passages as relevant or not, to classifying the relative importance of $n \times n$ passage pairs. Finally, PairRnk aggregates the results of all the pairwise comparisons into a relevance distribution over the passages that is used to guide the decoder during answer generation.

There are two main advantages of this method in contrast to the PointRnk of Masque:

- **Information: Global vs Narrow**. PairRnk uses a transformer encoder to update each passage representation with information from the rest of the passages in the example.
- **Setting: Relative vs Absolute**. PairRnk does ranking in a relative setting, by identifying the comparative advantage of each passage, through a series of pairwise comparisons.

Experiments in the MS-MARCO, show that an encoder-only model, which uses PairRnk, achieves an improvement of 2 points in Mean Average Precision over the PointRnk method. Although the gap between the two methods decreases to 0.5, with the decoder's inclusion in the multi-task question answering scenario, the improved ranking capabilities of the model translate to an increase of 0.74 ROUGE-L points in abstractive answer generation, as compared to a Masque baseline that uses PointRnk.

Multi-style learning is another key aspect of Masque, where the model learns to generate answers in both extractive and abstractive styles. The multi-style feature, apart from providing a flexible system for real-world applications, it additionally increases the performance of both styles, by learning style-independent answering. Although the ability to generate abstractive answers is of higher importance, since the model can provide a more human-like answer to a query, it is usually harder to develop. This difficulty is a direct consequence of a data shortage in abstractive answers since constructing such datasets is a more time-consuming and expensive process. The MS-MARCO falls into this category, having a well-formed, abstractive answer for only 30% of the answerable queries. This begs the question of whether the abstractive answering style of a QA model that uses MS-MARCO is under-trained, due to the fewer available examples. The abstractive data shortage becomes even more critical for certain query types, like "why" or "which", that make up only a tiny percentage of the total dataset (Figure 1.3).

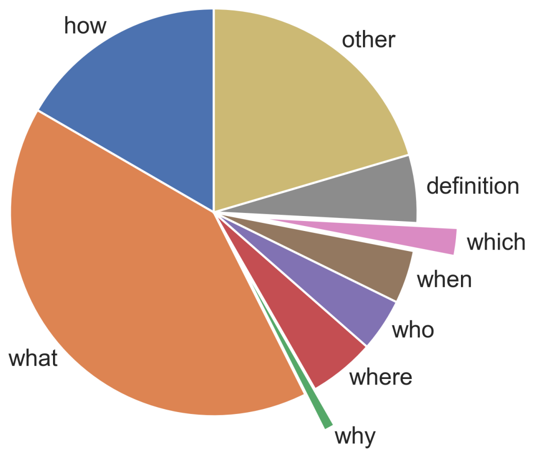


Figure 1.3: Query type distribution with abstractive answers.

The queries are categorized by whether they contain certain keywords, with the most notable categories being shown here. The total number of answerable queries with abstractive answers is 152 thousand, and for reference, the largest category, “what”, makes up 63 thousand of them, or 41%. A small overlap of 2 thousand queries between categories is omitted for this visualization.

It is hypothesized that, especially for these low-resource query types, there are not enough training examples to learn the specific patterns of their abstractive answering style. Thus, to enrich the abstractive style with more trainable examples, a Style-transfer, transformer-based model is proposed, that first learns a mapping from the extractive to the abstractive style, and then it is used to generate abstractive answers for all the answerable examples in the dataset. The proposed Style-transfer model can produce abstractive answers of high quality, with an overall ROUGE-L score of 87, and even above 90, for certain query types. A synthetic dataset, generated by the Style-transfer model is additionally used in multi-style training, improving the abstractive answer generation in low-resource query types, which consequently increases the average ROUGE-L score by 1.18 points, as compared to a Masque baseline that is trained on the non-augmented dataset. Furthermore, training on the synthetic data, aids in relieving a data bias that is absorbed by the artificial tokens in multi-style training. The unequal distributions of the answers that are available for each style cause the model to output completely different answers for the extractive and abstractive styles, given the same passages and query, in 6.6% of the cases. The inclusion of the synthetic abstractive answers bridges the gap between the styles and reduces the inconsistent generations to 4.8% of the cases.

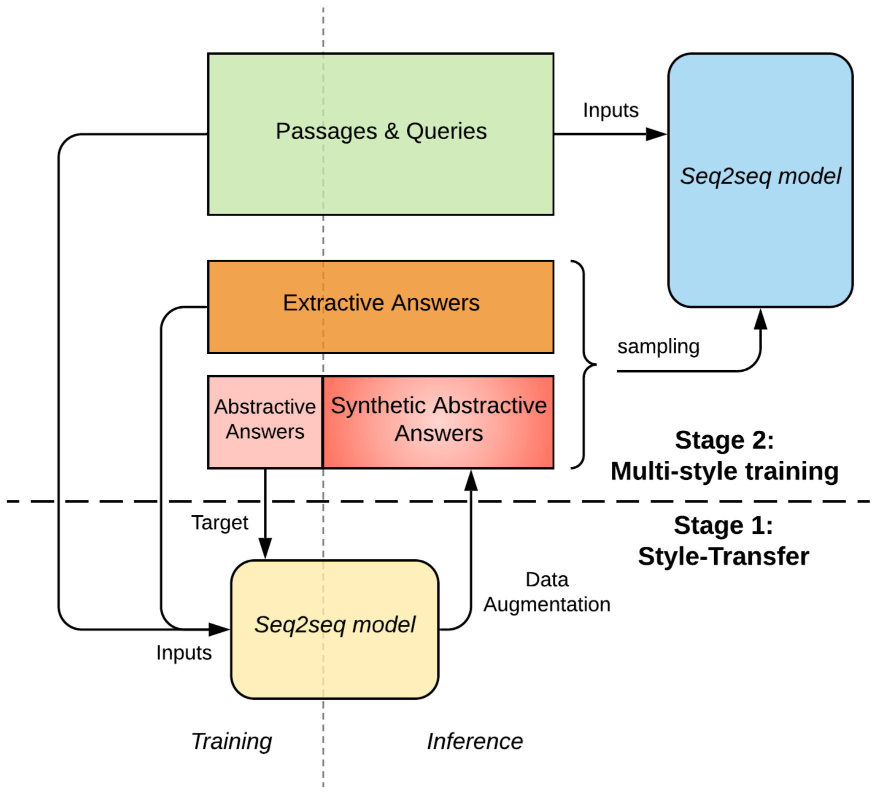


Figure 1.4: Data Augmentation through Style-Transfer.

Stage 1: A sequence-to-sequence model is trained on the subset of the examples that have both extractive and abstractive answers available. The trained model produces synthetic abstractive answers for the examples that only have extractive answers available.
Stage 2: A sequence-to-sequence model is trained on the question-answering task using multi-style training, where for each training example, the sampled target can be either extractive or abstractive, and for the latter one, either true or generated.

In the framework of multi-task learning, Masque classifies each example as answerable or not, by mapping the concatenated passage representations to a scalar through a linear layer. It is found that this method uses a very high amount of parameters, for a purely auxiliary task, taking away a significant percentage of the complexity from the encoder. Additionally, it introduces a bias regarding each passage's position in the example, which decreases the classifier's performance due to noisy training signals. To illustrate this bias, if the passages in the example are shuffled and given a decision boundary of 0.5, the classifier of Masque has different predictions in 2% of the cases. In this research, these issues are fixed by a much simpler, position-agnostic, max-pooling classifier that achieves significantly better classification accuracy compared to Nishida et al., 2019 and enable the encoder to produce higher quality representations.

Finally, all the proposed methods are combined in a Masque-inspired model, called *PREAST-QA* (Question-Answering by Pairwise Ranking and Extractive-Abstractive Style Transfer) that not only achieves high results in abstractive answer generation, but also is more effective in passage ranking, and answerability classification, in comparison with a vanilla Masque re-implementation. More specifically, PREAST-QA improves the ROUGE-L score by 0.87 in the NLG development set of MS-MARCO, and reduces the difference with the original implementation of Masque (Nishida et al., 2019), which additionally uses contextualized embeddings. Furthermore, PREAST-QA achieves comparable results with Nishida et al., 2019 in passage ranking and even superior, by almost 1 point in F1 score, in answerability classification, providing a competitive multi-task system, without relying on external information from pre-trained models like ELMo (Peters et al., 2018) or BERT (Devlin et al., 2018).

To summarize, the main contributions of this thesis are the following:

1. A **Pairwise Passage Ranker** is introduced that learns the comparative advantage of each passage to rest and enables better ranking, which directly increases the answer generation capabilities of a QA model. Its success is partly attributed to the use of a passage-to-passage transformer, which effectively fuses information from all the passages.

2. A **Style Transfer** model is proposed, which learns a mapping from the extractive to the abstractive answering styles and is used to generate high-quality answers for the latter one. The synthetic answers augment the multi-style learning procedure, further increasing the abstractive style's performance, with the improvements being more significant for low-resource query types.

3. **Masque's Answerability Classifier** is found to suffer from positional bias and is replaced by a simpler, position-agnostic classifier leading to substantially better accuracy.

1.5 Thesis Outline

The rest of the thesis is organized in 5 chapters. The related works in question-answering, encoder-decoder models, multi-task learning, and multi-style learning are covered in chapter 2. Following, in chapter 3, a background is provided in attention mechanisms, transformers, and pointer-generators. Then, chapter 4 builds on the background by firstly introducing the Masque architecture in section 4.1, which is the basic model used in this research. In the same chapter, sections 4.2, 4.3 and 4.4 are used to propose the new methods of this research, namely the pairwise ranker and the style-transfer model and resolve the issues of the answerability classifier of Nishida et al., 2019. Details on the dataset, and training and evaluation of the models are given in section 5.1 of chapter 5. Following, in section 5.2, are the main results and ablation studies regarding passage ranking and answer style transfer. An application of PREAST-QA in the domain of AI scientific literature is presented in section 5.3. Finally, the conclusions of this thesis and the possible directions of future research are discussed in chapter 6.

Chapter 2

Related Works

2.1 Question Answering and Machine Reading Comprehension

Following a comprehensive review study of Machine Reading Comprehension (MRC) and Question Answering (QA) in S. Liu et al., 2019, there are four essential parts of a QA system. These are the type of embeddings, the encoder feature extractor, the nature of the interaction between the passage and query, and the required answering type.

Word Embeddings: The choice of embeddings was traditionally one of the pre-trained distributed word embeddings GloVe (Pennington, Socher and Manning, 2014) or word2vec (Mikolov et al., 2013), which can be frozen or fine-tuned after initialization. In order to be able to deal effectively with out-of-vocabulary (OOV) words, character-level embeddings (Kim et al., 2015) can be additionally employed (Y. Wang et al., 2018). Recently, with the proven effectiveness of large scale pre-trained language models like ELMo (Peters et al., 2018), Bert (Devlin et al., 2018) and other variants (Lan et al., 2019), contextualized embeddings have become a crucial component of state-of-the-art QA systems. The next-word prediction and masked-language modelling tasks enable these models to gain language understanding, which can then be used by a QA system by transfer-learning. It is worth noting that QA models which heavily rely on the use of contextualized embeddings (Z. Zhang, Wu, Zhou et al., 2019, Z. Zhang, Wu, Zhao et al., 2019), have surpassed human performance on the SQuAD dataset (Rajpurkar et al., 2016). Although very effective, contextualized embeddings are out of scope for this research, due to their possible interference with the rest of the mechanisms and their heavy computational burden. Thus, the embeddings of this research are initialized with GloVe and fine-tuned during training.

Encoder: Modern QA systems process and understand the passage(s) and the query and learn to extract meaningful features. A standard choice for an encoder feature extractor is a Recurrent Neural Network (RNN), usually an LSTM (Hochreiter and Schmidhuber, 1997) or its lightweight version, a GRU (Chung et al., 2014). Apart from RNNs, a Convolutional Neural Network (CNN) (D. Chen, Bolton and Manning, 2016) can also be used to read and understand the inputs, as in Chaturvedi, Pandit and Garain, 2018, where a CNN encoder surpasses LSTM-based models in multiple-choice QA. To further increase the capabilities of the QA system, encoders can additionally combine an attention module (Bahdanau, Cho and Bengio, 2014). Since 2017, research has shifted towards fully attentional encoders, like the Transformer architecture (Vaswani et al., 2017), taking advantage of their ability to be parallelized (contrary to RNNs) and their global receptive field (contrary to CNNs). Furthermore, QA-Net (A. W. Yu et al., 2018) utilizes both the local features of CNNs and global features from multiple transformer layers, which enabled their model to achieve significant improvements in the SQuAD dataset. The encoders used in this research are transformer-based, which are shared or not between the sequences of the input side.

Passage-Query Interaction: The interaction between passage and query, is necessary for fusing information and identifying the parts of the passage that are necessary to answer the query. Once again, an attention module is adopted for this purpose, which can be either uni-directional, where the query informs a passage, or bi-directional where additionally, the passage informs the query. The Bi-Directional Attention Flow (BiDAF) (Seo et al., 2016) and the Dynamic Coattention Module (DCN) (Xiong, Zhong and Socher, 2016) are typical examples of the later, where passage-query similarity matrix is normalized across different dimensions to get two distinct attention matrices that update the representations for the context and the query. The use of multiple interactions between passage and query has also been proven effective, with implementations like DCN+ (Xiong, Zhong and Socher, 2017), that utilizes the co-attention module in two parts of the encoder or the Reinforced Mnemonic Reader (Hu, Peng and Qiu, 2017), that re-attends multiple times to past attentions, addressing the problems of attention redundancy and attention deficiency. For this research, the passage-query interactions are modelled with a Dual Attention (Nishida et al., 2019), which is a variant of BiDAF and DCN. Additionally, for the Style-transfer task, the interactions between three sequences are modelled by a fusion of three Dual Attention over the three possible sequence pairings, which takes place at two points during encoding, making it multi-hop.

Answering Type: The type of answer for a QA task can be distinguished in single-word prediction, multiple option selection, span extraction or free-form answer generation. Single-word prediction problems require the model to find the most probable word in the passage(s), that complete or fill a gap in the query. Pointer-Networks (Vinyals, Fortunato and Jaitly, 2015) have been successfully employed for the task, by defining a probability distribution over the tokens in the passage(s) (Kadlec et al., 2016). In multiple-option selection, a model reads and understands the passage(s) and query and selects the most probable answer from a given set of options. Robust methods for tackling this task usually include a similarity measure between a query-updated passage representation and the available options (Chaturvedi, Pandit and Garain, 2018), (Z. Chen et al., 2019). The span extraction answering task adds another level of complexity to single-word prediction, where the model has to predict the most probable span in the passage instead of a word. Again, Pointer-Networks are powerful methods for solving the problem, by defining two probability distributions over the passage, to infer the most probable start and end points of the answer span (S. Wang and Jiang, 2016, Xiong, Zhong and Socher, 2016). Finally, free-form answering is considered the most challenging task, where the answer is not necessarily part of a span in the passage and has to be generated from a learned vocabulary, token-by-token. An early approach to the task is the S-NET (Tan et al., 2017), where the model first extracts possible spans from the passages and the query using bidirectional GRUs and Pointer-Networks. The possible answer spans serve as an input to a decoder that synthesizes them into a single answer. This two-steps approach enabled S-NET to reach human-level performance in the QnA task of MS-MARCO. Pointer-Generator Networks (See, P. J. Liu and Manning, 2017) have also been successfully applied for free-form answering, where at each decoding step, the model can either copy a token from the inputs or generate a novel one from a learned vocabulary. This research deals with the latter answering type, which is approached by language generation, where a transformer decoder employs a multi-source pointer-generator, allowing the copying of tokens from multiple sources or the generation of novel tokens from the vocabulary.

2.2 Multi-task Learning and Ranking

Multi-task learning has shown great promise in combining several NLP tasks, using the high relatedness among them that provides an inductive bias, forcing models to learn more generally useful representations (Mitchell, 1980). Lately, this has become a quite common practice, with many applications in unified classic language tasks, like dependency parsing, Part-of-Speach (POS) tagging, Named Entity Recognition (NER) and inference (Collobert et al., 2011), (Hashimoto et al., 2016). The Natural Language Decathlon (DecaNLP) (McCann et al., 2018) has shown the levels at which multi-task learning is useful NLP problems, by jointly training a model that performs 10 tasks at once, including translation, question answering and summarization.

For question answering, multi-task learning has been effectively used in Y. Wang et al., 2018, where their QA model jointly learns to predict the answer span, the answer content and the cross-passage answer verification. More specifically, the answer span module predicts possible boundaries using a Pointer-Network, and the answer content module identifies the words in these spans that should be included in the answer. Finally, the cross-passage answer verification module, enables information exchange between the candidates, in order to verify or not each other, by obtaining passage-specific verification scores. The verification scores are subsequently used to rank the potential candidates and augment the token selection process. The joint learning of these three tasks enabled their model to surpass other QA models, like S-NET, in terms of both ROUGE-L and BLEU-1 for the MS-MARCO dataset. They additionally show how a heuristic-based passage ranker can significantly enhance the performance of BiDAF baseline in DuReader (W. He et al., 2018) and thus incorporate it into their final model to gain a further increase in terms of bleu-4 and ROUGE-L.

The Deep Cascade model (Yan et al., 2019) tackles the problem MRC over extensive text collections, achieving high results in the TriviaQA (Joshi et al., 2017) and DuReader datasets. They follow a three-step process, in which their parameters are shared and jointly learned. Firstly, a module ranks the available documents to their relevance with the query using a pointwise approach with traditional retrieval metrics, like BM25 and TF-IDF. Following, an XGBoost (T. Chen and Guestrin, 2016) pointwise ranker selects the most relevant paragraphs from the ranked documents of the previous stage. Finally, a Pointer-Network identifies the most probable answer from the ranked paragraphs.

In Reinforced Ranker-Reader (R^3) (S. Wang, M. Yu et al., 2017), they jointly train a reader and a ranker using reinforcement learning. The ranker module produces a probability distribution over the passages, using a softmax normalization over their concatenated representations. Subsequently, the reader module employs REINFORCE (Sutton et al., 2000) to select a passage based on the computed distribution and then selects an answer span from it. Masque (Nishida et al., 2019) also uses a pointwise ranker that shares the encoder with the rest of the QA model to obtain relevance probability for each passage. The relevance probabilities are passed on to the decoder, augmenting the generation process and preventing it from attending to irrelevant passages.

Contrary to the techniques above, the ranker proposed in this thesis solves a pairwise ranking task and then translates the pairwise comparison results to relevance probabilities for each passage. Furthermore, it combines the passage representations at a lower level than the R^3 model, by using a transformer layer to enable passage-to-passage information exchange. This idea is also explored in the Hierarchical Transformer (Y. Liu and Lapata, 2019) for multi-document summarization. They propose a Global Transformer layer to share information between passages and thus obtain richer representations.

2.3 Multi-style Learning and Style Transfer

Multi-style training was first proposed for Neural Machine Translation (NMT) (Johnson et al., 2016), where artificial tokens specific to each language control the output language of the translation, achieving state-of-the-art results at the time. Artificial tokens have additionally been used in NMT to enforce various constraints in the target sequences (Sennrich, Haddow and Birch, 2016), or control politeness (Takeno, Nagata and Yamamoto, 2017). Using artificial tokens and multi-style training was introduced into the field of question answering in Nishida et al., 2019, where they use two styles, an extractive and an abstractive to train an encoder-decoder transformer in the MS Marco dataset. They furthermore achieve state-of-the-art results the NarrativeQA dataset (Kočiský et al., 2017), by fine-tuning their model with the use of a separate style for the examples in it.

Style transfer is a concept explored in the context of formality transfer, where given an informal sentence, a model produces its formal counterpart. In Y. Zhang, Ge and Sun, 2020, they investigate three different approaches to the task, namely back translation, formality discrimination, and multi-task transfer. Back translation is widely used in NMT, where a sequence-to-sequence model produces synthetic parallel sentences to augment the translation data. A formality discriminator CNN-based model identifies whether an informal sentence has become formal after a round-trip translation, and appends it to an augmented dataset. Finally, multi-task transfer uses data from a Grammatical Error Correction (GEC) dataset to teach a sequence-to-sequence model how to translate informal sentences that contain grammatical errors to formal ones, by fixing their errors. Pre-training on the style-transferred augmented data created by these techniques and then fine-tuning on the original data improved the models' performance on the GYAFC dataset (Rao and Tetreault, 2018). In this research, similarly to Y. Zhang, Ge and Sun, 2020, a sequence-to-sequence learns how to translate extractive-style answers to abstractive ones and is subsequently used to augment the MS-MARCO dataset with synthetic abstractive answers that are used together with the original data during training.

Chapter 3

Background

3.1 Attention Mechanisms

Attention has revolutionized how sequence-to-sequence models work by effectively modeling in- and cross-sequence interactions. This research's methods rely heavily on attention mechanisms to model the interactions between three different types of sequences, namely, questions, passages, and answers. In this section, four essential attention mechanisms are presented, the Additive Attention (Bahdanau, Cho and Bengio, 2014), Bidirectional Attention Flow (Seo et al., 2016), Dynamic Co-attention (Xiong, Zhong and Socher, 2016) and Multi-Head Attention (Vaswani et al., 2017).

3.1.1 Additive Attention

The Additive attention (Bahdanau, Cho and Bengio, 2014) between a sequence representation $M^x \in \mathbb{R}^{\ell_x \times d}$ and the i-th token representation of a sequence y, $M_i^y \in \mathbb{R}^d$ is used to obtain vectors $c_i^x \in \mathbb{R}^d$ and $\alpha_i^x \in \mathbb{R}^{\ell_x}$. The vector c_i^x corresponds to the context of the i-th token in y informed by the the entirety of x, while α_i^x are the attention weights of the i-th token in y, which define a probability distribution over the tokens of sequence x.

Two linear layers map the sequence representation of x to the key $K \in \mathbb{R}^{\ell_x \times d}$ and the i-th token representation of y to the query[1] $Q \in \mathbb{R}^d$.

[1] This is a standard notation for the attention, not to be confused with the actual query in the question answering framework.

$$K = M^x \cdot W^K \tag{3.1}$$

$$Q = M^y \cdot W^Q \tag{3.2}$$

, where $W^K, W^Q \in \mathbb{R}^{d \times d}$ are learnable parameters.

In order to obtain the attention weight of the i-th token in y, an energy vector is computed via a non-linear transformation. Then a softmax normalization is applied across the sequence length dimension of x. Thus, for the i-th token in y and for the j-th token in x:

$$e_{ij} = \tanh(K_j + Q_i) \cdot w^e + b^e \tag{3.3}$$

$$\alpha_j^x = \text{softmax}(e_i) \tag{3.4}$$

, where $e_i \in \mathbb{R}^{\ell_x}$ is the energy vector, $\tanh(\cdot)$ is the hyperbolic tangent function and $w^e, b^e \in \mathbb{R}^d$ are learnable parameters.

Finally, the context vector $c_i^x \in \mathbb{R}^d$ is obtained by the inner product of the sequence representation and the attention weights.

$$c_i^x = (M^x)^T \cdot \alpha_i^x \tag{3.5}$$

, where T is the transpose operator.

Apart from the context, the attention weights, are also of use for modules outside of the Additive Attention. Thus, Additive Attention is defined as:

$$c_i^x, \alpha_i^x = \text{AddAttn}(M^x, M_i^y) \tag{3.6}$$

3.1.2 Bi-Directional Attention Flow

For sequence representations $H^x \in \mathbb{R}^{\ell_x \times d}$, $H^y \in \mathbb{R}^{\ell_y \times d}$, a Bi-Directional Attention Flow (BiDAF) module is used to fuse information from x to y and from y to x.

This process is especially useful for tasks where more than one sequence has to be encoded, as in the task of Question Answering. The advantage of BiDAF is that the bidirectionally updated representations originate from a common similarity matrix $U \in \mathbb{R}^{\ell_x \times \ell_y}$, in which, the element U_{ij} denotes the similarity between the i-th token in x and the j-th token in y.

$$U_{ij} = [H_i^x, H_j^y, H_i^x \odot H_j^y] \cdot w^u \tag{3.7}$$

, where $[\cdot, \cdot, \cdot]$ denotes horizontal concatenation, $\odot$ is the element-wise multiplication operator and $w^u \in \mathbb{R}^{3d}$ is a learnable parameter.

Following, the x-to-y attention weights $A \in \mathbb{R}^{\ell_x \times \ell_y}$ are obtained by a softmax normalization across the columns of U. Accordingly, the context vectors $\bar{H^y} \in \mathbb{R}^{\ell_x \times d}$ of x are obtained via a matrix multiplication.

$$A = \text{softmax}_{col}(U) \tag{3.8}$$

$$\bar{H^y} = A \cdot H^y \tag{3.9}$$

The y-to-x attention weights $b \in \mathbb{R}^{\ell_x}$ define a probability distribution of sequence y over each token in x. Using these attention weights, the updated context vectors $\bar{H^x} \in \mathbb{R}^{\ell_x \times d}$ are obtained, where for each token i:

$$b = \text{softmax}\Big(\max_{col}(U^T)\Big) \tag{3.10}$$

$$\bar{H}_i^x = \sum_i^{\ell_x} b_i \cdot H_i^x \tag{3.11}$$

Finally, the new representations are combined by vertical concatenation to yield the bidirectionally informed representation of the x sequence, $G \in \mathbb{R}^{\ell_x \times 4d}$.

$$G = \Big[H^x, \bar{H^y}, H^x \odot \bar{H^y}, H^x \odot \bar{H^x}\Big] \tag{3.12}$$

3.1.3 Dynamic Co-Attention

Dynamic Co-Attention Networks (DCN) have been used similarly to BiDAF, to model the interactions between two sequences x and y in the encoder. For sequence representations $H^x \in \mathbb{R}^{\ell_x \times d}$, $H^y \in \mathbb{R}^{\ell_y \times d}$, a similarity matrix $U \in \mathbb{R}^{\ell_x \times \ell_y}$ is computed as their dot product and the attention weights $A^y \in \mathbb{R}^{\ell_x \times \ell_y}$, $A^x \in \mathbb{R}^{\ell_y \times \ell_x}$ are obtained via normalization across rows and columns.

$$U = H^x \cdot (H^y)^T \tag{3.13}$$

$$A^y = \text{softmax}_{col}(U) \tag{3.14}$$

$$A^x = \text{softmax}_{col}(U^T) \tag{3.15}$$

Then context for the y sequence $C^y \in \mathbb{R}^{\ell_y \times d}$ can be computed by the dot-product of the attention weights and the sequence representation and the updated representation $\bar{H}^y \in \mathbb{R}^{\ell_y \times d}$ is the concatenation of the pre-DCN and x-attended representations.

$$C^y = M^x \cdot A^y \tag{3.16}$$

$$\bar{H}^y = \Big[H^y, C^y\Big] \tag{3.17}$$

Following, the context for the x sequence, $C^x \in \mathbb{R}^{\ell_x \times 2d}$ is computed similarly and the updated representation $\bar{H}^x \in \mathbb{R}^{\ell_x \times 3d}$ is obtained via concatenation.

$$C^x = M^y \cdot \bar{H}^y \tag{3.18}$$

$$\bar{H}^x = \Big[H^x, C^x\Big] \tag{3.19}$$

3.1.4 Multi-head Attention

The multi-head attention (MHA) module is the key component of the Transformer, where features are extracted from multiple subspaces of the input. Multi-head attention can model interactions between a sequence and itself or between two different sequences. Following is an overview of the more general case that deals with two sequences.

A sequence $H^x \in \mathbb{R}^{\ell_x \times d_x}$ is projected to via linear transformations to two different representations, $K \in \mathbb{R}^{\ell_x \times d_z}$ and $V \in \mathbb{R}^{\ell_y \times d_z}$, called key and value, where ℓ_x is the sequence length for x, d_x is the feature dimension of x and d_z is the attention dimensionality. Another sequence $H^y \in \mathbb{R}^{\ell_y \times d_y}$ is projected to a representation $Q \in \mathbb{R}^{\ell_y \times d_z}$, which is called query, through another linear transformation.

$$K = H^x \cdot W^K \tag{3.20}$$

$$V = H^x \cdot W^V \tag{3.21}$$

$$Q = H^y \cdot W^Q \tag{3.22}$$

, where $W^K, W^V \in \mathbb{R}^{d_x \times d_z}$ and $W^Q \in \mathbb{R}^{d_y \times d_z}$ are learnable parameters.

Each of the three representations is separated into h heads, by equally splitting the feature dimension, thus obtaining $K_i, V_i \in \mathbb{R}^{\ell_x \times d_{head}}$, and $Q_i \in \mathbb{R}^{\ell_y \times d_{head}}$, where $d_{head} = d_z/h$. The scaled dot-product attention is calculated, where each token in Q can attend to each token V, in h different subspaces.

$$\text{AttnHead}_i = \text{softmax}_{col}(\frac{Q_i \cdot K_i^T}{\sqrt{d_{head}}}) \cdot V_i \tag{3.23}$$

, where $\text{AttnHead}_i \in \mathbb{R}^{\ell_y \times d_{head}}$..

The modified representations of the h subspaces are combined into a final representation, by concatenation and another linear transformation.

$$\bar{H^y} = \Big[\text{AttnHead}_1; \ldots; \text{AttnHead}_h\Big] \cdot W^o \tag{3.24}$$

, where $[\cdot; \ldots; \cdot]$ indicates vertical concatenation and $W^o \in \mathbb{R}^{d_z \times d_y}$ is a learnable parameter.

There are three different use cases of Multi-head Attention in the Transformer.

- **Encoder Self-Attention**: In this case, a sequence representation in the Encoder H^x, can attend to itself, and thus the key, value, and query, all come from the same sequence.

$$\bar{H}^x = \overset{enc}{\text{MHA}}\,(H^x, H^x) \tag{3.25}$$

- **Decoder Self-Attention**: A sequence representation in the decoder H^y, attends to itself, in a similar way as the Encoder Self-Attention. A process called masking is additionally applied to prevent left-to-right information flow. Thus, a token can attend to all the tokens until and inclusive of its position. Masking is applied in the scaled dot-product attention of each head, before the softmax normalization, by setting the corresponding scores to $-\infty$.

$$\bar{H}^y = \overset{dec}{\text{MHA}}\,(H^y, H^y) \tag{3.26}$$

- **Encoder-Decoder Attention**: The final representation of an encoder sequence M^x is attending to a decoder sequence representation H^y. This case is the most general one, where x gives rise to the key and value, while y serves as the query.

$$\bar{H}^y = \overset{enc-dec}{\text{MHA}}\,(M^x, H^y) \tag{3.27}$$

3.2 Transformers

The success of Transformer architectures (Vaswani et al., 2017) is attributed mostly to their ability to handle sequence modeling on parallel by making use of attention mechanisms. The Transformer can be divided into two parts, the encoder, responsible for processing the input sequence, and the decoder, which, given the encoded representation of the input sequence, is responsible for decoding the output sequence. Before going through the encoder and decoder parts, it is useful to explain three position-wise modules that are at use.

3.2.1 Position-wise Modules

- **Positional Embeddings**

  Due to the non-autoregressive nature of the Transformer, it is necessary to provide information about each token's position in the sequence through the use of positional embeddings. The sinusoidal encoding is a common practice, where the relative position is encoded as a sine or cosine function of the position and dimension of the embedding. For a sequence x of length ℓ_x, the positional embedding matrix for a sequence $E^{pos} \in \mathbb{R}^{\ell_x \times d_{emb}}$

$$E^{pos}_{i,j} = \begin{cases} sin(i/10000^{j/d_{emb}}), \text{if } j \text{ is even} \\ cos(i/10000^{j/d_{emb}}), \text{if } j \text{ is odd} \end{cases} \tag{3.28}$$

, where d_{emb} is the dimensionality of the embeddings. The positional embeddings E^{pos} are added to the word embeddings, providing information about each token's position in the sequence.

- **Feed-forward Networks**

  The feed-forward networks in the Transformer have one hidden layer, with a non-linear activation, and are position-wise, treating each feature independently. This module is used identically in both encoder and decoder transformer layers. For a feature vector $x \in \mathbb{R}^d$, the position-wise feed-forward network performs the following operation.

$$\text{FFN}(x) = f(x \cdot W^{in} + b^{in}) \cdot W^{out} + b^{out} \tag{3.29}$$

  , where $f(\cdot)$ is a non-linear activation function, $W^{in} \in \mathbb{R}^{d \times d_h}$, $W^{out} \in \mathbb{R}^{d_h \times d}$, $b^{in} \in \mathbb{R}^{d_h}$, $b^{out} \in \mathbb{R}^d$.

- **Layer Normalization**

  Normalization techniques are usually used in large deep neural networks to normalize the activities of each neuron in a layer and ensure a more efficient and stable training. Layer Normalization (J. L. Ba, Kiros and Hinton, 2016) is the go-to technique for Transformer architectures. For each neuron, the layer normalization module encodes the distribution of its inputs by using adaptive parameters γ and β. For a feature vector $x \in \mathbb{R}^d$, Layer normalization performs the following operation.

$$\mathrm{LN}(x) = \frac{x - \mathbb{E}[x]}{\sqrt{\mathbb{V}[x] + \epsilon}} \odot \gamma + \beta \tag{3.30}$$

, where the gain, $\gamma \in \mathbb{R}^d$ and the bias, $\beta \in \mathbb{R}^d$ are learnable parameters, $\mathbb{E}[\cdot]$ is the average operator, $\mathbb{V}[\cdot]$ is the variance operator and $\epsilon \in \mathbb{R}$ is a small constant introduced for numerical stability.

3.2.2 Encoder

A transformer encoder is made up from several transformer encoder layers, where each layer has two sub-layers. The first is a multi-head attention module (Section 3.1.4) that extracts features from different parts of the input space, followed by a feed-forward network (Section 3.29). After each sublayer, a residual connection (K. He et al., 2015) and a layer normalization (Section 3.30) are applied. Thus, the i-th transformer encoder layer, for the sequence representation $H^{x^{(i-1)}} \in \mathbb{R}^{\ell_x \times d}$ of the previous layer, can be defined as:

$$\bar{H}^{x^{(i)}} = \mathrm{LN}^{(a)}\Big(\overset{enc}{\mathrm{MHA}} \left(H^{x^{(i-1)}}, H^{x^{(i-1)}} \right) + H^{x^{(i-1)}} \Big) \tag{3.31}$$

$$H^{x^{(i)}} = \mathrm{LN}^{(b)}\Big(\mathrm{FFN}\big(\bar{H}^{x^{(i)}}\big) + \bar{H}^{x^{(i)}}\Big) \tag{3.32}$$

, where the input for the first transformer encoder layer is the output of the embedding layer and thus $H^{x^{(0)}} = E^x$.

3.2.3 Decoder

A transformer decoder layer has a similar structure as the encoder. Each decoder layer is comprised of three sub-layer. First, a decoder self-attention module (Eq. 3.26), followed by an encoder-decoder attention module (Eq. 3.27) and lastly a feed-forward network (Eq. 3.29). Again, residual connections and layer normalization (Eq. 3.30) are applied after each sublayer. Thus, for the output of the encoder $M^x \in \mathbb{R}^{\ell_x \times d}$ and the result of the previous decoder layer $H^{y^{(j-1)}} \in \mathbb{R}^{\ell_y \times d}$ the j-th decoder layer is defined as:

$$\bar{H}^{y^{(i)}} = \mathrm{LN}^{(a)}\Big(\overset{dec}{\mathrm{MHA}} \left(H^{y^{(i-1)}}, H^{y^{(i-1)}} \right) + H^{y^{(i-1)}} \Big) \tag{3.33}$$

$$\bar{\bar{H}}^{y^{(i)}} = \mathrm{LN}^{(b)}\Big(\overset{enc-dec}{\mathrm{MHA}} \left(M^x, \bar{H}^{y^{(i)}}\right) + \bar{H}^{y^{(i)}}\Big) \tag{3.34}$$

$$H^{y^{(i)}} = \mathrm{LN}^{(c)}\Big(\mathrm{FFN}\Big(\bar{\bar{H}}^{y^{(i)}}\Big) + \bar{\bar{H}}^{y^{(i)}}\Big) \tag{3.35}$$

, where the input of the first layer, $H^{y^{(0)}}$ is the embeddings of the shifted-right y sequence. The final output of the transformer decoder is projected with a linear layer and a softmax normalization to a probability distribution over the tokens in the vocabulary.

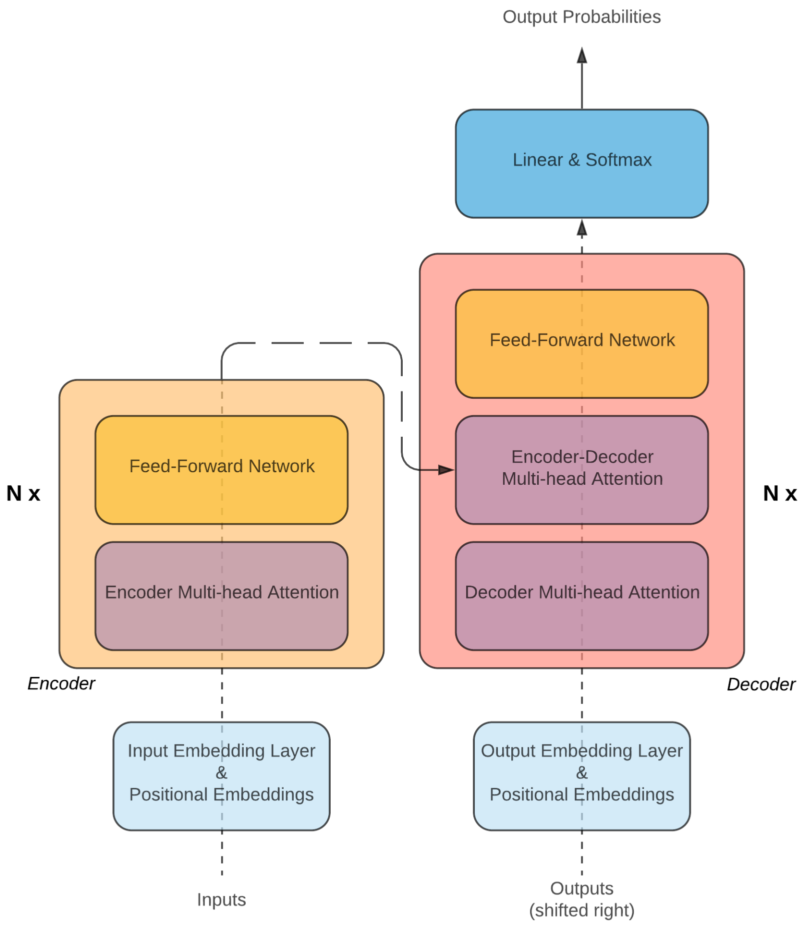

Output Probabilities
Linear & Softmax
Feed-Forward Network
Encoder-Decoder Multi-head Attention
Decoder Multi-head Attention
Feed-Forward Network
Encoder Multi-head Attention
N x
N x
Encoder
Decoder
Input Embedding Layer & Positional Embeddings
Output Embedding Layer & Positional Embeddings
Inputs
Outputs (shifted right)

Figure 3.1: The Transformer architecture.

3.3 Pointer-Generators

Pointer-Generator Networks (See, P. J. Liu and Manning, 2017) bridge the gap between extractive and abstractive language generation techniques, where for each position t in the target sequence, the model can either generate a token from the vocabulary or copy a token from the input sequence. The copying mechanism enables the inclusion of out-of-vocabulary (OOV) tokens in the output sequence, by operating on an extended vocabulary V^{ext}, which contains the tokens in the fixed vocabulary V^{fixed} and the tokens of the input sequence. Thus, the extended vocabulary is dynamically defined for each input sequence x and $V^{ext} \supset V^{fixed}$. Given the representation of the input sequence $M^x \in \mathbb{R}^{\ell_x \times d}$, coming from an encoder module and a representation of the t-th token in the target sequence $M_t^y \in \mathbb{R}^d$, an additive attention module (Section 3.1.1) is used to obtain a context vector $c_t^x \in \mathbb{R}^d$ and attention weights $a_t^x \in \mathbb{R}^{\ell_x}$.

$$c_t^x, a_t^x = \text{AddAttn}(M^x, M_t^y) \tag{3.36}$$

The attention weights define a probability distribution over the token positions in the input sequence, and the context vector is a summary of the input sequence information for the t-th position in the target sequence. The attention weights can be mapped into probabilities over the extended vocabulary by a dot-product with the one-hot encoded representation of the input sequence in the extended vocabulary $S^x_{(ext)} \in \{0,1\}^{\ell_x \times |V^{fixed}|}$. The probability distribution over the fixed vocabulary is obtained via a linear mapping and a softmax normalization.

$$P^{copy}(y_t) = a_t^x \cdot S^x_{(ext)} \tag{3.37}$$

$$P^{gen}(y_t) = \text{softmax}\Big(M_t^y \cdot (W^{out} + b^{out})\Big) \tag{3.38}$$

, where $W^{out} \in \mathbb{R}^{d \times |V^{fixed}|}$ and $b^{out} \in \mathbb{R}^{|V^{fixed}|}$ are learnable parameters.

To combine the two distributions, are probability $p_{gen} \in [0,1]$ is calculated by using the context of the source sequence and the representation t-th token of the target sequence.

$$p_{gen} = \text{sigmoid}\Big([c_t^x, M_t^y] \cdot w^{gen} + b^{gen}\Big) \tag{3.39}$$

, where $w^{gen} \in \mathbb{R}^{2d}$, $b^{gen} \in \mathbb{R}$ are learnable parameters.

Then, the final distribution over the extended vocabulary can be obtained by:

$$P^{final}(y_t) = p_{gen} \cdot P^{gen}(y_t) + (1 - p_{gen}) \cdot P^{copy}(y_t) \tag{3.40}$$

Chapter 4

Methodology

In this chapter, the Masque model (Nishida et al., 2019), as well as the new methods proposed in this research are introduced in more detail[1].

4.1 Masque

Masque (Nishida et al., 2019) is a sequence-to-sequence, encoder-decoder transformer (Vaswani et al., 2017) that additionally combines a Pointer-Generator (See, P. J. Liu and Manning, 2017). It uses multi-task learning by sharing the encoder part with a passage ranker and an answerability classifier and multi-style learning by sharing the whole model with two answering styles. The naming convention of the two styles from Nishida et al., 2019 is adopted, where QA refers to the extractive style and NLG to the abstractive one. The model is tasked with generating an answer, a probability that the query is answerable, and a relevance probability for each of the K passages, by using the query, the K passages, and a specified answer style. More formally, it maximizes the conditional probability $P\Big(y, \alpha, \{r\}_1^K \Big| q, \{p\}_1^K, s\Big)$, where:

- $y \in \mathbb{N}^T$ is the answer, represented by a sequence of tokens in the vocabulary with length T
- $\alpha \in \{0, 1\}$ is the answerability label, with 0 indicating that the answer is not answerable and 1 indicating that it is answerable
- $\{r\}_1^K$ are the relevance labels for each one of the K passages, where $r_k \in \{0, 1\}$ is the relevance of the k-th passage with respect to the query, with 0 indicating the non-relevant label and 1 the relevant label

[1] The code for all models, including the implementation of Masque will be available at https://github.com/johntsi/preast_qa

- $q \in \mathbb{N}^J$ is the query, represented by a sequence of tokens in the vocabulary with length J
- $\{p\}_1^K$ are K passages, with $p_k \in \mathbb{N}^{L_k}$ representing the k-th passage, with a sequence of L_k tokens in the vocabulary
- $s \in \{0, 1\}$ is the style label, with 0 indicating the QA style and 1 indicating the NLG style

Masque can be separated into five essential components (Figure 4.1). The implementation of Nishida et al., 2019 is closely followed, with the most notable difference between the two implementations lying in the Embedder (Section 4.1.1).

- The **Embedder** produces an embedding for a given one-hot encoded sequence and is shared among passages, queries, and answers.
- The **Encoder** combines three transformer encoders and a dual attention module that fuses information from each passage to the query and from the query to each passage. It produces a representation for the query and each passage in an example, given their embeddings.
- The **Rassage Ranker** generates a relevance probability $\beta_k \in \mathbb{R}$, given the representation of the k-th passage from the encoder. Thus, for the k-th passage, it maximizes the conditional probability $P\Big(r_k|p_k, q\Big)$
- The **Answerability Classifier** generates a probability that the query is answerable, given the K passage representations from the encoder, by maximizing $P\Big(a|\{p\}_1^K, q\Big)$
- The **Decoder** combines a transformer decoder and a multi-source pointer generator that mixes several distributions to obtain the final distribution for the t-th position in an answer. It uses the K passages and query representations, the relevance probabilities, and the answer embeddings until position $t-1$. The decoder is furthermore conditioned on the required answering style, indicated by a special token at the beginning of the answer. Thus, for position t in the answer, it maximizes the conditional probability $P\Big(y_t|\{p\}_1^K, q, y_{t-1}, \ldots, y_1, s\Big)$.

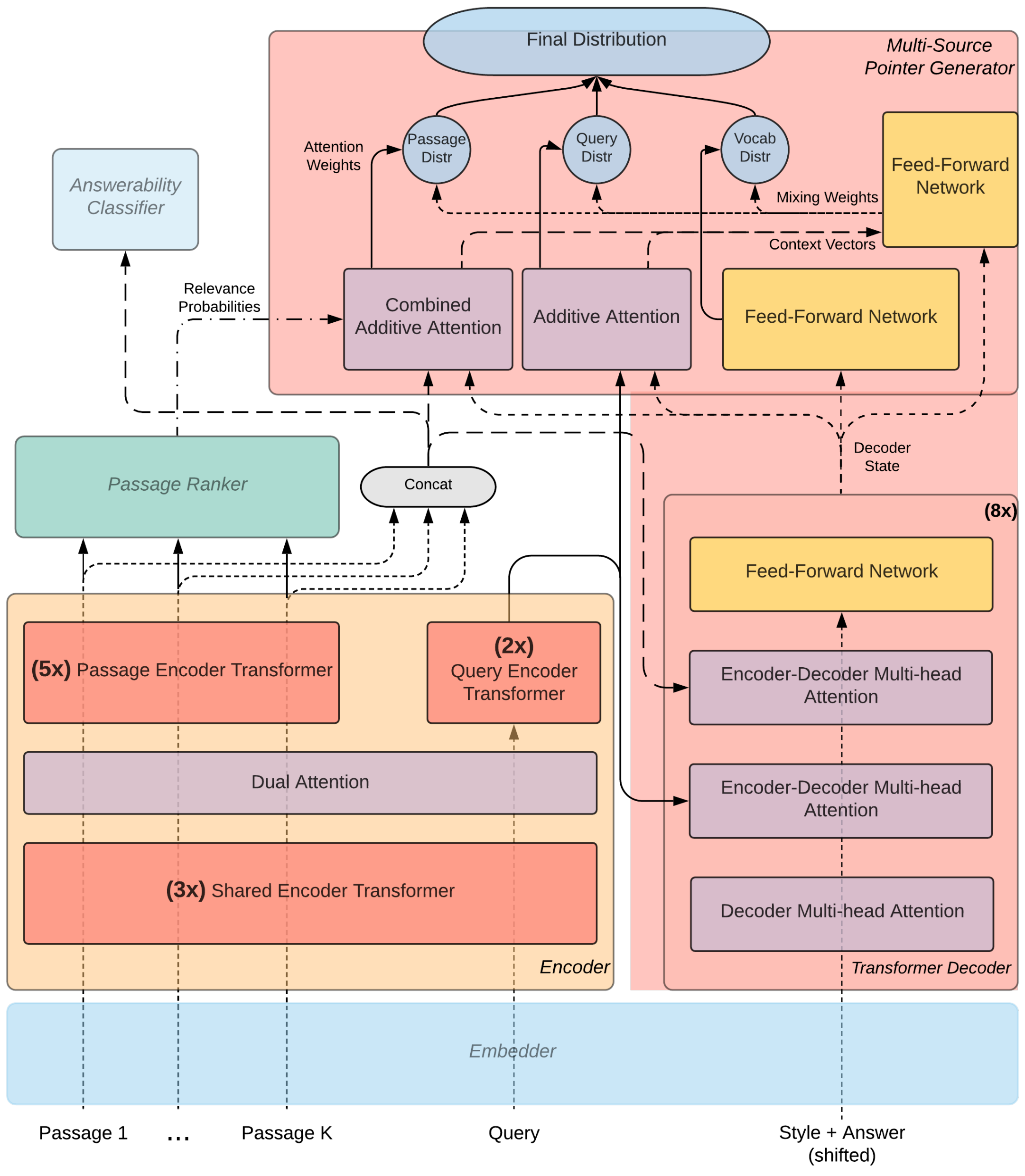


Figure 4.1: The Masque architecture.

4.1.1 Embedder

The embedder module is common for all types of sequences. Thus for an arbitrary sequence x, which is either a passage p_k, a query q or an answer y, its one-hot encoded representation, $S^x_{(fixed)} \in \{0,1\}^{\ell_x \times |V^{fixed}|}$ is mapped to an embedding matrix

$E^x \in \mathbb{R}^{\ell_x \times d_{emb}}$, where $|V^{fixed}|$ is the size of the fixed vocabulary, ℓ_x is the length of the sequence and d_{emb} is the dimensionality of the embeddings.

$$E^x = S^x_{(fixed)} \cdot W^{emb} + E^{x,pos} \tag{4.1}$$

, where $W^{emb} \in \mathbb{R}^{|V^{fixed}| \times d_{emb}}$ is the learnable embedding projection matrix and $E^{x,pos} \in \mathbb{R}^{\ell_x \times d_{emb}}$ is the positional embedding matrix, as defined in equation 3.28.

Here lies the most notable difference between the implementation of Masque in this research and Nishida et al., 2019. In this implementation, the embeddings are 300-dimensional vectors, initialized with GloVe (Pennington, Socher and Manning, 2014), and fine-tuned during training. The original implementation fuses the GloVe embeddings with the 512-dimensional ELMo embeddings (Peters et al., 2018) with a 2-layer Highway Network (R. K. Srivastava, Greff and Schmidhuber, 2015), adding approximately 3 million[2] more parameters to the model. Contextualized embeddings contribute to Masque's success, but they are omitted from this research's implementation for two main reasons. Firstly, although they contribute to increased model performance, they introduce outside information to the task and possibly interfere with the effect of the investigated methods. Secondly, they rely on large, computationally heavy models, which significantly increase training time and required resources. Finally, in Nishida et al., 2019, due to the use of ELMo embeddings, no positional encoding is needed.

4.1.2 Encoder

The Encoder is used to obtain the representation $M^{p_k} \in \mathbb{R}^{L \times d}$ for the k-th passage and $M^q \in \mathbb{R}^{J \times d}$ for the query. The embeddings of the passages and the query are passed through a shared transformer encoder to extract universal features. Following, a dual attention module fuses information from the query to each passage, and from all the passages to the query.

For a passage representations $H^{p_k} \in \mathbb{R}^{L_k \times d}$ and query representation $H^q \in \mathbb{R}^{J \times d}$

[2]It is not clear whether the highway network is shared between all the sequences. However, most probably, a different one is used for the decoder, making it in total four layers of size 812, which translates to $4 \times 812 \times 812$ more parameters. Additionally for each of the shared transformer encoder and the transformer decoder the initial mapping to d-dimensional vectors, thus additionally using another $2 \times 512 \times 304$ parameters.

obtained from the shared transformer encoder, a similarity matrix $U^k \in \mathbb{R}^{L_k \times J}$ is computed, as in BiDAF (Section 3.1.2). Thus, the similarity between the l-th token in the k-th passage and for the j-th token in the query is:

$$U^k_{lj} = [H^{p_k}_l, H^q_j, H^{p_k}_l \odot H^q_j] \cdot w^{dual} \tag{4.2}$$

, where $w^{dual} \in \mathbb{R}^{3d}$ is a learnable parameter.

The similarity matrix is normalized per columns and rows to obtain attention weights $A^k \in \mathbb{R}^{L_k \times J}$ and $B^k \in \mathbb{R}^{J \times L_k}$.

$$\begin{aligned} A^k &= \text{softmax}_{col}\Big(U^k\Big) \\ B^k &= \text{softmax}_{col}\Big((U^k)^T\Big) \end{aligned} \tag{4.3}$$

Finally, the bidirectionally informed representations for the K passages $G^{q \to p_k} \in \mathbb{R}^{L_k \times 5d}$ and for the query $G^{p \to q} \in \mathbb{R}^{J \times 5d}$ are obtained using dynamic co-attention (Section 3.1.3).

$$\bar{A}^k = A^k \cdot H^q \tag{4.4}$$
$$\bar{B}^k = B^k \cdot H^{p_k} \tag{4.5}$$
$$\bar{\bar{A}}^k = A^k \cdot \bar{B}^k \tag{4.6}$$
$$\bar{\bar{B}}^k = B^k \cdot \bar{A}^k \tag{4.7}$$

, where $\bar{A}^k, \bar{\bar{A}}^k \in \mathbb{R}^{L_k \times d}$ and $\bar{B}^k, \bar{\bar{B}}^k \in \mathbb{R}^{J \times d}$, which are then combined via horizontal concatenation as:

$$G^{q \to p_k} = \Big[H^{p_k}, \bar{A}, \bar{\bar{A}}, \bar{A} \odot H^{p_k}, \bar{\bar{A}} \odot H^{p_k}\Big] \tag{4.8}$$
$$G^{p \to q} = \Big[H^q, \bar{B}, \bar{\bar{B}}, \bar{B} \odot H^q, \bar{\bar{B}} \odot H^q\Big] \tag{4.9}$$

, where $\bar{B}, \bar{\bar{B}} \in \mathbb{R}^{J \times d}$ are the aggregated information from all the passages using a max function.

$$\bar{B} = \max_k \left(\bar{B}^k \right) \tag{4.10}$$
$$\bar{\bar{B}} = \max_k \left(\bar{\bar{B}}^k \right) \tag{4.11}$$

After the dual attention layer, the fused representations for the passages are passed through a transformer encoder, shared among all passages, and the query to a separate transformer encoder to obtain the final representations $M^{p_k} \in \mathbb{R}^{L_k \times d}$ and $M^q \in \mathbb{R}^{J \times d}$.

All the transformer encoders are as introduced in Section 3.2.2. Following the latest trends in transformer-like architectures in BERT (Devlin et al., 2018) and GPT-2 (Radford et al., n.d.), a GELU activation (Hendrycks and Gimpel, 2016) is used for the hidden layer of the feed-forward networks. Furthermore, for every transformer encoder, the input to the encoder is directly mapped to d-dimensional vectors with learnable parameters $W^{in} \in \mathbb{R}^{d_{in} \times d}$ and $b^{in} \in \mathbb{R}^d$. Thus, $H^{x^{(0)}}$ corresponds to the result of this linear transformation. The input to the shared transformer encoder is the output of the embedding layer, hence $d_{in} = d_{emb}$, while for the passage and query transformers it is the output of the dual attention, hence $d_{in} = 5d$.

4.1.3 Passage Ranker

In Nishida et al., 2019, obtaining relevance scores for each passage is approached as a pointwise ranking problem. For the k-th passage in an example, the pointwise ranker (PointRnk) receives as input the final encoder representation of the first token of each passage $M_1^{p_k} \in \mathbb{R}^d$, which corresponds to the [CLS] token. The [CLS] token is an artificial token, appended to the beginning of each passage and fine-tuned to gather task-specific information, such as the passage relevance. For the k-th passage in the example, the relevance probability $\beta^{p_k} \in \mathbb{R}$is obtained through a linear mapping with learnable parameter $w^r \in \mathbb{R}^d$ followed by a sigmoid function.

$$\beta^{p_k} = \text{sigmoid}(M_1^{p_k} \cdot w^r) \tag{4.12}$$

4.1.4 Answerability Classifier

The answerability classifier produces a probability $P(\alpha) \in \mathbb{R}$, indicating the confidence of the encoder that the query is answerable given the encoded passages. The classifier takes as input the horizontal concatenation of the K [CLS] token representation, which is a vector of dimensionality Kd, maps it to a scalar through a linear transformation and a sigmoid function.

$$P(\alpha) = \text{sigmoid}\Big(\Big[M_1^{p_1}, \ldots, M_1^{p_K}\Big] \cdot w^c\Big) \tag{4.13}$$

, where $w^c \in \mathbb{R}^{Kd}$ is a learnable parameter.

4.1.5 Decoder

The Decoder module is composed of a transformer decoder (Section 3.2.3) and a multi-source pointer generator (Section 3.3). It is used to obtain T probability distributions $P^{final}(y_t) \in \mathbb{R}^{|V^{ext}|}$ over the extended vocabulary for each position t in the answer, by utilizing the passages and query representations $M^{p_k} \in \mathbb{R}^{L \times d}$, $M^q \in \mathbb{R}^{J \times d}$ from the Encoder, the relevance scores $\beta \in \mathbb{R}^K$ from the ranker, as well as the answer embeddings $E^y \in \mathbb{R}^{T \times d_{emb}}$.

Similar, to the encoder transformer blocks, in the beginning of the decoder transformer, the answer embeddings are passed through a linear transformation to obtain $H^{y^{(0)}} \in \mathbb{R}^{T \times d}$.

$$H^{y^{(0)}} = E^y \cdot W^{dec} + b^{dec} \tag{4.14}$$

, where $W^{dec} \in \mathbb{R}^{d_{emb} \times d}$ and $b^{dec} \in \mathbb{R}^d$ are a learnable parameters.

A decoder layer is similar to the one introduced in section (Section 4.1.5), with an additional encoder-decoder attention to account for the extra input sequence. Thus, the i-th decoder layer is comprised of a multi-head attention module (Eq. 3.26) that performs masked self-attention to the output of the previous layer $H^{y^{(i-1)}}$. Following, there are another two multi-head attention modules (Eq. 3.27) that carry out encoder-decoder attention, first using the query representation and then using the passage representations. At the end of each decoder layer, there is a feed-forward network (Eq. 3.29) and, after each sublayer, a residual connection and layer normalization (Eq. 3.30) are applied.

$$\bar{H}^{y^{(i)}} = \mathrm{LN}^{(a)}\Big(\overset{dec}{\mathrm{MHA}} \Big(H^{y^{(i-1)}}, H^{y^{(i-1)}} \Big) + H^{y^{(i-1)}} \Big) \tag{4.15}$$

$$\bar{\bar{H}}^{y^{(i)}} = \mathrm{LN}^{(b)}\Big(\overset{enc-dec}{\mathrm{MHA}} \Big(M^q, \bar{H}^{y^{(i)}}\Big) + \bar{H}^{y^{(i)}} \Big) \tag{4.16}$$

$$\bar{\bar{\bar{H}}}^{y^{(i)}} = \mathrm{LN}^{(c)}\Big(\overset{enc-dec}{\mathrm{MHA}} \Big(M^{p_{cat}}, \bar{\bar{H}}^{y^{(i)}}\Big) + \bar{\bar{H}}^{y^{(i)}} \Big) \tag{4.17}$$

$$H^{y^{(i)}} = \mathrm{LN}^{(d)}\Big(\mathrm{FFN}\Big(\bar{\bar{\bar{H}}}^{y^{(i)}}\Big) + \bar{\bar{\bar{H}}}^{y^{(i)}} \Big) \tag{4.18}$$

, where $M^{p_{cat}} = [M^{p_1}; \ldots ; M^{p_K}] \in \mathbb{R}^{KL\times d}$.

A multi-source pointer generator combines the copy distributions of the concatenated passages $P^{p_{cat}}(y_t) \in \mathbb{R}^{|V^{ext}|}$ and the query $P^q(y_t) \in \mathbb{R}^{|V^{ext}|}$, as well as the fixed vocabulary distribution $P^v(y_t) \in \mathbb{R}^{|V^{fixed}|}$ to get a final distribution over the extended vocabulary.

The representation of the query and the t-th answer token are passed through an additive attention module (Section 3.1.1).

$$c_t^q, \alpha_t^q = \mathrm{AddAttn}^{(q)}(M^q, M_t^y) \tag{4.19}$$

, where $c_t^q \in \mathbb{R}^d$ is the context vector of the query for the t-th answer token and the attention weights $\alpha_t^q \in \mathbb{R}^J$ define a probability distribution over each position in the query.

A similar process is followed for the concatenated passages $M^{p_{cat}}$, where they are passed through a shared additive attention module:

$$c_t^{p_{cat}}, \alpha_t^{p_{cat}} = \mathrm{AddAttn}^{(p)}(M^{p_{cat}}, M_t^y) \tag{4.20}$$

, where $c_t^{p_{cat}} \in \mathbb{R}^d$ and $\alpha_t^{p_{cat}} \in \mathbb{R}^{KL}$.

To prevent the model from attending to tokens from irrelevant passages, the relevance probabilities β are used to modify the attention weights. Thus, before obtaining the context vector (Eq. 3.5), the alphas are re-normalized as:

$$\alpha_{ti}^{p_{cat}} = \frac{\alpha_{ti}^{p_{cat}} \cdot \beta_{k(i)}}{\sum_j \alpha_{tj}^{p_{cat}} \cdot \beta_{k(j)}} \tag{4.21}$$

, where $\beta_{k(i)}$ is the relevance probability of the passage in which the i-th token is part of.

To obtain the distribution over the fixed vocabulary $P^v(y_t) \in \mathbb{R}^{|V^{fixed}|}$, a linear layer projects the representation of the t-th answer token to a d_{emb}-dimensional vector, followed by another linear mapping to a vector dimensionality of $|V^{fixed}|$. Finally, a softmax operator transforms the vector to a probability distribution over the fixed vocabulary.

$$P^v(y_t) = \text{softmax}\Big((M_t^y \cdot W^1 + b^1) \cdot W^2\Big) \tag{4.22}$$

, where, $W^1 \in \mathbb{R}^{d \times d_{emb}}$, $b^1 \in \mathbb{R}^{d_{emb}}$ and $W^2 \in \mathbb{R}^{d_{emb} \times |V^{fixed}|}$ are learnable parameters.

Note that in Nishida et al., 2019 W^2 is tied with W^{emb} (Eq. 4.1), as proposed in Inan, Khosravi and Socher, 2016. Experiments with tying these parameters provided slightly worse results (Table A.2) and thus W^2 is not shared with W^{emb} of Eq. 4.1

The attention weights α_t^q and $\alpha_t^{p_{cat}}$ are used to define the probability distributions of the query $P^q(y_t) \in \mathbb{R}^{|V^{ext}|}$, and the concatenated passages $P^{p_{cat}}(y_t) \in \mathbb{R}^{|V^{ext}|}$ over the extended vocabulary.

$$P^q(y_t) = \alpha_t^q \cdot S_{(ext)}^q$$
$$P^{p_{cat}}(y_t) = \alpha_t^{p_{cat}} \cdot S_{(ext)}^{p_{cat}}$$

, where $S_{(ext)}^q \in \{0,1\}^{J \times |V^{ext}|}$ and $S_{(ext)}^{p_{cat}} \in \{0,1\}^{KL \times |V^{ext}|}$ are the one-hot encoded representations of the query and the concatenated passages in the extended vocabulary.

The two copy distributions and the fixed vocabulary distribution are combined with weight factors $\lambda_t^v, \lambda_t^q, \lambda_t^p \in \mathbb{R}$, which are computed using the context of the query, the combined context of the passages and representation of t-th answer token from the decoder.

$$\lambda_t^v, \lambda_t^q, \lambda_t^p = \text{softmax}\Big(\Big[M_t^y, c_t^q, c_t^{p_{cat}}\Big] \cdot W^\lambda + b^\lambda\Big) \tag{4.23}$$

, where $W^\lambda \in \mathbb{R}^{3d \times 3}$ and $b^\lambda \in R^3$ are learnable parameters.

The final distribution $P^{final}(y_t) \in \mathbb{R}^{|V^{ext}|}$ of the tokens in the extended vocabulary for the t-th answer token is:

$$P^{final}(y_t) = \lambda_t^v \cdot P^v(y_t) + \lambda_t^q \cdot P^q(y_t) + \lambda_t^p \cdot P^{p_{cat}}(y_t) \tag{4.24}$$

Note that the set the extended vocabulary, $V^{ext} \supseteq V^{fixed}$, is dynamically defined for each tuple $(S_{(ext)}^q, S_{(ext)}^{p_{cat}})$. Additionally, the probability of a token that is part of the extended vocabulary set ($y_t \in V^{ext}$), but is not part of the fixed one ($y_t \notin V^{fixed}$), is equal to zero in the fixed vocabulary distribution $P^v(y_t)$ for all positions t in the answer.

4.1.6 Loss function

The model is trained by three separate losses that define a weighted sum into a final loss.

$$\mathcal{L}^{total} = \gamma_{gen}\mathcal{L}_{gen} + \gamma_{rnk}\mathcal{L}_{rnk} + \gamma_{cls}\mathcal{L}_{cls} \tag{4.25}$$

, where $\mathcal{L}_{gen}$ is the answer generation loss, $\mathcal{L}_{rnk}$ is the relevance ranking loss, $\mathcal{L}_{cls}$ is the answerability classification loss and $\gamma_{gen}, \gamma_{rnk}, \gamma_{cls} \in \mathbb{R}$ are the weight factors that control their relative importance.

- **Answerability Classifier Loss**

  It is the negative binary cross-entropy between the true and predicted value, averaged for all the examples in the batch $\mathcal{B}$.

$$\mathcal{L}_{cls} = -\frac{1}{N} \sum_{\alpha \in \mathcal{B}} \Big(\alpha \cdot \log P(\alpha) + (1 - \alpha) \cdot \log \big(1 - P(\alpha)\big)\Big) \tag{4.26}$$

  , where $N = |\mathcal{B}|$, $a \in \{0, 1\}$ denotes the ground-truth for an example in the batch and $P(\alpha) \in (0, 1)$ is the probability of answerability for the same example, which is the output of the classifier (Section 4.1.4).

- **Relevance Ranking Loss**

  The ranking loss is the negative binary cross-entropy between the true and predicted relevance for each passage in an example, averaged for all the examples in the batch $\mathcal{B}$.

$$\mathcal{L}^{rnk} = -\frac{1}{N} \sum_{r,\beta,K \in \mathcal{B}} \frac{1}{K} \sum_{k=1}^{K} \Big(r_k \cdot \log \beta_k + (1 - r_k) \cdot \log (1 - \beta_k)\Big) \tag{4.27}$$

  , where for each example in the batch, K is the number of passages, $r_k \in \{0, 1\}$ is the ground truth relevance label, and $\beta_k \in (0, 1)$ is the relevance probability for the k-th passage.

- **Answer Generation Loss**

  This loss operates on token prediction level and is the average negative log probability of the ground-truth tokens, over the answer sequence, over the answerable examples.

$$\mathcal{L}^{gen} = -\frac{1}{N_{ans}} \sum_{\alpha,a,T,V^{ext} \in \mathbb{B}} \Big(\frac{\alpha}{T^v} \sum_{t=1}^{T} \Big(\mathbb{I}(y_t \in V^{ext}) \cdot \log \big(P_t^{final}(y_t)\big)\Big)\Big) \tag{4.28}$$

  , where $N_{ans} = \sum_{a \in \mathbb{B}} a$ is the number of answerable examples in the batch and for each example:

  - $a \in \{0, 1\}$ indicates whether the example is answerable

- T is the number of tokens in the sequence
- $y_t \in \{0,1\}^{|V^{fixed}|}$ is the ground truth token for the t-th position in the sequence
- $V^{ext} \supseteq V^{fixed}$ is the set of tokens in the extended vocabulary
- $T^v = \sum_t^T \mathbb{I}(y_t \in V^{ext})$ is the number of known tokens in the sequence
- $P_t^{final}(y_t) \in \mathbb{R}$ is the probability of the ground truth token (Eq. 4.24)

The condition $y_t \in V^{ext}$ covers the rare scenario where the ground-truth token is not part of the fixed vocabulary, and it is not contained in the passages nor the query.

4.2 Pairwise Passage Ranking

In the Pointwise Ranker (*PointRnk*) of Nishida et al., 2019, each passage is processed independently from the others to obtain $\beta^{p_k} \in \mathbb{R}$, indicating how relevant is the k-th passage only with respect to the query. More formally, PointRnk maximizes the conditional probability $P(r_k|M_1^{p_k})$, where $r_k \in \{0,1\}$ is the ground truth label for the relevance of the k-th passage and $M_1^{p_k} \in \mathbb{R}^d$ is the vector representation of the first token in the k-th passage, which corresponds to the [CLS] token, which is tuned for the ranking and classification tasks.

It is important to note that although the [CLS] token representation of each passage incorporates query-specific information from the dual attention (Section 4.1.2), it does not do contain information from the rest of the passages since there is no cross-passage communication in the encoder. Thus, $M_1^{p_k}$ is a function of the k-th passage and the query, and as a result PointRnk maximizes the conditional probability $P(r_k|p_k, q)$.

In order to approach ranking in a relative context and additionally use global information from all the passages, a Pairwise Ranker (*PairRnk*) is proposed. PairRnk, instead of predicting whether a passage is relevant or not (absolute ranking), it learns to identify the most relevant in a pair of passages (relative ranking). Formally, for a pair of passages i, k in an example, PairRnk maximizes the conditional probability $P(r_{ki}^{pair}|M_1^{p_k}, M_1^{p_i})$, where $r_{ki}^{pair} \in \{0,1,2\}$ is the ground truth label for the pair, indicating which passage is more relevant or if they are of equal relevance. Furthermore, PairRnk utilizes a transformer encoder to globally update the [CLS] token of each passage with information from the rest. Consequently, each [CLS] token representation is a function of all passages and the query, and thus for the k-th and i-th passages, PairRnk maximizes the conditional probability $P(r_{ki}^{pair}|p_1, \ldots, p_K, q)$.

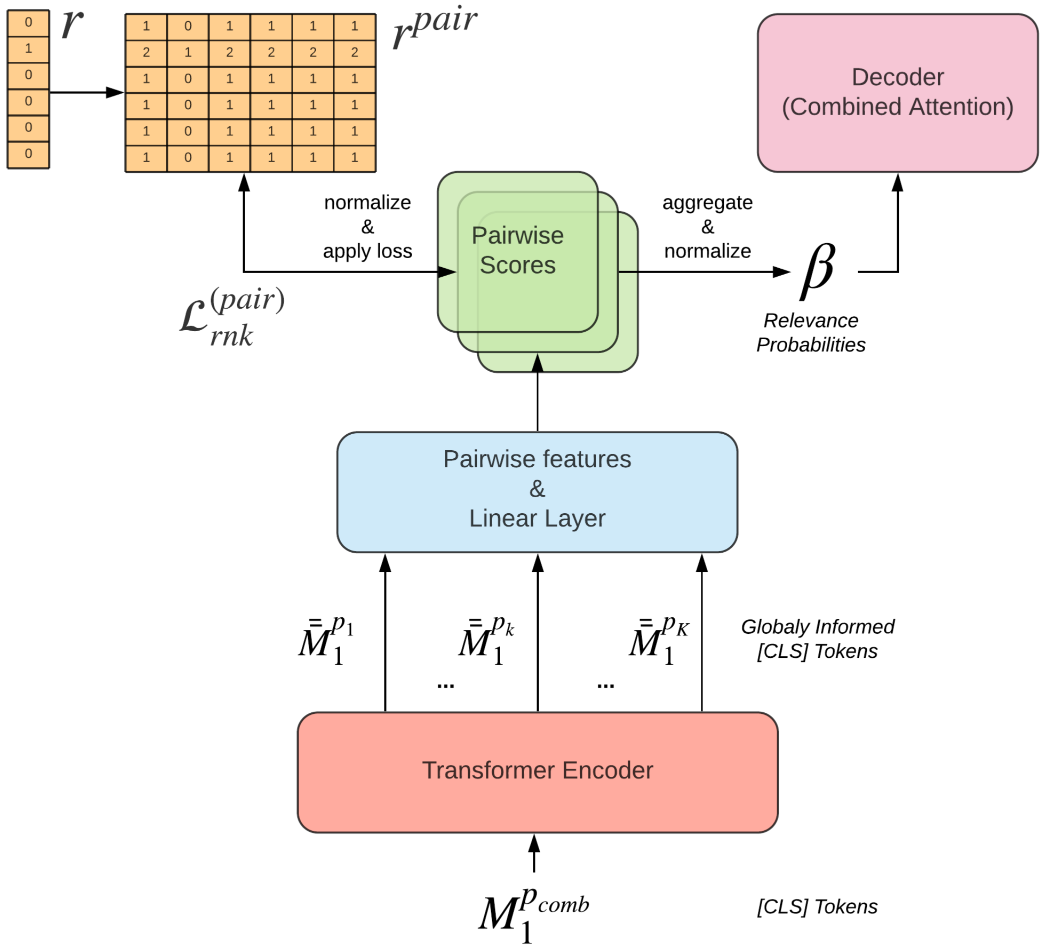


Figure 4.2: Pairwise Ranking Scheme.

A Transformer Encoder updates the [CLS] tokens with global information. For each pair of [CLS] tokens, the pairwise features are passed through a linear layer to obtain the pairwise class scores. Left path: The pairwise scores are normalized along the class dimension and the $\mathcal{L}_{rnk}^{(pair)}$ loss is applied. Right path: The pairwise scores are aggregated along the class dimension and normalized along the columns to obtain the relevance probabilities $\beta \in \mathbb{R}^K$, which are passed to the Decoder of the QA-model. With yellow are the ground-truth relevance labels, where the pointwise labels $r \in \{0,1\}^K$ are transformed into pairwise labels $r^{pair} \in \{0,1,2\}^{K \times K}$ (Eq. 4.29). For ease of representation the example in the figure has $K = 6$ passages available, where the second passage is relevant.

The methodology of PairRnk is given in more detail below.

The problem is transformed from absolute to relative by defining the pairwise ground-truth labels for each pair k, i of passages.

$$r_{ki}^{pair} = \begin{cases} 0, & \text{if } r_k > r_i \\ 1, & \text{if } r_k = r_i \\ 2, & \text{if } r_k < r_i \end{cases} \tag{4.29}$$

, where label 0 indicates that passage k is more relevant than i, label 1 indicates that they are equally relevant and finally label 2 indicates that passage k is less relevant than i.

A transformer encoder layer (Section 3.2.2) updates the [CLS] token representation of each passage with information from the rest of the passages.

$$\bar{M_1}^{p_{comb}} = \text{LN}^{(a)}\Big(\overset{enc}{\text{MHA}} \left(M_1^{p_{comb}}, M_1^{p_{comb}}\right) + M_1^{p_{comb}}\Big) \tag{4.30}$$

$$\bar{\bar{M}}_1^{p_{comb}} = \text{LN}^{(b)}\Big(\text{FFN}\Big(\bar{M_1}^{p_{comb}}\Big) + \bar{M_1}^{p_{comb}}\Big) \tag{4.31}$$

, where $M_1^{p_{comb}}, \bar{M_1}^{p_{comb}}, \bar{\bar{M}}_1^{p_{comb}} \in \mathbb{R}^{K\times d}$ are the concatenated representations of the [CLS] token for all the passages in the example, as output by the encoder, with $M_1^{p_{comb}} = [M_1^{p_1}; \ldots ; M_1^{p_K}]$.

PairRnk combines the globally informed [CLS] token representations of the k-th and i-th passages to obtain a feature vector $M_{ki}^{pair} \in \mathbb{R}^{4d}$ and maps it to the pairwise scores $\hat{r}_{ki}^{pair} \in \mathbb{R}^3$.

$$M_{ki}^{pair} = \left[\bar{\bar{M}}_1^{p_k}, \bar{\bar{M}}_1^{p_i}, \text{abs}(\bar{\bar{M}}_1^{p_k} - \bar{\bar{M}}_1^{p_i}), \bar{\bar{M}}_1^{p_k} \odot \bar{\bar{M}}_1^{p_i}\right] \tag{4.32}$$

$$\hat{r}_{ki}^{pair} = M_{ki}^{pair} \cdot W^r \tag{4.33}$$

, where $\text{abs}(\cdot)$ is the element-wise absolute value operator and $W^r \in \mathbb{R}^{4d\times 3}$ is a learnable parameter.

The scores are normalized to obtain a probability distribution over the three classes, $\bar{r}^{pair}_{ki} \in \mathbb{R}^3$.

$$\bar{r}^{pair}_{ki} = \text{softmax}(\hat{r}^{pair}_{ki}) \tag{4.34}$$

Finally, the pairwise passage ranking loss is defined as the correct class's negative log-likelihood, averaged per comparison, and per example.

$$\mathcal{L}^{rnk}_{(pair)} = -\frac{1}{N} \sum_{r^{pair}, \bar{r}^{pair}, K \in \mathcal{B}} \frac{2}{K(K+1)} \sum_{k=1}^{K} \sum_{i=1}^{k} \sum_{j=0}^{2} \mathbb{I}(r^{pair}_{ki} = j) \log (\bar{r}^{pair}_{kij}) \tag{4.35}$$

, where for an example in the batch, K is the number of passages, $r^{pair}_{ki} \in \{0, 1, 2\}$ is the ground truth comparable relevance between the k-th and i-th passage, and $\bar{r}^{pair}_{kij} \in (0, 1)$ is the probability of the label j, between the k-th and i-th passage. $\mathbb{I}(\cdot)$ is the indicator function, which is 1 if the expression is true and 0 otherwise. Note that due to the symmetrical nature of the pairwise comparison matrix, to avoid redundancy, only the elements that correspond to the lower triangular matrix are taken into account, which are equal to $K(K+1)/2$.

The pairwise scores are aggregated into a probability distribution over the passages, which is utilized by the combined attention in the decoder (Eq. 4.21). For the k-th passage, an overall advantage score $b_k \in \mathbb{R}$ is computed by averaging the net relevant scores of each comparison in which k is involved. The net relevant score of each comparison is the prediction score for the more-relevant class, minus the prediction scores for the equally-relevant and less-relevant classes.

$$b_k = \frac{1}{K} \sum_{i=1}^{K} (\hat{r}^{pair}_{ki1} - \hat{r}^{pair}_{ki2} - \hat{r}^{pair}_{ki3}) \tag{4.36}$$

The relevance probabilities $\beta \in \mathbb{R}^K$ are obtained by a softmax normalization over the advantages of all the passages.

$$\beta = \text{softmax}(b) \tag{4.37}$$

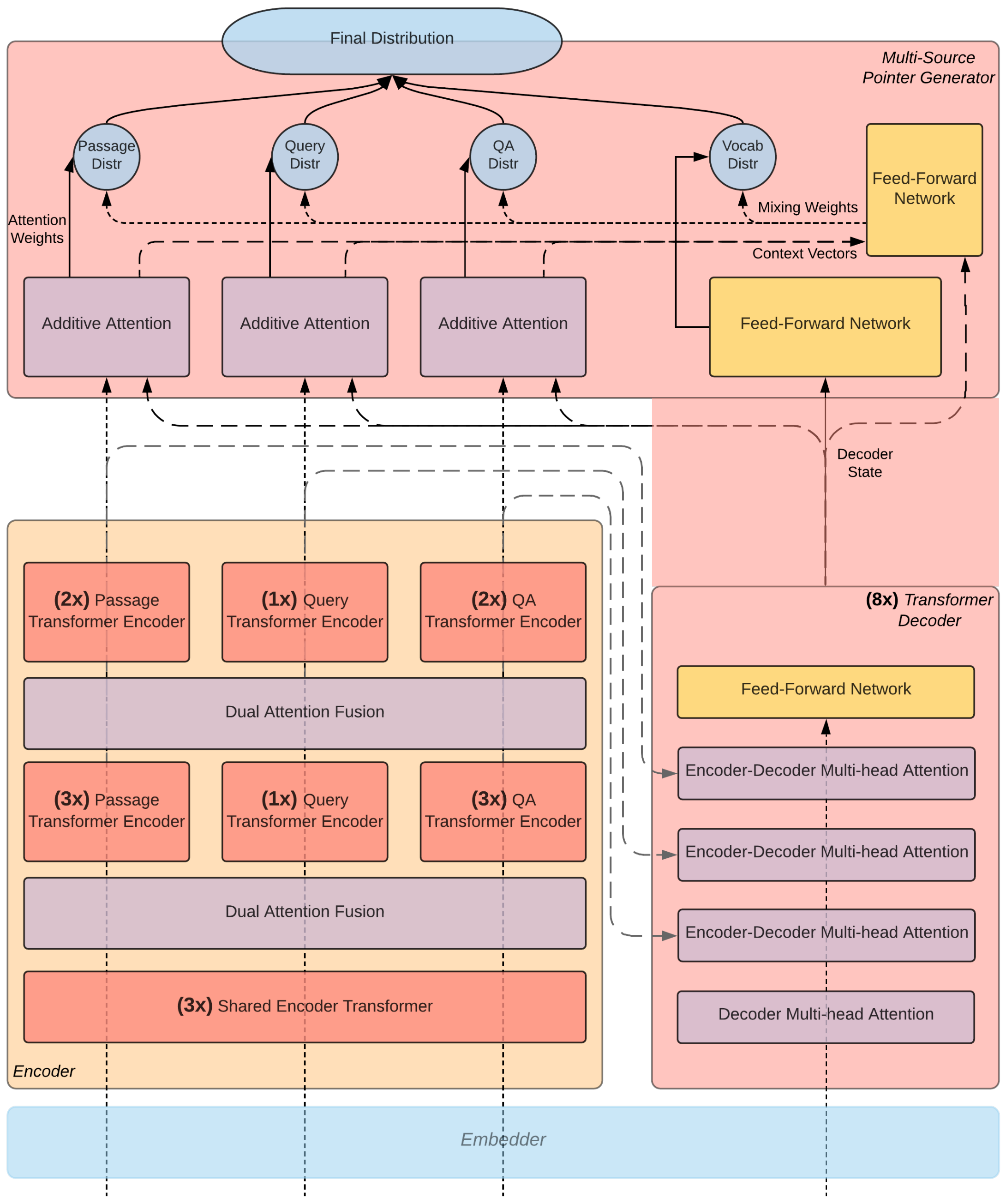
It is easy to identify the differences in complexity between the PointRnk and the PairRnk methods, which are exponential for PairRnk and linear for PointRnk. Usually, in multi-task learning scenarios, it is ideal to keep the auxiliary task modules, as simple as possible, to force all the complexity in the shared parts of the model. This way, the main task will benefit the most. Unlike most multi-task learning scenarios, passage ranking is not precisely an auxiliary task since the probabilities β are essential for channeling the decoder's attention to the most relevant passages. In other words, the passage ranker is not an end-point of the model, but just a middle one, and thus increasing its complexity is beneficial for the main task of answer generation (Table 5.7).

4.3 Extractive-Abstractive Answer Style Transfer

This model is used to learn a mapping between answer styles, and more specifically, from the QA (extractive) to the NLG (abstractive) answer style. To achieve that, information from the relevant passage of each example and the query is additionally used. Thus, the model learns a mapping from a triplet of QA Answer, relevant passage, and query to an NLG answer. More formally, for the t-th position in the NLG answer the Style-transfer model maximizes the conditional $P\Big(y_t|e, p, q, y_{t-1}, \ldots, y_1\Big)$, where:

- $y \in \mathbb{N}^T$ is the NLG answer tokens in the vocabulary, with length T
- $e \in \mathbb{N}^N$ is the QA answer tokens in the vocabulary, with length N
- $p \in \mathbb{N}^L$ are the tokens of the relevant passage that corresponds to the QA answer and has a length of L
- $q \in \mathbb{N}^J$ is the query tokens in the vocabulary, with length J

Style-transfer is carried out by a transformer-based encoder-decoder model. The encoder reads and fuses information from the three input sequences (e, p, q), and the decoder models their interactions with the target sequence by applying multiple attention modules. The decoder also employs a multi-source pointer generator that combines the three copy distributions of the input sequences and the fixed vocabulary distribution, obtained from the decoder state, to produce a final distribution over the extended vocabulary (Figure 4.3).



Final Distribution
Multi-Source Pointer Generator
Passage Distr
Query Distr
QA Distr
Vocab Distr
Feed-Forward Network
Attention Weights
Mixing Weights
Context Vectors
Additive Attention
Additive Attention
Additive Attention
Feed-Forward Network
Decoder State
(2x) Passage Transformer Encoder
(1x) Query Transformer Encoder
(2x) QA Transformer Encoder
Dual Attention Fusion
(3x) Passage Transformer Encoder
(1x) Query Transformer Encoder
(3x) QA Transformer Encoder
Dual Attention Fusion
(3x) Shared Encoder Transformer
Encoder
(8x) Transformer Decoder
Feed-Forward Network
Encoder-Decoder Multi-head Attention
Encoder-Decoder Multi-head Attention
Encoder-Decoder Multi-head Attention
Decoder Multi-head Attention
Embedder
Relevant Passage
Query
QA Answer
NLG Answer (shifted)

Figure 4.3: Style-transfer Encoder-Decoder Transformer.

An embedder module (Section 4.1.1) combines word and positional embeddings and is shared between all four types of sequences. The embeddings of the QA answer, the relevant passage and the query are passed through shared transformer encoder (Section 3.2.2), common for all sequences from the encoder side. Then three dual attention layers (Section 4.1.2), fuse information between each pair of sequences, namely $(p, q), (p, e), (q, e)$, with learnable parameters $w^{pq}, w^{pe}, w^{qe} \in \mathbb{R}^{3d}$. Thus, for the pair of relevant passage and query and for their representations from the shared transformer encoder, $H^p \in \mathbb{R}^{L\times d}$, $H^q \in \mathbb{R}^{J\times d}$, first their similarity matrix $U^{pq} \in \mathbb{R}^{L\times J}$ (Eq. 4.2) is normalized along its two dimensions to get attention weights $A^{pq} \in \mathbb{R}^{L\times J}$, $B^{pq} \in \mathbb{R}^{J\times L}$ (Eq. 4.3). The attention weights are subsequently used to obtain the updated representations:

$$\bar{A}^{pq} = A^{pq} \cdot H^q \tag{4.38}$$

$$\bar{B}^{pq} = B^{pq} \cdot H^p \tag{4.39}$$

$$\bar{\bar{A}}^{pq} = A^{pq} \cdot \bar{B}^{pq} \tag{4.40}$$

$$\bar{\bar{B}}^{pq} = B^{pq} \cdot \bar{A}^{pq} \tag{4.41}$$

Which are concatenated to get the passage-updated query and query-updated passage representations $G^{q\to p} \in \mathbb{R}^{L\times 4d}$, $G^{p\to q} \in \mathbb{R}^{J\times 4d}$.

$$G^{q\to p} = \left[H^p, \bar{A}^{pq}, \bar{\bar{A}}^{pq}, \bar{A}^{pq} \odot H^p, \bar{\bar{A}}^{pq} \odot H^p\right] \tag{4.42}$$

$$G^{p\to q} = \left[H^q, \bar{B}^{pq}, \bar{\bar{B}}^{pq}, \bar{B}^{pq} \odot H^q, \bar{\bar{B}}^{pq} \odot H^q\right] \tag{4.43}$$

The same process is repeated for the dual attentions between passage with QA answer and query with QA answer. Finally, the corresponding updated representations from every pair are concatenated to their pre-attentional representations to get $G^{(q,e)\to p} \in \mathbb{R}^{L\times 9d}$, $G^{(p,e)\to q} \in \mathbb{R}^{J\times 9d}$ and $G^{(p,q)\to e} \in \mathbb{R}^{N\times 9d}$.

$$G^{(q,e)\to p} = [H^p, G^{q\to p}, G^{e\to p}] \tag{4.44}$$
$$G^{(p,e)\to q} = [H^q, G^{p\to q}, G^{e\to q}] \tag{4.45}$$
$$G^{(p,q)\to e} = [H^e, G^{p\to e}, G^{q\to e}] \tag{4.46}$$

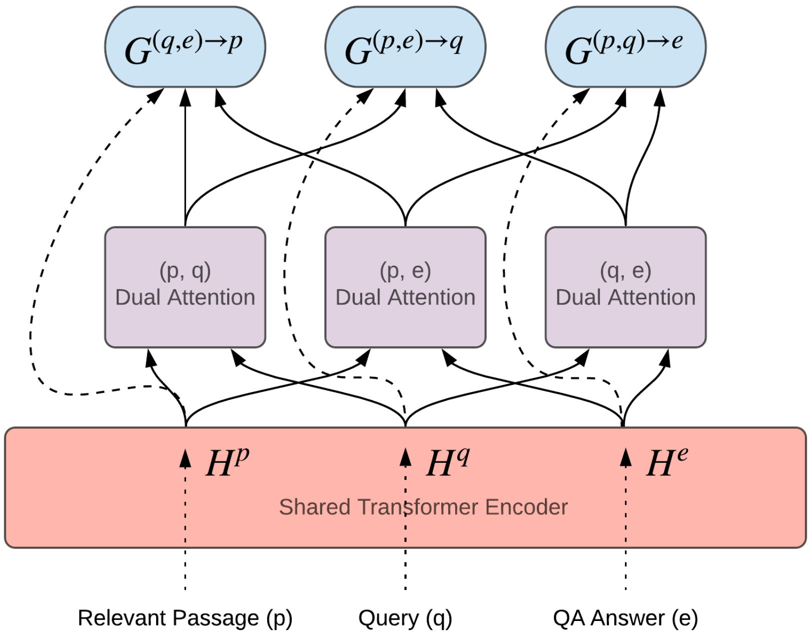


Figure 4.4: Dual Attention Fusion.

The representations of the previous transformer layer are fed in pairs into dual attention modules. Then the corresponding updated representations are concatenated to the pre-attentional representations. The outputs of this layer are now informed from all sequences of the encoder side.

The fused representations are passed through transformer encoders, one for each type of sequence. Finally, there is another block of dual attention fusion and sequence-specific transformers, to obtain representations $M^p \in \mathbb{R}^{L\times d}$, $M^q \in \mathbb{R}^{J\times d}$ and $M^e \in \mathbb{R}^{N\times d}$, which are passed to the decoder module.

The decoder is made up of a transformer decoder and a multi-source pointer generator (Section 4.1.5). Each transformer decoder layer a masked self-attention (Eq. 3.26) for modelling the interactions within the NLG answer, three encoder-decoder attention modules (Eq. 3.27), one for each sequence from the encoder side and a feed-forward network (Eq. 3.29). All modules are followed by residual connections and layer normalization layers. The output of the transformer decoder $M^y \in \mathbb{R}^{T \times d}$ is passed to the multi-source pointer generator, which combines the copy distributions from the three input sequences, together with the fixed vocabulary distribution. Three additive attention modules are utilized to obtain the context and attention vectors for the relevant passage, query, and QA-answer for t-th token in the NLG-answer.

$$c_t^e, \alpha_t^e = \text{AddAttn}^{(e)}(M^e, M_t^a) \tag{4.47}$$

$$c_t^q, \alpha_t^q = \text{AddAttn}^{(q)}(M^q, M_t^a) \tag{4.48}$$

$$c_t^p, \alpha_t^p = \text{AddAttn}^{(p)}(M^p, M_t^a) \tag{4.49}$$

The representation of the input sequences in the extended vocabulary map attention weight into the extended vocabulary distributions, while a feed-forward network defines the distribution of the fixed vocabulary.

$$P^e(y_t) = \alpha_t^e \cdot S^e_{(ext)} \tag{4.50}$$

$$P^q(y_t) = \alpha_t^q \cdot S^q_{(ext)} \tag{4.51}$$

$$P^p(y_t) = \alpha_t^p \cdot S^p_{(ext)} \tag{4.52}$$

$$P^v(y_t) = \text{softmax}\Big((M_t^a \cdot W^1 + b^1) \cdot W^2\Big) \tag{4.53}$$

, where $W^1 \in \mathbb{R}^{d \times d_{emb}}$, $b^1 \in \mathbb{R}^{d_{emb}}$ and $W^2 \in \mathbb{R}^{d_{emb} \times |V^{fixed}|}$ are learnable parameters.

The mixing weights $\lambda \in \mathbb{R}$ of each distribution are calculated by 4-way softmax normalization on the concatenation of the context vectors and the NLG representation for the t-th token.

$$\lambda_t^v, \lambda_t^e, \lambda_t^q, \lambda_t^p = \text{softmax}\Big(\Big[M_t^a, c_t^q, c_t^e, c_t^p\Big] \cdot W^\lambda + b^\lambda\Big) \tag{4.54}$$

, where $W^\lambda \in \mathbb{R}^{4d \times d}$ and $b^\lambda \in \mathbb{R}^d$ are learnable parameters.

The final distribution of the t-th token over the extended vocabulary is a weighted combination of all the four distributions.

$$P^{final}(y_t) = \lambda_t^v \cdot P^v(y_t) + \lambda_t^e \cdot P^e(y_t) + \lambda_t^q \cdot P^q(y_t) + \lambda_t^p \cdot P^p(y_t) \tag{4.55}$$

The loss of the Style-transfer model is the negative log probability of the correct tokens (Eq. 4.28), averaged over the length of each sequence and the number of examples in the batch.

$$\mathcal{L} = -\frac{1}{N} \sum_{y,T,V^{ext} \in \mathbb{B}} \left(\frac{1}{T^v} \sum_{t=1}^{T} \left(\mathbb{I}(y_t \in V^{ext}) \cdot \log \left(P_t^{final}(y_t) \right) \right) \right) \tag{4.56}$$

The trained Style-transfer model can be used on inference mode to generate an abstractive NLG answer for any triplet of relevant passage, query, and extractive QA answer in the train set. The augmented NLG dataset is made up of approximately 350,000 generated NLG answers and is denoted by NLG$_{aug}$.

4.4 Issues with the Answerability Classifier

In Nishida et al., 2019, classification of an example, as answerable or not, is done by a linear layer with learnable parameter $w^c \in \mathbb{R}^{Kd}$, where K is the number of passages. The input to the classification layer, $[M_1^{p_1}, \ldots, M_1^{p_K}] \in \mathbb{R}^{Kd}$, is the concatenation of the first token representation, corresponding to the [CLS] token (Eq. 4.13). Several issues arise with this approach, and namely, a restriction on the maximum number of passages K, a positional bias, and an unnecessarily increased complexity by the Kd number of parameters. More specifically:

- **The classifier can handle a maximum of K passages per example**, since the number of passages K is included in the dimensionality of the learnable parameter w^c. Although this might not be an important issue for training and evaluating the classifier on a dataset with a fixed number of passages, the model cannot be applied to more general scenarios where the number of passages is greater than K.

- **There is a positional bias towards the order of the passages in the example.** The parameter w^c, can be decomposed to K different vectors, with $w^c = [w_1^c; \ldots; w_K^c]$. Thus, the classifier learns parameters $w_k^c \in \mathbb{R}^d$, which are specific to passages placed in the k-th position of each example. Consequently, if the order of the passages is shuffled in the example, the prediction of the model changes, which should not be happening since the information stays the same. One may argue, that the order of the passages in each example is not completely random since the initial ordering has a mean average precision (MAP) of 34.62. However, that value is quite low to enforce the model to learn specific parameters for each position in the ordering. It is hypothesized that this dependency results in significant noise during training, restricting the classifier's accuracy.

- **The number of learnable parameters is very high for a purely auxiliary task**. Unlike the relevance ranking task, which is not an end-point, the answerability classifier is. That means that the information encoded by the classifier does not have a direct connection to the decoding, but an indirect one. Thus, ideally, the classifier should be as simple as possible to force a substantial part of the complexity to the encoder. In other words, a simpler classifier would result in the encoder producing representations that carry more information.

In this research, the classifier is simplified to a smaller linear layer on the maximum of the vertical concatenation of the [CLS] token representations.

$$P(a) = \text{sigmoid}\Big(\max_k\Big(\Big[M_1^{p_1}; \ldots; M_1^{p_K}\Big]\Big) \cdot w^c\Big) \tag{4.57}$$

, where $w^c \in \mathbb{R}^d$ is a learnable parameter.

This approach addresses all the issues mentioned above since the classifier works for an arbitrary number of passages K, there is no dependency on the ordering of the passages in the example, and the number of learnable parameters is reduced by a factor of K. For the rest of the thesis, this classifier is referred to as *MaxCls*, while the one of Masque is referred to as *LinCls*.

Chapter 5

Experiments

5.1 Setup

5.1.1 Dataset

The dataset used for this research is the MS MARCO v2.1 (Nguyen et al., 2016), which contains approximately 1 million queries that were inserted by users in the Bing search engine. Each data-point has the following information:

- **Query**
- **Query Type**, with the majority of the cases (53%) is labelled as description, 26% is of numerical type and the rest equally distributed among Entity, Person and Location. The full distribution of the query types also based on whether they contain certain keywords can be found in Table 5.1.
- **Passages** that were retrieved with the Bing search engine using the query. Each passage has three fields, which are text, URL and an indication of whether it is relevant or not for answering the query. The number of passages in 97.28% of the cases equals to 10, in 2.66% of the cases to less than 10. Finally, in some rare cases, which amount to 0.048%, there are more than 10 passages available.
- **Answers**, which are one or more, short, extractive answers for the query, if the query is answerable. These answers are referred to as QA-styled answers. The queries contain a QA answer and are considered answerable in about 55% of the total examples.
- **Well-Formed Answers**, which are longer, abstractive answers and are re-

ferred to as NLG-styled answers. Note that these answers exist only for about 30% of the answerable queries.

(A) Keyword		**(B) Type**	
What	34.96%	Description	53.12%
How	16.8%	Numeric	26.12%
Where	3.46%	Entity	8.81%
Who	3.33%	Location	6.17%
When	2.71%	Person	5.78%
Which	1.79%		
Why	1.67%		
Other	27.83%		

Table 5.1: Query distribution in train set of MS-MARCO.

In part (A) the queries are categorized by the keywords they contain. In part (B) the queries are categorized by the query type section included in each example of MS-MARCO.

The dataset was constructed by human annotators, where they were presented with a query and K passages, retrieved from Bing. Each passage was labelled as relevant or not for answering the query. If there was at least one relevant passage, one or more answers were provided for the query. In some selective cases, there was a second round of answer construction, where human annotators re-wrote answers which did not resemble full sentences, contained grammatical and syntactical errors or were incomplete. These answers are the “Well-Formed Answers” of an example and are at the centre of focus for this research.

Query	albany mn population		
Query Type	Numeric		
Passages	**relevant**	**text**	**url**
	0	...	...
	1	Albany, Minnesota,as per 2017 US Census estimate, has a community population of 2,662 people. Albany is located in Stearns County, 20 miles west of St. Cloud and 80 miles northwest of Minneapolis/St. Paulon Interstate 94 (I-94). Albany has direct access to State Highway 238, which originates in Albany.	...
	0	...	...
	...	...	...
	0	...	...
Answers	['2,662']		
Well-Formed Answers	['The population of Albany, Minnesota is 2,662.']		

Table 5.2: Example of a datapoint in MS-MARCO.

The query text, query type, passage, list of answers and list of well-formed answers of an example, with the text of the non-relevant passages and the URLs being omitted for ease of representation.

The complexity of this dataset, especially working with the "Well-Formed Answers", stems from several reasons.

1. The vast majority of the queries are non-factoid (Table 5.1), requiring complex reasoning to form a comprehensive answer.

2. Answers are presented in an abstractive way, meaning that purely extractive techniques would fail in producing a complete answer.

3. The context for each query is approximately 10 passages, which requires a selection mechanism that focuses the attention of the model on the most relevant ones.

4. A query is not necessarily answerable given the available passages.
5. There are cases with more than one relevant passage and/or more than one answer for each query.

Two subsets can be distinguished from each dataset: The ANS set, which includes all the answerable examples and the NLG set, which includes all the answerable examples that have an NLG answer, with NLG $\subset$ ANS $\subset$ ALL (Table 5.3).

	train	dev
ALL	808731	101093
ANS	502937	55578
NLG	152551	12448

Table 5.3: Datasets and Subsets in MS-MARCO.

The dataset contains four types of sequences, namely passages, queries, QA answers and NLG answers, which vary significantly in length (Figure 5.1). The mode of the QA-style length distribution is located at the first bin (between 0 and 4 sub-tokens) and signifies the extractive nature of the style. On the other side, the mode of the NLG-style length distribution is located higher, with approximately 15 sub-tokens.

The dataset is preprocessed to remove noise and inconsistencies. Whole examples are removed when they include an answer, but no indication of the relevant passage is given and similarly when there is a relevant passage, but no answer is given. Furthermore, within each example, duplicate passages, answers and well-formed answers are removed. Lastly, a small percentage of queries contain substantial noise in the form of either multiple continuous underscores or multiple continuous question and exclamation marks. These queries push up the complexity during training, especially for the dual attention[1], (Section 4.1.2). For these cases, the noise is reduced to a single instance, avoiding random rises in complexity and without any loss of information.

[1]The complexity of the dual attention lies in computing the similarity matrix, which is $|\mathcal{B} \times K \times L \times J$, where $\mathcal{B}$ is the batch size, K is the number of passages, L is the maximum passage length in an example and J the maximum query length. When an example with a noisy query is sampled, this complexity may increase even by a factor of 4, causing out-of-memory issues.

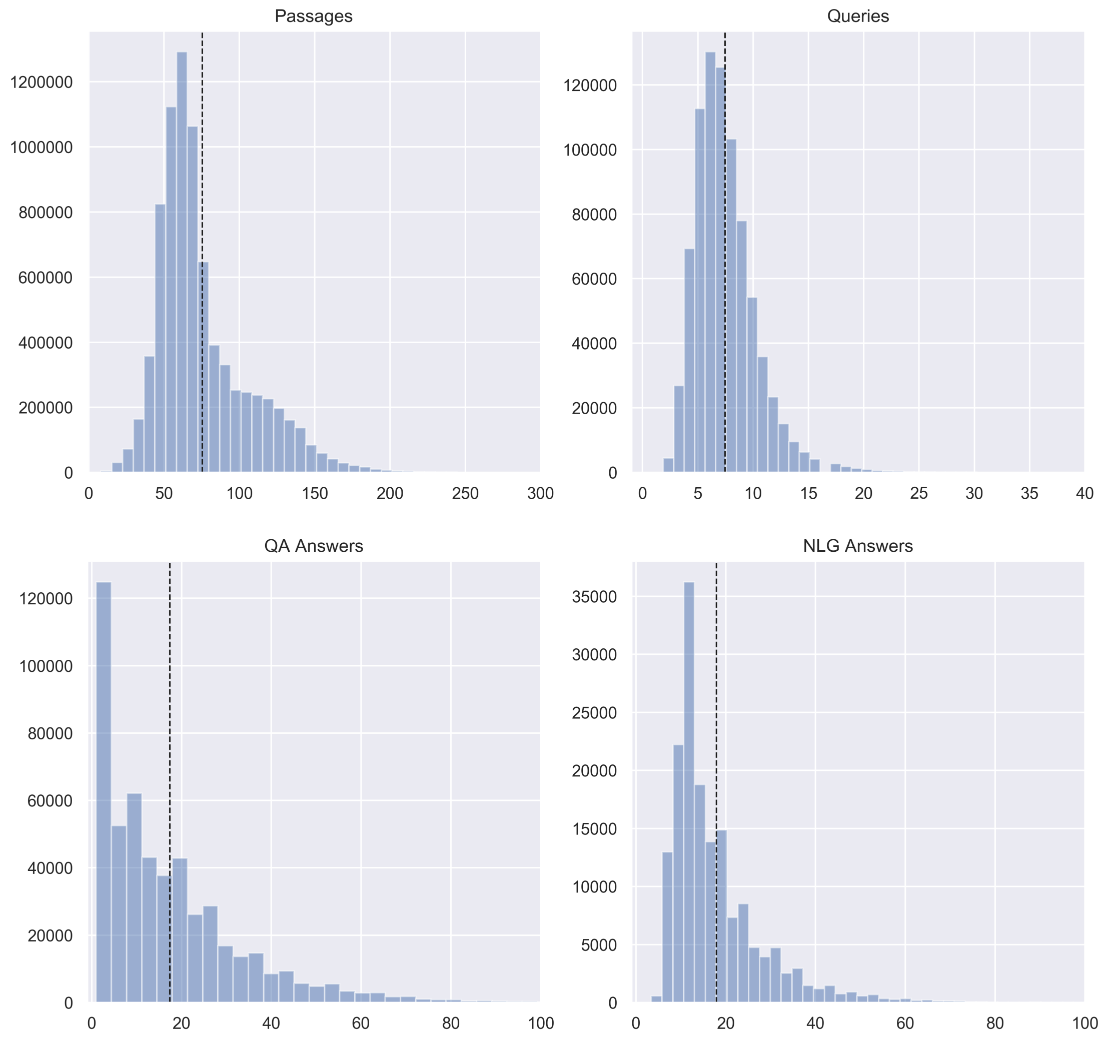


Figure 5.1: Sequence length distributions in the train set of MS-MARCO.

The x-axis corresponds to the length in sub-word tokens and the line denotes the average value for each type.

5.1.2 Training

This section provides all the details regarding the training and implementation of Masque (Section 4.1), the Style-transfer model (Section 4.3) and PREAST-QA, where the hyper-parameters of Nishida et al., 2019 are in most cases adopted. Masque and PREAST-QA are referred to as QA-models. The multi-task models that were trained without the decoder (Table 5.7) are referred to as Encoder-only models.

Hardware and training time: All models were trained with four NVIDIA TITAN V GPUs, using data parallelization, for eight epochs. A 12-core CPU machine was utilized for preparing each batch, where six workers were employed. For the implementation of Masque and PREAST-QA, training took 29 hours. Note that the implementation of Masque in this research is trained in just 10% of the time reported[2] in Nishida et al., 2019. The reasons for this reduction are not only the absence of contextualized embeddings but also extensive vectorization and a custom sampler that balances GPU-allocation. Furthermore, the training of the encoder-only models (Table 5.7), took approximately 8 hours and the training of the Style-transfer model took roughly 5 hours, using the same hardware.

Model Size: The QA-models have 3 shared encoder layers, 5 passage encoder layers, 2 query encoder layers and 8 decoder layers. The number of attention heads h is 8, the model dimensionality d is 296 and the size of the hidden layers in the feed-forward networks d_h is 256. The hidden size was chosen to be smaller than the model size in order to compress information. Finally, the dimensionality of the embeddings d_{emb} is 300. The encoder-only models have the same structure in the encoder and also the same dimensionalities.

The Style-transfer model has the same dimensionalities as the QA-models, with 8 decoder layers, but its encoder is structured differently. It has 3 shared encoder layers, followed by a layer of dual attention fusion, 3 passage encoder layers, 1 query encoder layer and 3 QA-style encoder layers. After, another layer of dual attention fusion is applied, followed again by another 2 passage encoder layers, 1 query encoder layer and 2 QA-style encoder layers.

Initialization: In Nishida et al., 2019, weights were initialized by sampling from $\mathcal{N}(0, 0.02)$ and zero vectors for all biases. Contrary, in this research sampling both weights and biases from $\mathcal{U}(-\frac{1}{\sqrt{d}}, \frac{1}{\sqrt{d}})$, where d is the size of the layer, was found to accelerate training in the beginning.[3] The gain parameter of normalization layers was initialized with vectors of ones and their biases with zeros.

[2]In Nishida et al., 2019 they trained on 8 NVIDIA P100 GPUs for six days, making the total of GPU hours 1152. In this research, the total GPU hours per run are just 116 (which can be increased by another 20 hours for the training of the Style-transfer model).

[3]This conclusion was based only on experiments with a smaller portion of the dataset.

Furthermore, the layers of the Style-transfer model that were common with the Masque implementation were initialized with a converged checkpoint of Masque. Thus, the Style-transfer model makes use of the whole MS Marco dataset, including non-relevant passage and non-answerable queries.

Tokenization and Embeddings: All sequences were lowercased and the Bert-Tokenizer of the HuggingFace Transformer library (Wolf et al., 2019) was used for the tokenization. The size of the fixed vocabulary $|V^{fixed}|$ is roughly 30,000 and the dimensionality of the embedding d_{emb} is 300. The majority of the tokens in the fixed vocabulary, namely around 90%, are initialized with GloVe (Pennington, Socher and Manning, 2014). For those tokens that are not part of GloVe or are special tokens, initialization is done by sampling from $\mathbb{N}(0, 0.01)$. The same procedure was followed by both the QA-models and the Style-transfer model.

Sampling a Batch: The Encoder-only models use a batch size of 80, while the QA-models were trained with a batch size of 44. Note that in terms of computational resources and memory, the answerable examples are much heavier than the non-answerable ones, since they determine the total number of tokens for which a decoder loss is computed. To optimize the GPU usage, and facilitate an efficient training procedure, the number of answerable examples in a batch was tuned to 27. For an answerable example, an original NLG answer is always sampled if available. For Masque, if an NLG answer does not exist, the QA answer is sampled, while For PREAST-QA, the answer is sampled randomly from its QA answers or its generated NLG answers. In Nishida et al., 2019 is not clear how the sampling process is carried out, but in the experiments of this research, it was found that prioritizing NLG answers is always better when abstractive answer generation is the primary goal. For the passages, if more than 10 are available, all the relevant ones are chosen, and the rest are randomly sampled. No custom batch sampler was used for the encoder-only models.

The Style-transfer model is trained with a batch size of 80. As mentioned in Section 5.1.1, some examples contain multiple relevant passages, multiple QA answers and multiple NLG answers. For those examples, which were approximately 5% of the NLG set, it would create a significant amount of noise to just randomly sample the triplet of relevant passage, QA answer and NLG answer, since the combination might contain inconsistent information. To prevent the adverse effect training the Style-transfer with this noise, for each such example, the ROUGE-L score is used to find the correct mappings within the set of all possible combinations of triplets. Thus, during training, given a sampled NLG answer, the relevant passage, with the highest ROUGE-L score with the NLG answer, is sampled. The same process is carried out for the QA answer to complete the sampled triplet for the example.

Special Tokens: The fixed vocabulary for the QA-models includes six special tokens in total. The [CLS] token is appended at the beginning of every passage, and it is fined-tuned for answerability classification and passage ranking during training. The [QA] and [NLG] tokens are appended at the beginning of every QA and NLG-styled answer accordingly. They are fine-tuned during training to absorb the features of each answer distribution and can subsequently be used during inference to condition the decoder and control the desired style. The [EOS] token is appended at the end of the target sequence and is fine-tuned to signify the end of the sequence generation process. The [UNK] token is used to replace all the tokens that are not part of the vocabulary. Finally, [PAD] tokens are appended to every type of sequence, to match the maximum size of each type in a batch.

The Style-transfer model uses a fixed vocabulary with four special tokens. The [CLS] token is not used since there is no classification nor ranking taking place. Furthermore, decoding is done only on the NLG-style answer, and thus the [QA] token is also not needed.

Truncation: During training, for the QA-models, each query is truncated to 40 tokens and each passage and answer to 100. The maximum number of passages, K, was set to 10, while for cases where $K < 10$, a vector with the [CLS] token and [PAD] tokens is used instead.

Since only one passage is used in each example for the Style-transfer model (the relevant one), due to the increased availability of memory, the relevant passage, QA-style answer and NLG-style answers were truncated to 120 instead of 100. The queries were still truncated at 40 tokens.

Optimizer: All models in this research were trained with an Adam optimizer (Kingma and J. Ba, 2014), with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. Warm-up training was additionally employed, where the learning rate is linearly increased from zero to $2.5 \cdot 10^{-4}$ in the first steps and then back to 0, by cosine annealing. In

Nishida et al., 2019, Masque was trained with 2000 warm-up steps, which is adjusted here to 4000, to account for the smaller batch size. Again, adjusting for the number of steps and batch size, the warm-up steps for Encoder-only models are 2000 and 1000 for the Style-transfer model. Gradient clipping with a maximum norm of 1 was used. Experiments with an increased learning rate of 0.3 and gradient accumulation were also conducted.

Regularization: All models in this research, use dropout (N. Srivastava et al., 2014) with a rate of 0.3 in all dot-product and dual attention operations. Additionally, dropout, with a decreased rate of 0.1, was applied to the output of the embeddings[4]. Furthermore, the modified version of $L2$ regularization (Loshchilov and Hutter, 2017) was used, with $w = 0.01$ to all non-bias parameters. Finally, for all models that use an answerability classifier or a pointwise ranker, label smoothing on the positive class label was applied, making it equal to 0.9.

Loss: The QA-models are trained with a weighted sum of the answer generation, passage ranking and answerability classification losses (Eq. 4.25) in a multi-task framework. The weighting scheme of Nishida et al., 2019 is adopted for the Masque implementation, namely with factors $\gamma_{gen} = 1$, $\gamma_{rnk}^{(point)} = 0.5$ and $\gamma_{cls} = 0.1$. The PairRnk method uses a different loss function (Eq. 4.35) than the PointRnk of Masque, which results in a different loss scale. In order to keep the importance of the ranking task equal for both ranking scenarios, the ranking weighting factor is adjusted to $\gamma_{rnk}^{(point)} = 0.265$. The Encoder-only models are trained with a weighted combination of the passage ranking and answerability classification losses. Similarly, to keep the relative and absolute importance of each task similar, the weight factors are set to $\gamma_{rnk}^{(point)} = 1$, $\gamma_{rnk}^{(pair)} = 0.53$ and $\gamma_{cls} = 0.2$. Finally, the Style-transfer model is trained with the loss defined in Eq. 4.56. Details for deriving the weight factor adjustments for PairRnk and Encoder-only models can be found in the Appendix C.

5.1.3 Inference and Evaluation

The evaluation of the answerability classifier is done in the ALL subset of the development set. To provide results comparable to Nishida et al., 2019, performance

[4]In Nishida et al., 2019 this is also 0.3 but applied to the fused GloVe and ELMo embeddings. Since in this implementation, only GloVe was used, applying a higher rate than 0.1 was found to significantly decrease performance, due to the high restrictions on the information for each embedding.

is measured by first tuning the decision boundary to maximize the precision-recall trade-off, and then the f1 score that corresponds to that decision boundary is reported.

For the passage ranking task, evaluation is done in the ANS set of the development set, which includes all the answerable examples. The metrics used are the Mean Average Precision (MAP) and the Mean Reciprocal Rank (MRR). For each example, $r \in \{0,1\}^K$ is the ground truth relevance of the passages in descending order, according to the predictions $\beta \in \mathbb{R}^K$ of the ranker, where K is the number of passages in an example.

$$\text{MAP} = \frac{1}{|\mathcal{D}|} \sum_{r,K \in \mathcal{D}} \left(\frac{1}{\sum_k r_k} \sum_{k=1}^{K} \mathbb{I}(r_k = 1) \frac{\sum_{i=1}^{k} r_k}{k} \right) \tag{5.1}$$

$$\text{MRR} = \frac{1}{|\mathcal{D}|} \sum_{r,K \in \mathcal{D}} \left(\sum_{k=1}^{K} \mathbb{I}\Big(r_k = 1\Big) \cdot \mathbb{I}\Big(\beta_k = \max_k(\beta)\Big) \frac{1}{k} \right) \tag{5.2}$$

The difference between the two metrics is that the MRR takes into account only the highest-ranked relevant passage, while MAP provides a more general view, by it considering the ranking of all relevant passages.

Answer generation is done with a greedy algorithm that keeps generating tokens in an autoregressive way until the [EOS] token is generated or the maximum number of tokens, 100, is reached. The style of the answer is controlled by initializing the generation with the [QA] or the [NLG] special token. For both the QA-models and the Style-transfer model, evaluation is done in the NLG subset of the development set, which includes all the answerable examples that have an NLG-style answer. The widely used BLEU-1 (Papineni et al., 2002) and ROUGE-L (Lin, 2004) are reported for evaluation, which measure the overlapping n-grams between the reference answer and the generated answer. More specifically, bleu focuses on precision, while rouge on recall. Note that decoding with a beam search did not provide any meaningful increase[5] for the development set of MS-MARCO and thus was not used.

[5] A beam search of size 5, where the average log-likelihood of each sequence determines the fitness of the answer, provided an increase in ROUGE-L, but a considerably more substantial decrease in BLEU-1.

5.2 Results

5.2.1 Style-Transfer and Data Augmentation

The Style-transfer model (Section 4.3) is trained on the NLG subset of the train set, and its performance is evaluated on the NLG subset of the development set. Results show that it can produce synthetic NLG answers of high quality, from a triplet of relevant passage, query and QA answer, achieving a ROUGE-L score of 87.02 and a BLEU-1 score of 87.09. Additionally, for certain query types, such as "where" and "who", the generation scores are higher than 90 (Table 5.4). It is also worth noting that the Style-transfer model can exactly map the QA answers to their NLG-styled counterparts, in 37.6% of the total cases, achieving a perfect ROUGE-L score of 1. An essential part of the Style-transfer model is its use of two dual attention fusion layers that enable the relevant passage, query and QA answer to simultaneously update each other. Including only one of those, significantly decreased the quality of the generated NLG answers, by 1.5 ROUGE-L points.

The trained model is used to augment the MS-MARCO dataset by generating a synthetic NLG answer for all the answerable examples that were lacking one (ANS$\setminus$ NLG). The augmented NLG set is denoted NLG$_{aug}$ and has a size of approximately 350,000. To further assess the quality of the NLG$_{aug}$ set, the Masque implementation of this thesis (Section 4.1) uses it as additional training data. During training on the ALL + NLG$_{aug}$, multi-style training is used, where an original NLG answer is always sampled if available and otherwise there is a 50-50 sampling probability between the QA answer and NLG$_{aug}$ answer. Results indicate that synthetic data provided by Style-transfer model increase the abstractive answer generation abilities of the Masque, raising the overall ROUGE-L score by 0.65 points (Table 5.4).

	Masque			Style-Transfer
Trained on	ALL	ALL + NLG$_{aug}$	ALL + NLG$_{aug}^{(-)}$	NLG
overall ROUGE-L				
	67.20	67.85 (+0.65)	**68.38 (+1.18)**	87.01
per-type ROUGE-L				
Description (55%)	60.53	60.51(-0.02)	**61.62 (+1.09)**	82.62
Numeric (23%)	72.38	73.20 (+0.82)	**73.49 (+1.11)**	86.52
Entity (9%)	63.95	**65.24 (+1.29)**	65.21 (+1.26)	84.21
Location (7%)	79.62	**81.51 (+1.89)**	80.97 (+1.35)	91.17
Person (6%)	72.14	73.53 (+1.39)	**73.76 (+1.62)**	88.70
per-keyword ROUGE-L				
What (39%)	66.22	66.93 (+0.71)	**67.64 (+1.42)**	87.21
How (13%)	64.57	65.09 (+0.52)	**65.46 (+0.99)**	84.66
Definition (6%)	64.62	62.28 (+0.66)	**67.19 (+2.57)**	89.63
Where (4%)	74.57	**75.82 (+1.25)**	74.99 (+0.42)	91.59
Who (4%)	74.40	75.59 (+1.18)	**76.58 (+2.18)**	90.86
When (3%)	71.13	72.15 (+1.02)	**72.25 (+1.12)**	87.26
Which (2%)	70.45	**72.02 (+1.57)**	71.75 (+1.30)	85.31
Why (1%)	47.89	**50.49 (+2.6)**	49.84 (+1.95)	83.65
Other (21%)	67.74	67.93 (+0.19)	**68.26 (+0.52)**	85.86

Table 5.4: Results of Masque with augmented data on the MS-MARCO dev set.

ROUGE-L score for Masque (Section 4.1) and the Style-transfer model (Section 4.3) on the MS-MARCO development set. Apart from the overall ROUGE-L score, the ROUGE-L score for each query type and keyword (Table 5.1) is also reported. Masque is trained with three different set configuration. ALL corresponds to the entirety of the train set. NLG$_{aug}$ corresponds to the synthetic abstractive answer set, generated by the Style-transfer model. NLG$_{aug}^{(-)}$ indicates the reduced version of synthetic abstractive answer set, without the high-resource type queries that are underlined (An example is removed from the NLG$_{aug}$ if it is of type Description and contains either of the underlined keywords). The Style-transfer is by default trained on the NLG subset. The number in the parenthesis of each type and keyword indicates their percentage in the NLG$_{aug}$ set. The parenthesis in the scores of Masque with the NLG$_{aug}$ and NLG$_{aug}^{(-)}$ sets indicate the absolute improvement in the ROUGE-L score from the model that was trained on the ALL set. With bold are the best scores for each type and keyword among the Masque runs. There is a negative correlation of 0.7 between the percentage of the queries in the dataset and the absolute improvement brought by training on the NLG$_{aug}$ set. A positive correlation, of 0.18, is observed between this improvement and the ROUGE-L score of the Style-transfer model.

Analyzing the per-type and per-keyword ROUGE-L scores, it is found that the improvement brought to by the NLG$_{aug}$ set, is more significant for low-resource queries, where the correlation between the absolute improvement of Masque trained on the ALL+NLG$_{aug}$ set from Masque trained only on ALL set, and the percentage of queries in the NLG$_{aug}$ set is equal to -0.7. It is worth noting that for description type queries, which are by far the most resourceful queries, there is a slight drop in of 0.02 when using additional data. Thus, to further increase the positive effects of training on synthetic data, a smaller synthetic NLG dataset, denoted by NLG$_{aug}^{(-)}$ is used for training, excluding high-resource queries that had little or no improvement. More specifically, an example is excluded if it is labelled as description type and additionally contains either of the keywords "what", "how" and "definition" or "define". Results on training with the selective synthetic NLG data indicate significantly more substantial improvements for the high-resource query types that have a higher contribution to the overall score, with an absolute improvement, from Masque trained on the ALL set, reaching 1.18 ROUGE-L points. As a consequence of the fewer synthetic NLG data, there is a slight decrease in the improvement of the low-resource queries, such as "which" and "why".

5.2.2 Abstractive Question Answering

The efficacy of the proposed model of this thesis, PREAST-QA is compared against the Masque baseline (Section 4.1) on the main task of abstractive answer generation and on the tasks of passage ranking and answerability classification (Table 5.5). PREAST-QA uses PairRnk (Section 4.2) for passage ranking and MaxCls (Section 4.4) for answerability classification, as opposed to Masque that uses PointRnk (Section 4.1.3) and LinCls (Section 4.1.4). Additionally, PREAST-QA is trained on the joint set of ALL and NLG$_{aug}^{(-)}$, where the latter one is composed of the synthetic abstractive answers, generated by the Style-transfer model (Section 4.3). To ensure that the results are less affected by the choice of hyper-parameters, both Masque and PREAST-QA are run in three different settings, in terms of learning rate and batch size, and the best results for each model are reported in Table 5.5, which is a part of the full results, available in Table A.1. All the rest of the hyper-parameters and procedures are as described in Section 5.1.2 and both models are trained with multi-task and multi-style learning.

Results show that the combined methods of this thesis, provide a significant improvement over the Masque baseline in abstractive answer generation, as indicated primarily by the ROUGE-L score and ultimately also by the BLEU-1 score (Table 5.5). Furthermore, PREAST-QA obtains consistently higher results than Masque, as indicated by the more extensive experiments presented in Appendix A. The gains of PREAST-QA over Masque can be first attributed to its use of high quality synthetic abstractive answers of $\text{NLG}_{aug}^{(-)}$ (Table 5.4), proving that multi-style training can be further enhanced and by including an intermediate step of style-transfer. Secondly, the gains can also be traced to its superior ranking capabilities, as indicated by the 0.65 increase in both MAP and MRR. Although the original Masque implementation makes use of contextualized embeddings form ELMo, PREAST-QA bridges the difference in abstractive answer generation to 1.3 ROUGE-L points, obtains competitive results in passage ranking and finally surpasses it in terms of answerability classification, by a large margin.

	NLG Answer Generation		Passage Ranking		Answerability Classification
	ROUGE-L	BLEU-1	MAP	MRR	F1 @t
Masque (Nishida et al., 2019)	69.77†	65.56†	69.51†	69.96†	78.93
Masque (this implementation)	67.60	63.46	68.49	68.96	78.41
PREAST-QA	**68.47**	**64.14**	**69.16**	**69.64**	**79.81**†

Table 5.5: Results for Masque and PREAST-QA on the the MS-MARCO dev set.

Results on NLG answer generation, passage ranking and answerability classification of the re-implementation of Masque and the proposed model of this thesis, PREAST-QA that combines the mechanisms of PairRnk (Section 4.2) and MaxCls (Section 4.4) and is additionally trained with an augmented dataset $\text{NLG}_{aug}^{(-)}$, generated by the Style-Transfer model (Section 4.3). Both Masque and PREAST-QA were run in different hyper-parameters settings, and the best runs for each model were selected for this table. The extended version of these results can be found in Appendix A. The results of Nishida et al., 2019 are provided for reference. The main difference between the Masque implementations is that Nishida et al., 2019 uses contextualized embeddings from ELMo (Peters et al., 2018). MAP and MRR are measured @K, where K is the number of passages in each example and F1 @t, where t is the optimized decision boundary. With bold are best scores among the models implemented in this research and with † are the overall best.

(A) **Query**: we elect a u.s. representative for how many years?

Relevant Passage: Two (2) years. You elect a member of the US House of Representatives every two years. Senate members serve for six years. Representatives of the US Congress serve two-year terms, with elections being held in all years that are even numbered. Two (2) years.

Reference Answer: You elect a member of the US House of Representatives every two years

Masque Answer: we elect a u. s. representative for 2 years.

PREAST-QA Answer: we elect a member of the us house of representatives every two years.

(B) **Query**: chlorophyll can be found in ________.

Relevant Passage: Chlorophyll (also chlorophyl) is a term used for several closely related green pigments found in cyanobacteria and the chloroplasts of algae and plants. Its name is derived from the Greek words chloros (green) and phyllon (leaf).

Reference Answer: Chlorophyll can be found in cyanobacteria and the chloroplasts of algae and plants.

Masque Answer: chlorophyll can be found in cyanobacteria.

PREAST-QA Answer: chlorophyll can be found in cyanobacteria and the chloroplasts of algae and plants.

(C) **Query**: why don't cats like water

Relevant Passage: The main reason, that cats don't like water, is that it causes them to lose body heat. However, this is only if they are forced into a body of water, most cats are ok with getting a little wet if they can help it. Some cats actually enjoy water, to the point where they will happily take baths, and swim around.

Reference Answer: The main reason, that cats don't like water, is that it causes them to lose body heat.

Masque Answer: don't cats like water because of their sensitivity to its odor.

PREAST-QA Answer: cats don ' t like water because it causes them to lose body heat.

Table 5.6: Generated Answers from Masque and PREAST-QA.

Examples from MS-MARCO development set with the query, the relevant passage, the reference NLG answer and the generated answers from Masque and PREAST-QA using the NLG style. Example (A): Both models have a correct answer, but Masque's does not have proper form. Example (C): Masque stopped generation early, and its answer is incomplete, while PREAST-QA generated the complete answer. Example (C): The answer of Masque to the "why" query is wrong in content and form. The answer of PREAST-QA has the correct content and form.

Table 5.6 provides examples from the development set, that show how training on the synthetic data aided in PREAST-QA having better form in abstractive answer generation. More specifically, in example (A), while Masque just reversed the query and added the required answer "2", PREAST-QA chose to copy from the second sentence, providing, in general, a more well-formed answer to the query. In example (B), PREAST-QA continues generating after the point that Masque stopped, thus having a complete answer. Finally, example (C) is a low-resource "why" query with negation, which was confusing for Masque and thus answered it improperly both in form and content. Contrary, PREAST-QA reversed the negation correctly from the query and provided not only the proper content but also the correct answering form. It is hypothesized that the differences in the answers of the two models for example (C) are a direct consequence of the additional 5,000 synthetic abstractive answers for "why" queries, that PREAST-QA was trained with.

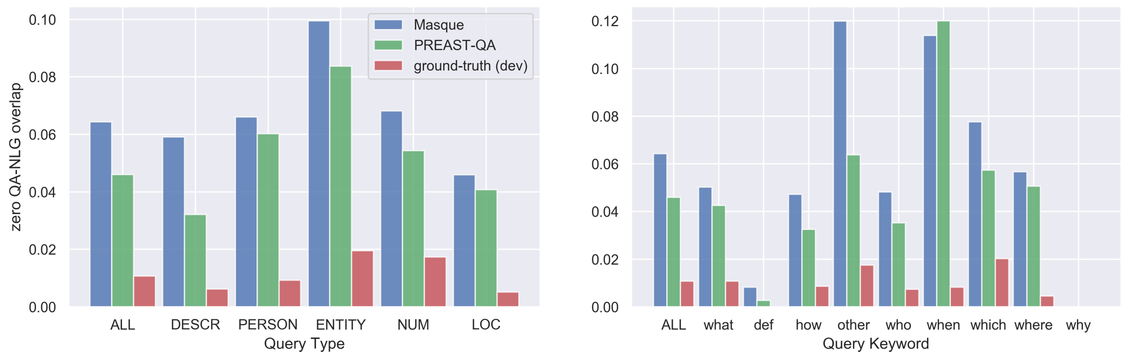


Figure 5.2: Zero overlap between QA and NLG styles.

Y-axis measures the percentage of cases where the QA and NLG style predictions had zero overlapping n-grams between them, for Masque and PREAST-QA. The percentage of cases with zero overlaps between the two styles in the ground-truth answers of the development set is also given for reference. All measurements were made in the NLG subset of the development set. In the first figure, cases are categorized by type and in the second one by keyword. ALL is the same for both figures. Predictions are made by the best models of Masque and PREAST-QA that were included in Table 5.5.

Another interesting of training on the $\text{NLG}^{(-)}_{aug}$ set is that it alleviates a data bias introduced by the multi-style training. More specifically, it is observed that the artificial style tokens absorb certain biases caused by the differences in the QA and NLG answer distributions. Consecutively, given the same passages and query, there are cases, where different styles produce entirely different answers. The scale of this adverse effect can be quantified by measuring the percentage of QA and NLG answer pairs that have zero overlapping n-grams. Overall, the synthetic NLG answers balanced the training data of the two styles and brought the zero overlapping cases from 6.4% to 4.6%, with the effect being more significant for description type queries the non-keyword category "other". Although there was a 1.8% improvement, the zero overlap between the ground-truth answers of the styles is at 1%, indicating that a significant amount of bias still exists, even after the inclusion of the style-transfer step into the framework of multi-style training. Also, note that the percentage of zero-overlapping QA-NLG pairs indicates the minimum[6] number of cases where the QA answer is different from the NLG one.

5.2.3 Pairwise Ranking in Multi-task Learning

	NLG Answer Generation		Passage Ranking		Answerability Classification
	ROUGE-L	BLEU-1	MAP	MRR	F1 @t
$\mathcal{L}_{rnk}, \mathcal{L}_{cls}$					
PointRnk + LinCls	-	-	65.35	65.83	77.77
PointRnk + MaxCls	-	-	64.80	65.29	78.96
PairRnk + LinCls	-	-	**67.23**	**67.72**	77.61
PairRnk + MaxCls	-	-	67.08	67.56	**79.26**
$\hookrightarrow$ w/o p2p Transformer	-	-	66.14	66.63	79.18
$\mathcal{L}_{gen}, \mathcal{L}_{rnk}, \mathcal{L}_{cls}$					
Masque	67.20	62.54	68.49	68.96	78.41
Masque with PairRnk + MaxCls	**67.94**	**63.29**	**69.12**†	**69.56**†	**79.72**†

Table 5.7: Results on Multi-task learning with/without answer generation.

Ranking and Classification results on the development set of MS-MARCO v2.1 for different combinations of ranking and classifying mechanisms. PointRnk (Section 4.1.3) and LinCls (Section 4.1.4) refer to the ranker and classifier of Masque (Nishida et al., 2019). PairRnk (Section 4.2) and MaxCls (Section 4.4) refer to the ranker and classifier introduced here, while p2p Transformer refers to the passage-to-passage transformer layer of the PairRnk method. In the first part of the table, models use the same encoder (Section 4.1.2) and are trained with a weighted sum of the ranking and classification losses. In the second part, models use the same encoder and decoder (Section 4.1.5) and are trained with a weighted sum of the answer generation, ranking and classification losses. Apart from the changes in the weights of multi-task learning, all other hyper-parameters were kept fixed. With bold are the best scores in each section and with †, the overall best.

To investigate in more depth the effect of the ranking and classification mechanism proposed in this thesis, they are compared with the ones of Masque in scenarios where the models are trained jointly for the tasks of passage ranking and answerability classification. All the possible ranking and classification technique combinations are examined in the first part of Table 5.7. An experiment where the passage-to-

[6]A non-zero overlap might be caused by shared stopwords, such as“and”, that do not contribute to the semantics of the answer.

passage transformer is excluded from the PairRnk method is additionally conducted to gain a better knowledge of its influence. The second part of the same table includes an experiment where the ranking and classification techniques of Masque are benchmarked against those of this research, with the decoder also being active, to understand the actual impact that the secondary tasks have on the main task of abstractive question answering.

The main findings of the experiments of Table 5.7 can be summarized as follows:

- **PairRnk is strictly better than PointRnk**, as indicated in terms of both MAP and MRR, and with or without the answer generation task. Subsequently proven is the importance of the information exchange between passages, enabled by the transformer layer in PairRnk as well as the advantage of casting the pointwise ranking task to a pairwise problem, which can then be solved by comparing the relevance of each passage to the rest. Additionally, the exclusion of the transformer layer reveals that the pairwise nature of the ranker is powerful enough by itself, and still surpasses PointRnk by a significant margin in the corresponding metrics.

- **MaxCls is strictly better than LinCls**, as indicated again by the tuned F1 score in the encoder-only and full-model scenarios. Consequently, the classification results confirm the hypothesis of Section 4.4, that the positional bias introduced by horizontally concatenating the passages in LinCls, produces significant noise during training, ultimately setting an upper limit to its accuracy.

- **PairRnk and MaxCls lead to better answer generation**, demonstrated by the increase of 0.75 points in ROUGE-L and BLEU-1 in the scenario where all tasked are jointly trained. This increase is primarily a direct effect of the substantially more accurate relevance probabilities that the decoder receives from the ranker, which lead to better guidance of the generation process through the combined attention (Eq. 4.21). Furthermore, it is an indirect effect of the increased quality of representations that the decoder receives from the encoder, due to the absence of the noisy signals of LinCls.

Apart from the main findings discussed above, it is observed that the inclusion of the answer generation task significantly improves the results of the ranking and classification tasks, independently of the mechanisms used. Furthermore, it is noted that the PairRnk method is combined better with the LinCls, at least in the experiments of the encoder-only models. It can be hypothesized that this is due to the upper bound in the classification accuracy, introduced by the positional bias of LinCls, which creates more room for improvement in the ranking task (at the expense of the classification task).

5.3 Application in a Scientific Domain

Research in Artificial Intelligence (AI) has been growing rapidly in recent years. Only in 2019, the newly submitted AI-related papers in arXiv[7] surpassed the levels of 30,000, with their submission numbers growing at an exponential rate[8]. Thus, methods of navigating this vast amount of research are becoming increasingly important. For that end, the proposed model of this thesis, PREAST-QA, is applied as a complex question answering system in the domain of artificial intelligence literature. More specifically, The model is included as a component in the AI Research Navigator[9] of Zeta Alpha, the company where this research was carried out. PREAST-QA receives as input a query along with K passages, retrieved with vector-based search from a space of more than 100 thousand AI research papers, and provides an abstractive answer to the query.

The performance of PREAST-QA on the out-of-domain scientific text of AI literature was assessed with a small hand-crafted set of 270 AI-related questions from Quora[10]. These questions differ significantly from the ones in MS-MARCO, averaging in length at 15 sub-tokens, with some of them containing even more than 30 sub-tokens. The increased question length is a result of the AI terms that are usually included, which are not part of the known vocabulary and are thus split into numerous parts. Such an example is the term "convolutional" which is tokenized into "con-vo-lu-tion-al". Furthermore, the vast majority of the questions are descriptive, containing "what", "how" or "why" keywords (Figure 5.3). Each question was coupled with a set of 10 passages retrieved using Zeta Alphas' vector-based search system, which were subsequently truncated at 200 sub-tokens. Since these are unlabeled data, and no gold

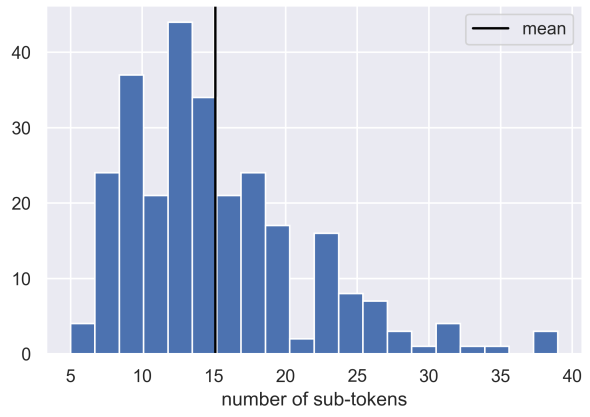
[7] https://www.arxiv.org

[8] https://www.zeta-alpha.com/post/growth-of-ai-research-in-2020-steady-on-the-exponential-path-in-times-of-crisis

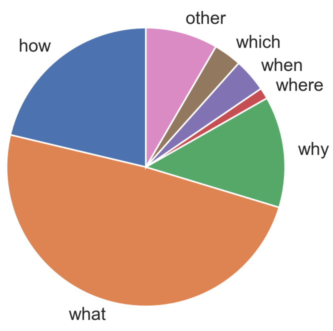
[9] https://search.zeta-alpha.com

[10] https://www.quora.com/

answers exist, traditional metrics like ROUGE-L and BLEU-1 can not be reported. Consequently, the performance of PREAST-QA is evaluated manually. Every case was labelled as either "good", if the question was answered properly, "acceptable", if at least some relevant information was provided, or "bad" if the generated answer was irrelevant to the question.



(a) Length distribution



(b) Type distribution

Figure 5.3: Length and Type distribution of the Quora questions.

Results of the manual evaluation process showed that PREAST-QA generates "good" answers in approximately 17% of the cases, "acceptable" answers in 21% of the cases and "bad" answers for the rest. The answer quality distribution changes drastically if only the cases where the answer exists in the passages are considered. Under those circumstances, "good" answers are a majority with 39%, while the "acceptable" ones are at 36% and finally, the answer is of "bad" quality in 25% of the answerable questions. Examples of different answer qualities are given in Table 5.8.

Contrary to examples from MS-MARCO, beam search was found to increase the quality of the generated answers significantly, in the domain of AI research. The positive effect of the beam search is to enable longer answers, since the model was ending generation early, mainly due to the many unknown terms in the questions and passages. The beam size was set to 5, and the average log-likelihood of its tokens measured the fitness of each sequence. Since the model does not suffer from repetition in neither MS-MARCO nor the Quora questions, n-gram blocking was not applied. Interestingly, it was noted that for the examples, where the classifier of PREAST-QA had enough confidence that the question is answerable, longer answers were preferable, even though they did not have the best log-likelihood among the candidates. Thus, for the cases with $P(a) > 0.3$, beam search selects the longest answer that is within reasonable bounds of log-likelihood, compared to the best one. Furthermore, it was observed that PREAST-QA relied significantly more on copying than generating, which is explained from the uncertainty brought by the multiple unknown words and the different language features of scientific text. Finally, PREAST-QA performed poorly in answering very long questions, since too much context was provided by them, forcing it to transform the question to an answer, disregarding the passages.

Answer Quality: Good
(A) What is the difference between logistic regression and Naive Bayes?
$\hookrightarrow$ The difference between naive bayes and logistic regression is due only to the fact that the first is generative and the second discriminative; the two classifiers are, for discrete input, identical in all other respects.
(B) Why is the pooling layer used in a convolution neural network?
$\hookrightarrow$ A pooling layer in a convolutional neural network enforces that the representation of the data at that layer is invariant to small translations of the input data.
Answer Quality: Acceptable
(C) What is the Hessian matrix? What is it used for and for what reason?
$\hookrightarrow$ The hessian matrix is computationally and spatially expensive, especially for large models.
(D) Why do we use Naive Bayes classifiers?
$\hookrightarrow$ Naive bayes classifiers apply bayes â^{TM} theorem with the assumption of conditional independence between every pair of features. We chose this classifier because it has shown to perform surprisingly well on text problems.
Answer Quality: Bad
(E) What is an intuitive explanation of stochastic gradient descent?
$\hookrightarrow$ An intuitive alternative 6 to circumvent the need to solve this is to consider an iterative solution, called gradient descent / ascent.
(F) Why do deep learning models use convolutional layers instead of fully-connected layers?
$\hookrightarrow$ Deep learning models use convolutional layers instead of fully connected layers.

Table 5.8: Generated answers by PREAST-QA for the Quora questions.

Two examples for each answer quality. (A) and (B) can be considered complete. In (C) there is relevant information but not sufficient to answer the question and in (D) there is copied noise. In (E) the answer is incorrect and in (F) the question is inverted since the model was unable to find an answer.

To conclude, PREAST-QA provides on average acceptable results for the task of complex question answering in the domain of AI research literature. The most important limitations are the following:

- There is barrier introduced by the different language of the scientific documents, being substantial more formal than the text of MS-MARCO.
- Scientific documents contain various terms that are unknown to the model or even confusing, such as “CNN”, where the model has learned it as the media channel and not a class of neural networks. The use of contextualized embeddings can potentially alleviate this negative effect.
- Passages retrieved from scientific documents additionally contain many references and mathematical equations that introduce a certain degree of noise to the model.
- A bias towards the beginning of each passage was observed since PREAST-QA is trained on passages that average in length at 75 sub-tokens. Consequently, even if the answer is included in the second half of the passage, it was rarely identified by the model.

Chapter 6

Conclusions and Further Research

6.1 Conclusions

This research is focused on the task of Question-Answering (QA), and more specifically in answering complex queries with abstractive language generation. The task is investigated in a scenario where multiple passages are available, queries are not always answerable, and answers are primarily extractive, as facilitated by the MS-MARCO dataset. In such a setting, a high-quality QA model has to be able to identify the most relevant passages and effectively learn how to produce abstractive answers by using knowledge from extractive answers. To provide a better framework for solving the task, PREAST-QA, a transformer encoder-decoder, is proposed, which is based on the Masque multi-task and multi-style model of Nishida et al., 2019. PREAST-QA achieves higher results than Masque in the MS-MARCO NLG task, by employing a pairwise ranker and introducing a Style-transfer model that augments the dataset with more abstractive training data.

The proposed ranker of this research, PairRnk casts the pointwise problem of labelling each passage as relevant or not, to a pairwise one, where the ranker has to identify the most relevant passage in each pairwise comparison in the example. To achieve that, PairRnk additionally employs a transformer encoder layer that enables passage-to-passage information exchange. Part of the success of PREAST-QA is traced to the guidance that it receives from the relevance probabilities obtained by PairRnk. The pairwise approach is systematically better than the pointwise in many multi-task learning experiments, and even in the absence of the passage-to-passage transformer layer.

For augmenting the MS-MARCO dataset with 3x more abstractive answers, a Style-transfer model, which is a modified encoder-decoder transformer, learns a mapping between a triplet of relevant passage, query and extractive answer to an abstractive answer. To achieve that, it employs several transformer layers and dual attention fusions in the encoder, as well as a 4-way multi-source pointer-generator in the decoder. The Style-transfer model learns to produce high-quality synthetic abstractive answers that are subsequently used by PREAST-QA in training. The inclusion of these augmented data, as an additional step in multi-style learning, is found to significantly increases the abstractive answering abilities of a QA model. The effect is even more substantial after a selection process in the synthetic data, excluding answers for high-resource query-types. Furthermore, training on the synthetic abstractive answers helped alleviate a data bias absorbed by the artificial tokens during multi-style training, which is causing the predictions of the two answering styles to be completely different in certain cases.

This research additionally solves a positional bias found in the large, linear classifier of Masque, which was introducing significant noise during training, restricting its performance. PREAST-QA instead uses a simpler max-pooling classifier that outperforms the one of Masque by a considerable margin and enables classification with a variable number of passages.

The capabilities of the proposed model were also assessed in an out-of-domain application scenario, and more specifically, that of AI scientific research. Certain limitations, like the poor understanding of scientific language and terms, were identified. These limitations forced the model to rely more on copying than generating, in order to obtain on average good results.

To conclude, firstly, the results of this thesis indicate that a pointwise ranker is not adequate for identifying the relevant passages for answering a query, whereas the PairRnk method is better at modelling the problem. Secondly, multi-style learning is found to leave the abstractive answering task under-trained, especially in low-resource scenarios, which can be alleviated by using synthetic data from a style-transfer model.

6.2 Further Research

The high-quality of the synthetic data generated by the Style-transfer model had a crucial role in the abstractive answering performance of PREAST-QA. There are two possible ways to increase their quality further, subsequently improving future Question-Answering systems. Firstly, the Style-transfer model proposed here is trained only for uni-directional mapping, from the extractive to the abstractive style. This process can become bi-directional by the use of multi-style learning, where similarly to the QA models described in this thesis, the Style-transfer model can sample one of the styles as input and the other one as a target. Learning a bi-directional mapping between styles, would increase the style-agnostic generation abilities of the Style-transfer model and also provide the option to use synthetic extractive answers in the training of QA models.

In this research, the style-transfer task and the QA task are approached separately and independently of each other. Thus, another possible direction for future works is to train the Style-transfer model end-to-end with the QA system, in a scenario similar to the training of Generative Adversarial Networks (Goodfellow et al., 2014). In that case, the Style-transfer model plays the role of the generator, providing synthetic answers to the QA-model, which additionally uses a discriminator for identifying them from the original answers. This process would force the Style-transfer model to generate progressively better synthetic answers, with the goal of making them indistinguishable from the true ones.

Apart from increasing the quality of the synthetic data, research on methods to further decrease the bias of multi-style training should also be conducted. Ideally, the two styles should be able to produce the same content in different styles. A potential solution is to include a term in the loss that penalizes abstractive generations that do not overlap with the extractive gold answer. Excluding the 1% of the examples in the training set of MS-MARCO, where the gold extractive and abstractive answer have zero overlap with each other, should also aid in the resolution of this bias.

The methods proposed in this research were only applied for the case of MS-MARCO, and thus, it would be interesting to assess their effect on other Question Answering datasets. The recently proposed ELI5 (Fan et al., 2019) makes for a good challenge, having queries that are more complex than those of MS-MARCO and usually require much lengthier answers. Additionally, multiple passages are provided for the queries, along with extractive and abstractive answers. Furthermore, the PairRnk mechanism proved to be successful not only the multi-task scenario, where it is jointly trained with a decoder but also in the purely ranking tasks (Table 5.7), indicating that it can be useful for more general applications in Information Retrieval. A certain limitation lies in the fact that its efficacy was tested for small-scaled retrieval scenarios, usually with ten available passages for each query and thus its performance for large-scale retrieval has to be first assessed.

Finally, issues with proper query and passage understanding are observed for PREAST-QA (Table B.1), probably due to the absence of contextual information. This absence was more significant for the application of the model in the domain of AI scientific literature (Section 5.3). As also indicated by the gap of more than 2 ROUGE-L points between the implementation of Masque here and the one of Nishida et al., 2019, the use of contextualized embeddings plays a crucial role in obtaining state-of-the-art performance in abstractive question answering. It can be hypothesized that the use of embeddings from ELMo (Peters et al., 2018) or BERT (Devlin et al., 2018) will increase further the performance of both the Style-transfer model and PREAST-QA.

References

Ba, Jimmy Lei, Jamie Ryan Kiros and Geoffrey E. Hinton (2016). *Layer Normalization*. arXiv: 1607.06450 [stat.ML] (cit. on p. 24).

Bahdanau, Dzmitry, Kyunghyun Cho and Yoshua Bengio (2014). 'Neural Machine Translation by Jointly Learning to Align and Translate'. In: *CoRR* abs/1409.0473 (cit. on pp. 3, 13, 18).

Caruana, Rich (1997). 'Multitask learning'. In: *Machine learning* 28.1, pp. 41–75 (cit. on p. 4).

Chaturvedi, Akshay, Onkar Pandit and Utpal Garain (July 2018). 'CNN for Text-Based Multiple Choice Question Answering'. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Melbourne, Australia: Association for Computational Linguistics, pp. 272–277. DOI: 10.18653/v1/P18-2044. URL: https://www.aclweb.org/anthology/P18-2044 (cit. on pp. 13, 14).

Chen, Danqi, Jason Bolton and Christopher D. Manning (Aug. 2016). 'A Thorough Examination of the CNN/Daily Mail Reading Comprehension Task'. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, pp. 2358–2367. DOI: 10.18653/v1/P16-1223. URL: https://www.aclweb.org/anthology/P16-1223 (cit. on p. 13).

Chen, Tianqi and Carlos Guestrin (Aug. 2016). 'XGBoost'. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. DOI: 10.1145/2939672.2939785. URL: http://dx.doi.org/10.1145/2939672.2939785 (cit. on p. 16).

Chen, Zhipeng et al. (July 2019). 'Convolutional Spatial Attention Model for Reading Comprehension with Multiple-Choice Questions'. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33, pp. 6276–6283. ISSN: 2159-5399. DOI: 10.1609/aaai.v33i01.33016276. URL: http://dx.doi.org/10.1609/aaai.v33i01.33016276 (cit. on p. 14).

Chung, Junyoung et al. (2014). 'Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling'. In: *CoRR* abs/1412.3555. arXiv: 1412.3555. URL: http://arxiv.org/abs/1412.3555 (cit. on p. 13).

Collobert, Ronan et al. (2011). 'Natural Language Processing (almost) from Scratch'. In: *CoRR* abs/1103.0398. arXiv: 1103.0398. URL: http://arxiv.org/abs/1103.0398 (cit. on p. 15).

Devlin, Jacob et al. (2018). 'Bert: Pre-training of deep bidirectional transformers for language understanding'. In: *arXiv preprint arXiv:1810.04805* (cit. on pp. 10, 12, 34, 76).

Fan, Angela et al. (2019). *ELI5: Long Form Question Answering*. arXiv: 1907.09190 [cs.CL] (cit. on p. 75).

Goodfellow, Ian J. et al. (2014). *Generative Adversarial Networks*. arXiv: 1406.2661 [stat.ML] (cit. on p. 75).

Hashimoto, Kazuma et al. (2016). 'A Joint Many-Task Model: Growing a Neural Network for Multiple NLP Tasks'. In: *CoRR* abs/1611.01587. arXiv: 1611.01587. URL: http://arxiv.org/abs/1611.01587 (cit. on p. 15).

He, Kaiming et al. (2015). 'Deep Residual Learning for Image Recognition'. In: *CoRR* abs/1512.03385. arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385 (cit. on p. 25).

He, Wei et al. (July 2018). 'DuReader: a Chinese Machine Reading Comprehension Dataset from Real-world Applications'. In: *Proceedings of the Workshop on Machine Reading for Question Answering*. Melbourne, Australia: Association for Computational Linguistics, pp. 37–46. DOI: 10.18653/v1/W18-2605. URL: https://www.aclweb.org/anthology/W18-2605 (cit. on p. 15).

Hendrycks, Dan and Kevin Gimpel (2016). 'Bridging Nonlinearities and Stochastic Regularizers with Gaussian Error Linear Units'. In: *CoRR* abs/1606.08415. arXiv: 1606.08415. URL: http://arxiv.org/abs/1606.08415 (cit. on p. 34).

Hochreiter, Sepp and Jürgen Schmidhuber (1997). 'Long short-term memory'. In: *Neural computation* 9.8, pp. 1735–1780 (cit. on pp. 3, 13).

Hu, Minghao, Yuxing Peng and Xipeng Qiu (2017). 'Mnemonic Reader for Machine Comprehension'. In: *CoRR* abs/1705.02798. arXiv: 1705.02798. URL: http://arxiv.org/abs/1705.02798 (cit. on p. 13).

Inan, Hakan, Khashayar Khosravi and Richard Socher (2016). 'Tying Word Vectors and Word Classifiers: A Loss Framework for Language Modeling'. In: *CoRR* abs/1611.01462. arXiv: 1611.01462. URL: http://arxiv.org/abs/1611.01462 (cit. on pp. 37, 85).

Johnson, Melvin et al. (2016). *Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation.* arXiv: 1611.04558 [cs.CL] (cit. on p. 16).

Joshi, Mandar et al. (2017). *TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension.* arXiv: 1705.03551 [cs.CL] (cit. on p. 15).

Kadlec, Rudolf et al. (2016). *Text Understanding with the Attention Sum Reader Network.* arXiv: 1603.01547 [cs.CL] (cit. on p. 14).

Kim, Yoon et al. (2015). *Character-Aware Neural Language Models.* arXiv: 1508.06615 [cs.CL] (cit. on p. 12).

Kingma, Diederik P. and Jimmy Ba (2014). *Adam: A Method for Stochastic Optimization.* arXiv: 1412.6980 [cs.LG] (cit. on p. 58).

Kočiský, Tomáš et al. (2017). *The NarrativeQA Reading Comprehension Challenge.* arXiv: 1712.07040 [cs.CL] (cit. on p. 17).

Lan, Zhenzhong et al. (2019). *ALBERT: A Lite BERT for Self-supervised Learning of Language Representations.* arXiv: 1909.11942 [cs.CL] (cit. on p. 12).

Lin, Chin-Yew (July 2004). 'ROUGE: A Package for Automatic Evaluation of Summaries'. In: *Text Summarization Branches Out.* Barcelona, Spain: Association for Computational Linguistics, pp. 74–81. URL: https://www.aclweb.org/anthology/W04-1013 (cit. on p. 60).

Liu, Shanshan et al. (2019). 'Neural Machine Reading Comprehension: Methods and Trends'. In: *CoRR* abs/1907.01118. arXiv: 1907.01118. URL: http://arxiv.org/abs/1907.01118 (cit. on p. 12).

Liu, Yang and Mirella Lapata (2019). *Hierarchical Transformers for Multi-Document Summarization.* arXiv: 1905.13164 [cs.CL] (cit. on p. 16).

Loshchilov, Ilya and Frank Hutter (2017). *Decoupled Weight Decay Regularization.* arXiv: 1711.05101 [cs.LG] (cit. on p. 59).

McCann, Bryan et al. (2018). 'The Natural Language Decathlon: Multitask Learning as Question Answering'. In: *CoRR* abs/1806.08730. arXiv: 1806.08730. URL: http://arxiv.org/abs/1806.08730 (cit. on p. 15).

Mikolov, Tomas et al. (2013). 'Distributed Representations of Words and Phrases and their Compositionality'. In: *Advances in Neural Information Processing Systems 26.* Ed. by C. J. C. Burges et al. Curran Associates, Inc., pp. 3111–3119 (cit. on p. 12).

Mitchell, Tom M. (1980). *The Need for Biases in Learning Generalizations.* Tech. rep. (cit. on p. 15).

Nallapati, Ramesh et al. (Aug. 2016). 'Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond'. In: *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning.* Berlin, Germany: Association for Computational Linguistics, pp. 280–290. DOI: 10.18653/v1/K16-1028. URL: https://www.aclweb.org/anthology/K16-1028 (cit. on p. 3).

Nguyen, Tri et al. (2016). 'MS MARCO: A Human Generated MAchine Reading COmprehension Dataset'. In: *CoRR* abs/1611.09268. arXiv: 1611.09268. URL: http://arxiv.org/abs/1611.09268 (cit. on pp. 5, 51).

Nishida, Kyosuke et al. (2019). 'Multi-style Generative Reading Comprehension'. In: *CoRR* abs/1901.02262. arXiv: 1901.02262. URL: http://arxiv.org/abs/1901.02262 (cit. on pp. iii, 4–6, 10, 11, 14, 16, 29, 30, 32, 34, 37, 40, 49, 55–57, 59, 64, 67, 73, 76, 85, 88).

Papineni, Kishore et al. (July 2002). 'Bleu: a Method for Automatic Evaluation of Machine Translation'. In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics.* Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, pp. 311–318. DOI: 10.3115/1073083.1073135. URL: https://www.aclweb.org/anthology/P02-1040 (cit. on p. 60).

Pennington, Jeffrey, Richard Socher and Christopher D. Manning (2014). 'Glove: Global vectors for word representation'. In: *In EMNLP* (cit. on pp. 12, 32, 57).

Peters, Matthew E. et al. (2018). 'Deep contextualized word representations'. In: *CoRR* abs/1802.05365. arXiv: 1802.05365. URL: http://arxiv.org/abs/1802.05365 (cit. on pp. 10, 12, 32, 64, 76).

Radford, Alec et al. (n.d.). 'Language models are unsupervised multitask learners'. In: () (cit. on p. 34).

Rajpurkar, Pranav et al. (Nov. 2016). 'SQuAD: 100,000+ Questions for Machine Comprehension of Text'. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing.* Austin, Texas: Association for Computational Linguistics, pp. 2383–2392. DOI: 10.18653/v1/D16-1264. URL: https://www.aclweb.org/anthology/D16-1264 (cit. on p. 12).

Rao, Sudha and Joel Tetreault (2018). *Dear Sir or Madam, May I introduce the GYAFC Dataset: Corpus, Benchmarks and Metrics for Formality Style Transfer.* arXiv: 1803.06535 [cs.CL] (cit. on p. 17).

See, Abigail, Peter J. Liu and Christopher D. Manning (2017). 'Get To The Point: Summarization with Pointer-Generator Networks'. In: *CoRR* abs/1704.04368. arXiv:

1704.04368. URL: http://arxiv.org/abs/1704.04368 (cit. on pp. 3, 14, 27, 29).

Sennrich, Rico, Barry Haddow and Alexandra Birch (June 2016). ‘Controlling Politeness in Neural Machine Translation via Side Constraints’. In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, pp. 35–40. DOI: 10.18653/v1/N16-1005. URL: https://www.aclweb.org/anthology/N16-1005 (cit. on p. 16).

Seo, Min Joon et al. (2016). ‘Bidirectional Attention Flow for Machine Comprehension’. In: *CoRR* abs/1611.01603. arXiv: 1611.01603. URL: http://arxiv.org/abs/1611.01603 (cit. on pp. 3, 13, 18).

Srivastava, Nitish et al. (2014). ‘Dropout: A Simple Way to Prevent Neural Networks from Overfitting’. In: *Journal of Machine Learning Research* 15.56, pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html (cit. on p. 59).

Srivastava, Rupesh Kumar, Klaus Greff and Jürgen Schmidhuber (2015). ‘Highway Networks’. In: *CoRR* abs/1505.00387. arXiv: 1505.00387. URL: http://arxiv.org/abs/1505.00387 (cit. on p. 32).

Sutskever, Ilya, Oriol Vinyals and Quoc V Le (2014). ‘Sequence to Sequence Learning with Neural Networks’. In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., pp. 3104–3112 (cit. on p. 2).

Sutton, Richard S et al. (2000). ‘Policy gradient methods for reinforcement learning with function approximation’. In: *Advances in neural information processing systems*, pp. 1057–1063 (cit. on p. 16).

Takeno, Shunsuke, Masaaki Nagata and Kazuhide Yamamoto (Nov. 2017). ‘Controlling Target Features in Neural Machine Translation via Prefix Constraints’. In: *Proceedings of the 4th Workshop on Asian Translation (WAT2017)*. Taipei, Taiwan: Asian Federation of Natural Language Processing, pp. 55–63. URL: https://www.aclweb.org/anthology/W17-5702 (cit. on p. 16).

Tan, Chuanqi et al. (2017). *S-Net: From Answer Extraction to Answer Generation for Machine Reading Comprehension*. arXiv: 1706.04815 [cs.CL] (cit. on p. 14).

Vaswani, Ashish et al. (2017). ‘Attention Is All You Need’. In: *CoRR* abs/1706.03762. arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762 (cit. on pp. 4, 13, 18, 23, 29).

Vinyals, Oriol, Meire Fortunato and Navdeep Jaitly (2015). ‘Pointer Networks’. In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes et al.

Curran Associates, Inc., pp. 2692–2700. URL: http://papers.nips.cc/paper/5866-pointer-networks.pdf (cit. on pp. 3, 14).

Wang, Shuohang and Jing Jiang (2016). *Machine Comprehension Using Match-LSTM and Answer Pointer*. arXiv: 1608.07905 [cs.CL] (cit. on p. 14).

Wang, Shuohang, Mo Yu et al. (2017). *R^3: Reinforced Reader-Ranker for Open-Domain Question Answering*. arXiv: 1709.00023 [cs.CL] (cit. on p. 16).

Wang, Yizhong et al. (2018). 'Multi-Passage Machine Reading Comprehension with Cross-Passage Answer Verification'. In: *CoRR* abs/1805.02220. arXiv: 1805.02220. URL: http://arxiv.org/abs/1805.02220 (cit. on pp. 12, 15).

Wolf, Thomas et al. (2019). 'HuggingFace's Transformers: State-of-the-art Natural Language Processing'. In: *ArXiv* abs/1910.03771 (cit. on p. 57).

Xiong, Caiming, Victor Zhong and Richard Socher (2016). 'Dynamic Coattention Networks For Question Answering'. In: *CoRR* abs/1611.01604. arXiv: 1611.01604. URL: http://arxiv.org/abs/1611.01604 (cit. on pp. 3, 13, 14, 18).

– (2017). 'DCN+: Mixed Objective and Deep Residual Coattention for Question Answering'. In: *CoRR* abs/1711.00106. arXiv: 1711.00106. URL: http://arxiv.org/abs/1711.00106 (cit. on p. 13).

Yan, Ming et al. (July 2019). 'A Deep Cascade Model for Multi-Document Reading Comprehension'. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33, pp. 7354–7361. ISSN: 2159-5399. DOI: 10.1609/aaai.v33i01.33017354. URL: http://dx.doi.org/10.1609/aaai.v33i01.33017354 (cit. on p. 15).

Yu, Adams Wei et al. (2018). 'QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension'. In: *CoRR* abs/1804.09541. arXiv: 1804.09541. URL: http://arxiv.org/abs/1804.09541 (cit. on p. 13).

Zhang, Yi, Tao Ge and Xu Sun (2020). *Parallel Data Augmentation for Formality Style Transfer*. arXiv: 2005.07522 [cs.CL] (cit. on p. 17).

Zhang, Zhuosheng, Yuwei Wu, Hai Zhao et al. (2019). *Semantics-aware BERT for Language Understanding*. arXiv: 1909.02209 [cs.CL] (cit. on p. 12).

Zhang, Zhuosheng, Yuwei Wu, Junru Zhou et al. (2019). *SG-Net: Syntax-Guided Machine Reading Comprehension*. arXiv: 1908.05147 [cs.CL] (cit. on p. 12).

Appendices

Appendix A
Additional Results

	NLG Anwser Generation		QA Anwser Generation		Passage Ranking		Answerability Classification	
	ROUGE-L	BLEU-1	ROUGE-L	BLEU-1	MAP	MRR	F1 @t	threshold
$lr = 2.5e^{-4}$, $bs = 44$								
Masque	67.20	62.54	50.64	40.72	68.67	69.16	78.51	0.4857
$\hookrightarrow$ with PairRnk + MaxCls	67.94	63.29	51.24	41.37	69.12	69.56	79.72	0.5298
$\hookrightarrow$ with NLG$^{(-)}_{aug}$	68.37	62.96	50.91	41.13	68.76	69.24	78.57	0.4393
PREAST-QA	**68.47**†	**64.14**†	**51.43**†	**43.49**†	**69.16**†	**69.64**†	**79.81**†	0.5131
$lr = 3e^{-4}$, $bs = 44$								
Masque	67.47	63.59	51.18	41.45	68.64	69.13	78.42	0.4678
$\hookrightarrow$ with PairRnk + MaxCls	67.85	**63.80**	**51.21**	**41.88**	68.84	69.31	79.56	0.4926
$\hookrightarrow$ with NLG$^{(-)}_{aug}$	67.82	62.96	50.8	41.59	68.60	69.09	78.42	0.4440
PREAST-QA	**68.02**	63.35	51.15	41.45	**68.92**	**69.40**	**79.76**	0.5126
$lr = 2.5e^{-4}$, $bs = 88$								
Masque	67.60	**63.46**	50.85	**42.05**	68.49	68.96	78.41	0.4522
PREAST-QA	**68.29**	63.15	**51.05**	41.35	**68.93**	**69.42**	**79.78**	0.5267

Table A.1: Additional results on the MS-MARCO dev set.

NLG-style and QA-style answer generation are measured in the NLG subset, Passage Ranking is measured in the ANS subset and Answerability Classification is measured in the ALL subset. Apart from the indicated hyper-parameters, *lr* for learning rate and *bs* for batch size, the rest are the same. Masque refers to the implementation of this research (Section 4.1) and PREAST-QA to the proposed model that combines PairRnk (Section 4.2), MaxCls (Section 4.4) and is additionally trained on the augmented abstractive answers NLG$^{(-)}_{aug}$ generated by the Style-transfer model (Section 4.3). MAP and MRR are measured @K, where K is the number of passages in each answerable example. F1 score is measured @t, where t is the optimal decision threshold. With bold are the best scores for each hyper-parameter setting and with † are the best scores overall.

To prove the efficacy of PREAST-QA, extensive experiments are run using different hyperparameter settings, in terms of learning rate and batch size. PREAST-QA is consistently better than Masque in all settings as measured primarily by the ROUGE-L score in NLG answer generation. Additionally, although PREAST-QA uses less QA-styled answer during training, in favour of the synthetic NLG-styled answers, it still achieves competitive results in QA answer generation, which are even better than Masque in many cases. All the models that use the proposed ranking (PairRnk) and classification (MaxCls) techniques of this research surpass those of Masque by a large margin in the corresponding metrics, MAP, MRR and F1@t. The increased performance of the methods is independent of the hyper-parameters used. Finally, there is considerable variance in BLEU-1 among the models, since the primary metric for selecting the best models is ROUGE-L. A slight decrease in the BLEU-1 of the models that are trained on the synthetic NLG data is attributed to the brevity penalty of the metric since the synthetic answers are slightly shorter than the original ones.

	NLG Anwser Generation		Passage Ranking		Answerability Classification
	Rouge-L	Bleu-1	MAP	MRR	F1 @t
PREAST-QA	**68.47**	**64.14**	**69.16**	**69.64**	**79.81**
$\hookrightarrow$ with tied embeddings	68.13	63.66	68.91	69.39	79.73

Table A.2: Results of PREAST-QA with tied input and output embeddings.

Comparison between PREAST-QA (1) without and (2) with sharing the parameters of the input (Eq. 4.1) and output (Eq. 4.22) embedding matrix. Both models were trained in the same hyper-parameter setting. The PREAST-QA run without the tied embeddings is the same as the one of Table A.1. Results are on the development set of MS-MARCO. Abstractive Answer generation is evaluated on the NLG subset, passage ranking on the ANS subset and answerability classification on the ALL subset. With bold are the best for every metric between the two runs.

In Nishida et al., 2019, the method of tying the embeddings of the input and output is followed, which has been shown to regularize the learned embeddings and lead to better performance (Inan, Khosravi and Socher, 2016). By tying the embeddings of PREAST-QA, a slight drop in the performance of all task was observed (Table A.2). This is probably caused by substantial differences in the input and output language that over-regularize the single embedding matrix, eventually leading to a worse understanding of the input text and worse predictions.

Appendix B
Additional Generated Answers

(A) **Query**: what is office depot return policy
Relevant Passage: 100% Satisfaction Guarantee: All returns and exchanges must be in original condition and include all accessories. Office Depot reserves the right to deny any return or exchange and may request identification as a condition of your return or exchange.
Reference Answer: As per the office depot return policy, office Depot reserves the right to deny any return or exchange and may request identification as a condition of your return or exchange.
Masque Answer: office depot return policy is a delaware corporation, including its subsidiaries, parent companies, or affiliates.
PREAST-QA Answer: office depot return policy is the right to deny any return or exchange and may request identification as a condition of your return or exchange.

(B) **Query**: is argintina european country
Relevant Passage: 1 Argentina is South American South America is made up of a dozen independent countries and three dependencies, with one of the largest nations being Argentina. 2 Many individuals are unfamiliar with this c.... 3 Europe on a Budget: 21 Free Walking Tours in Europe Walking tours can be a great way to get to know a new city.
Reference Answer: No, Argentina is not an European Country.
Masque Answer: no, argintina is not european country.
PREAST-QA Answer: yes, argintina is european country.

(C) **Query**: how many days are in three years
Relevant Passage: assuming that there are 365 days in each year, multiply that by 3 for the number of days in three years.... 365*3 = 1095 days in three years. In cases where there might be a leap year — simply add 1 to the outcome, thus answer is 1096 days in 3 years. aloofnerd · 9 years ago.
Reference Answer: There are 1095 days in three years.
Masque Answer: there are 365 days in three years.
PREAST-QA Answer: there are 1095 days in three years.

(D) **Query**: ok city population
Relevant Passage: Oklahoma City Population 2017. Oklahoma City is the capital and largest city of Oklahoma, and ranks 29th in the US in terms of population. At the 2010 census, its population was 579,999, but a metropolitan population exceeding 1.25 million. In 2014, Oklahoma City has an estimated population of 1.31 million people.
Reference Answer: The population of Oklahoma City is 1.31 million.
Masque Answer: the population of oklahoma city is 579,999.
PREAST-QA Answer: the population of ok city is 579,999.

(E) **Query**: how long was the battle of passchendaele timeline
Relevant Passage: Battle of Passchendaele: 31 July - 6 November 1917. Officially known as the Third Battle of Ypres, Passchendaele became infamous not only for the scale of casualties, but also for the mud.
Reference Answer: The battle of Passchendaele timeline was 31 July to 6 November 1917.
Masque Answer: the battle of passchendaele was timeline for five miles.
PREAST-QA Answer: the battle of passchendaele timeline was 31 july to 6 november 1917.

Table B.1: Additional Generated Answers from Masque and PREAST-QA.

(A): Masque has wrong content, providing an answer about “Home Depot” and not about its cancellation policy. PREAST-QA has the correct content, but the wording is not proper. (B): Correct yes/no prediction by Masque and incorrect by PREAST-QA, but both classifiers assigned a low probability of answerability, indicating that none of the models has a proper understanding of the query. The reasoning that Argentina is South America, and thus not in Europe was required. In general, an increased difficulty in answering yes/no questions was observed for both models. (C): Incorrect copying of numerical value from Masque, probably confused by 365 (wrong) being earlier in the sentence than 1095 (correct). (D): Answer is incorrect for both models in a considerably complicated numerical query, in which reasoning over time had to be made and choose the most recent estimate of population. (E): Incorrect understanding of the query by Masque, leading to an answer about length in space and not in time.

Appendix C Derivation of Multi-task Loss Weights

To ensure that the importance of the ranking task in the multi-task framework remains the same, when either of the RnkPoint (Section 4.1.3) or PairRnk (Section 4.2) methods are used, different weight factors have to be chosen. The weight factors of Nishida et al., 2019 are adopted also for this thesis, which are $\gamma_{gen} = 1$ for the answer generation loss (Eq. 4.28), $\gamma_{rnk}^{(point)} = 0.5$ for the pointwise ranking loss (Eq. 4.27) and $\gamma cls = 0.1$ for the answerability classification loss (Eq. 4.26). Thus, $\gamma_{rnk}^{(pair)}$ has to be set proportionally to $\gamma_{rnk}^{(point)}$ by adjusting it with the ratio of the expected losses for RnkPoint and PairRnk.

$$\gamma_{rnk}^{(pair)} = \frac{\mathbb{E}\Big[\mathcal{L}_{rnk}^{(point)}\Big]}{\mathbb{E}\Big[\mathcal{L}_{rnk}^{(pair)}\Big]}\gamma_{rnk}^{(point)} \tag{C.1}$$

$$\tag{C.2}$$

By assuming for simplicity, that every example has the same number of passages K, the expected losses are expected negative log-likelihood for each passage.

For RnkPoint, with $r = 0$ indicating the non-relevance class and $r = 1$ the relevant one:

$$\mathbb{E}\Big[\mathcal{L}_{rnk}^{(point)}\Big] = -\Big(P(r=0)\log P(r=0) + P(r=1)\log P(r=1)\Big) \tag{C.3}$$

, where $P(r = 1) = 1 - P(r = 0)$ is equal to the ratio of relevant passages to all the passages in the dataset, which is $\frac{N_{rel}}{N_{all}} = 0.066$.

$$\begin{aligned}\mathbb{E}\Big[\mathcal{L}_{rnk}^{(point)}\Big] &= -\Big((1-0.066)\cdot\log(1-0.066) + 0.066\cdot\log(0.066)\Big) && \text{(C.4)}\\ &= -0.2432 && \text{(C.5)}\end{aligned}$$

Similarly, for PairRnk, with $r^{pair} = 0$ indicating the more-relevant class, $r^{pair} = 1$ indicating the equal-relevance class and $r^{pair} = 2$ indicating the less-relevant class:

$$\mathbb{E}\left[\mathcal{L}_{rnk}^{(pair)}\right] = -\sum_{j \in \{0,1,2\}} P(r^{pair} = j) \log P(r^{pair} = j) \tag{C.6}$$

For passages k and i, the probability that they are of equal relevance is:

$$P(r^{pair} = 1) = P(r = 0)^2 + P(r = 1)^2 = 0.8767 \tag{C.7}$$

Then for the other two classes, and due to symmetry:

$$P(r^{pair} = 0) = P(r^{pair} = 1) = \frac{1 - P(r^{pair} = 1)}{2} = 0.0617 \tag{C.8}$$

And the expected pairwise loss is calculated as:

$$\mathbb{E}\left[\mathcal{L}_{rnk}^{(pair)}\right] = -\Big(2 \cdot 0.0617 \cdot \log(0.0617) + 0.8767 \cdot \log(0.8767)\Big) = -0.4589 \tag{C.9}$$

Thus, the weighting factor of the PairRnk loss is:

$$\begin{aligned}
\gamma_{rnk}^{(pair)} &= \frac{\mathbb{E}\left[\mathcal{L}_{rnk}^{(point)}\right]}{\mathbb{E}\left[\mathcal{L}_{rnk}^{(pair)}\right]} \gamma_{rnk}^{(point)} && \text{(C.10)} \\
&= \frac{-0.2432}{-0.4589} \cdot 0.5 && \text{(C.11)} \\
&= 0.265 && \text{(C.12)}
\end{aligned}$$

Finally, for the encoder-only models (Table 5.7), in the absence of an answer generation loss, the weight factor for the pointwise ranking method is scaled to 1, and the rest are adjusted accordingly.

	γ_{gen}	γ_{rnk}	γ_{cls}
Encoder-Only			
PointRnk + CatCls	0	1	0.2
PointRnk + MaxCls	0	1	0.2
PairRnk + CatCls	0	0.53	0.2
PairRnk + MaxCls	0	0.53	0.2
QA-models			
Masque	1	0.5	0.1
PREAST-QA	1	0.265	0.1

Table C.1: Loss weights in Multi-task learning.