A General Machine Reading Comprehension pipeline

Mémoire

Roxane Debruyker

Maîtrise en informatique - avec mémoire
Maître ès sciences (M. Sc.)

Québec, Canada

Résumé

Savoir lire est une compétence qui va de la capacité à décoder des caractères à la compréhension profonde du sens de textes. Avec l'émergence de l'intelligence artificielle, deux questions se posent : Comment peut-on apprendre à une intelligence artificielle à lire ? Qu'est-ce que cela implique ? En essayant de répondre à ces questions, une première évidence nous est rappelée : savoir lire ne peut pas se réduire à savoir répondre à des questions sur des textes.

Étant donné que les modèles d'apprentissage machine apprennent avec des examples d'essai-erreur, ils vont apprendre à lire en apprenant à répondre correctement à des questions sur des textes. Cependant, il ne faut pas perdre de vue que savoir lire, c'est comprendre différents types de textes et c'est cette compréhension qui permet de répondre à des questions sur un texte. En d'autres termes, répondre à des questions sur des textes est un des moyens d'évaluation de la compétence de lecture plus qu'une fin en soi.

Aujourd'hui, il existe différents types de jeux de données qui sont utilisées pour apprendre à des intelligences artificielles à apprendre à lire. Celles ci proposent des textes avec des questions associées qui requièrent différents types de raisonnement : associations lexicales, déductions à partir d'indices discéminés dans le texte, paraphrase, etc. Le problème est que lorsqu'une intelligence artificelle apprend à partir d'un seul de ces jeux de données, elle n'apprend pas à lire mais est plutot formée à répondre à un type de question, sur un certain type de texte et avec un certain style d'écriture.

Outre la problématique de la généralisation des compétences de lecture, les modèles d'intelligence artificielle qui apprennent à lire en apprenant à répondre à des questions retournent des réponses sans systématiquement indiquer sur quelles phrases du texte sources ils se basent. Cela pose un problème d'explicabilité et peut entrainer une mécompréhension des capacités de ces modèles.

Dans ce mémoire, nous proposons de résoudre le problème de généralisation de l'apprentissage en proposant une méthodologie générale adaptée à n'importe quel jeu de données. Ainsi, en ayant une méthodologie commune à tous les types de jeux de données pour apprendre à répondre à tout type de question, sur tout type de texte, nous pourrions apprendre aux modèles d'intelligence artificielle à se concentrer sur les compétences générales de lecture plutôt que sur la capacité spécifique à répondre aux questions. Afin de résoudre également le problème de

l'explicabilité, la métholodogie que nous proposons impose à tout modèle de compréhension de lecture automatique de renvoyer les extraits du texte source sur lequel ces réponses sont basées.

Abstract

Reading is a skill that ranges from the ability to decode characters to a deep understanding of the meaning of a text. With the emergence of artificial intelligence, two questions arise: How can an artificial intelligence be taught to read? What does this imply? In trying to answer these questions, we are reminded of the obvious: knowing how to read cannot be reduced to knowing how to answer questions about texts.

Since machine learning models learn with trial-and-error examples, they will learn to read by learning to answer correctly questions about the text they read. However, one should not forget the fact that knowing how to read means understanding different types of texts sufficiently well, and it is this that enables answering questions about a text. In other words, answering questions about texts is one of the means of assessing reading skills rather than an end in itself.

Today, there are different types of datasets that are used to teach artificial intelligences to learn to read. These provide texts with associated questions that require different types of reasoning: lexical associations, deductions from discrete clues in the text, paraphrasing, etc. The problem is that when an artificial intelligence learns from only one of these datasets, it does not learn to read but is instead trained to answer a certain type of question, on a certain type of text and with a certain writing style.

In addition to the problem of generalizing reading skills, artificial intelligence models that learn to read by learning to answer questions return answers without systematically indicating which sentences in the source text they are based on. This poses a problem of explicability and can lead to a misunderstanding of the capabilities of these models.

In this thesis, we propose to solve the generalization issue of learning from one dataset by proposing a general methodology suiting to any machine reading comprehension dataset. Thus, by having a methodology common to all types of datasets to learn how to answer any type of question, on any type of text, we could teach artificial intelligence models to focus on general reading skills rather than on the specific ability to answer questions. In order to also solve the issue of explanability, the methodology we propose impose any machine reading comprehension model to return the span of the source text its answers are based on.

Contents

Résumé ii

Abstract iv

Contents v

List of Tables vi

List of Figures vii

Glossary viii

Acknowledgment xii

Introduction 1

1 Language representation as machine reading skill 3
- 1.1 Static high-dimensional token representation . . . 4
- 1.2 Static dense word embedding . . . 5
- 1.3 Contextualized word embeddings . . . 8

2 Machine Reading Comprehension 15
- 2.1 Machine Reading Comprehension challenges . . . 15
- 2.2 Machine Reading Comprehension models and architectures . . . 22

3 A pipeline architecture for generalisation in MRC skills 37
- 3.1 Methodology . . . 39
- 3.2 Experiments . . . 43
- 3.3 Implementation details . . . 44
- 3.4 Results . . . 50

4 Discussion 61
- 4.1 Report on the hypothesis . . . 61
- 4.2 Analysis and future works . . . 65

Conclusion 72

Bibliography 74

List of Tables

1.1 Occurrence of each entry of the vocabulary within a corpus of 4 sentences. The last column gives the occurrence of each entry in a window of 2 tokens around the token "bark". 6

2.1 Illustration of each reasoning type with an example, extracted from Trischler et al. (2016a) 16
2.2 Overiew of existing MRC datasets. 21

3.1 Phases of the pipeline associated with the models used to accomplish them. . 40
3.2 Example extracted from ROPES development set 45
3.3 The table provide respectively the average (avg.) number (#) of paragraphs (par.), the minimum (min.) number (#) of paragraphs (par.), the maximum (max.) number (#) of paragraphs (par.), the average (avg.) , the average (avg.) number (#) of answer spans (ans.), the minimum (min.) number (#) of answer spans (ans.), the maximum (max.) number (#) of answer spans (ans.) per question and finally the percentage of questions for which no span has been retrieved (% No ans). 49
3.4 SQuAD evaluation. 52
3.5 COQA evaluation. 53
3.6 RACE evaluation. For the SOTA and the baseline, the only measure available is the accuracy 54
3.7 WikiHop evaluation. 55
3.8 NarrativeQA over summaries evaluation. 56
3.9 NarrativeQA over stories evaluation. 57
3.10 ROPES evaluation. 58
3.11 Natural Question evaluation on short answers. 60

4.1 The table provides on the left part the average (avg.) number (#) of answer spans (ans.), the minimum (min.) number (#) of answer spans (ans.), the maximum (max.) number (#) of answer spans (ans.) per question, the percentage of questions for which at least one span has been retrieved (% >0 ans). The right part displays the F1-score and exact match (EM) scores of the answer spans retrieved compared with the ground truth answers 66
4.2 Scores of span extracted with each individual model normalized on the score of their weakly retrieved spans. 68
4.3 Scores of answers generated with UnifiedQA for each dataset using the complete passage as context for the left section and the extracted span as context for the right section. 69

List of Figures

1.1 RNN general architecture (better with color) with one layer (light orange). . . 11
1.2 BiRNN general architecture (better with color) with one bi-directional layer . . 11
1.3 BERT general architecture (better with color). 13

2.1 Typical algorithm to solve MRC tasks. The data states are represented by rectangle while the processing phases are represented by hexagons. The dotted-line arrow indicate phases or representations which are not systematic but specific to some tasks. 23
2.2 Similarity Matrix S computed in Seo et al. (2016) 27
2.3 BERT for Question Answering (better with color) with an example of passage/question extracted from Trivia QA. The blue rectangles represent each word representation while the dark pink rectangles represent $\mathbf{A}_1$ and the lila ones represent $\mathbf{A}_2$: the probability distribition for each word to be the beginning or the end of the answer span. The ground truth answer to the question is "Elton John". 30
2.4 Example of multi-mentioning extracted from TriviaQA. 31
2.5 Transformers general architecture (better with color) with an example extracted from TriviaQA 33

3.1 Our Pipeline. The optional phases (changing from a dataset or from one model to another) are represented with dashed lines while the mandatory ones are depicted with plain ones. The red rectangle emphases the phases which are specific to our pipeline. 38

Glossary

For sake of readibility, along the thesis, acronyms are used. Each acronym which have been used is listed here with its complete name in alphabetical order. Also, words and concepts which could be used with a specific meaning in this thesis or which could have an ambiguous meaning or which are used as synonyms are also listed here with a short definition.

Acronyms

- CNN : Convolutional neural network
- DL : Deep learning
- E-MRC : extractive machine reading comprehension
- EM : Exact-match
- FCNN ; Fully-connected neural network
- G-MRC: generative machine reading comprehension
- GRU : Gated recurrent unit
- HRC : Human reading comprehension
- IR : information retrieval
- LM : Language model
- LSTM : Long-short term memory
- MC-MRC: multi-choice machine reading comprehension
- ML : Machine learning
- MLM : Masked language model
- MR : Machine reading

- MRC : Machine reading comprehension
- NLP : Natural language processing
- NN : Neural network
- NR : Naive retrieval
- NSP : next sentence prediciton
- OOV : out of vocabulary
- OR : Oracle retrieval
- RC : Reading Comprehension
- RNN : Recurrent neural network
- SOTA : State-of-the-art
- ZSL : zero-shot learning

Definitions and synonyms

- Workers are human worker which label data. In this thesis, the workers mainly refer to the human who have been hired in order to retrieve, chose or write the answers to the questions in order to constitute the MRC datasets.
- Words and token are, in this thesis, used as synonyms. For sake of readability, in this thesis, those two terms also refer to punctuation signs, number and special characters.
- A pipeline is a succession of independent, reusable, modular phases that can then be pipelined together to create a combination of models which solve a task end-to-end.
- An architecture refers, in this thesis, to a nerual network architecture. A neural network solution is composed of artificial neurons which are organised in architecture. An architecture is then the general structure of the neural network (amount of layers, number of neuron per layer, kind of activation function used, etc).
- A model is the instance of the architecture trained on data; it is composed of the architecture with the weights which have been learned. and model
- Seq2Seq and encoder-decoder are used as synonym. They are an architecture desgigned for text generation with an first sub-architecture which encodes a sequence and another sub-architecture which decode another sequence. More details are given in Section 2.2.2.

- dataset, task and challenge are used as synonyms in this thesis. Indeed, an MRC dataset propose texts and questions about this text to be answered, so it is a task and a challenge as well.
- Step and phase are used in this thesis as synonyms.
- A cloze test is a task where some words in a text are masked and have to be retrieved.

Apprendre à lire, c'est allumer du feu ; toute syllabe épelée étincelle.

Victor Hugo - Les Misérables

Acknowledgment

Je lis souvent dans des sections de remerciements des phrases telles que "Cette thèse n'aurait pas été possible sans le soutien de X, Y et Z.", avec X, Y et Z pouvant etre des collègues, des membres de la famille et des ami.e.s. Avant de commencer à travailler sur ce mémoire, je ne réalisais cependant pas l'importance de cet entourage pour accomplir ce long et difficile, bien qu'enrichissant, travail. Qu'il soit professionnel ou personnel, chaque personne qui m'entoure aura su me donner la motivation de continuer.

Merci à Richard Khoury qui a continué de me soutenir malgré les changements de directions de mon projet de recherche, malgré les épreuves administratives et personnelles qui m'ont fait douter plus d'une fois. Merci à Thalès, qui m'a accueilli et apporté toute l'aide nécessaire via une supervision sans faille. Rana et Maryam, avoir des femmes aussi compétentes à mes coté ne m'a pas aidé que d'un point de vue technique mais m'a aussi et sourtout apporté des figures inspirantes. Grace à vous deux ainsi qu'à Audrey Durand et Anne Lauscher, les quatre personnes les plus compétentes avec qui il m'a été donné de travailler, je n'ai jamais eu à douter que l'on a notre place dans un domaine pourtant réputé si masculin. Merci à l'universié de Mannheim et particulièrement à Goran Glavaš qui m'a accueilli plusieurs mois dans son équipe et aidé à redéfinir mon projet de recherche et me donner les outils pour le mener à bien. Évidemment, les collègues du GRAAL qui n'ont jamais cessé de m'impressionner par leur zèle pour la recherche, presque égale à leur zèle pour l'humour douteux. La bonne humeur et les pauses volley ont vraiment manqué après le début de la pandémie.

Merci aussi aux 18 différents colocataires que j'ai eu depuis le début de ce travail. Je n'aurais définitivement pas réussi à finir ce travail au travers des multiples déménagements sans avoir un entourage compréhensif et à l'écoute de chaque moment de doute, toujours prêt.e à cuisiner et me forcer à prendre des pauses parfois, ne pas en prendre d'autre fois. Enfin, Ellie, merci infiniment pour la relecture et les commentaires.

Introduction

The ability to read refers to the ability to decode the content of a written text, while reading comprehension (RC) ability is the ability to understand its meaning. This goes from extracting its explicit content given a question to interpreting its underlying meanings using also the context of the story, such as the year it has been written or the country where the author comes from. RC is thus influenced by three interrelated factors: the reader, the text and the context (Wixson and Lipson (1991)).

In the educational path, novice readers learn first to understand the global meaning of a text and then move towards more and more detailed analysis (Grellet (1986)). This evolution is the consequence of several factors including (1) reading experience, which leads to reading fluency, (2) RC supervision, which is usually performed by a teacher, and (3) prior general knowledge acquisition (Kozminsky and Kozminsky (2001)). Given the depth and influence of each of these factors, RC can mean different things to different people.

In the area of computer science and artificial intelligence, RC skills are a focus of the natural language processing (NLP) community, and one of their main challenges is retrieving information through question-answering (QA) tasks. The basic machine reading comprehension (MRC) dataset templates provide texts with question/answer pairs about their explicit content. Some datasets with paragraph-sized texts and factual questions that can be mostly answered by returning spans or paraphrases (extractive MRC) are most of the time already well-handled. For instance, state-of-the-art results using SQUAD ((Rajpurkar et al. (2016a)) reach a F1-score of 93% and TriviaQA (Joshi et al. (2017a)) 83% . [1] However, there are also MRC challenges that require the selection of their answer among several candidates (multi-choice MRC) or the generation of free-form answers (generative MRC). Multi-choice MRC datasets are usually oriented on deep reasoning skills (Lai et al. (2017a), Trischler et al. (2016b)), while generative MRC datasets include also writing skills.

Until recently, each MRC dataset or at least each category of MRC tasks (i.e extractive, generative, and selective) would have specific architectures associated in order to learn to answer the questions. But since 2019, models such as the one proposed in Raffel et al. (2019)

[1] Based on leaderboards of the datasets at the date of March 2022

have been showing the efficiency of common learning over different kind of NLP tasks. With this current master's work, we aim also for more generalisation but by proposing a unique solution for any existing MRC tasks.

In this thesis, we thus propose a general methodology to tackle the main existing MRC tasks based on hytpohesis. This methodology consists of designing a general pipeline composed of several stages with the hypothesis that any MRC tasks, including those requiring reasoning over underlying narrative elements, will benefit from a systematic span extraction to locate where the answer elements are in the provided passages. The context on which the models would base themselves to answer the questions could then be reduced to the extracted spans. In addition to the hypothesis that the presence of this extractive stage would improve the results, compared to when the same models would use the original complete passage, user's understanding would be improved by the clear quotation, showing what information the models based themselves on to provide an answer. In the same way that a pupil is asked to justify their answers in order to enable the teacher to evaluate their comprehension, the MRC skills could be assessed from this information. This gives the model explainability and makes it possible to track where and why it succeeds or fails.

Many ways to implement practically solutions which follow this methodology are possible. And of course not any solution could verify our three hypothesis. Some research and trials have to be done to chose the relevant architectures and training them with appropriate datasets. This thesis is laying the foundation stone by proposing a first combination of architectures trained on various dataset. Our work can then be used as a baseline to orientate the future implementation which would validate our three hypothesis.

This thesis is composed of four chapters. Chapter 1 is a refresher on the fundamental works in the field of language representation, which is necessary to understand what machine reading (MR) skills are and trace their evolution through the years. Chapter 2 details the current state of the field of MRC by providing the different challenges it covers and explaining how those different challenges have been tackled. Based on the knowledge gained in the previous two chapters, the description, application, and evaluation of our methodology in handling any existing MRC challenge is given in Chapter 3 using one concrete solution proposition tested on several datasets. The analysis of those experiments is given Chapter 4.

Chapter 1

Language representation as machine reading skill

The first step of reading comprehension is reading, or the ability to translate written words or symbols into meaningful representations. For a machine, those meaningful representations are commonly vectors that enable computations between them, and which are helpful to perform well on a wide variety of NLP tasks.

Language modeling (LM) tasks are associated with the MR skills since they work on language deciphering and enable downstream NLP tasks. LM tasks aim at predicting a word following a sequence of words. When the word to predict has to be predicted using preceding and following words, this is a cloze-test task. [1]. So, LM tasks are included into cloze-test ones. Therefore, LM models are taught to assign scores or probabilities to target units of text based on the context in which they use to appear. For instance, in the sentence "My dog hates children, he [PLACEHOLDER] as soon as he sees one", a word replacing the placeholder must be returned based on the beginning of the sentence. The most common LM models are token-based LM [2] , which have tokens as target units, and they thus return the probability that a token occurs, knowing its context. The predictive models are usually trained using preceding and succeeding elements of the target word as context (cloze-test) but the goal of a LM is to be able to predict, after training, using only preceding elements. Other units of texts can be used for LM instead of tokens, such as characters (Zhang and LeCun (2015)) or sentences (Pichotta and Mooney (2016)). In this thesis, we will use the token-level LM as a pre-training step to get representations.

The probabilities that a word occurs in a given context are computed based on the distributional hypothesis (Joos (1950); Harris (1954)), which assumes that words appearing in a similar context share a similar meaning. Words are represented in a vector space and are geometrically

[1] The definition of cloze-test task is given in the part

[2] Throughout this thesis, the expression "word" and "token" are used as synonyms.

near words with a similar distribution. In other words, words with similar meanings will be represented by vectors with low euclidean distance while words which are not related will have high euclidean distance (Mikolov et al. (2013)). The resulting vectors are called semantic vectors, or word embeddings for word-based LM. For example, one would expect that a good model creates word embeddings which are close for related words such as "bark" and "meow" while being more distant to unrelated words, such as "bark" and "tractor".

The transcription of a text into semantic vectors is the first step for any NLP task which uses neural networks (NN), for the simple reason that textual entities must be numerically translated while preserving as much meaning as possible. With a dictionary of tokens belonging to a language and each token associated a static ID number, a NN would take a text composed of n words as input represented by a list of n ID numbers. The input is transformed in a word embedding thanks to the embedding layers. The output of the embedding layers is a n d-dimensional word embedding is represented by $n \times d$ neurons.

The word embeddings are trained from a high quantity of texts and are self-supervised. Indeed, the training data collection consists of extracting complete sentences and randomly removing some words from it (Mikolov et al. (2013)). Since such models do not require manual data annotation, the amount of data available, thanks to internet resources, is abundant (at least in English). This high quantity enables the models to learn from a wide variety of examples and results in high-quality representations. Such models would then learn from a given training set to estimate the function to translate correctly a word into its numerical representation.

In this chapter, we present the main steps in the evolution of methods to compute word embeddings: Section 1.1 presents static high-dimension token representation techniques, Section 1.2 static low-dimensional token representation ones, and Section 1.3 dynamic low-dimensional token representation ones. Each method relies on the distributional hypothesis. The threshold to distinguish between high and low dimension is defined here as follows: if a word is represented by a vector which has as many dimensions as the size of the vocabulary it learned from, it is a high-dimensional vector. Otherwise, it is a low-dimensional vector.

Both static and dynamic vectors learn how to use the surrounding context (i.e the surrounding words) of a target word in order to predict the masked target word. The difference lays on the fact that static word embeddings model form then a static dictionary with one vector per token of the dictionary based on the training while dynamic word embeddings provide an adapted representation of it in each new context it appears.

1.1 Static high-dimensional token representation

Based on the distributional hypothesis, one can look the window of words which surround a target word (its context) and represent the target word with the list of words which compose

the context. In order to illustrate better how high-dimensional token representations are computed, an example composed of the 4 following sentences is proposed:

1. My dog hates children, he barks as soon as he sees one
2. The tree in my garden is losing its bark, is it serious?
3. My dog barks in his sleep
4. My dog can't stand our absence. He barks all day when we leave for work

The complete vocabulary in this example, is composed of the 33 lemmas which are listed in Table 1.1. With the target word "bark" and the context defined as a window of two words before and two words after the target word, the "context occurrence" column is built. For the remaining part of this section, the text is processed so each word is lowered and only its lemma is kept. However, for the sake of readability, we keep using "word" or "token" to refer their post-processed forms. Usually numerical values, special characters, as well as punctuation, could be considered as tokens. For this example, commas and points are not considered as words. Therefore, "he" appears in the context of "bark" two times within the four sentences. "he" also appears two times in total within the four sentences.

One possible way to represent a target word numerically is to create co-occurrence vectors with one value per word of the vocabulary and to give the occurrence of the word in the context (n surrounding words) as value. . With our example, this would be a 33 dimensional vector with the entries of the column "context occurrence" of Table 1.1 as values. Representing a word with such vector, when they are trained on big corpus of texts, offers the possibility to capture semantics and syntactic elements of a vocabulary. Indeed, words with the same part of speech and close context apparition, such as "cat" and "dog" can be easily matched (i.e by using cosine similarity between vectors). Each dimension is understandable by a human because each of them is associated with a word. Theoretically, the bigger the corpus and the vocabulary is, the better the representation. In practice, knowing that the number of words in English is estimated to be 1,022,000 (Michel et al. (2011)), it results in very large and sparse vectors. This is problematic computationally: co-occurrence matrices take a lot of memory space, and this space is mainly filled with 0, which makes the computation inefficient. Solutions are presented in Section 1.2 in order to reduce the dimensionality and the sparsity of those vectors.

1.2 Static dense word embedding

Since 2014, several methods which solve the sparse, dimensional, and sequence issues by learning to represent words of a vocabulary with dense low-dimensional vector (dense) have been published.

Vocabulary	**total occurrence**	**context occurrence**
my	4	1
dog	3	1
hate	1	0
child	1	1
he	2	2
bark	4	0
as	2	1
soon	1	1
see	1	0
one	1	0
the	1	0
tree	1	0
in	2	1
garden	1	0
is	2	1
lose	1	1
its	1	1
it	1	1
serious	1	0
his	1	1
sleep	1	1
can	1	0
not	1	0
stand	1	0
our	1	0
absence	1	1
all	1	1
day	1	1
when	1	0
we	1	0
leave	1	0
for	1	0
work	1	0

Table 1.1: Occurrence of each entry of the vocabulary within a corpus of 4 sentences. The last column gives the occurrence of each entry in a window of 2 tokens around the token "bark".

One solution is to compute a co-occurrence matrix, as described in Section 1.1 and to reduce the dimensionality via singular value decomposition. Pennington et al. (2014) also propose to use the co-occurrence matrix in a way that it generates significantly better results than SVD methods. They propose the global vector (GloVe) method. Vectors are created thanks to a machine learning model which learns to predict occurrence probability. The probabilities are computed using the co-occurrence matrix. The probability that the word i occurs knowing that the word k is in the sliding window is defined as being equal to the co-occurrence amount of i and k divided by the total amount of occurrence of i in the corpus($\mathbb{P}(i \mid k) = \frac{X_{i,j}}{X_i}$). For example, the probability that the word "dog" appears when we know that "bark" is in the context would be $\mathbb{P}(i = \text{dog} \mid k = \text{bark}) = \frac{1}{3}$. Similarly, $\mathbb{P}(i = \text{absence} \mid k = \text{bark}) = 1$ The authors want to learn word embedding by learning to predict $\frac{\mathbb{P}(i|k)}{\mathbb{P}(j|k)}$ for each triple i, j, k. Indeed, two words appearing in the same context or not appearing in the same context would have $\frac{\mathbb{P}(i|k)}{\mathbb{P}(j|k)}$ close to one, while two words which appear in different contexts will have values which get either close to 0 or to ∞. The recommendations of the authors for semantically efficient representation is to use context of 10 words (5 before and 5 after the target word).

While the GloVe vector's computation is based on global context observation (a word vector is designed based on a co-occurrence matrix which is computed on a large corpus), Mikolov et al. (2013) propose word2vec, which are vector representations based NN models which learn from the local context of the words. In other words, a pre-representation of a target word in its overall context is not computed in word2vec but its representation is rather based on probabilities computed example by example, as it is the case for any other ML models based on NN.

word2vec is a predictive model that can either predict the word based on the context (Continuous Bag of Words —CBoW) or predict the context based on a word (Skip-Grams —SG). The usual context is a four-word context window (two before and two after). The SG (CBoW) configuration first transforms a word (a context) into an n-dimensional vector through the representation layer, then using the output vector from the representation layer, computes the probability for each word to be found in the context (to be the expected word) via the prediction layer. The model learns in a self-supervised way to maximize the probability to retrieve the actual context by adjusting the vector output by the representation layer. The representation layers consist of a two-layer fully connected neural network architecture (FCNN). For the CBOW model, each word of the context gets a d-dimensional representation thanks to the representation layer. The representations among the context words are averaged. After the training phase the weights are frozen, the prediction layer is disregarded, and the word vector is given by the output of the representation layer.

Despite their different philosophies, Levy and Goldberg (2014) prove that word2vec implicitly factorizes a word-context matrix, which brings the two models closer together since both are based on the co-occurrence counts between words. In terms of performance, GloVe usually

performs better on similarity tasks while word2vec is stronger for analogy tasks (Levy and Goldberg (2014)).

Other techniques have been proposed to represent words with static dense vectors but GloVe and word2vec remain the most popular. For instance, context2vec (Melamud et al. (2016)) is a variation of word2vec's CBOW. context2vec applies one BiRNN layer (RNN are explained in Section 1.3), which gives two vectors, followed by a FCNN as representation layer. One of its main advantages is that the context is not limited to a fixed window of k-words but is rather sentential. Only one d-dimensional vector is computed to represent the context instead of k-vectors which need to be averaged. With a sentence composed by N tokens, the BiRNN layer computes a contextualized representation of the token t at the position i, with $1 \leq i \leq N$, which is the concatenation of the representation of the words appearing on the left of the target word with those on the right. The left representation is computed by applying the RNN from the first token to the $i - 1^{\text{th}}$; the resulting $i - 1^{\text{th}}$ vector is returned. The right representation operates from the N^{th} to the $i + 1^{\text{th}}$ one and returns the $i + 1^{\text{th}}$ representation. The concatenated vector is sent to a FCNN which helps to represent non-trivial dependencies between the two sides of the context and returns the contextualized d-dimensional representation for the token i.

Being able to turn an arbitrary word into a meaningful and computable vector representation has created a revolution in NLP because their semantic and computational structures resulted in significant capacity improvement in many downstream tasks, such as MRC. However, issues remain, such as the difficulty to represent polysemous words, and, in a more general way, the fact such modeling systems attribute one vector per token while its meaning can vary highly depending on the context in which it appears. Also, handling out of vocabulary (OOV) words, i.e words that are not present in the texts used for learning the pre-trained embeddings, is problematic.

1.3 Contextualized word embeddings

Static word embeddings use the context in which each word appear to infer its meaning, but this context is only used during the training phase. After the training phase, one static representation per word is given as a dictionary (the word i is represented by $\mathbf{w}_i \in \mathbb{R}^d$). Contextualized word embeddings (CWE) aim at representing the meaning of words in their context both during the inference time and during the training time, rather than just during the training time. The representations of tokens are then dynamic from one occurrence to another, compared to the previous word representations that we have seen until now, which were static. Before describing how contextualized representations can be computed, the difference between the static word embeddings and the contextualized one is illustrated with an example.

Illustration of the difference between static and dynamic representation

Let's reuse the toy example given in Section 1.1. The token "bark" would always be represented with the same vector in static word embedding applications, regardless of whether it refers to the peel of the tree or the sound of the dog. Contextualized word embedding representation would vary in each sentence. The BERT-base model of the *transformers* library [3] [4] has a dictionary of word pieces [5] which associate each word piece to an ID number. "bark" is then represented with the unique token ID 12001 but varying representations from one context to another. In the following sentences, we compare the cosine similarity of the BERT token representation "bark" between the first sentence and the three next ones.

1. My dog hates children, he barks as soon as he sees one (baseline sentence)
2. The tree in my garden is losing its bark, is it serious? (similarity : 0.4490)
3. My dog barks in his sleep (similarity : 0.5292)
4. My dog can't stand our absence. He barks all day when we leave for work (similarity : 0.5982)

In the second sentence, where "bark" refers to the tree instead of the dog's sound, the distance with the similarity with the first sentence is lower in comparison with the sentences 3 and 4. Also sentences 1 and 4 have more similarity in their syntax and semantics, which leads to higher similarity score, compared with the two other sentences.

Model architectures

Word representation have benefited from the progress made in the field of deep learning architectures. Two families of deep learning architectures are used for learning word representation : recurrent neural networks (RNN) and transformers.

RNNs' general architecture for LM tasks is depicted in Figure 1.1. LSTM (Hochreiter and Schmidhuber (1997)) and GRU (Cho et al. (2014)) architectures belong to this family. In those architectures, texts are encoded word by word, following the temporal order of language. Passing through the encoder, the information accumulates along the processing of the sentence and can give meaning to the sequence while remembering important parts and forgetting the unimportant ones (relatively to a given task and corresponding annotations). At each new word seen composing a sequence, the new information is encoded and computed with the previous information.

[3] https://pypi.org/project/transformers/

[4] BERT is a contextualized word embedding model which is going to be detailed in the next pages, within the same section.

[5] the difference with tokens are explained below in the same section; but for the example of "bark" the word piece and token are the same.

Some RNN use a BiRNN, which uses also the sequence backward : it takes as input the sequence from its last word until the target word. The target word is predicted concatenating the forward and backward representation of the target word as depicted in Figure 1.2. By using BiRNN, as context2vec described in Section 1.2, the context on the left and on the right of a pivot word can be captured. ELMo (Peters et al. (2018)) apply L BiRNN layers as representation layers. ELMo has then $1 + 2L$ representation layers (rectangles concatenated by row in Figure 1.2), the first one being a static word embedding representation and 2 per BiRNN layer. Although context2vec and ELMo have close architectures, context2vec saves the word embedding statically after training and its pre-trained embedding can be used as input of a downstream architecture, while ELMo freezes the weights of the model, removes the softmax layer and connects the architecture to another architecture designed for a downstream task to be performed. So for each new entry, $1 + 2L$ context representations will be generated. Depending on the task which has to be accomplished, ELMo learns during the training phase which linear combination between the $1+2L$ representation is the most helpful for the task. For instance, for tasks which are oriented on syntax comprehension, such as name entity recognition, models are shown to emphasize more the lower layers while tasks which require reasoning skills, such as MRC, would put more weights on higher layers. From this observation, the authors interpret that the lower levels of the representation layers encode more syntactical information and by getting deeper, semantics is more and more encoded. For each NLP architecture which would intend to use static word embeddings as input to accomplish a task, pre-trained ELMo can be used instead. The training cost of the specific models would not be too much higher than using static word embedding regarding the fact that the weights of the representation layers are frozen and that only the weights of the linear combination ($1 + L$ parameters) must be learned.

Even though ELMo demonstrates a high increase in capacities compared with static word embedding methods, its capacities are limited because of the RNN architecture: as the input texts get longer, information gets lost because of the vanishing gradient effect (Hochreiter (1998)). Later models preferred using an attention mechanism to fix this issue.

Attention is a mechanism which provides insight through distributional weights about the words in a text which are relevant to a target word. The meaning of "relevant" depends on the context. In the case of machine translation, an attention mechanism will put high attention weights on the direct translation of the word in an input language to translate it in the target language. While this example computes attention across two different texts (i.e text A and text B), self-attention consists in computing the weights for words of one text (text A). With the example "My dog hates children, he barks as soon as he sees one", a self-attention layer would result in a 12-dimensional vector for each word, with each value of the vectors being the self-attention score with the target word. For instance, one can imagine that the word "he" would attribute a high self-attention score to "dog" to which it refers.

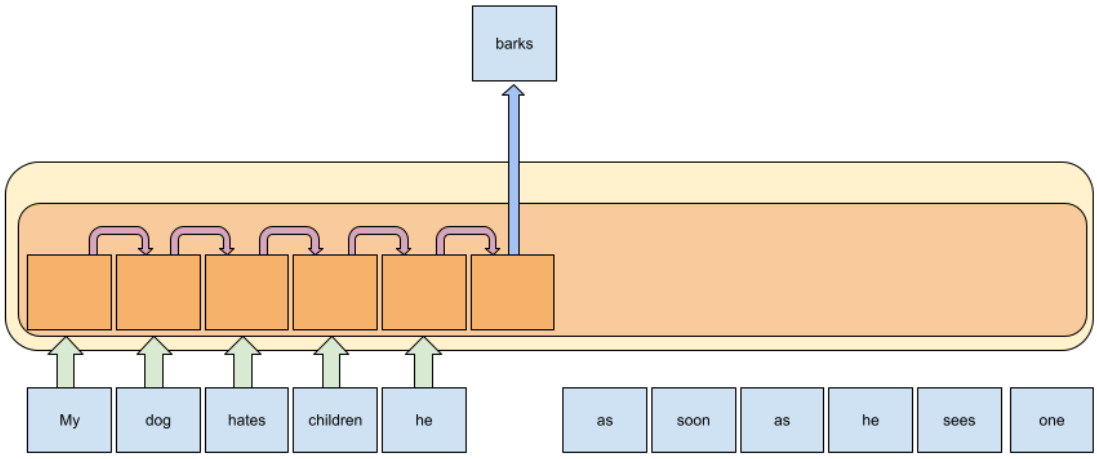

Figure 1.1: RNN general architecture (better with color) with one layer (light orange).

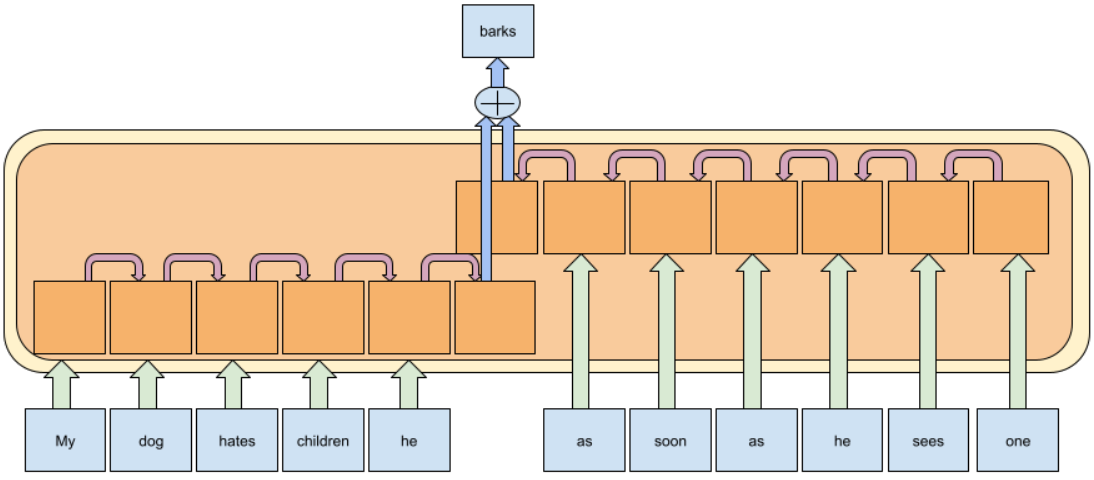

Figure 1.2: BiRNN general architecture (better with color) with one bi-directional layer

There are several ways to compute attention weights, some of which are described in Section 2.2.1. Attention weights can be used to create contextualized representations since they can weigh the importance of the context to define a target word. The weights can be learned and applied within an architecture, like after RNN layers, in order to overcome vanishing gradient issues or they can be used as the central architectural component, as in the transformer architecture (Vaswani et al. (2017)).

Tranformers are also deep learning architecture widely used for learning contextualized language representation. They contrast with the RNN with their main feature: they are attention-based. This means that their architectures are only composed of attention blocks, which is the reason why authors have called their paper presenting transformers "Attention is All You Need" (Vaswani et al. (2017)). Tranformers are encoder-decoder models which can be used for language generation. For language representation, only the transformer encoder can be trained, and it results in BERT Devlin et al. (2018), depicted in Figure 1.3.

Compared with the FCNN or the RNN, the transformers do not get the input sequentially, thus it has a lack of information regarding the position of the tokens one to another. This is why positional encoding has been proposed thanks to word pieces. The positional encoding is an unsupervised text tokenizer, which produces word pieces: known tokens and also out-of-vocabulary sub-word tokens. Using WordPiece (Socher et al. (2013)) has the advantage of reducing the vocabulary size and collecting more examples for each instance. A first representation per word piece is then given as input to the transformer encoder (light yellow in Figure 1.3).

The output of the pre-trained model is a transformed representation of the input accordingly to a task. For language representation, BERT is trained on masked LM and on next sentence prediction tasks.

Masked LM Task. Traditional language models predict a target word in a sentence by using its predecessors in the sentence. Compared to ELMo, which can be described as "shallowly bidirectional" because it predicts a target word using its predecessors *and* successors separately before concatenating the representation, BERT is "deeply bidirectional". Indeed, it trains one deep model using both sides simultaneously thanks to its masked language model task (MLM), which consists in masking with random words or placeholders the words to be predicted instead of stopping the model before seeing the target word. This difference allows BERT to look at the representations from both sides simultaneously, which increases its analytical power.

Next Sentence Prediction. BERT trains on a next sentence prediction in addition to MLM, which is particularly useful for question answering (QA) tasks. Given two sentences, the model predicts whether the second sentence is a logical continuation of the first or not. This is

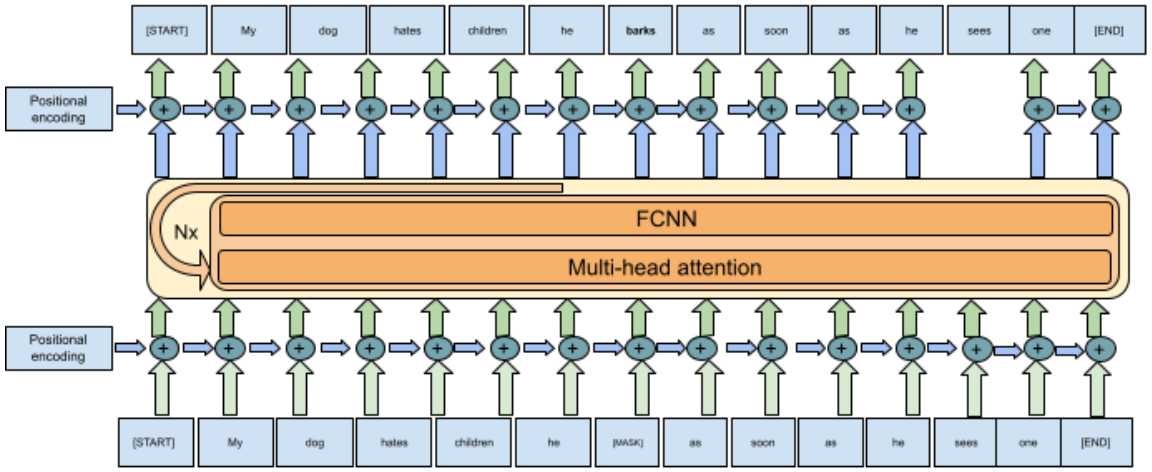

Figure 1.3: BERT general architecture (better with color).

a binary classification task, predicting if yes/no the second sentence is the direct succeeding sentence of the first one. This task helps in understanding the relationship between two sentences, which is particularly useful in the context of QA, where the first sentence is then the question and the second is a paragraph.

The transformer's encoder is composed of several attention blocks (light orange in Figure 1.3), which are composed of the multi-head attention layer and the FCNN layer. A multi-head attention layer is composed of several attention heads which are concatenated and reduced with linear projections. This combination of multi-head attention layer and FCNN repeats N times, where N is a hyper-parameter to be set during training. At the $N + 1$th iteration, the final representation is used to return the final output (target word for LM, classification for next sentence prediction, answer for MRC, etc.).

BERT is far deeper than its predecessor with its 12 layers for BERT-base and 24 for BERT-large. As a comparison, ELMO had 5 layers. As explained above, the first layers of a deep learning model tend to represent simple structures while deeper layers will be able to encode more complex concepts. This hierarchy is well shown in Peters et al. (2018) where named entity extraction tasks, which has to focus more on character-representation to fulfill the task, would mostly use the first layers while sentiment analysis tasks, which require deep semantic understanding of a text, use deeper layers. Having a deep architecture is then a feature which enables having a complex function represent an input and which can lead, with a good architecture, to semantically precise representation.

BERT have been adopted and demonstrate better performance than other models to embed words according to given NLP tasks, including MRC. Thanks to the evolution of word representation techniques, reading skills have greatly evolved over the last few years and

so did the performances of downstream tasks.

In addition to the kind of architecture, two learning approaches can be distinguished when it comes to CWE: the feature-based approach and the fine-tuning approach. In the feature-based approach, an LM is first trained and then integrated to a downstream model, but its weights are frozen (Koehn et al. (2003)). In ELMo, during the training phase of the downstream task, the pre-trained weights of the embedding models are frozen, but it is possible to combine the output of the respective layers. Indeed, the final vector representing a token is a linear combination of these several layers where weights are learned depending on the downstream task during training. The inclusion of lower-level information and the adapted combination depending on the downstream task is a major feature that is beneficial to word representations. By contrast, the fine-tuning approach also pre-trains LM but their weights are not frozen, which means that a few task-specific parameters must be added during the specialization learning phase (Dai and Le (2015)). For instance, in the case of binary sentence pair classification as a downstream task, only one additional classification layer is required. Therefore, the computational cost of the fine-tuning approach is higher and the LM architecture has to be adapted for downstream tasks with only a few additional layers, which can be complex to design.

For both techniques, it has been noticed that the highest levels of the network tend to represent better semantic information and the lowest levels highlight syntactic specificities (Peters et al. (2018); Devlin et al. (2018); Peters et al. (2019)).

Chapter 2

Machine Reading Comprehension

The research area of MRC focuses on the ability of AI to understand written text documents. There are several levels of comprehension of a text, ranging from direct understanding of the explicit content to the interpretation of the underlying narrative elements. Many different datasets and challenges have been proposed to train and evaluate MRC systems, emphasizing on various aspects of text comprehension. We expose the existing variety of challenges in Section 2.1 and, in Section 2.2, the proposed solutions which have been designed to tackle them.

2.1 Machine Reading Comprehension challenges

Machine reading skills of MRC models are conventionally tested through question-answer (QA) challenges: agents are asked to answer questions about specific texts. Answering those questions require different types of reasoning, initially presented in Chen et al. (2016) as:

1. **Word Matching** This is the simplest level of reasoning, where the agent retrieves a span which contains the exact same words as the question.
2. **Paraphrasing** This is a form of reasoning that allows the agent to retrieve an answer span which uses a different wording from the question.
3. **Inference** This is when the agent can retrieve an answer span that does not explicitly contain the elements asked in the question at all. The agent thus needs to reason to connect the question to a textually unrelated answer.
4. **Synthesis/Multi-Hop Reasoning** This is when the agent needs to piece together an answer from information found in several different spans of text.
5. **Detecting unanswerable questions** This is the level of reasoning needed for an agent to realize that the information in a text is too ambiguous or insufficient to answer the question.

Reasoning type	Example
Word matching	Q : When were the findings published ? A : Both sets of research findings were published Thursday
Paraphrasing	Q: Who is the struggle between in Rwanda? A: The struggle pits ethnic Tutsis, supported by Rwanda, against ethnic Hutus, backed by Congo.
Inference	Q: Who drew inspiration from presidents? A: Rudy Ruiz says the lives of US presidents can make them positive role models for students.
Synthesis/ MH Reasoning	Q : Where is Brittanee Drexel from? A : The mother of a 17-year-old Rochester, New York high school student ... says she did not give her daughter permission to go on the trip. Brittanee Marie Drexel's mom says...

Table 2.1: Illustration of each reasoning type with an example, extracted from Trischler et al. (2016a)

Table 2.1 gives an example from the NewsQA dataset (Trischler et al. (2016a)) of a question-answer pair for the each type of reasoning (since unsanswerable questions have no answer, no example is proposed for that case). MRC datasets usually contain question-answer pairs that cover several of those categories but most of them tend to specialize in a specific type of reasoning. For instance, Ostermann et al. (2018) encourage both inference and disambiguation skills by proposing questions about short daily life situation scripts which necessitate commonsense knowledge to be answered; Welbl et al. (2017) force the development of synthesis by ensuring that pieces of answers are spread in different documents; Rajpurkar et al. (2016b) and Trischler et al. (2016a) include unanswerable questions to teach machines to detect ambiguity and insufficiency; and both Kočiskỳ et al. (2018) and Trischler et al. (2016a) try to avoid the over-exposure to word matching by creating questions about passages from their summaries.

In addition to the reasoning skills mentioned above, QA challenges [1] that test MRC often include other complementary skills, such as conversational skills (Reddy et al. (2018); Choi et al. (2018)), text generation (NLG) (Nguyen et al. (2016); Kočiskỳ et al. (2018)) or information retrieval (IR) (Nguyen et al. (2016); Kočiskỳ et al. (2018)). This thesis does not address the conversational skills but challenges with NLG and IR are included in our study, and are respectively detailed in Section 3.1.3 and 3.1.4.

The skills that the challenges aim to address can, in some cases, be inferred from some of their features. The most remarkable feature is answer type: spans and cloze-style lead to word matching, while multi-choice and human-written answers are likely to address either paraphrasing, inference, or synthesis. Although multi-choice and human-writen answers can tackle both the same reasoning skills, human-writen answers include the necessity to develop

[1] Within this thesis, the words tasks, challenges and datasets are interchangeable and are all three used for the sake of readability.

NLG skills to fulfil correctly the tasks. Concerning IR skills, datasets which have several passages per question may need them.

The existing datasets and their features are listed in Table 2.2 in order to get an overview of the evolution and state of MRC, and to understand the needs behind the architectures which have been designed to learn MRC skills, as it is depicted in Section 2.2.

In each row, a dataset is depicted through a short description of its main characteristics:

- The column *Dataset* gives the name of the dataset and the reference of the article in which it has been published.
- The column *Passages* displays information, when available, about the amount of passages in the dataset, their sources (i.e childrens stories, newspapers, ...) and their size (i.e number of tokens, sentences, ...).
- The column *Questions* displays the number of questions and the way they have been collected (i.e written by humans, extracted from the passage, ...)
- The column *Answers* displays the type of answers which are requested (i.e multiple-choice, extracted from the passage, abstractly generated, ...). This feature, as it is explained in Section 2.2.2, determines the last processing phase of the model and its output.
- The column *Psg:Qst* displays the amount of passage per question. 1:N means that the answer to a question is contained in one given passage and that there are several questions asked per passage. N:N means that the answer of a question can necessitate to explore several passages (the datasets with N:N Psg:QA usually imply that the models which tackle them develop IR skills in addition to the MRC ones). 1L:N is similar to 1:N but the passage given with the question is long, which means that it cannot be entirely processed by models as is (one solution would be to shorten it). For instance, the passage may be an entire book. The reference for saying if a passage is long or not is the maximal capacity of BERT, which is 512 tokens.

In the following section, it is shown that the answer type of a challenge has a high influence on the architecture that is used to solve it. We propose to separate the MRC datasets in three groups in order to ease the MRC architectures' description understanding. Those three groups are described briefly below in order to offer better reading of the Table 2.2 and will be extensively described, with their influence on the architecture design, in Section 2.2.2.

- Extractive MRC (E-MRC) datasets answer a question using a span in the passage, which is why those datasets are also called "span-based question answering".

- Multiple-choice MRC (MC-MRC) datasets usually consist of a text passage, a question, and a set of candidate answers, with one of them being the target answer.
- Generative MRC (G-MRC) datasets answer a question with generated text. There are two different kind of G-MRC tasks : the extractive generation (EG), which only use the words which are already in the passage (such as Vinyals et al. (2015)), and the abstractive generation (AG), which pick words in the entire vocabulary (such as See et al. (2017); Devlin et al. (2018)).

Cloze tests are a specific kind of MRC task which consist in retrieving a word passage masked in a text by a placeholder. If we consider each word from a vocabulary to replace a masked one as an answer candidate, then this task can be seen as a MC-MRC task; otherwise, if there are several consecutive masked tokens, the task can be seen as a G-MRC task.

Dataset	Passages	Questions	Answers	Psg:Qst
MC Test Richardson et al. (2013a)	660 fictional stories for children, 150/300 tokens	2640, written by human	multiple choice	1:N
CNN/Daily Mail Hermann et al. (2015)	313K anonymised news articles avg. 750 tokens, max 2k	386k, sentence extract from abstractive summary	anonymized entity, cloze-style	1:N
CBT Hill et al. (2015)	687K snippet from 108 children's book 20 sentences per snippet	687k, 21st sentence	entities or common nouns, cloze-form with multiple choice	1:1
MSMarco Nguyen et al. (2016)	top-10 ranked passages for each queries (1M passages) from 200k+ documents	100k search queries	human generated	N:N
NewsQA Trischler et al. (2016a)	13K CNN news articles avg. 750 tokens, max 2k	120k, human inspired on headline and summary point	span	1:N
SQuAD 2.0 Rajpurkar et al. (2018)	23k paragraphs from 536 Wikipedia articles, $<$ 500 characters	140k, human 54k unanswerable	span	1:N
NarrativeQA over summaries	1,572 summaries of book or movie scripts avg. 650 tokens, max 1,2k	46k, human, based on summaries	human (abstractive), based on summaries	1:N
NarrativeQA over stories Kočiskỳ et al. (2018)	1,572 books or movie script avg. 62k tokens, max 430k	same as summary	same as summary	1L:N

RACE Lai et al. (2017b)	28k variable types (news, stories, ads, biography, philosophy, etc.) avg. 321.9 tokens	98k sentences from tests, human	multiple choice cloze-style	1:N
CLOTH Xie et al. (2017)	7k narratives avg. 313 tokens	99k sentences directly from the passages	multiple choice cloze-style	1:N
TriviaQA Joshi et al. (2017a)	662k passages avg. 6 passages retrieved from questions, multiple sources avg. 2.9k tokens	95k, human from trivia and quiz-leagues websites	span	
WikiHop Welbl et al. (2017)	avg. 13 documents per questions, 1st wikipedia article paragraph 100 tokens	51k triples giving an entity and a relation	the last entity of the triple, require multi-hop over several documents, multiple-choices	N:N
CoQA Reddy et al. (2018)	8.4 diverse passages avg. 271 tokens	127k, human, multi-turn avg. 15.2 turn/passage	human generated and span	1:N
QuAC Choi et al. (2018)	8.8k wikipedia passages avg. 401 tokens	98k, human, multi-turn avg. 7.2 turn/passage	human generated	1:N
MCScripts Ostermann et al. (2018)	2.1k human-written texts from 110 script scenarios 196 tokens	27k, human, half are unanswerable	multiple choice (2), 27% require commonsense inferences	1:N

ROPES Lin et al. (2019)	1915 background information from science textbooks and Wikipedia; with situations related to the background which are written by humans Backgrounds have in avg. 120 tokens and 60 tokens for the situation	14,322 human; 12 avg. tokens	1.4 avg. spans from either the situation or the question, based on reasoning	1:N
BookQA Angelidis et al. (2019)	614 books from NarrativeQA	3.4k Who-questions	human-generated (generally entity)	1L:N

Table 2.2: Overiew of existing MRC datasets.

2.2 Machine Reading Comprehension models and architectures

In this section, we describe the architectures and the models which have been designed to tackle MRC challenges. In the previous chapters, explanations remained shallow with the goal to give an overview of the context in which MRC tasks take place. From this chapter, we enter in detail in the design of the MRC architectures and provide mathematical explanations how they work. A short reminder of the definition of a neural network architecture and a neural network model is given in the Glossary.

While some NN architectures are designed for one specific challenge, more general architectures that can be applied to any challenges of the same answer type have been proposed over the years. There are also solutions that aim at addressing any MRC, NLU or even any NLP tasks with one unique architecture. In this section, the different types of architectures, specific or general, that have been used solve MRC are analyzed. During our research, we identified three common phases across them which lead from a passage and a question till an answer. This result in a MRC pipeline which is depicted in Figure 2.1. Instead of presenting sequentially each existing architecture, we detail and compare the sub-architectures associated with each phase in the following sub-sections. Doing so, identifying the common phases among existing architecture is made easier and allow a deeper comprehension. The phases which imply knowledge which have not been described previously in this thesis have a dedicated subsection.

The first phase is the *machine reading phase*, which can also be called *contextualized embedding phase*. This is the first reading of a passage and of a question independently of each other. The contextualisation, as we could see in Section 1.3, means that the representation of the tokens within a passage are not independent but are influenced by their surrounding tokens. The natural language texts are transformed into dense representations using techniques such as the ones described in Chapter 1. If, during this phase, the embedding method used is part of the contextualized word embedding methods (see in Section 1.3), it completely fulfills the machine reading phase. Otherwise, if the embedding method does not include contextualization (such a static word embedding models described in Section 1.1 and 1.2), a contextualization phase follows the embedding phase and both would be part of the machine reading phase. The contextualization techniques which can be used for the contextualisation phase are similar as the ones found in the biased machine reading phase (described in Section 2.2.1).

The second phase is the *question-aware passage representation phase*, in which a representation of the passage contextualized to its associated question is computed. As a result, the question-aware representation (also called contextualized representation) of one passage, is different for each question. The question-aware passage representation phase is composed of two sub-phases which are the alignment phase and the biased machine reading phase. The two sub-phases and the associated techniques to compute those representations are described in Section 2.2.1.

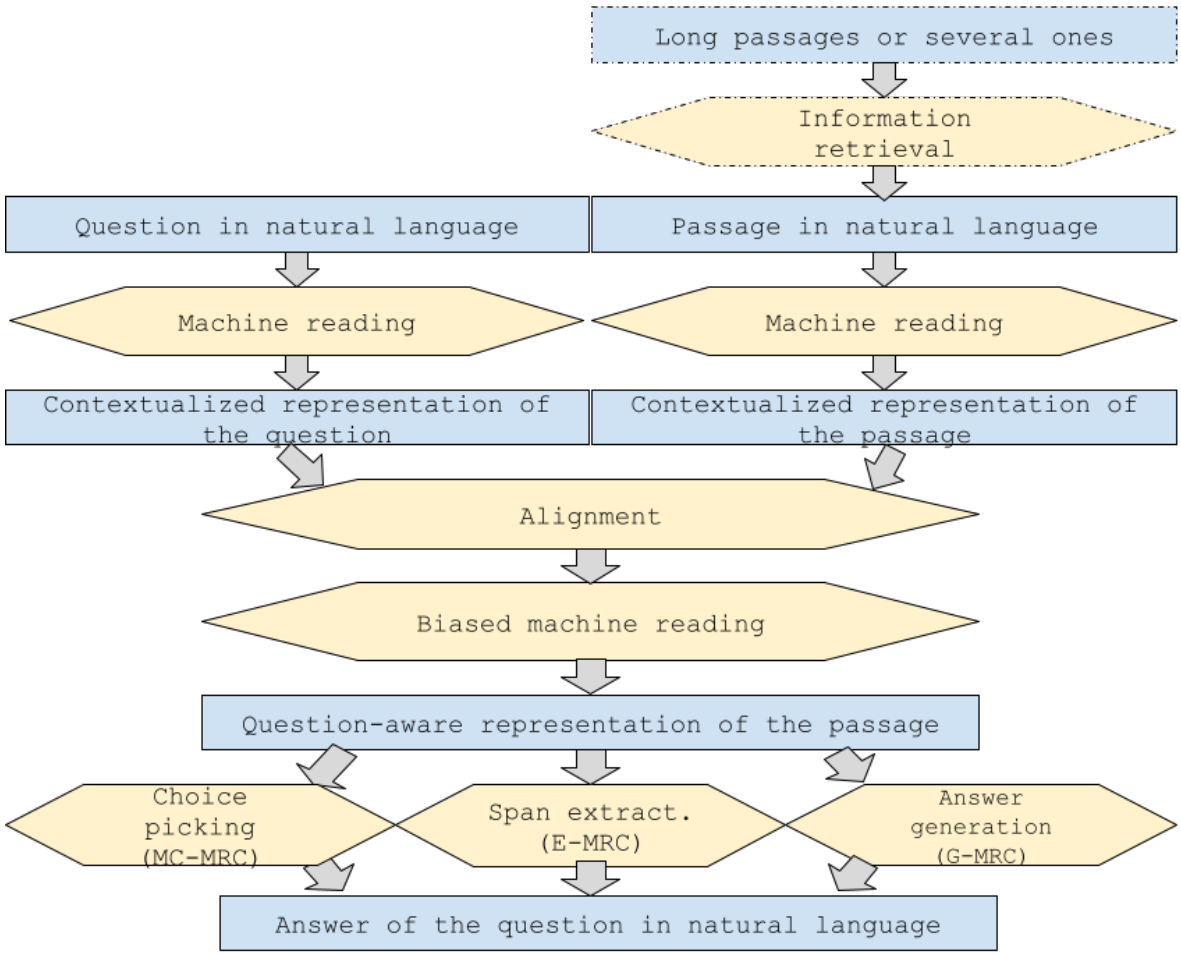

Figure 2.1: Typical algorithm to solve MRC tasks. The data states are represented by rectangle while the processing phases are represented by hexagons. The dotted-line arrow indicate phases or representations which are not systematic but specific to some tasks.

The third and last phase is the *answer finding phase*, in which the answer may be extracted, selected or generated, depending on the challenge (respectively E-MRC, MC-MRC and G-MRC, described in Section 2.1). The three ways of finding an answer have generated fundamentally different kind of architectures, which are each explained in Section 2.2.2.

2.2.1 Question-aware passage representation phase

As input data, an MRC task receives a passage and an associated question. The first phase consists of transforming both the passage and the question independently into a meaningful representation. This is done during the machine reading phase, which is extensively described along the Chapter 1. The output of this machine reading phase is the contextualized representations of the question and the passage. Those contextual representations are composed of

one d-dimensional vector per token. So, with a passage containing N tokens and a question containing M tokens, we would have respectively the passage representation $\mathbf{P} \in \mathbb{R}^{N \times d}$ and a question representation $\mathbf{Q} \in \mathbb{R}^{M \times d}$.

The next phase, the one described in this section, consists of building a question-aware representation of the passage from the contextualized representation of the passage and of the question. The question-aware representation of the passage is done by integrating the information of the questions into the passage in order to highlight the elements of the passages which are related to the question.

Two sub-phases exist in the literature to build the question-aware passage representations, and are commonly applied sequentially in MRC-specific models: the alignment phase and the biased machine reading phase.

The alignment phase is presented in Section 2.2.1. It links and fuses the information between the passage's words and the question's ones with the purpose to highlight the words in the passage which could help answering the question. To do so, it aligns the representation of the passage, the question, and optionally additional external information, such as knowledge bases.

The biased machine reading phase is presented in Section 2.2.1 and consists of updating the passage representation by integrating the results of the alignment phase. We name it "biased" because the MR phase is no longer based only on the language skills and general knowledge but is influenced by what has been asked.

The transformer-based architectures, such as BERT and its variations, which have been described in Section 1.3, also generated a query-aware representation of the context but in a more abstract way. Indeed, for MRC tasks, the models would receive the passage and the question as input, which would be separated by a separation mark. In comparison with the previous systems described in this Section, BERT's encoder computes only one contextualized representation for both the passage and the question instead of computing it separately and then getting a question-aware representation of the passage. Therefore, BERT's architecture gathers the machine reading phase, the alignment and the biased machine reading phase within its encoder. This encoder would be represented by several attention blocks in its architecture which would generalize more and more the information as we go further in the architecture.

Alignment phase

The alignment phase can be found in most MRC architectures and systematically uses attention mechanisms to update the representation of the passage based on the question. The shared idea among existing techniques is to encode how related the question is to each passages word. However, there are variations in their way of computing the relation between the two. We detail below a few existing variations. As already mentioned a few paragraphs above, each alignment

phase takes as input the output of the machine reading phase: a passage representation $\mathbf{P} \in \mathbb{R}^{N \times d}$ and a question representation $\mathbf{Q} \in \mathbb{R}^{M \times d}$. The output of the alignment phase is a context-query aligned representation stored in $\mathbf{G} \in \mathbb{R}^{N \times k}$, with k varying depending on the method used.

Kadlec et al. (2016) use a RNN (more precisely, a bidirectional GRU network) to encode the question with a unique d-dimensional vector $\mathbf{q} \in \mathbb{R}^d$. Then, it aligns the question with the context by applying a dot product between them:

$$\mathbf{C2Q}_i = \mathbf{P}_i \cdot \mathbf{q} \tag{2.1}$$

Here, $\mathbf{G} = \mathbf{C2Q} \in \mathbb{R}^{N \times d}$ A softmax layer can be directly applied to $\mathbf{C2Q} \in \mathbb{R}^N$ in order to rate the importance of each passage's word i.

Chen et al. (2017) chose another way of computing C2Q which avoids losing information from the question's words by encoding them in a unique vector. Instead, each word from the passage is represented by the sum of each contextual representation of question words weighted by $a_{i,j}$:

$$\mathbf{C2Q}_i = \sum_j a_{i,j} \cdot \mathbf{Q}_j \tag{2.2}$$

where

$$a_{i,j} = \text{softmax}(\alpha(\mathbf{P}_i) \cdot \alpha(\mathbf{Q}_j)) \tag{2.3}$$

In this example, the softmax is computed among each question word j and its result is stored in $a_{i,:} \in \mathbb{R}^M$, which gathers all the attention scores over the question for the passage word i. α is learned with a feed forward neural network. Its learned weights $\mathbf{w}^T$ are shared for both passage and question α computation.

$$\alpha(\mathbf{V}_k) = \text{ReLU}(\mathbf{w}^T(\mathbf{V}_k) \tag{2.4}$$

$\mathbf{V}_k$ represents the vector either from a passage's word or a question's word. Chen et al. (2017) enhance the aligned representation of the context with the query $\mathbf{C2Q} \in \mathbb{R}^{N \times d}$ with a 303-dimensional vector so $\mathbf{G} \in \mathbb{R}^{N \times (d+303)}$. This vector is composed of (1) the 300-dimensional representation of this passage's word computed during the MR phase (word embedding), (2) a scalar indicating if the word i can be exactly found in the question and (3) a 3-dimensional vector which is composed of part-of-speech numerically represented, a binary scalar indicating if it is recognized as being a name entity, and a normalized term-frequency value among the passage and question.

Wang and Jiang (2016) also compute C2Q to get an aligned question/passage representation thanks to Equation 2.2, but in their case the attention score $a_{i,j}$ results from an RNN computation instead of a feed-forward neural network:

$$a_{i,j} = \text{softmax}(\alpha(\mathbf{P}_i, \mathbf{Q}_j); \tag{2.5}$$

$$\alpha(\mathbf{P}_i, \mathbf{Q}_j) = \mathbf{w} \cdot \tanh(\mathbf{W}^p \mathbf{P}_i + \mathbf{W}^q \mathbf{Q}_j + \mathbf{W}^a \mathbf{h}^a_{j-1}) \tag{2.6}$$

where $\cdot$ is the dot-product between two vectors; the vector $\mathbf{w} \in \mathbb{R}^d$; and all matrices $\mathbf{W}^* \in \mathbb{R}^{d \times d}$ contain weights to be learned. $\mathbf{h}^a_{k1}$ is the hidden state of the previous question word.

Seo et al. (2016) propose the bi-directional attention flow (BiDAF) architecture, which is probably the most widely-used as alignment phase for MRC tasks but also the more complex. In this paper, the alignment phase is called "attention flow layer". To get an aligned question/passage representation, the authors first compute a similarity matrix $\mathbf{S}$ by measuring each passage-question word pairs using a trainable scalar function α that encodes similarity between two vectors. The similarity matrix $\mathbf{S}$ is depicted in Figure 2.2. In addition to compute a context-to-query (C2Q) representation, a query-to-context (Q2C) representation is proposed. [2] As we have already defined, C2Q representation encodes which question words are most relevant to each context word. Concerning Q2C, it identified which passage's words are critical for answering the question.

C2Q is also computed with Equation 2.2, but the attention score which link a passage's word i to a question word j is given with

$$\begin{aligned} a_{i,j} &= \text{softmax}(\mathbf{S}_{i,j}); \\ &= \text{softmax}(\alpha(\mathbf{P}_i, \mathbf{Q}_j)) \end{aligned} \tag{2.7}$$

where

$$\alpha(\mathbf{P}_i, \mathbf{Q}_j) = \mathbf{w}^T [\mathbf{P}_i; \mathbf{Q}_j; \mathbf{P}_i \circ \mathbf{Q}_j] \tag{2.8}$$

where $[;]$ is a vector concatenation across row operation and $\circ$ is the elementwise multiplication operator. The softmax is computed among each question word j and stored in $a_{i,:} \in \mathbb{R}^M$, which gathers all the attention scores $\alpha(\mathbf{P}_i, \mathbf{Q}_:)$ over the question for the passage word i. $\mathbf{w} \in \mathbb{R}^{3d}$ is a vector containing learned weights.

Q2C vectors emphasise the most important words in the passage with respect to the question: it computes $b_: = \text{softmax}(\max(\mathbf{S}_{i,:})) \in \mathbb{R}^N$, which first retrieves the highest similarity scores across passage words for each question word j and, in a second time, performs a softmax across those maximum values. The final Q2C vector is a sum of each word from the passage weighted by b_i;

$$\mathbf{Q2C}_i = \sum_i b_i \cdot \mathbf{P}_i. \tag{2.9}$$

The C2Q, Q2C and passage representation P of each word from the passage i are used to compute the aligned representation:

$$\mathbf{G}_i = \beta(\mathbf{C2Q}_i, \mathbf{Q2C}_i, \mathbf{P}_i) \tag{2.10}$$

[2] For sake of readability, we will continue using "passage" instead of "context" to avoid confusion with the contextualized embedding. The word "question" is used instead of "query".

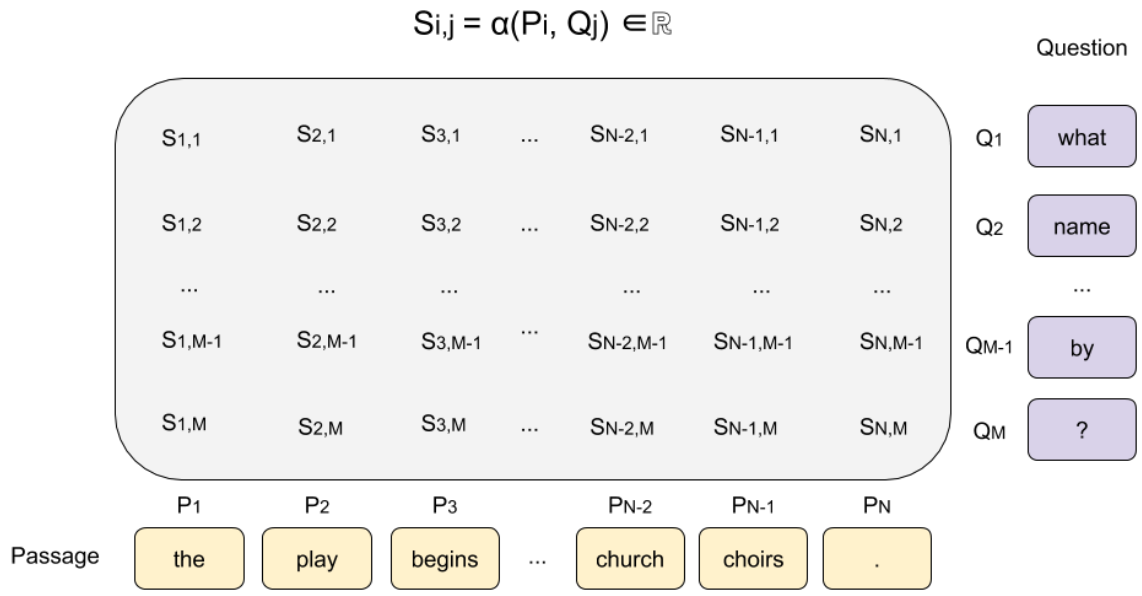

Figure 2.2: Similarity Matrix S computed in Seo et al. (2016)

Although beta can represent a neural network that fuses the vectors given as parameters, it can also be a simple concatenation. That is what is proposed in the original paper Seo et al. (2016).

$$\beta(\mathbf{C2Q}_i, \mathbf{Q2C}_i, \mathbf{P}_i) = [\mathbf{C2Q}_i; \mathbf{Q2C}_i; \mathbf{C2Q}_i \circ \mathbf{Q2C}_i; \mathbf{C2Q}_i \circ \mathbf{P}_i \tag{2.11}$$

which result to $\mathbf{G} \in \mathbb{R}^{N \times 4d}$.

MRC task instances can also be augmented by extending the passage with general knowledge. In some case, such as Mihaylov and Frank (2018), entity-relation-entity triplets related to the passage are retrieved from knowledge bases by selecting facts that contain lemmas from the passage or question and adding them at the end of the passage before any machine reading phase. Another option proposed by Bauer et al. (2018) is to incorporate the knowledge right after the passage/question alignment phase. A second alignment occurs, this time to align the retrieved triplets and $\mathbf{G}$, thanks to a classic attention layer.

In case of multi-hop reasoning MRC tasks, a popular strategy is to repeat the alignment phase k times, using as input the representation of the machine reading phase as and the output of the $k-1$ alignment phase (Dhingra et al. (2017); Bauer et al. (2018); Feldman and El-Yaniv

(2019)).

This sample of alignment architectures gives an overview of the first steps to represent the passage regarding a question. The output is a matrix $\mathbf{G} \in \mathbb{R}^{N \times k}$, which is minimally composed of a context-to-query $\mathbf{C2Q} \in \mathbb{R}^{N \times d}$ matrix and by the concatenation with other elements, such as, as in the case we exposed here: feature vectors, static word embedding, contextualized word embedding, external knowledge from knowledge bases or query-to-context vectors. $\mathbf{G}$ can already provide a question-aware representation of the context, but several research suggest to enhance it thanks to a *biased reading* phase.

Biased machine reading phase

Following the alignment phase, critical parts of the passage to answer the question have been highlighted. By integrating a new reading of the passage, the model builds a new overview of a passage, which is now aware of the query. The core idea of this phase is to get a new contextual representation of the passage which integrates the newly generated information. The techniques are thus similar to the ones used for contextual word embedding described in Section 1.3 and are thus less extensively described.

The two architectures that are usually used for this phase are RNN layers and self-attention blocks. For instance, Seo et al. (2016) call the biased machine reading phase the *modeling layer* and is composed of two BiLSTM layers. Clark and Gardner (2017) enhanced it with a *residual static self-attention mechanism*: a bi-GRU layer (Cho et al. (2014)) is applied over the aligned question/passage representation before passing it sequentially through a self-attention layer and a fully-connected layer.

The biased machine reading phase is not systematic but present in many MRC architectures. However, it cannot occur without having before an alignment phase: it would then not be a biased reading phase since there would not be any awareness of the question. The output is a matrix $\mathbf{H} \in \mathbb{R}^{N \times k'}$, with k' being dependant on the architecture chosen.

Once the question-aware representation of the passage phase is computed, its output representation can be directly used in order to find the answer.

2.2.2 Answer finding phase

As reported in Table 2.2, MRC tasks can differ by the type of answer which is expected. We identify three types, as described in Section 2.1 : E-MRC, MC-MRC and G-MRC. Each type of answer requires a different architecture and is described in the following subsections. In the last subsection called *Transfer learning*, we present a shared architecture among the three types of tasks.

Span extraction for E-MRC tasks

E-MRC tasks take as input a passage concatenated with a question and return the probabilities for each token of the passage to be the start or the end of the answer span (Seo et al. (2016); Chen et al. (2017)).

For challenges which are known to contain only one token in their answer (Hermann et al. (2015); Hill et al. (2015)), a popular technique, called Attention Sum Reader (ASR) (Kadlec et al. (2016)), consists of multiplying each question-aware token representation of the passage with the question representation from the machine reading phase and passing the output through a softmax layer. ASR can be adapted for tasks which return answers with more than one token by learning two probability distributions: the output of the model is then composed by two vectors : $\mathbf{A}_1$ and $\mathbf{A}_2 \in \mathbb{R}^N$. They represent the probability distribution for each token-to-be, respectively, the start of the answer span and the end of the answer span. In order to increase the span likelihood, Chen et al. (2017) constrained the system by limiting the distance between the start and the end to 15 tokens and by adding to the objective function the constraint to maximize the multiplication between both probabilities whereas Tay et al. (2018) add the constraint to minimize the sum of negative log probabilities between the start and end indices.

Since the publication of the transformers (see Section 1.3), the most common way to learn those vectors is to fine-tune pre-trained transformers with question-answering tasks, as it is done in Devlin et al. (2018) with BERT for Question-Answering, as depicted in Figure 2.3. To do this, the authors propose to use the pre-trained original architecture, as described in Section 1.3 and to add one or two independent FCNN, depending if the span must be followed by a softmax which would return respectively $\mathbf{A}_1$ and $\mathbf{A}_2$. During the fine-tuning, only the two output vectors are learned while the other weights of the model are frozen.

In the case where the token corresponding to the answer is present several times in the given passage (multi-mentioning), Min et al. (2019) make the hypothesis that only one mention (that we will call from now on answer mention) is really answering the question, the other mentions are distractors (mentions). An example of this situation is depicted in Figure 2.4 with an example extracted from TriviaQA, where the first mention of the answer in the passage is the ground truth answer mention and the other, according to the hypothesis of Min et al. (2019), is considered as a distracting mention. Unfortunately, except for the E-MRC datasets which provide the exact place of the spans through index, MRC datasets do not provide the ground truth answer mention. It has then to be guessed or weakly retrieved. In classical models (Joshi et al. (2017b); Tay et al. (2018); Talmor and Berant (2019)), the answer mention is also unique but chosen either by selecting the first mention or at random. As an alternative, Min et al. (2019) propose a weak supervision, which, first, computes the likelihood of each answer candidate given the question and the passage with a learned function, and second, selects

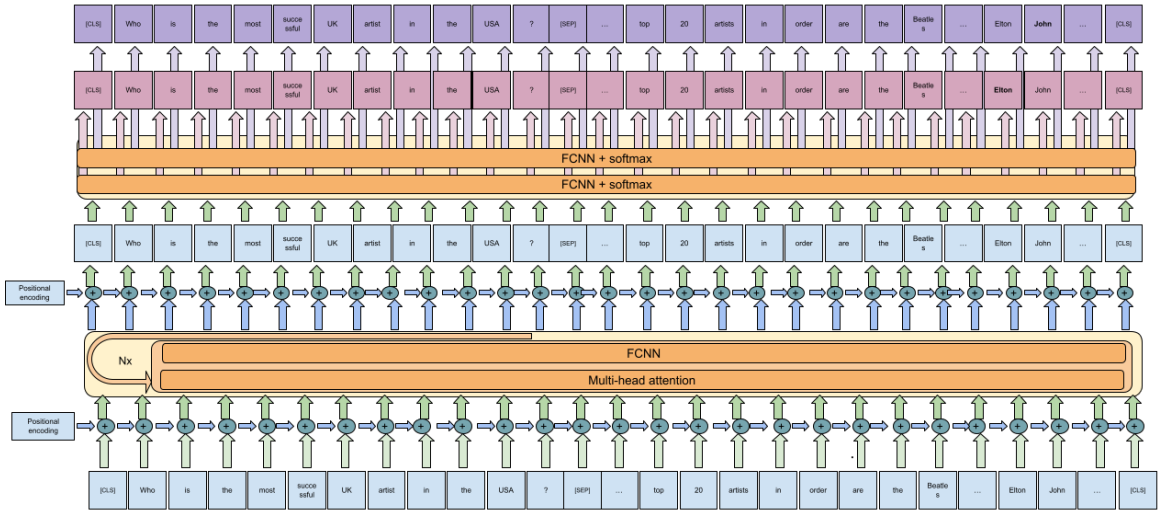

Figure 2.3: BERT for Question Answering (better with color) with an example of passage/question extracted from Trivia QA. The blue rectangles represent each word representation while the dark pink rectangles represent $\mathbf{A}_1$ and the lila ones represent $\mathbf{A}_2$: the probability distribition for each word to be the beginning or the end of the answer span. The ground truth answer to the question is "Elton John".

the answer with the highest likelihood as being the answer mention. The likelihood can be computed with any model which gives the probability to have an answer based on a passage and a question. For instance, one can use the output of ASM or BERT for Question Answering and only check the probability for the answer and distractors mentions. Whichever weak supervision method is used to select the answer mention (i.e first answer, random answer or most likely answer), Min et al. (2019) propose a HARD-EM loss which maximize the likelihood to retrieve the answer mention and minimize the weakly retrieved distractors.

Multi-choice answers

Depending on the challenge, the candidate answers can be directly extracted from the passage (Richardson et al. (2013b); Hermann et al. (2015); Hill et al. (2015)) or not (Lai et al. (2017a)). In any case, the most common methodology consists of training a system to score each candidate answer for the same passage/question and picking the highest scoring one.

When the MC-MRC tasks can be accomplished by extracting its answer candidates from the passage, the same architecture can be used as for E-MRC tasks to compute the probability of each answer candidate. Such as for E-MRC, in case of multi-mentioning, each occurrence's probability can be summed, averaged, or only one can be selected. In the field of MC-MRC, the most common technique is to sum it, as it is proposed in Kadlec et al. (2016) with their pointer sum attention mechanism (Dhingra et al. (2016); Cui et al. (2016)).

Question	Who is the most successful UK solo artist in the USA?
Answer	Elton John
Passage	The top 20 artists, in order, are The Beatles, Michael Jackson, Madonna, Led Zeppelin, **Elton John**, Pink Floyd, Mariah Carey, Celine Dion, AC/DC, Whitney Houston, The Rolling Stones, Queen, ABBA, The Eagles, U2, Billy Joel, Phil Collins, Aerosmith, Frank Sinatra, and Barbra Streisand. The list is perfectly split between 10 solo artists and 10 groups. Eight of the 10 solo artists are from North America, while eight of the 10 bands are from outside America, the majority being British. Remarkably, the country that invented rock and roll has not produced any of the top seven rock bands. America's strongest contender, in at No. 8, is often-derided soft-rock stalwarts The Eagles. (...) It's hard to avoid wondering whether political/social mores play a role in the dichotomy. America, after all, likes to think of itself as a land of individualists. Elvis, Jackson, and Madonna all came from humble beginnings, surrounded by poverty and family tragedy. They epitomized the American dream, and so you might argue that the more left-leaning Europeans are happier to celebrate the collectivism of a band. If we look to what's thought to be the most ideologically "right" genre, this theory holds true: Of the 25 greatest selling country-music stars of all time, all are solo artists. The UK's two bestselling solo stars, meanwhile, do not fit the rags-to-riches mold of the American singers, but are rather privileged virtuosos who were in stage school from a very young age (Phil Collins, **Elton John**.)

Figure 2.4: Example of multi-mentioning extracted from TriviaQA.

In Trischler et al. (2016b), a parallel architecture called a "reasoner" is added to the extractive one. In this module, each candidate answer is concatenated with its associated question to form a set of hypotheses. With the example given in Figure 2.4, one can imagine to have two answer candidates to the question "who is the most successful UK solo artist in the USA ?" : Phil Collins and Elton John. In that case, the first hypothesis would be "who is the most successful UK solo artist in the USA ? Phil Collins" and the second one "who is the most successful UK solo artist in the USA ? Elton John". In that case, the authors propose an adaptation of textual entailment tasks (TE). TE tasks consist of recognizing when a first text precedes a second one. Such as LM, TE training does not require manual annotation, and so the amount of data available to train TE models is large. It can be adapted to MC-MRC challenges by replacing the first text by the passage and the question, and the second text by a candidate answer. The score for each candidate answer is compared to the other and thanks to a a softmax layer over the batch of candidates.

The final score for each the candidate answer is computed by multiplying the probability of each candidate answer gotten through the extractive architecture with the entailment score for each candidate gotten from the reasoner (Trischler et al. (2016b)). The final choice returned is the candidate the highest score.

An alternative to teaching a model to select the best answer from a set is to present each

<passage, question, candidate answer> independently for each candidate answer and to attribute a good score for sets containing right answers and a low score for those that do not (Jiang et al. (2020)). In this case, a model sees several times one passage and its questions, sometimes with the right answer, sometimes with a distractor (untrue candidate answer). This task requires evaluating each candidate individually and not just emphasize finding the right candidate from a batch. After rating all the candidates for a question/passage pair, during the test phase the candidates are ranked based on their score and the best one is returned. One advantage of this architecture is that it can be applied to MC-MRC problems that have several valid answers among their candidates. In that case, the best k ranked candidates are returned. The score is attributed right after the question/answer-aware representation phase by using a fully connected layer.

Based on the hypothesis that, for all MC-MRC models (1) the valid answer refers to a specific passage's span and that (2) this specific passage is a unique sentence, Jiang et al. (2020) propose an alternative architecture which is trained to select, in a first step, for each candidate answer, the span in the passage which is the most likely to contain it and, in a second step, trained to identify the right <span, question, candidate answer> triplet. The advantage of filtering the relevant spans by answer is to reduce the amount of text (and noise) that the model receives as input, which, in addition to simplifying the task, makes this task close enough to a textual entailment one for using some as pre-trained models before fine-tuning with multiple-choice tasks.

Generative answers

Cloze tests are a specific kind of MRC task which consist of retrieving a word passage masked in a text by a placeholder. It is important to emphasize that, for cloze tests, there are no questions. If we consider each word from a vocabulary as candidate to replace a masked one as an answer candidate, then this task can be solved with MC-MRC models (Lai et al. (2017b); Xie et al. (2017)). However, if there are several consecutive masked tokens, the task is a generative one.

In any case, the inputs of cloze test models are a passage with placeholders, and its outputs are the probability distributions for each placeholder over candidate answers, words from a passage, or the complete language vocabulary.

Generative MRC (G-MRC) is similar to cloze tests on many points except that the answer answers a question instead of replacing a placeholder. In G-MRC, the input is a passage followed by a question and the output remains the same as for cloze tests.

The usual architecture used for G-MRC are Seq2Seq (also called encoder-decoder) architectures. In Section 1.3 and Section 2.2.2, we explored the encoders. Generative tasks require adding a decoder. The original transformers (Vaswani et al. (2017)) such as GPT (Radford et al. (2018))

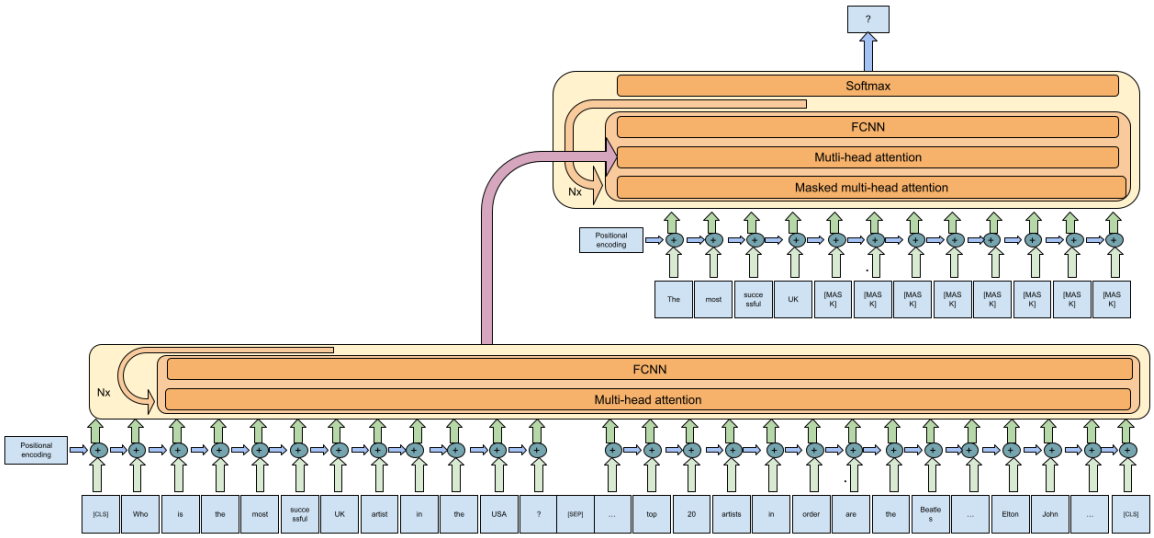

Figure 2.5: Transformers general architecture (better with color) with an example extracted from TriviaQA

or BART (Lewis et al. (2019)) are Seq2Seq. Their architecture is sketched in Figure 2.5

In parallel to input processing in the encoder, the elements of the output text which have already been generated (i.e the first words of an answer for MRC) pass by a masked multi-head attention layer. Afterwards, the newly computed representation of the encoder and the masked multi-head attention feed conjointly in the decoder in a new multi-head attention layer. Then, a FCNN layer is applied. The combination of those three layers repeats N times, where N is the same hyperparameter as for the encoder. At the $N+1$ times, the computed representation is sent to a softmax layer, which will give the most likely word to be picked as continuing the answer. In the case represented in Figure 2.5, the expected word replacing the "?" placeholder is "artist".

There are two types of generative tasks. The extractive generation (EG), which only use the words which are already in the passage (such as Vinyals et al. (2015)), and the abstractive generation (AG), which pick words in the entire vocabulary (such as See et al. (2017); Devlin et al. (2018)).

G-MRC can also be seen as a kind of text summarization task: an input text must be summarized but with a specific focus, namely the question. For this challenge, the answers generated are usually sequences of words that must be linguistically coherent in addition to returning the relevant information to answer the question correctly.

Summarization tasks can also be tackled with either the EG approaches or the AG ones. The advantages and disadvantages of EG approaches versus AG ones are the same for both summarization and for G-MRC tasks: EG are easier because they do not require generating

grammatically coherent sequences but are limited regarding paraphrasing, generalization, or the incorporation of real-world knowledge.

An EG problem can be seen as a multiple-choice MRC task with

1. a choice to repeat over each word to generate
2. as many candidate answers, for each word, as the amount of unique token in the passage and question.

PtrNET (Vinyals et al. (2015)) is an EG model which generates each word of the answer thanks to a Seq2Seq model, and which uses an attention mechanism as a pointer to the passage/questions words to use as answer.

An AG problem can also be seen as a multiple-choice MRC but with more answer candidates than EG: the whole language vocabulary. However, even if each single word of the answer is not coming from the passage and the question, the information to answer the question must be contained into the passage, otherwise, answering a question using uniquely general knowledge, no MRC skills are engaged. That is why AG should still be encouraged to use words from the passage and question. Picking words in the language vocabulary helps building better sentences linguistically and offer more freedom to paraphrase and summarize the collected information.

PGNet (See et al. (2017)) is a system which combines PtrNet and a generator for text summarization. Its architecture is also a Seq2Seq network. Each new answer token generated through the decoder of the Seq2Seq model uses the pointer mechanism to weight each passage token. The distribution of the importance of the passage words is then used to chose which one of the words from the complete vocabulary should be used. To the best of our knowledge, although there are PGNet variations, its principles are re-used in each current well-performing G-MRC solution (Nishida et al. (2019); Indurthi et al. (2018); Bauer et al. (2018)). The biggest competitor of PGNet is GPT-2 Radford et al. (2019), which is transformer-based and is adaptable to generative tasks while limiting the possibility of repetition. However, it is still far from performing as well as the PGNet-based models.

G-MRC tasks may require concise answers which do not repeat the context of the question or elaborated answers. For the question "Who is the most successful UK solo artist in the USA ?", a concise answer would be "Elton John" while an elaborate one would be "The most successful solo artist in the USA is Elton John". In order to be trained and to perform on various G-MRC tasks with different answer styles, Masque (Nishida et al. (2019)) indicates which answer style is desired as input and uses this information to adapt the answer.

Transfer learning

Because each task and dataset has their particularities, models are usually trained from scratch for each task (Trischler et al. (2016b); Chen et al. (2016)). GPT (Radford et al. (2018)) and BART (Lewis et al. (2019)) are transformer-based Seq2Seq architectures, which can solve both solve EG and AG tasks. BART is pre-trained by corrupting documents (token masking, token deletion, sentence permutation, document rotation, text infilling) and thus extending its generalisation capacities. Thanks to the fact that such pre-trained models have their parameters publicly available, groups with low computational resources now have access to a trained high-performing model they can fine-tune to their specific purposes. Indeed, learning from scratch to solve a problem is computationally expensive, so sharing pre-trained parameters makes deep network learning quicker and more accessible. However, the number of parameters to fine-tune remains large and fine-tuning still requires high computational resources.

The development of NLP capacities thanks to the emergence of fine-tuned models has generally increased the interest in transfer-learning models. The idea that a real understanding machine should handle correctly a task even if it has not specifically been trained for it (zero-shot learning, ZSL) has become popular with GPT-2 (Radford et al. (2019)), and, since then, generalisation of models is often evaluated through ZSL. One advantage of models that perform well on ZSL evaluation set-ups is that those models have high generalisation capacities. Another one is that it decreases the dependency on the data. However, ZSL over a wide variety of NLP tasks is too general and does not perform as well as task-specific models.

As is shown in Section 2.1, each dataset, by capturing a special aspect of MRC, has its own particularities. Also, depending on the way the data has been collected, the style of the text, etc., the models cannot adapt well to new types of data. Multi-task models using uniquely MRC models have proven to be efficient at generalizing: Talmor and Berant (2019) report that training a model on several different datasets provides better results than on one target alone. They also found that using examples from different datasets converges quicker: they reach the performance of a task-specific model using one-third the number of examples. Furthermore, they showed that with good multi-task pre-trained models, the fine-tuning step may be skipped entirely. By finely pre-training one single BERT-large model with 10 MRC datasets, Talmor and Berant (2019) obtain close to or better than state-of-the-art performance on each of those datasets, without fine-tuning to the targeted ones. Thus, in addition to generalizing well, such models require neither large datasets nor large computational resources to train for specific tasks.

T5 (Raffel et al. (2019)) is another example of a multi-task model which does not focus only on one general task as MRC but on all text-based language problems. This system converts each problem into a text-to-text format, in contrast to BERT which can take also any text-based language problem as input but has to adapt the output format to the downstream task. It is a

transformer-based architecture which differs from BERT by the fact that, instead of masking isolated words in the MLM task, it masks sequences. Doing so allows the model to handle a wide variety of tasks, including NLG tasks such as summarization, question-answering, and machine translation. The model knows which information to return (answer to a question, translation of a passage, class to pick up, etc.) thanks to a text prefix added to the original input sequence. However, although a multi-task pre-trained model has many advantages, the results of T5 under-perform the standard pre-training plus fine-tuning setup (Raffel et al. (2020)). One hypothesis is that multi-task pre-training on a too wide variety of tasks prohibits the model from getting very good at any one task as it would hinder performance on other tasks.

UnifiedQA (Khashabi et al. (2020)) proposes a multi-task pre-training based on T5 but limited to QA tasks. The authors propose a format-agnostic question-answering system, arguing that the ability to comprehend a text is not supposed to be governed by the answer format. The model takes as input the question followed optionally by additional information associated with each answer type (i.e candidates answers for MC-MRC) and learns how to use the given additional information to return an appropriate answer. Unlike the original T5, UnifiedQA is pre-trained only with MRC tasks. It uses 8 seed datasets for the pre-training of 4 different formats (E-MRC, M-MRC, G-MRC and a yes/no-MRC, which we have not discussed in this thesis). UnifiedQA performs almost as good or better than individual T5 models targeted to each dataset, and no example of failure has been reported.

Basing natural language learning on a variety of tasks requires the AI to acquire general NLP knowledge. However, having too general NLP skills without emphasizing one specific task via fine-tuning may prevent the AI from excelling on any of them but simply perform correctly, as it is shown in Raffel et al. (2019). The diversity of tasks to use for training a model without requiring fine-tuning must be then neither too broad nor too narrow. In this section, we have seen that sharing a same architecture but also the same model models for all the MRC datasets is very promising.

Chapter 3

A pipeline architecture for generalisation in MRC skills

The overarching goal of MRC is to enable machines to read and comprehend a text rather than learning to answer specific kinds of questions on specific kind of texts. In Chapter 2, the general methodology to treat MRC tasks from the existing architectures has been studied and depicted in Figure 2.1. It has to be noted that the stages that compose it are recurring steps in the literature rather than explicit systematic ones: not all architectures follow those steps. By identifying which steps are recurrent, we have systematized the MRC learning process. Hypotheses regarding which sub-tasks could share one architecture and eventually, also, one common training can now be posed.

We propose a new pipeline in Figure 3.1 based on the following hypothesis:

First hypothesis. Using a shared MRC pipeline composed of models trained with challenge-specific data can perform at least as good on any given challenge as the task-specific architecture baseline proposed by the research groups which have released the datasets.

Second hypothesis. Sharing the training weights (i.e model) in addition to the architecture among several MRC datasets could improve the performance on new datasets (seen for the first time during test time) compared to the results obtained by having unique architecture among datasets but separate training. For instance, a model which is trained just with SQuAD and RACE examples would perform systematically better on the NarrativeQA dataset than a model trained just with SQuAD examples.

Third hypothesis. The performance of generative MRC tasks can increase thanks to the addition of an extractive phase (span extraction phase in Figure 3.1). Indeed, since the complex tasks of retrieving the information and of generating the answer text are split in two

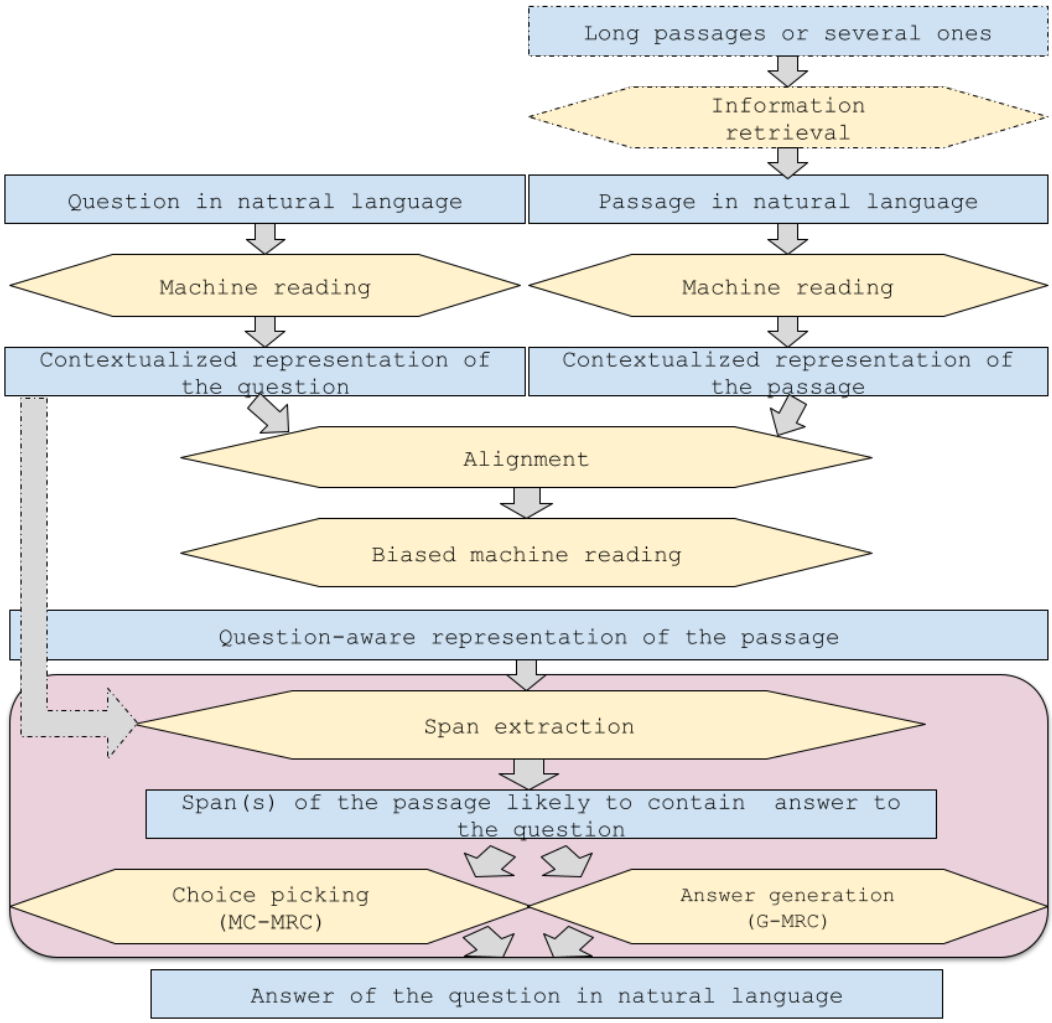

Figure 3.1: Our Pipeline. The optional phases (changing from a dataset or from one model to another) are represented with dashed lines while the mandatory ones are depicted with plain ones. The red rectangle emphases the phases which are specific to our pipeline.

different steps, the system would just have to generate from a few well-selected spans instead of complete passages.

In addition to an improvement in the performance of the models, having a performing extraction phase which reduces the context to a few spans makes the MRC systems more explainable by returning which part of the passage the answer is based on. In case of failure, such as in the HRC tests, we can distinguish whether the model understood the text but is unable to generate a correct answer or if it managed to answer correctly from false clues.

The drawback of this method is that if a step does not perform well, all following steps would be negatively impacted. However, our modular architecture makes it easier to determine which

component failed and to replace it with an improved version.

In this chapter, we propose an updated version of the pipeline described in Chapter 2 which includes a mandatory extracting phase for each MRC challenge. A first example of a combination of architectures which compose the pipeline trained with various dataset is proposed in section 3.1. Section 3.2 propose several experiments in order to evaluate this baseline. Section 3.3 gives the datasets used for our experiments and how to adapt them. Finally we showcase the results of the experiments in Section 3.4.

3.1 Methodology

The pipelines extracted (Figure 2.1) and proposed (Figure 3.1) are abstract procedures which both generalize a procedure to perform in a MRC task. In practice, a combination of pre-trained models are going to follow this procedure, with the possibility that one model achieves several steps. An architecture is here defined as the organisation of a neural network (i.e., the amount of layers, the size of each layer, how they are connected one to another, etc.) in order to accomplish a task. A model is the architecture with the weights trained. As it is shown with the extracted pipeline in Chapter 2, many different combination of architectures can follow a unique pipeline despite their difference. Also, one architecture can include several steps of the pipeline.

This is the case, for instance, in BERT Devlin et al. (2018): the model gets as input a question and a passage and returns their question-aware representation of the passage and their passage-aware representation of the question. When a BERT is intended to perform a question-answering task, two additional layers are added to the original BERT architecture in order to return the span in the passage that is likely to contain the answer to the question, as it is explained in Section 2.2.2. In the case of BERT then, a unique architecture performs the machine reading phase, the alignment phase, the biased machine reading phase and the span extraction.

In this section, we propose a combination of architectures which follows the pipeline depicted in Figure 3.1. The different architectures which compose it and the steps of the pipeline they are associated to are summarized in Table 3.1 and detailed in the following subsections.

Since each dataset has its specificities, extra phases can still be added before, in-between, or after the pipeline. For instance, some MRC challenges require handling several passages or very long ones (Kočiskỳ et al. (2018)). In such cases, we can simply add an additional initial information retrieval phase over the set of passages (or the long passages) and form a shortened passage composed of the retrieved information.

Pipeline phase	Architecture selected
Machine reading phase Alignment phase Biased machine reading phase Span extraction phase	BERT-base for Question Answering, english (Devlin et al. (2018)), enhanced with HARD-EM loss (Min et al. (2019))
Choice Selection (MC-MRC) / Answer generation (G-MRC)	UnifiedQA (Khashabi) et al. (2020)

Table 3.1: Phases of the pipeline associated with the models used to accomplish them.

3.1.1 Span retrieval

The particularity of our pipeline is the mandatory span extraction phase. The challenge here is to teach models to retrieve spans for tasks, such as MC-MRC and G-MRC, which do not provide the spans on which their ground truth answers are based on. That is why we have to use weak supervision techniques to retrieve those spans in the passages. The weak supervision consists in retrieving automatically the spans in the passage which are close to the ground truth answers. Using the ROUGE score and BLEU score can help retrieving linguistically close spans (Kočiskỳ et al. (2018); Mou et al. (2020)). Techniques which are based on the semantics should also be used to retrieve semantically close spans. One solution could be to average the word embedding of the ground truth answers obtained with techniques described in Chapter 1 and to retrieve spans which are close in the vectorial space. However, in our experimentation, only lexical-based weak supervision has been explored due to time constraints.

The Rouge score used is the Rouge-L score (Kočiskỳ et al. (2018); Mou et al. (2020); Min et al. (2019)). Thus, the spans in the passages which have a Rouge-L score superior or equal to θ_r compared to the ground truth answers are extracted. Kočiskỳ et al. (2018), Mou et al. (2020) and Min et al. (2019) use 0.5 as a threshold θ_r, but we found that using 0.6 instead avoids the situation where a two-token answer span which includes a weak word is picked.

To the best of our knowledge, we have not found any reference using BLEU score as weak supervision technique for span retrieval. However, since BLEU-score is usually associated with Rouge score to assess the proximity between two texts, this metric has been used as well as weak supervision for span retrieval. The BLEU score can be computed by comparing unigrams (BLEU-1), bigrams (BLEU-2), trigrams (BLEU-3), etc. Based on our observations, when using n-grams with $n > 2$, the vocabulary size turns out to be too big and therefore the scores less meaningful.

We only compute BLEU-1 and BLEU-2. After applying a grid search on the coefficient of each of the two BLEU scores and looking at the amount of span retrieved, we have decided to compute the BLEU-score by with a linear combination between BLEU-1 (B1 in the following

equation) and BLEU-2 (B2) so that $B = B1 * 0.8 + B2 * 0.2$). So the spans in the passages which have a BLEU score $B \geq \theta_b$ compared to the ground truth answers are extracted. θ_b has been set to 0.5 after manually testing the quality of the retrieved spans on 100 random samples of our model using the values 0.4, 0.5, 0.6 and 0.7.

The statistics regarding the amount of spans retrieved per question are given in Table 3.3

3.1.2 From machine reading to span extraction with BERT

The machine reading phase, the alignment phase, the biased machine reading phase, and the span extraction phase are all taken care of by a BERT for question answering described by Devlin et al. (2018) but adapted with the HARD-EM loss of Min et al. (2019), which suits the cases of multi-mentioning, as already described in Section 2.2.2. By following Min et al. (2019), we consider that only one span (or none when there is no span retrieved) can be the correct the answer. Although we may get more spans which could answer the question correctly or contribute to the answer, this simplification helps to identify noisy answers and lower their influence.

3.1.3 Choice selection and answer generation with UnifiedQA

The choice selection for MC-MRC as well as the answer generation for G-MRC is operated by UnifiedQA, described in Section 2.2.2. This model has been chosen because it does not require to be re-trained for specific MRC datasets to perform on any QA tasks (Khashabi et al. (2020)). It is also possible to use UnifiedQA to perform each step of the baseline pipeline depicted in Figure 2.1. With such application, UnifiedQA takes as input the question, the passage, and the answer candidates for MC-MRC. It returns the predicted answer on the form of free text.

When it is used after the span extraction phase (our pipeline, Figure 3.1), the paragraphs are replaced by the span of the passage likely to contain the answer to the question. The question-and-answer candidates, and ground truth remain the same as for the baseline pipeline. For our experiments, we use the pre-trained model referred to as UnifiedQA-BART in Khashabi et al. (2020). We use the cased version of the model, even though it is suggested to use the uncased model by the authors and our data are pre-processed to be lower-cased. The reason is simply that we noticed a high amount of "No answer" with the lower-case model which are well-handled by the cased model.

3.1.4 Information Retrieval Phase

This phase is operated when the number of paragraph is too large to be processed by the models, such as the datasets in 2.2 which contain "x:1L" or "x:N" in the "Psg:Qst", with "x" which can be any value. Datasets with paragraph-size passages (i.e with "x:1" in the column "Psg:Qst") do not require this IR phase.

In the case of NarrativeQA over stories, the text on which the machine has to retrieve the answer to the question is the complete book, which is too long to be handled either by generative or extractive architecture. Indeed, models like BERT-base cannot handle more than 512 tokens as input (i.e the question plus the passage) (Devlin et al. (2018)). In this situation, a common approach is to process in two steps: reduce the book to a few paragraphs relevant to the question and then answer the questions based on this reduced context (Chen et al. (2017); Lin et al. (2018)).

Since NarrativeQA is the only dataset that we are going to use for training that is affected by this issue and because our approach differs slightly from the existing studies which work on this challenge (QA over stories) (Kociský et al. (2017); Tay et al. (2019); Mou et al. (2020)), we describe the IR phase applied to this dataset.

Chunk creation. Existing methods to split a too-long document into smaller chunks use the number of tokens as a marker. Indeed, Kočiskỳ et al. (2018) and Mou et al. (2020) consider chunks of 200 tokens and Tay et al. (2019) considers different sizes of chunks (from 50 to 500) and determined during training which value improves the validation score the most. This technique leads to information loss due to the tendency to split paragraphs and sentences in the middle. That is why we adjust the chunk size to the size of the paragraph so we can keep together elements meant to be read together. In the case of very short paragraphs (i.e with less than 25 tokens), we group them together so that their size exceeds 100 tokens. This makes it possible to keep the number of chunks reasonable, for example in the case of scripts composed of lines of dialogue. For paragraphs exceeding 500 tokens, we split them approximately in the middle but take care to keep sentences complete.

Chunk Ranking. Once a story is split into chunks, we score the similarity of each of them to a query and rank them. This is a common challenge in the IR field and several methods exist to compute the similarity score. The query can be either either a question (this setup is called *naive ranking (NR)* setup along the thesis) or a question *with* its possible answers (*oracle ranking (OR)* setup). The OR setup performs usually better since it has access to the possible answers and cannot be used at test time. In our case, the OR results aim at defining the best score a perfect chunk ranking could reach based on the given information.

Regarding the similarity metrics which are used, we compute both the cosine similarity between the TF-IDF representation of the chunk and the query, such as Kociský et al. (2017) and Tay et al. (2019), but we also use ranking techniques which have not yet been applied to NarrativeQA over stories such as BM25 (Yang et al. (2018)) and a version of BERT trained for ranking (Nogueira and Cho (2019)). We combine the three methods and get the top-N chunks for each metric. [1].

[1]We use the cosine similarity on the TFIDF vectors from scikit-learn, the BM25 Okapi toolkit (Trotman et al. (2014)) and the pre-trained version on MSMarco Dataset (Nguyen et al. (2016)), with the checkpoint of the model available at `https://github.com/nyu-dl/dl4marco-bert`

So we have [BERTTop$_1$, BERTTop$_2$, ...BERTTop$_N$], [BM25Top$_1$, BM25Top$_2$, ..., BM25Top$_N$] and [TFIDFTop$_1$, TFIDFTop$_2$, ..., TFIDF$_N$] lists that we concatenate so to obtain [BERTTop$_1$, BM25Top$_1$, TFIDFTop$_1$, BERTTop$_2$, BM25Top$_2$, TFIDFTop$_2$, ..., BERTTop$_N$, BM25Top$_N$, TFIDFTop$_N$]. In case of duplicate chunks, only the first occurrence will be kept. Consequently, the size of the list can vary from N to 3N.

Context creation. For the training phase, we use the union of the three top-ranked chunks of each of the three ranking methods and the OR setup, which produce a context with three to nine paragraphs. At test time, we use the five top-ranked chunks of BPRR and BM25 with the NR setup.

3.2 Experiments

Each of the three hypotheses that we have made at the beginning of this chapter are tested through three experiments which are described below.

3.2.1 First hypothesis: one pipeline can perform well on any task

As a reminder, the first hypothesis is that using a shared MRC pipeline composed of models trained with challenge-specific data can perform at least as good on any given task as the task-specific architecture baseline proposed by the research groups which have released the datasets. To test this hypothesis, we train one model per dataset with the same architecture from the machine reading phase to the span extraction phase of our pipeline; we have then one E-MRC architecture but task-specific models. The architecture which performs this training is BERT-base for Question Answering enhanced with HARD-EM loss, as has been described in Section 3.1.2. The datasets which are used to test this hypothesis are listed in Section 3.3. For E-MRC tasks, the output is the predictions of the span extraction phase For MC-MRC and G-MRC tasks, the output is the predictions of the choice selection/answer generation through UnifiedQA, which has already been trained on many QA datasets, as detailed in Section 3.3.3. We compare the predictions from the n models to each of the n datasets' baseline. The baseline is the result that is found in the original paper of each dataset. The hypothesis would be validated if the results are better for each of the datasets tested.

3.2.2 Second hypothesis: a common training gets better generalisation skills than specialized training

Our second hypothesis is that sharing one model in addition to the architecture among n MRC datasets improves the performance on new datasets (datasets seen for the first time at testing time) compared to the results obtained by sharing a unique architecture but using task-specific models. To test this, we compare the predictions of the n models obtained in first hypothesis on m new datasets and we compare it with the predictions when we train a new

model but with the same architecture which include the shuffled data of the n datasets (i.e common training). The n datasets which are use for the training and the m ones used for the test are described in 3.3. The hypothesis would be validated if the results of the common training are better for each of the m datasets than any of the n specialised training.

3.2.3 Third hypothesis: G-MRC performance increases with a span prediction add-on phase

Finally, we hypothesise that the performance of generative MRC tasks increases thanks to the addition of an extractive phase. To test this, if the second hypothesis is validated for the G-MRC datasets, we compare the results of the common training (described in Section 3.2.2) on the G-MRC datasets to the results computed with a model which does not explicitly include a span retrieval phase, i.e following the extracted pipeline (as in Figure 3). To have comparable results, we have chosen to use UnifiedQA as the unique model of our architecture. The hypothesis would be validated if the results of the training with extraction phase gets better results on each G-MRC datasets, regardless of whether they have been seen during the training or not.

3.3 Implementation details

We ran the experiments proposed in the Section 3.2 on several datasets in order to verify our hypothesis. The datasets which are used to train our models are given in Section 3.3.1 and the ones used uniquely for testing purposes are given in Section 3.3.2. Since we use a similar pipeline for each MRC dataset, we need to adapt each datasets' format, so they all fit the unique architecture. This data conversion is described for each MRC dataset in Section 3.3.3. Other implementation details are given in Section 3.3.4.

3.3.1 Datasets for training

In order to test our three hypotheses, we have selected six different datasets among the list given in Table 2.2 that we use for both training and testing. The datasets have been chosen based on their amount of data (to have as much as data as possible for training) and on their diversity (to train robust models). Since there are three different kinds of MRC datasets, namely E-MRC, MC-MRC and G-MRC, we have selected datasets from these three categories. We have SQuAD and COQA as E-MRC. COQA provides the span answer as well as an answer written by a worker based on the answer span; it could thus also be seen as a G-MRC dataset. We have decided to classify it here as a E-MRC because we can use the provided official spans as ground truth answers without having to use any of the weak supervision techniques (listed in Section 3.3). However, we compare the final results with the humanly generated answer as it is more meaningful to judge the quality of the system. RACE and WikiHop are the two

Question	Who is more likely to get an injury during the race?
Passage	<Wikipedia> Sometimes muscles and tendons get injured when a person starts doing an activity before they have warmedup properly. A warmup is a slow increase in the intensity of a physical activity that prepares muscles for an activity. Warming up increases the blood flow to the muscles and increases the heart rate. Warmed-up muscles and tendons are less likely to get injured. For example, before running or playing soccer, a person might jog slowly to warm muscles and increase their heart rate. Even elite athletes need to warm up (Figure below). <Situation> Greg and Carl and about to do a marathon. Greg sees Carl doing some warmups and laughs to himself and thinks it is silly. They both want to get a good time, and are both avid runners.
Answer	Greg

Table 3.2: Example extracted from ROPES development set

datasets chosen for MC-MRC tasks. The difficulty of RACE lies on the reasoning skills that it requires to gather several pieces of evidence from a text to answer a general question : the answer cannot be lexically retrieved in the given passage. By contrast, WikiHop is designed to find which entity, among those in a given passage, correctly completes a triplet. Thus, WikiHop has E-MRC features in addition to MC-MRC ones. Finally the G-MRC task we train on is NarrativeQA, which has two different kind of setups. With the same question and answer, one is given the summary of a book (the challenge over summaries) and the other is given the complete book (the challenge over stories). Since the humans who have written the questions and answers only had access to the summaries and that the summaries are much shorter than the complete books, the task over summaries is easier and close to an E-MRC task. Not many studies tackled the task over stories, which require IR skills as well as generative skills.

3.3.2 Datasets for testing

We use additional datasets uniquely for testing so we can test the generalisation skills of our models for the second and third hypotheses. Those datasets are ROPES and NaturalQuestion. ROPES (Lin et al. (2018)) is a G-MRC dataset which has many questions asked in a MC-MRC. Indeed, two options are usually offered within the question, such as in the example : "Would location A have faster or slower clicks than location B?". The original paper does not provide the number of dual-choice questions. In this dataset, there are two texts as input: a passage extracted from a webpage defined as "background", such as a Wikipedia article, and a situation written by a worker based on this passage. The question is asked based on those two texts and answering it requires collecting information from both texts and to apply multi-hop reasoning. An example is displayed in Table 3.2 in order to illustrate the nature of the background and of the situation. Often, several questions refer to a same passage and situation, but with only one word changing from a question to another. For instance, one could find the question ""Would location A have more or less clicks than location B?".

NaturalQuestions (Kwiatkowski et al. (2019)) is originally not a MRC task but an open-domain QA task; that is why it has not been listed into Table 2.2. However, by finding articles thanks to IR and using them as passages, this task can be seen as a G-MRC one. This IR task is already operated by Khashabi et al. (2020), where the authors also only select the questions with short answers (up to 5 tokens).

3.3.3 Data formatting

The experiments that we lead following the pipeline displayed in Figure 3 use UnifiedQA's architecture to complete each step of the pipeline. In that case, the data only needs to be formatted once at the entrance of the architecture, as explained in Khashabi et al. (2020). The experiments that we lead following our pipeline use the architectures listed in Table 3.1 The first step consists of passing through BERT-base architecture with the modifications proposed by Min et al. (2019) in order to get the span which is predicted as being related to the answers. Since the format is different from the original format from BERT, the architecture is named from here "QA-Hard-EM" for sake of readability, but it refers to the adaptation of BERT as implemented in Min et al. (2019). The MRC datasets listed in Section 3.3 first have to be formatted to fit with the code of the authors. The way to convert each of the selected datasets is explained in the next subsection. The outputs of those models have then to be converted in order to feed UnifiedQA's original code and accomplish the choice selection/answer generation phase. The method to convert the original datasets and replace the original passage by the extracted spans is detailed in the last subsection named *From QA-Hard-EM to UnifiedQA*

Data formatting for QA-Hard-EM

Since we use the model from Min et al. (2019), as explained in Section 3.1.2, we use the same data format as in that paper for the input dataset. Each sample consists of a question, one or several passages, the weakly-supervised answers extracted from each passage, and the final answer. The dataset is stored in a json file containing all samples. An example of a sample taken from COQA with the QA-Hard-EM format is given in Listing 3.1.

Listing 3.1: Formatted example with a unique context passage from COQA dataset

```
{
    "id": "3dr23u6we5exclen4th8uq9rb42tel_6",
    "question": "Was Cotton happy that she looked different than the rest
        of her family?",
    "context": [
        ["once", "upon", "a", "time", ",", "in", "a", "barn", "near", "a"
            , "farm", "house", ",", "there", "lived", "a", "little", "
            white", "kitten", "named", "cotton", ".", "cotton", "lived", "
            high", "up", "in", "a", "nice", "warm", "place", "above", "the
            ", "barn", "where", "all", "of", "the", "farmer", "'", "s", "
```

```
            horses", "slept", ".", "but", "cotton", "wasn", "'", "t", "
            alone", "in", "her", "little", "home", "above", "the", "barn",
             ",", "oh", "no", ".", "she", "shared", "her", "hay", "bed", "
            with", "her", "mommy", "and", "5", "other", "sisters", ".", "
            all", "of", "her", "sisters", "were", "cute", "and", "fluffy",
             ",", "like", "cotton", ".", "but", "she", "was", "the", "only
            ", "white", "one", "in", "the", "bunch", ".", "the", "rest", "
            of", "her", "sisters", "were", "all", "orange", "with", "
            beautiful", "white", "tiger", "stripes", "like", "cotton", "'"
            , "s", "mommy", ".", "being", "different", "made", "cotton", "
            quite", "sad", ".", "she", "often", "wished", "she", "looked",
             "like", "the", "rest", "of", "her", "family", ".", "so", "one
            ", "day", ",", "when", "cotton", "found", "a", "can", "of", "
            the", "old", "farmer", "'", "s", "orange", "paint", ",", "she"
            , "used", "it", "to", "paint", "herself", "like", "them", ".",
             "when", "her", "mommy", "and", "sisters", "found", "her", "
            they", "started", "laughing", ".", "\"", "what", "are", "you",
             "doing", ",", "cotton", "?", "!", "\"", "\"", "i", "only", "
            wanted", "to", "be", "more", "like", "you", "\"", ".", "cotton
            ", "'", "s", "mommy", "rubbed", "her", "face", "on", "cotton",
             "'", "s", "and", "said", "\"", "oh", "cotton", ",", "but", "
            your", "fur", "is", "so", "pretty", "and", "special", ",", "
            like", "you", ".", "we", "would", "never", "want", "you", "to"
            , "be", "any", "other", "way", "\"", ".", "and", "with", "that
            ", ",", "cotton", "'", "s", "mommy", "picked", "her", "up", "
            and", "dropped", "her", "into", "a", "big", "bucket", "of", "
            water", ".", "when", "cotton", "came", "out", "she", "was", "
            herself", "again", ".", "her", "sisters", "licked", "her", "
            face", "until", "cotton", "'", "s", "fur", "was", "all", "all"
            , "dry", ".", "\"", "don", "'", "t", "ever", "do", "that", "
            again", ",", "cotton", "!", "\"", "they", "all", "cried", ".",
             "\"", "next", "time", "you", "might", "mess", "up", "that", "
            pretty", "white", "fur", "of", "yours", "and", "we", "wouldn",
             "'", "t", "want", "that", "!", "\"", "then", "cotton", "
            thought", ",", "\"", "i", "change", "my", "mind", ".", "i", "
            like", "being", "special", "\"", ".", "; ", "what", "color", "
            was", "cotton", "?", "; ", "white", "; ", "where", "did", "she
            ", "live", "?", "; ", "in", "a", "barn", "; ", "did", "she", "
            live", "alone", "?", "; ", "no", "; ", "who", "did", "she", "
            live", "with", "?", "; ", "with", "her", "mommy", "and", "5",
            "sisters", "; ", "what", "color", "were", "her", "sisters",
            "?", "; ", "orange", "and", "white"]
    ],
    "answers": [
```

```
        [{
            "text": "being different made cotton quite sad",
            "word_start": 116,
            "word_end": 122
        }]
    ],
    "final_answers": ["no"]
}
```

For each dataset, the ID of the sample is the same as in the original dataset. The question is taken from the original dataset and lower-cased.

The passages are listed under the label "context". Each passage is lower-cased and tokenized. [2] COQA's passages are composed of a unique paragraph, but other datasets such as Wikihop or NarrativeQA over stories have passages composed of multiple paragraphs. Since COQA is a multi-turn QA dataset, we include the previous questions and answers at the end of the passage. In the example given in Listing 3.1, there are five questions/answers which precede the current question. SQuAD associates one paragraph of an article to each question. Since QA-Hard-EM learns to retrieve the right answer span among several mentions and several paragraphs, it has been decided to include all the paragraphs of the article as context to the model for the training set in order to enable it to perform better negative sampling. Regarding NarrativeQA over stories, which requires an information retrieval phase, we use the context created by our context creation phase described in Section 3.1.4.

Final answer. The final answer consists of the text of the ground truth answer, if one is provided (i.e if this is an answerable question). It can be different from the extracted span, as in the case of COQA, where the answer span is given in addition to a human-written answer based on the span but syntactically and grammatically adapted in order to answer the question. For instance, to the question "Was Cotton happy that she looked different than the rest of her family?", the answer span is "being different made cotton quite sad" while the human-written answer is "No". By contrast, SQuAD only provides the span, so the final answer is the span provided. For RACE and Wikihop, the final answer is the right option from the multiple choices. The NarrativeQA dataset provides two human-written answers which both compose the final answer set. Finally, MSMARCO provides "answers" and "well-formed answers". The "answers" are the answers written by humans, and the "well-formed answers" are those same answers reformulated when :

- the answer does not have proper grammar

[2] The tokenizer used is the BasicTokenizer, which can be found at https://github.com/shmsw25/qa-hard-em/blob/master/tokenization.py

Dataset	avg # par.	min # par.	max # par.	avg # ans.	min # ans.	max # ans.	% No ans.
SQuAD	54.06	10	149	1.0	1	1	0%
CoQA	1.0	1	1	1.0	1	1	0%
RACE	1.0	1	1	0.35	0	20	78.91%
WikiHop	13.67	3	63	4.35	0	109	1.09%
NarQA sum.	2.0	2	2	1.85	0	42	18.75%
NarQA stor.	13.01	2	18	5.4	0	115	32.78%

Table 3.3: The table provide respectively the average (avg.) number (#) of paragraphs (par.), the minimum (min.) number (#) of paragraphs (par.), the maximum (max.) number (#) of paragraphs (par.), the average (avg.) , the average (avg.) number (#) of answer spans (ans.), the minimum (min.) number (#) of answer spans (ans.), the maximum (max.) number (#) of answer spans (ans.) per question and finally the percentage of questions for which no span has been retrieved (% No ans).

- there is a high overlap in the answer and one of the provided passages (because it may indicate that the human did not write the answer but simply pasted a passage from the passage)
- the answer returns an entity without sentence to give context. With the example given in Listing 3.1, "No" would not be accepted as being a well-formed answer but rather "Cotton was not happy that she looked different than the rest of her family."

We take the well-formulated answer when it is provided, otherwise the normal answer.

Answers. The elements in the section "answers" are spans extracted from the section "context". The information it contains is the non-tokenized text of the span as well as the token indices from the "context" section where it starts and ends. For E-MRC, the span is already provided. Sometimes the indices provided in the original datasets are character indices instead of token indices. In that case, we need to recompute the indices. For MC-MRC and G-MRC, the spans are retrieved with the weak supervision described in Section 3.1.1. There are as many lists of answers as there are paragraphs in the context. In the case where there is no possible answer, the lists remain empty. So, if there is no answer possible for any paragraph, the answer will be a list of several empty lists.

Table 3.3 displays statistics about the number of paragraphs composing the context for each question as well as the number of spans retrieved per question (through the entire context).

As a reminder, COQA and Squad, which are E-MRC tasks, already provide spans with their positions, so we do not search for the possible answers but directly use the provided spans; that is why we have a unique answer per sample in those datasets. RACE, especially the high school version, formulates high-level questions and we can see that our weakly-supervised span retrieval techniques, which do not consider the semantics of the words, fail to retrieve a relevant answer span for one-third of the questions. For cases like WikiHop, which has entities

as answers, most of the spans are retrieved. While having a rather high number of paragraphs per context, NarrativeQA over stories also collects many span answers.

From QA-Hard-EM to UnifiedQA

UnifiedQA takes as input examples with the text format

```
<question> \n <candidate answers>  \n <passage> \t < gt_answer>
```

The candidate answers are optional and only used in case of MC-MRC tasks.

The output of QA-Hard-EM model is a json file containing the id of each example (referring to the ID met in Listing 3.1) with one predicted span per sample. The predicted span is used as passage.

The question, the candidate answers for the MC-MRC tasks and the ground truth answers that we provide as input to UnifiedQA are retrieved from the input file described in Section 3.3.3. Since it is only possible to provide one answer as ground truth answer to UnifiedQA, so we give as answer the first element of the "final_answers".

3.3.4 Implementation details

In addition to the data formatting, we trained seven models on QA-Hard-EM: one per training datsaset (individual models) and one common with all the datasets together. We keep the exact same parameters as those proposed in Min et al. (2019). For the testing phase, it is possible to choose how many paragraphs in the "context" to use. For the six individual models, we use the maximum number of paragraphs per question of the concerned dataset, given in Table 3.3. For the common model, we use the maximum number of paragraphs in a context among the six training datasets; which is thus the maximum for SQuAD, equal to 149.

We do not train UnifiedQA further but simply use the trained models described in Khashabi et al. (2020).

3.4 Results

In this section, the results of the experiments are displayed with a first shallow analysis. A deeper and more general analysis held in the Chapter 4. The results are grouped by category: E-MRC, MC-MRC and G-MRC. Within each category, we adapt the evaluation of each dataset depending on whether or not they have been used not for training the span extractive phase (i.e., if they follow our pipeline or not). For each dataset, we present the results after the span

extraction phase and after the choice picking/answer generation phase (see Section 3.1 to be reminded the phases and their associated models) [3].

Each table provides the SOTA's scores at the time of writing [4] as well as the baseline's score. [5] For each dataset also used for training (Section 3.3.1), we show the results when models are trained with their own training set and when models are trained with the common training. For the datasets used uniquely for testing (Section 3.3.2), we show the results when models are trained with the six individual models and when models are trained with the common model. In order to know the performance of the weak supervision and thus the limit of the training, we provide also the score of the weak labels: the span retrieved using weak supervision (see Section 3.1.1). This score is the top score achievable with this weak supervision technique for the span extraction phase. However, for the answer generation phase, the "weak label" score indicates that we directly used the weak label retrieved as passage instead of the span output by the span extraction model. We evaluate the answers returned by the answer generation phase when the weakly retrieved answer span is used as passage in order to provide the top score achievable by the answer generation model by using the weakly supervised span. This is thus a way to assess the quality of the span retrieval weak supervision. Finally, we provide for the answer generation phase the score using the complete passage; so without the extraction phase applied on it, as described in 3.2.3. This score is then the score obtained by applying the pipeline depicted in Figure 3.

Regarding the evaluation measures, we provide five common evaluation measures for all kinds of MRC datasets, namely the F1 score, the exact match (EM), Rouge-L, Bleu-1 and Bleu-4. We run the evaluation on the span extracted and on the final generated answer. In both cases, we compare the results to the ground truth answer.

3.4.1 E-MRC

For both E-MRC tasks, since the answer span are already given, we do not have to apply our span retrieval algorithm (as already explained in Section 3.1.1). For E-MRC tasks which only provide answer spans as ground truth and no human-written answer, such as SQuAD, we use the answer span provided as the ground truth answer for both the span extraction phase and the answer generation phase. However, for E-MRC tasks which provide a human-generated answer in addition to the answer span, as COQA, we use the human-generated answer as the ground truth answer for evaluating both phases. For training the span extraction phase, COQA's ground truth answer span is obviously used. The evaluation of the answer span, since

[3] From now on, for sake of readability, we will use "answer generation phase" instead of "choice picking/answer generation phase". The meaning stays the same because the model that we use, UnifiedQA, takes each MRC task as a G-MRC one.

[4] The SOTA values have been taken from the official leaderboard when it exists or from paperwithcode as of 27/08/2021. We select the best performing results which are linked to a scientific article.

[5] The baseline's scores are the ones given in the original publication presenting the dataset.

SQuAD	F1	EM	Rouge-L	Bleu-1	Bleu-4
Baseline Rajpurkar et al. (2018)	66.3	63.4	-	-	-
SOTA Zhang et al. (2020)	92.97	90.57	-	-	-
Span ext.					
Individual	73.61	59.56	73.76	70.01	29.55
Common	57.91	40.51	58.03	54.04	25.05
Answer span	100	100	100	100	58.48
Answer gen.					
No span pred.	82.31	67.05	82.43	78.78	36.97
Individual	59.33	47.04	59.47	56.04	23.80
Common	48.91	34.26	49.02	45.47	21.01
Answer span	65.91	52.31	66.06	62.26	26.25

Table 3.4: SQuAD evaluation.

it is not retrieved through weak supervision, is named here "ground truth" instead of "weak label". For the answer generation phase, the ground truth answer span of COQA is used as passage for the "answer span" evaluation.

SQuAD

SQuAD tasks expect spans as answers and it would be non-representative to judge the quality of the pipeline by using its evaluation on the generated answer created during the answer generation phase. But for curiosity reasons, the results are still computed and given after this phase.

Since the answer span for SQuAD is also the ground truth answer, the weak label score has to reach 100% for F1, EM, Roue-L and Bleu-1. The reason why the Bleu-4 only gets 58.48% is because many spans have less than 4 tokens.

The performances are significantly better for the spans extracted than for the generated answers based on the spans extracted: for the individual model, the spans extracted are better of 14.28% on F1 score, 11.0% better for the common model and 34.09% when the answer generated are based on the ground truth answer span. This decrease in performance is mainly since the evaluation is based on extracted spans and not on generated answers: the results of the span extracted have then to be better than the generated answer. However, this does not mean that the generated answers are bad, we simply miss ground truth data to assess the quality of the extracted spans.

We can observe that the better the span extracted, the stronger the negative impact of applying an answer generation model over it. Also, we can see that the best score over all trained set-ups is achieved with the answer generation when no span is extracted beforehand. In other words, using the ground truth spans as context and generating an answer from it provides worse results than generating an answer using the entire passage. This counter intuitive result

COQA	F1	EM	Rouge-L	Bleu-1	Bleu-4	
Baseline Reddy et al. (2018)	65.1	-	-	-	-	-
SOTA Ju et al. (2019)	90.7	-	-	-	-	-
Span ext.						
Individual	28.40	11.11	28.56	23.84	10.25	
Common	33.79	15.50	33.87	30.43	12.51	
Answer span	42.28	19.19	42.73	35.95	17.22	
Answer gen.						
No span pred.	42.96	28.94	43.78	39.84	15.38	
Individual	37.64	26.54	37.80	34.74	11.51	
Common	37.87	27.92	37.98	35.30	12.01	
Answer span	85.97	82.32	86.04	84.79	35.50	

Table 3.5: COQA evaluation.

is discussed in Chapter 4.

If we now compare the individual model and the common model, we can notice that the common model achieves results 15.7% lower than the individual model on the span extracted and 10.42% lower on the generated answers.

COQA

COQA provides a ground truth answer span and a ground truth human-written answer, based on the answer span. The lexical closeness of the ground truth answer span with the ground truth human-written answer is given in the "answer span" row of the span extraction phase. The answers generated with UnifiedQA using the ground truth answer span as context is given in the row "answer span" of the answer generation phase.

For this dataset, generating answers based using the extracted spans as passage gives better results than only using the extracted spans: the individual model gains 9.24%, the common one gains 4.07%. The answer generated based on the ground truth spans provides results at least two times better for each evaluation metric than comparing the ground truth span answers with the ground truth human-written spans.

As explained in Section 3.3.3, we include the previous questions and answers to the passage for the span extraction phase. However, we did the same for the answer generation phase and noticed that the results were worse in terms of performance than if we do not include them. Indeed, the F1-scores of the answer generated were respectively for the no span prediction, the individual model, the common model and the answer span setups: 39.83%, 26.75%, 21.92% and 25.76%. The previous questions and answers have thus been excluded from the passage when generating answer with UnifiedQA.

The common model gets better results than the individual one for the span extraction phase;

RACE	F1	EM	Rouge-L	Bleu-1	Bleu-4
Baseline Lai et al. (2017b)	44.1	-	-	-	-
SOTA Jiang et al. (2020)	91.4	-	-	-	-
Span ext.					
Individual	13.21	0.93	13.08	9.07	3.17
Common	13.58	1.43	13.45	9.41	3.49
Weak label	22.31	7.57	22.16	19.07	9.55
Answer gen.					
No span pred.	23.13	19.27	23.01	22.43	17.20
Individual	17.64	12.59	17.46	16.80	11.77
Common	17.25	12.45	17.07	16.46	11.86
Weak label	24.12	21.02	24.04	23.39	18.37

Table 3.6: RACE evaluation. For the SOTA and the baseline, the only measure available is the accuracy

but this advantage vanishes on the generated answer. Indeed, for this phase, both models perform similarly. Regarding the performance of UnifiedQA using only the ground truth answer span as context, the performance of both models is disappointing and stays well below the baseline.

Finally, it has to be noted that using only the ground truth answer span as context for the answer generation provides results which are below the SOTA. For E-MRC models, we notice that the span extraction model, however it is trained, does not perform better than using the complete original context and generating the answer from it.

3.4.2 MC-MRC

RACE

The RACE dataset is a challenge based on questions asked to pupils of two different levels: middle-school group and high-school group. The answers to the questions can be answered using word marching 29.4% of the time for the middle-school group and 11.3% of the time for the high-school group. The other questions require paraphrasing and single and multi-sentence reasoning in order to be answered (Lai et al. (2017a)).

Since our weakly supervised technique to retrieve spans is based on word matching (i.e lexical-based weak supervision), it makes sense to notice that the performance on that dataset was very low and lower than the baseline scores, for both stages. Still, we can notice that the weakly supervised answer spans are more accurate than the spans retrieved with the models: around 9% better in average for the span extraction phase and 6.3% for the answer generation phase on the F1-score. For each challenge, but especially for this one, it would be relevant to choose a semantic-based weakly supervised technique.

WikiHop	F1	EM	Rouge-L	Bleu-1	Bleu-4
Baseline Welbl et al. (2017)	54.5	-	-	-	-
SOTA Beltagy et al. (2020)	81.9	-	-	-	-
Span ext.					
Individual	59.58	51.29	59.56	57.35	18.12
Common	44.15	33.12	44.04	41.73	13.75
Weak label	98.80	98.42	98.80	98.66	30.23
Answer gen.					
No span pred.	4.04	1.18	4.04	3.20	0.88
Individual	33.28	28.15	33.26	31.87	10.32
Common	26.03	18.85	25.98	24.37	8.09
Weak label	55.49	53.30	55.50	54.73	17.16

Table 3.7: WikiHop evaluation.

We cannot see significant differences in the results between the common model and the individual one, although the common model includes training data with ground truth answer spans. We can conclude from this observation that the common model is not appropriate to retrieve answer spans requiring deep reasoning skills. This is also due to the poor quality of the extracted spans. Answers generated using the complete passage perform better than answers generated based on irrelevant spans.

WikiHop

As already mentioned in Section 3.3.1, WikiHop is close to the E-MRC because the answer candidates are each extracted from the given passage. Although we know that the answer cannot be found for each example (Welbl et al. (2017)) due to inconsistency between Wikipedia and Wikidata or because the information is contradictory between the two sources, we always find at least one mention of a candidate answer among the set of passages (see Table 3.3). And as we can see in Table 3.7, the weak supervision allows the system to retrieve the exact answer in 98.42% of the cases.

Consequently, we can observe that the span extraction model performs better than the answer generation phase. However, it does not mean necessarily that the quality of the generated answers is bad. Similarly, as for SQUAD's results, we miss data and qualitative evaluation metrics to judge the quality of those generated answers.

Because of the relevance of a Hard-EM approach to tackle this multi-mentioning task (see Section 2.2.2), the spans extracted with the individual model perform significantly better than those obtained with the common model (+15.43% on the F1-score). Still, the performance is only 5.48% better for the individual model compared to the baseline.

As a final point, we can note that the answer generation phase completely fails (Rouge-L score of 4.04%) when the complete passage is used as context. Reducing the context to one possible

Nar.QA sum.	F1	EM	Rouge-L	Bleu-1	Bleu-4
Baseline Kociský et al. (2017)	-	-	36.30	33.72	15.53
SOTA Nishida et al. (2019)	-	-	59.87	54.11	30.43
Span ext.					
Individual	54.98	32.67	54.84	49.09	19.41
Common	53.54	30.54	53.29	48.01	19.99
Weak label	73.71	53.08	73.30	70.71	31.84
Answer gen.					
No span pred.	46.15	21.38	45.71	40.47	15.28
Individual	48.28	27.24	48.16	42.78	16.63
Common	46.74	25.59	46.55	41.71	16.92
Weak label	63.85	40.42	63.98	59.58	24.91

Table 3.8: NarrativeQA over summaries evaluation.

answer span reduces the chances of error drastically in this experiment.

3.4.3 G-MRC

NarrativeQA over summaries

As already mentioned, NarrativeQA over summaries is a G-MRC challenge which is close to an E-MRC challenge since the workers writing and answering the questions rely on the summary provided, without any creativity constraints. This proximity to E-MRC is assessed by the scores of the weak labels: at least 53.08% of the human-written ground truth answers can be retrieved exactly (EM) in the summaries.

The scores of both the answer span predicted and the answer generated using the span predicted as context outperform the baseline, with generated answers getting better results than the spans extracted. The span extracted performance is under the SOTA from only 5.03% (Rouge-L score) for the individual model and 6.58% for the common model.

The format of this task is the one fitting the most with COQA to our architecture combination (see Table 3.1) since the evidence for the answer to the question can be found in one specific place in the text (so one span can be retrieved containing the answer), and the answer required is not a span but a sentence written by a human referring to the question using the retrieved answer span. Therefore, for this challenge, the results of the generated answer using the extracted span is better than using the complete summary(+2.45%).

We can also notice that the individual model is slightly better, though by less than 2%, for both span extraction phase and the answer generation phase.

Nar.QA sto.	F1	EM	Rouge-L	Bleu-1	Bleu-4
Baseline Kociský et al. (2017)	-	-	14.03	19.09	1.81
SOTA Mou et al. (2020)	22.10	7.34	24.28	26.62	5.03
Span ext.					
Individual	17.94	5.63	17.97	11.49	2.98
Common	19.55	6.73	19.45	15.60	4.41
Individual IR weak label	27.50	15.89	27.51	21.45	5.51
Weak label	52.30	30.81	52.03	46.30	13.68
Answer gen.					
No span pred.	11.83	2.60	11.51	9.61	2.34
Individual	13.10	5.00	13.05	10.17	2.65
Common	13.48	4.54	13.44	11.02	3.05
Weak label	39.74	15.38	41.70	34.14	9.80

Table 3.9: NarrativeQA over stories evaluation.

NarrativeQA over stories

NarrativeQA over stories has been tackled only a few times in the literature (Tay et al. (2019), Mou et al. (2020)) because of the size of its text passages: complete books or complete film scripts. So as a first step, a few paragraphs which are likely to contain the answer to the question are selected because (1) no computer available can handle such a big number of text memory-wise and (2) the complete passage would provide too much misleading information. In Table 3.9, the individual model uses the NR setup explained in Section 3.1.4 to execute the IR phase before giving the top-10 paragraphs to the individual model.

The "individual IR weak label" works the same way, but the ground truth answer is used in addition to the question to retrieve interesting paragraphs with the OR setup. The weak labels, in the span extraction section of Table 3.9, are retrieved from the paragraphs obtained with the OR setup.

This task is complex due to its multiple challenging aspects, and it is difficult to know exactly how to improve the scores. Still, we know this task is doable and offers a great challenge in terms of MRC since we know that many answers can be retrieved with lexical clues in the text; the weak labels correspond exactly to the ground truth answer in 30% of the cases and the average Rouge-L reaches 52%. In comparison, COQA has 19.19% of EM, RACE 7.57%, and NarrativeQA over summaries 53.08%.

Despite the potential of the weak supervision, the models do not perform: the weak labels are more than three times closer to the ground truth answers than the outputs of the common model on the span extraction phase. With an additional IR phase, we can make the hypothesis that the pipeline structure has the drawback of accumulating errors with each change of architecture. Moreover, errors that are made in an early phase make it impossible for the later stages to return a good answer. Despite those drawbacks, the individual and common models

ROPES	F1	EM	Rouge-L	Bleu-1	Bleu-4
Baseline Lin et al. (2019)	61.6	55.5	-	-	-
SOTA Khashabi et al. (2020)	80.25	74.80	-	-	-
Span ext.					
SQuAD	36.42	25.47	36.47	35.45	8.66
COQA	16.34	2.36	16.45	12.53	3.68
NarQA sum.	11.85	3.14	11.87	10.87	3.17
RACE	0.99	0.18	1.02	0.72	0.20
WikiHop	5.45	2.55	5.45	5.12	1.20
NarQAStory	7.41	3.14	7.42	6.18	1.57
Common	15.26	5.09	15.30	14.24	3.85
Answer gen.					
No span pred.	34.50	28.61	34.51	34.12	8.64
SQuAD	49.33	38.33	49.42	48.38	11.22
COQA	35.87	25.59	35.95	33.87	8.11
NarQA sum.	44.80	31.75	45.04	43.57	10.12
RACE	40.54	28.38	40.91	39.36	8.93
WikiHop	46.91	33.95	47.31	45.60	10.27
NarQAStory	44.50	31.40	44.70	43.26	9.88
Common	32.28	20.32	32.47	31.31	7.64

Table 3.10: ROPES evaluation.

achieve better results after the span extraction phase than the baseline on the ROUGE-L scores and the Bleu-4 scores.

For this task, the common model provides better predictions than the individual model, especially for the generated answer. Although the generated answers are less good than the predicted spans, one can notice that they still perform better when they are based on the predicted spans than when they are based on the complete context.

ROPES

ROPES has not been used during any model training but has been used exclusively for testing the generability skills of our models, architecture and pipeline (as explained in Section 3.3.2).

The first thing that one can notice is that the difference in performance between the span extracted and the answer generated is very high and variable from a dataset to another. For instance, the individual COQA and the common model are respectively the second and third best models after the span extraction phase and the two worst ones after the answer generation phase. Also, the RACE model got a ROUGE-L score of 1.02 for it extracted spans but 40.91 for its generated answers. A reason for this discrepancy is that the answer generation phase can pick tokens from the questions as well as the passage to generate the answers. As detailed in Section 3.3.2, the questions of this dataset often include the answer to the question by offering several answer candidates, and that is why we classified it as being close to a MC-MRC

challenge. It appears that, for this challenge, the questions are therefore more helpful than either the extracted spans or complete passages. This could be the reason why the results of the generated answers based on the complete passage would be lower than any test using extracted span as context : this configuration gives more text from which to pick tokens for the answer, so the model would have less chance to simply pick words from the passage. The results of the span extraction are very low : the F1-score is under 4% for the individual models trained with RACE, WikiHop and NarrativeQA over stories. The EM is below 6% for each model of those models. The results of span extracted with the individual models trained with NarrativeQA over summary and stories, RACE, WikiHop are so low, and the answer generated out of those non-performing span are, in comparison, so high (over 40%), that we can infer that the answer generation model picks the elements of the question randomly to generate its answer. The scores would then reflect the behavior of a model answering randomly to a MC-MRC with two answer candidates.

Comparing the individual model of COQA and the common model, we can see that the span extracted can be relevant to answer the questions in 16.34% and 15.26% of cases respectively. However, those relevant spans may confuse the answer generation phase, which would perform better by picking one of the answer candidates given in the question than the extracted span. This could explain the fact those two spans extracted used as context for the answer generation lead to worse result compared with the other models.

The individual model trained with SQuAD distinguished itself with a F1-score equal to 36.42% and an EM equal to 25.47%. It also provides the best answer generation score but with less distance between the scores than the span extracted. We can understand that this is the only model which does not answer randomly between the options offered in the question. Its results are nevertheless far under the baseline.

NaturalQuestion - Short Answers

With NaturalQuestion, the best score among our models is reached by the span extraction model trained individually on SQuAD, as it was for ROPES. The second and third best models after SQuAD are the common model and individual COQA model.

Contrary to ROPES, in this case, keeping the complete context (which is a paragraph from a Wikipedia page retrieved with the DPR engine, as explained in Section 3.3.2) and giving it completely to the answer generation model leads to better results (+8.94%) than SQuAD on span extraction. Similarly, to the Wikihop evaluation, which requires short answers and does not contain answer elements within the question, the performance drops from the extracted span to generated answers. This decrease of performance appears to be due to the training of UnifiedQA, which may be more helpful by creating complete sentences.

In order to understand better the drop of performance between the extracted span and the

Nat.QA	F1	EM	Rouge-L	Bleu-1	Bleu-4
Baseline Kwiatkowski et al. (2019)	31.5	-	-	-	-
SOTA Wang et al. (2020)	63.4	-	-	-	-
Span ext.					
SQuAD	23.36	14.71	23.52	21.17	7.17
COQA	13.71	5.68	13.76	11.43	3.92
NarQA sum.	17.98	9.74	18.03	16.00	5.51
RACE	5.77	1.67	5.79	4.45	1.32
WikiHop	4.30	2.23	4.30	3.66	1.20
NarQAStory	11.94	6.36	11.97	10.20	3.26
Common	18.47	10.30	18.52	16.52	5.75
Answer gen.					
No span pred.	26.30	16.85	26.37	23.62	8.23
SQuAD	13.50	8.78	13.50	12.20	3.92
COQA	9.28	5.09	9.30	8.09	2.63
NarQA sum.	10.81	6.28	10.81	9.63	3.18
RACE	3.22	1.23	3.24	2.65	0.80
WikiHop	2.51	1.46	2.51	2.18	0.67
NarQAStory	7.32	4.27	7.33	6.39	1.97
Common	10.90	6.38	10.91	9.72	3.23

Table 3.11: Natural Question evaluation on short answers.

answer generated on which it is based, we have extracted a sample of span extracted by the common model on both NarrativeQA over stories and NaturalQuestion. NarrativeQA over stories has been chosen for its similar Rouge-L score on their extracted spans with the common model (19.45 for NarrativeQA over stories and 18.52 for NaturalQuestions). We passed through the first 100 questions/span predicted/ground truth answer triples to try to get a clue of the possible issue. The two models seem to provide spans which are related to the question in many cases. For instance, if a character, a date or a place is required, the model returns respectively a character, a date or a place. Often the character/date/place is wrong but from the right type. But when we look at the generated answer, a big difference occurs: the answer generated for NaturalQuestions are irrelevant answers which have nothing to do with the span proposed while NarrativeQA use the elements of the spans provided. The most noticeable mistake is the abundance of "yes" and "no" answer (41.28% of the predictions) in the answer predictions of NaturalQuestion although no questions are of the yes/no form. In comparison, NarrativeQA over stories only gets 12.09% of yes and no together for no yes/no answers. UnifiedQA seems then to be particularly not adapted to NaturalQuestion format.

Chapter 4

Discussion

This master's thesis aims at proposing a general methodology to solve any MRC task which would consist of a pipeline (i.e., on a succession of phases) operated by one or a combination of architecture. In addition, the hypotheses have been made that even one combination of pre-trained models could perform well on any MRC task. The goal would then not be to outperform each SOTA with such general model but rather to acquire general MRC skills.

Experiments have been proposed with architectures which could follow the proposed pipeline as baseline. Since there is no solutions existing yet which follow this methodology, the propositions of architectures and models tested have a baseline status. Our goal is to provide a baseline which would identify the challenges in order to design performing solutions in future works. First insights regarding our hypothesis are then given in Section 4.1. A broader analysis of the results accompanied by some propositions and propositions given in Section 4.2.

4.1 Report on the hypothesis

It has to be noted that the hypothesis which have been made would be validated in case of success of our experiments. However, if the results of our experiments would not verify the hypothesis, this would not invalidate it but rather mean that new experiments with new propositions or architecture combination and newly trained model would must be proposed.

4.1.1 First hypothesis: one pipeline can perform well on any task

As a reminder from Section 3.2.1, the first hypothesis is that using a shared MRC pipeline composed of models trained with challenge-specific data can perform at least as good on any given task as the task-specific architecture baseline proposed by the research groups which have released the datasets. We have thus proposed to use two architectures following the pipeline: one which operates from the machine reading phase to the span extraction phase and one for the answer generation phase. To verify this hypothesis, we look at the results for each

challenge listed in Section 3.3.1 after passing through the pipeline. As a reminder, the first architecture is fine-tuned on each dataset. The second one is already pre-trained and has not been fine-tuned for each dataset (see Section 3.2), which means that in addition to use the same architecture the models share the same weights.

As we can notice in Section 3.4, apart from NarrativeQA over summaries, which gets a better Rouge-L and Bleu-4 score than the baseline (but a worst Bleu-1), the other challenges generate answers with our pipeline which reach a lower score than the baselines. Therefore, we cannot validate our hypothesis with the current results. However, this does not mean that the hypothesis could not be validated with other architecture combination and models following our pipeline. Other combination or architecture with different models must be proposed.

We can notice that the span extracted with 4 different individual models out of 6 get a better score than the associated baselines (i.e., SQUaD, WikiHop, NarrativeQA over stories and over summaries). The new combination of architecture to build has then, in a firsthand, to extract at least as good spans for the 4 succeeding challenge and to enhance its capacities by retrieving spans which are less close lexically but as close semantically. Indeed, for both COQA and ROPES, the weakly supervised answer spans reach themselves less good scores than the baselines. Techniques to enhance the weak supervision in Section 4.2.1.

Also, in future work, in order to have a combination of architecture which validate the hypothesis, the answer generation phase should not generate worse results than the ones obtained with the answer span they are based on. In our experiments, we have used a model pre-trained with UnifiedQA architecture and using paragraph-size passages to answer questions. However, we use this pre-trained model by giving sentence-size passages to answer questions. Fine-tuning an individual model for each dataset which would take weakly retrieved answer span as context would must be tested. Other suggestions regarding the answer generation phase are given in Section 4.2.2.

4.1.2 Second hypothesis: a common training leads to better generalization abilities than specialized training

As a reminder from Section 3.2.2, the second hypothesis is that sharing one model in addition to the architecture among n MRC datasets improves the performence on new datasets (datasets seen for the first time at testing time) compared to the results obtained by sharing a unique architecture but using task-specific models. We have thus trained one span extraction model with the concatenated data from the 6 training datasets listed in Section 3.3.1 (i.e., common model) and re-used UnifiedQA pre-trained using a variety of MRC datasets (Khashabi et al. (2020)). Compared with the pipeline tested for the first hypothesis, only the span extraction models are different. To test this hypothesis, we compare the results of the span extracted using the common model as well as the answers generated based on those with the same outputs

using the 6 individual models described for the first hypothesis, in Section 3.2.1. In order to test the generalization capacities of the common model, we test the outputs on datasets which have not seen during the training. Those 2 datasets are ROPES and NaturalQuestions, which are described in Section 3.3.2.

Merging the dataset to train a model on it, as it is done with the common model leads to better result than the average of the models. Indeed, knowing from each dataset's original paper the number of question-answer pairs in that dataset gives us their distribution in the common training set. In the common training set:

- $\frac{2}{10}$ of the data come from SQUAD
- $\frac{3}{10}$ of the data come from COQA
- $\frac{1}{10}$ of the data come from NarrativeQA, for both over summary and stories (so $\frac{2}{10}$ for both datasets)
- $\frac{2}{10}$ of the data come from RACE
- $\frac{1}{10}$ of the data come from WikiHop

Based on these values, we can compute the average performance of each of the 6 models. For RACE, this average is equal to 15.64 for the F1 score, which is slightly superior to the result of the common model (15.26) (see Section 3.6). For Natural Questions, this average is equal to 13.36 for the F1 score while the common model gets a score of 18.47 (see 3.11). It appears then that the common model increases the generalization skills for NaturalQuestions but not for RACE.

Although our second hypothesis cannot be validated with one positive result on two, a common model trained differently could validate it in future works. In this first experiment, we used the 6 datasets in the common training set regardless of their size or their performance on their own test set. In Section 4.2.3, we give further analysis and suggestions for better composition of the common training set.

4.1.3 Third hypothesis: G-MRC performance increases with a span prediction add-on phase

As a reminder from Section 3.2.3, the third hypothesis is that the performance of generative MRC tasks increases thanks to the addition of an extractive phase. To test this, we compare the answers generated from the common and individual training of the second hypothesis on the G-MRC datasets with the answers generated computed with a model which does not explicitly include a span retrieval phase, so following the extracted pipeline (as in Figure 2.1).

In order to have comparable results, we have chosen to use UnifiedQA as the unique model for the extracted pipeline. All the results which are discussed here are presented in Section 3.4.3.

Although the performance of the answer generated using the span extracted as context are lower than the results of the span extracted themselves for both NarrativeQA over summaries and over stories and for both individual models and the common model, they remain higher than the ones using the complete passage as context. Also, the span extracted with both individual and common model get better results than the baseline on each score except for NarrativeQA over summary which gets a lower Bleu-1 score than the baseline. The extraction phase appears thus to be helpful for those datasets.

Regarding ROPES, the best-performing model is the individual one trained with SQUAD. The spans it provides lead to far better answers than when the complete passage is used as context. When the models are trained on other datasets, drawing conclusion is harder because, as already mentioned in Section 3.4.3, the quality of the spans extracted is low and one can interpret that the model uses mainly the question to generate an answer. In that case, this is more helpful than using the complete context in terms of performance but we cannot qualify this score to result of MRC skills.

NaturalQuestions is the only G-MRC challenge for which neither the answer spans nor the generated answers using the answer span as context gets better results than the answers generated using the complete passage as context. However, the results of the span extracted using SQUAD model (i.e., . 23.36% F1-score) is close to the performance of the answer generated using the complete passage (i.e., 26.30% F1-score)as context compared with the other performances with other set-ups (which are all below 19%). For this dataset, the answer generation phase is deteriorating the transformation of the span extracted. For instance, the span extracted with SQUAD have a F1-score of 23.36% while the answers generated using it have a F1-score if 13.50%. Although we proposed a first shallow qualitative observation in Section 3.4.3 in order to try to understand where does the model fails, the reason of this failure remains unknown. Further investigation is required in future work.

The results on NaturalQuestions does not allow us to assess this third hypothesis. The hypothesis is thus neither validated nor rejected. Therefore, extended tests and analysis must be led in future works. The efforts for future works should be focused on the weak supervision and the evaluation metrics. Indeed, both weak supervision and evaluation metrics are lexical-based and not at all semantic-based. This is especially problematic in the case of the G-MRC challenges where paraphrasing skills are required. Further analysis and propositions for future work are given in Sections 4.2.1 and 4.2.4 for respectively answer span retrieval weak supervision techniques and evaluation metrics. As already suggested for the first hypothesis, the answer generation phase should be either fine-tuned to handle better sentence-size context or another architecture should be used. The suggestions to improve the answer generation

phase are given in Section 4.2.2.

The results of our experiments do not give us enough clue to either validate nor reject the three hypothesis. Our experiments have then more a baseline status which aim at being improved to be able to validate the hypothesis. They give a first overview of how the pipeline can be designed and give the directions how to improve it. In Section 4.2, propositions of future work based on the results of our experiments are given to enable the design of a pipeline able to solve lean general MRC skills.

4.2 Analysis and future works

We have proposed a pipeline to solve any MRC task in Chapter 3 and a first implementation following this pipeline. This implementation is composed of several parts and the choice made for it are described in Section 3.2. This first implementation is used as a baseline and the results of the experiments are thus used as basis to design further combination of architecture and model which could learn general MRC skills. Since the baseline does not allow us to validate or reject our hypothesis, new solutions must be implemented. This section aims at analyzing each implementation decision which have been made to create our baseline and propose accordingly propositions for the design of future solutions.

Section 4.2.1 proposes modification for the weak supervision, Section 4.2.2 for the architectures which are used, Section 4.2.3 for the datasets to use for further fine-tuning and Section 4.2.4 for the evaluation metrics to judge of the quality of the results.

4.2.1 Weak supervision for span retrieval

The technique used to retrieve answer span is described in Section 3.1.1. Briefly, the chosen way to retrieve spans, based on the literature, is to find the spans on the passages which are close to the ground truth answers by using lexical-based metrics such as Rouge-L and Bleu-1 and Bleu-2 scores. Semantic-based metric using the cosine similarity between the word representations to retrieve spans have not been used because they showed limited performance in Kočiskỳ et al. (2018). Other semantic-based metrics have not been tested in this work by lack of reference found on the subject and because a first overview of the capacities of lexical-based metrics were wished to build a baseline.

We could see in Table 3.3 the statistics for the number of answer spans retrieved thanks to the weak supervision (right side on the table), and in each table of Section 3.4, for the datasets belonging to the training datasets, the performance of those answer spans ("weak label" row of the span extraction tables). For sake of readability, Table 4.1 gathers the relevant the results.

For some datasets, such as WikiHop, which requires to return an entity based on several articles, the weak supervision technique is efficient to return in average 4.35 answer spans which

Dataset	avg # ans.	min # ans.	max # ans.	% >0 ans.	F1-score	EM
SQuAD	1.0	1	1	100%	100%	100%
CoQA	1.0	1	1	100%	42.48%	19.19%
RACE	0.35	0	20	22.09%	22.31%	7.57%
WikiHop	4.35	0	109	98.91%	98.80%	98.42%
NarQA sum.	1.85	0	42	81.25%	73.71%	53.08%
NarQA stor.	5.4	0	115	67.22%	52.30%	30.81%

Table 4.1: The table provides on the left part the average (avg.) number (#) of answer spans (ans.), the minimum (min.) number (#) of answer spans (ans.), the maximum (max.) number (#) of answer spans (ans.) per question, the percentage of questions for which at least one span has been retrieved (% >0 ans). The right part displays the F1-score and exact match (EM) scores of the answer spans retrieved compared with the ground truth answers

correspond exactly to the ground truth answer for 98.42% of the question. In comparison, for RACE's challenge, which require multi-hop, paraphrasing and inference skills , the weak supervision returns spans for 21.08% of the questions and 0.35 answers in average per question. The spans retrieved obtain a rouge-L score of 22.16% when compared with the gold answer of the test set.

NarrativeQA over summaries does not require high reasoning skills and the gold answers are expected to be lexically and semantically close to the given passage while being a G-MRC challenge. In that case, our weak supervision retrieves answers for 81.25% of the questions (i.e., the reverse case of retrieving no answer span at all) based on two propositions of ground truth answers. The F1-score that the weakly retrieves answer span reach, compared with the ground truth answer, is equal to 73.30%. Indeed, each span retrieve is cannot and do not aim at matching perfectly the ground truth answer since the answers are humanly written. Still, this score is at least 14% better than any of the three oral IR models of the original paper (Kočiskỳ et al. (2018)), which shows the relevance of our weak supervision to retrieve spans lexically. However, knowing the little number of reasoning skills that it requires to retrieve answer spans while being a G-MRC task, a future weak supervision model should include semantics skills and aim to get a score close to 100%.

Regarding the challenge over stories, we observe that at least one span answers are weakly retrieved in 67.22% of the cases and that in general, far more than one span is retrieved per question: on average, 5.4 answers are retrieved over an average of 13.01 paragraphs. Those retrieved span a Rouge-L score of 52.03 and exact match of 30.81%. Compared with the oracle IR Rouge-L given answer from Kociský et al. (2017), we retrieve 2.09% more spans.

From the results listed above, we can see that our weak supervision technique to retrieve the answer span is efficient to retrieve spans which are lexically close to the ground truth. However, when the questions require paraphrasing, inference or multi-hop reasoning, the metrics used to retrieve the spans is not enough. Semantic-based metrics must be used to complete the

current lexical-based one. Retrieval techniques borrowed from IR should be used and neural networks-based weak supervision should be explored. In that case, COQA dataset could be used as training dataset since it provides ground truth answer spans as well as ground truth human-written spans. Moreover, the lexical distance between those two datasets (the answer spans are exactly the same as the human-written answer for 19.19% of the dataset and reaches a F1-score of 42.48%) would ease the learning of paraphrasing skills.

Designing a performing weak supervised model which retrieve answer spans based on the ground truth answer is the biggest priority on which one should focus on future work since this is the only way to retrieve the spans in the passage on which a machine should rely to return an answer.

4.2.2 Architectures used for training

In this section, the quality of the architectures used for the span extraction phase and the answer generation phase are discussed. Those architecture have been described in Sections 2.2.2, 2.2.2 and 3.1. As a reminder, Bert-base for Question Answering adapted for multi-mentioning with an Hard-EM loss is used as architecture for the span extraction phase. For the answer generation phase, Unified QA architecture is used. As the IR architecture just applies to NarrativeQA over stories and that no data are provided by the original dataset to know the quality of its retrieved passage, the IR phase is not discussed.

Span extraction: BERT-Base for Question Answering with Hard-EM loss

The best dataset that we must judge of the quality of this architecture is SQUAD. Indeed, this is the only dataset for which the ground truth answer is also a span answer. Therefore, we can compare fairly the spans extracted with the ground truth answers. With a F1-score of 66.6% and 92.97% for respectively the baseline and the SOTA (Table 3.4), the architecture trained uniquely with SQUAD (i.e., individual model) perform reasonably good with 73.61% but stays far under the best model.

Wikihop, while being a MC-MRC dataset, has ground truth answers which can be extracted from the spans thanks our weak supervision for 98.91% of the questions, as we can see in Table 4.1. Since the weak supervision works well for this dataset, the performance of the span extraction phase can be interpreted as being the only entirely responsible for the results. Such as for SQUAD, with a F1-score of 54.4% and 81.9% for respectively the baseline and the SOTA (Table 3.7), the individual model perform reasonably good with 59.58% but stays far under the best model.

The other datasets have ground truth answers which are not span based. Evaluating the span extraction architecture without being biased by the weakly retrieved spans it is based on is not possible. Also, the evaluation of the extracted span is done by comparing it with the ground

Dataset	F1	EM	Rouge-L	Bleu-1	Bleu-4
SQuAD	73.61	59.56	73.76	70.01	50.53
COQA	66. 69	57.84	65.23	66.31	59.23
RACE	59.11	12.28	59.02	47.61	33.19
WikiHop	60.30	52.11	60.28	58.12	59.90
NarQA sum.	71.58	61.54	74.81	69.42	60.96
NarQAStory	34.30	18.27	34.53	24.81	21.78

Table 4.2: Scores of span extracted with each individual model normalized on the score of their weakly retrieved spans.

truth answers which are not spans, which means that the best model could not possibly reach maximum reachable score a 100% score. Scores of the extracted spans make sense then only relatively to the scores of the weakly supervised spans. For that reason, we have divided each span extraction score with the individual model for each dataset by the score of their weakly supervised answer spans. In future work, we could measure the evolution of performance of new span extraction models by comparing their normalized results by the ones we got for our 6 individual models, displayed in Table 4.2. The individual models seem to perform with a low variance among the datasets except for NarrativeQA over stories. Indeed, each dataset except NarrativeQA over stories has its F1-score included between 59.11% and 73.61%. NarrativeQA gets lower results with only 34.30%. Further interpretation on the span extraction model based on those results require to be compared with other span extraction models.

Either new architecture for the span extraction should be tested or the same architecture with different training data. According to the second hypothesis, it has been proposed to train the span extraction phase with data coming from different MRC datasets. That is what we refer as our "common training" along the thesis. The analysis of results of this common training with ideas for improvement are detailed in Section 4.2.3. Although the BERT-base architecture that we have used has the advantage to adapt well to the variety of MRC datasets and to provide acceptable spans, new architectures should be tested with the goal to improve the results.

The current span extraction architecture returns only one span. However, for future work, it would be recommended to use or adapt an architecture which is able to return several spans. Indeed, in case of challenges which require multi-hop reasoning, it would be necessary to have an architecture able to return each relevant passages to answer a question.

Answer generation: UnifiedQA

In order to analyze the quality of UnifiedQA architecture, we compare first the results gotten with UnifiedQA when it is used as an end-to-end architecture from the machine reading phase to the answer generation (i.e., when the models take as input the complete passages as context). In a second time, we also compare the results gotten using the answer spans predicted by the individual models for the training dataset and the individual model trained on SQUAD for

	Complete passage					extracted span				
Dataset	F1	EM	R-L	Bleu-1	Bleu-4	F1	EM	R-L	Bleu-1	Bleu-4
SQuAD	82.31	67.05	82.43	78.78	36.97	59.33	47.04	59.47	56.04	23.80
COQA	42.96	28.94	43.78	39.84	15.38	37.64	26.54	37.80	34.74	11.51
RACE	23.13	19.27	23.01	22.43	17.20	17.64	12.59	17.46	16.80	11.77
WikiHop	4.04	1.18	4.04	3.20	0.88	33.28	28.15	33.26	31.87	10.32
NarQA sum.	46.15	21.38	45.71	40.47	15.28	48.28	27.24	48.16	42.78	16.63
NarQAStory	11.83	2.60	11.51	9.61	2.34	13.10	5.00	13.05	10.17	2.65
ROPES	34.50	28.61	34.51	34.12	8.64	49.33	38.33	49.42	48.38	11.22
NatQA	26.30	16.85	26.37	23.62	8.23	13.50	8.78	13.50	12.20	3.92

Table 4.3: Scores of answers generated with UnifiedQA for each dataset using the complete passage as context for the left section and the extracted span as context for the right section.

the two testing datasets (see Section 3.3). As a difference with the span extraction phase, the models are not fine-tuned for each dataset but benefit from a common training with several MRC datasets (Khashabi et al. (2020)). Therefore, the analysis reports also the generalization skills of the model. The results spread along Section 3.4 are reminded in Table 4.3 for sake of readability.

The results obtained by taking the complete passage as context has the advantage to show the performance of the model alone, without being impacted by the performance of any previous one. Compared with the performances of the span extraction phase, we can see that the variation in the performance of the answer generates is high. Indeed, the F1-scores varies from 4.04% for WikiHop to 82.31% for SQUAD. Those scores outperform the dataset's baseline only for SQUAD and NarrativeQA over summaries. Indeed, the baseline's F1-score scores is equal to 66.6% for SQUAD and NarrativeQA over summary's Rouge-L score is equal to 36.30%. [1]

In our experiments, we have not fine-tuned the UnifiedQA model proposed in Khashabi et al. (2020) for each dataset. Fine-tuning the model on each dataset has not been done yet because it goes against the wish to train a unique model, according to the second hypothesis.

The main proposition to improve the answer generation is to reduce the context from the complete passage, which consist on one or several paragraphs to the relevant answer spans. That is what we have tried to do through our pipeline. With our weak supervision technique and our span extraction models, we can see an improvement (Table 4.3) for 4 datasets on 8. However, the average difference between the F1-score on the answer generated using the extracted span as context and the answer generated based on the complete context through the 8 datasets is equal to -0.638. In other words, with the actual models, it does not help in average to reduce the context from the complete passage to extracted spans. In order to improve the performance of the answer generation using the extracted spans as context, the span extraction has reach higher quality, with the propositions mentioned in Section 4.2.1

[1] The F1-score is not given in the original paper for NarrativeQA

and 4.2.2. Moreover, UnifiedQA could be also fine-tuned to be more adapted to sentence-size context rather than paragraph-size contexts. Datasets which could be used for fine-tuning such a model are given in Section 4.2.3.

4.2.3 Datasets used for training

One of our hypothesis is that having one model trained on several MRC challenges would learn general MRC skills better than models trained individually on different datasets (See Section 3). We also proposed a pipeline to solve any MRC challenge. The main innovation of this pipeline is to impose an extraction phase. Therefore, we need at least two different models to solve a MRC task, each being trained on several MRC datasets: a span extraction model and an answer generation model based on the span extracted. Both have received a "common" training, using several MRC datasets as training data.

The common span extraction model has been fine-tuned with the 6 datasets detailed in Section 3.3.1. However, in Section 4.2.2, it has been proposed to change the composition of the datasets to use for training. SQUAD and COQA remain two necessary datasets to use for training since they provide ground truth answer spans. Through the evaluation of the individual models on the testing datasets (ROPES and NaturalQuestions, described in Section 3.3.2 and with their results given in Sections 3.10 and 3.11), they are also the two individual models performing the best. Other E-MRC datasets should be used to have a bigger variety of reasoning. As a more efficient weak supervision will be implemented (see Section 4.2.1), MC-MRC and G-MRC datasets with their weakly-supervised answer spans should also be used. Indeed, E-MRC datasets may focus on lexical matching reasoning while other types of dataset would include more easily deeper MRC skills, such as paraphrasing and multi-sentence reasoning.

The answer generation model is trained with 7 MRC datasets including E-MRC, MC-MRC and G-MRC (Khashabi et al. (2020)). However, in Section 4.2.2, we have seen that a fine-tuning with sentence-size passage instead of paragraph-size must be tested. Therefore, we need to fine-tune UnifiedQA with datasets which have both answer spans and human-written answers. Having a discrepency between the answer span and the human-written answer would also be important so the system learn not just to copy a span but to generate grammatically complete answers which answer the questions. Among the datasets that we have presented in this thesis , only COQA would be a candidate dataset to fine-tune such a model. Indeed, it has both ground truth answer spans and human-written passages. SQUAD and WikiHop's span are also the ground truth answer, so they are not good candidate. RACE, NarrativeQA over summaries and stories, with a performing and reliable weakly supervised answer span retrieval, could be integrated to the training data.

4.2.4 Evaluation metrics

In section 4.2.1, we concluded that the weak supervision should be improved by using semantic-sensitive metrics instead of lexical-semantic metrics, such as Bleu and Rouge scores. The same remark can be done regarding the evaluation metrics since they use the same techniques. The issue is that there is until now no other standard solution which is widely used to evaluate NLG tasks. On the one hand, human evaluation would be the most reliable, but it is the most expansive in addition to being difficult to reproduce (Celikyilmaz et al. (2020)), and there exists no standard criteria for human evaluation methods Celikyilmaz et al. (2020). On the other hand, any machine-learning-based evaluation methods has the risk of being biased through over-fitting or gaming the metrics if they are trained on specific tasks. In future work, we recommend then to use task-agnostic machine-learning-based evaluation metrics which would be trained on many data and many tasks. For instance, MoverScore (Zhao et al. (2019)) compute the Euclidian distance between the contextualized representation of the predicted answer and the ground truth. MoverScore captures both the number of shared data between the predicted answer and the ground truth, but it also captures how much the one text has derived from the other one. Each word of the ground truth is associated with one or more word of the predicted answer and then, the distance is calculated.

Once performing weak supervision will be available, another proposition to improve the evaluate MRC models would be to integrate the evaluation of the span extracted into the global answer evaluation. The result prediction of an answer would be a linear combination between the F1-score to the ground truth span and the newly implemented semantic-based evaluation score to the ground truth answer.

Conclusion

In this thesis, we have proposed and reported the evaluation of a general methodology to tackle the main existing MRC tasks. Our methodology is based on three hypotheses :

- One pipeline can perform well on any MRC task
- A common training gets better generalization skills than specialized training
- G-MRC performance increases with a span prediction add-on phase

We have proposed a first implementation composed of several model which follow this methodology. This baseline is a pipeline which use BERT for question-answering trained with a Hard-EM loss in order to extract spans in a passage which are likely to answer a question. In order to test if one model trained with several MRC datasets would be able to generate good answers on any MRC datasets, we have trained this model using 6 different MRC datasets: SQUAD, COQA, RACE, WikiHop, NarrativeQA over stories and over summaries. 4 of those 6 datasets do not provide ground truth answer spans. Therefore, we proposed a weakly-supervised weak supervision to retrieve them. Then, an answer is generated from the question and the span previously extracted thanks to UnifiedQA. However, the model we use is the one proposed by the authors and this model is trained on the complete passages instead of only a few relevant spans included in it.

Although the results on this baseline does not allow us to validate any of the hypothesis, it provides enough insights to know how to adapt the models in future work in order to improve the baseline and verify the hypothesis. The challenges regarding the weak supervision, the architectures to use, the datasets on which to train each model and the evaluation metrics have been clearly identified so that the hypothesis could be validated in future work.

The weak supervision to be adopted as well as the evaluation metrics must go beyond the word matching but also consider the meaning of the words to be able to identify relevant spans which require deep reasoning skills. Regarding the span extraction architecture, it has to evolve compared to the baseline in order to be able to return several relevant spans and thus allow reasoning over several sentences. Both training of the span extraction architecture and

answer generation architecture must be adapted. The span extraction by using more reliable data. In a first place, this could mean using uniquely datasets which provide ground truth answer spans (i.e., E-MRC). As a reliable weak supervision technique and way to evaluate it will be found, MC-MRC and G-MRC will must be used since they offer questions which require deeper reasoning skills. The level of intelligence would then be improved by using such datasets. Regarding the answer generation architecture, it should be fine-tune on data which have sentence-size context instead of paragraph-size context. Indeed, since our experiments show that using paragraph-size context for the training and testing with sentence-size context does not provide satisfying results, an adapted training has to be implemented.

As a conclusion, this thesis proposes a general methodology how to solve MRC tasks, a baseline which implement it and insights for future work based on this baseline to get a performing and general MRC pipeline.

Bibliography

Angelidis, S., Frermann, L., Marcheggiani, D., Blanco, R., and Màrquez, L. (2019). Bookqa: Stories of challenges and opportunities. *arXiv preprint arXiv:1910.00856*.

Bauer, L., Wang, Y., and Bansal, M. (2018). Commonsense for generative multi-hop question answering tasks. *CoRR*, abs/1809.06309.

Beltagy, I., Peters, M. E., and Cohan, A. (2020). Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Celikyilmaz, A., Clark, E., and Gao, J. (2020). Evaluation of text generation: A survey. *arXiv preprint arXiv:2006.14799*.

Chen, D., Bolton, J., and Manning, C. D. (2016). A thorough examination of the cnn/daily mail reading comprehension task. *arXiv preprint arXiv:1606.02858*.

Chen, D., Fisch, A., Weston, J., and Bordes, A. (2017). Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Choi, E., He, H., Iyyer, M., Yatskar, M., Yih, W., Choi, Y., Liang, P., and Zettlemoyer, L. (2018). Quac : Question answering in context. *CoRR*, abs/1808.07036.

Clark, C. and Gardner, M. (2017). Simple and effective multi-paragraph reading comprehension. *arXiv preprint arXiv:1710.10723*.

Cui, Y., Chen, Z., Wei, S., Wang, S., Liu, T., and Hu, G. (2016). Attention-over-attention neural networks for reading comprehension. *arXiv preprint arXiv:1607.04423*.

Dai, A. M. and Le, Q. V. (2015). Semi-supervised sequence learning. In *Advances in neural information processing systems*, pages 3079–3087.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Dhingra, B., Li, L., Li, X., Gao, J., Chen, Y.-N., Ahmed, F., and Deng, L. (2017). Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–495.

Dhingra, B., Liu, H., Yang, Z., Cohen, W. W., and Salakhutdinov, R. (2016). Gated-attention readers for text comprehension. *arXiv preprint arXiv:1606.01549*.

Feldman, Y. and El-Yaniv, R. (2019). Multi-hop paragraph retrieval for open-domain question answering. *arXiv preprint arXiv:1906.06606*.

Grellet, F. (1986). *Developing Reading Skills: A practical guide to reading comprehension exercises.* Ernst Klett Sprachen.

Harris, Z. S. (1954). Distributional structure. *Word*, 10(2-3):146–162.

Hermann, K. M., Kociský, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. (2015). Teaching machines to read and comprehend. *CoRR*, abs/1506.03340.

Hill, F., Bordes, A., Chopra, S., and Weston, J. (2015). The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301*.

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9:1735–80.

Indurthi, S., Yu, S., Back, S., Cuayáhuitl, H., et al. (2018). Cut to the chase: a context zoom-in network for reading comprehension. Association for Computational Linguistics.

Jiang, Y., Wu, S., Gong, J., Cheng, Y., Meng, P., Lin, W., Chen, Z., et al. (2020). Improving machine reading comprehension with single-choice decision and transfer learning. *arXiv preprint arXiv:2011.03292*.

Joos, M. (1950). Description of language design. *The Journal of the Acoustical Society of America*, 22(6):701–707.

Joshi, M., Choi, E., Weld, D. S., and Zettlemoyer, L. (2017a). Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *CoRR*, abs/1705.03551.

Joshi, M., Choi, E., Weld, D. S., and Zettlemoyer, L. (2017b). Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.

Ju, Y., Zhao, F., Chen, S., Zheng, B., Yang, X., and Liu, Y. (2019). Technical report on conversational question answering. *arXiv preprint arXiv:1909.10772*.

Kadlec, R., Schmid, M., Bajgar, O., and Kleindienst, J. (2016). Text understanding with the attention sum reader network. *arXiv preprint arXiv:1603.01547*.

Khashabi, D., Khot, T., Sabharwal, A., Tafjord, O., Clark, P., and Hajishirzi, H. (2020). Unifiedqa: Crossing format boundaries with a single qa system. *arXiv preprint arXiv:2005.00700*.

Kociský, T., Schwarz, J., Blunsom, P., Dyer, C., Hermann, K. M., Melis, G., and Grefenstette, E. (2017). The narrativeqa reading comprehension challenge. *CoRR*, abs/1712.07040.

Kočiskỳ, T., Schwarz, J., Blunsom, P., Dyer, C., Hermann, K. M., Melis, G., and Grefenstette, E. (2018). The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. Technical report, UNIVERSITY OF SOUTHERN CALIFORNIA MARINA DEL REY INFORMATION SCIENCES INST.

Kozminsky, E. and Kozminsky, L. (2001). How do general knowledge and reading strategies ability relate to reading comprehension of high school students at different educational levels? *Journal of Research in Reading*, 24(2):187–204.

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., et al. (2019). Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Lai, G., Xie, Q., Liu, H., Yang, Y., and Hovy, E. (2017a). Race: Large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794.

Lai, G., Xie, Q., Liu, H., Yang, Y., and Hovy, E. H. (2017b). RACE: large-scale reading comprehension dataset from examinations. *CoRR*, abs/1704.04683.

Levy, O. and Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2019). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Lin, K., Tafjord, O., Clark, P., and Gardner, M. (2019). Reasoning over paragraph effects in situations. *arXiv preprint arXiv:1908.05852*.

Lin, Y., Ji, H., Liu, Z., and Sun, M. (2018). Denoising distantly supervised open-domain question answering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1736–1745.

Melamud, O., Goldberger, J., and Dagan, I. (2016). context2vec: Learning generic context embedding with bidirectional lstm. In *Proceedings of the 20th SIGNLL conference on computational natural language learning*, pages 51–61.

Michel, J.-B., Shen, Y. K., Aiden, A. P., Veres, A., Gray, M. K., Team, G. B., Pickett, J. P., Hoiberg, D., Clancy, D., Norvig, P., et al. (2011). Quantitative analysis of culture using millions of digitized books. *science*, 331(6014):176–182.

Mihaylov, T. and Frank, A. (2018). Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge. *CoRR*, abs/1805.07858.

Mikolov, T., Chen, K., Corrado, G. S., and Dean, J. (2013). Efficient estimation of word representations in vector space.

Min, S., Chen, D., Hajishirzi, H., and Zettlemoyer, L. (2019). A discrete hard em approach for weakly supervised question answering. *arXiv preprint arXiv:1909.04849*.

Mou, X., Yu, M., Yao, B., Yang, C., Guo, X., Potdar, S., and Su, H. (2020). Frustratingly hard evidence retrieval for qa over books. *arXiv preprint arXiv:2007.09878*.

Nguyen, T., Rosenberg, M., Song, X., Gao, J., Tiwary, S., Majumder, R., and Deng, L. (2016). Ms marco: A human-generated machine reading comprehension dataset.

Nishida, K., Saito, I., Nishida, K., Shinoda, K., Otsuka, A., Asano, H., and Tomita, J. (2019). Multi-style generative reading comprehension. *arXiv preprint arXiv:1901.02262*.

Nogueira, R. and Cho, K. (2019). Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*.

Ostermann, S., Modi, A., Roth, M., Thater, S., and Pinkal, M. (2018). Mcscript: A novel dataset for assessing machine comprehension using script knowledge. *CoRR*, abs/1803.05223.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *In EMNLP*.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Peters, M. E., Ruder, S., and Smith, N. A. (2019). To tune or not to tune? adapting pretrained representations to diverse tasks. *CoRR*, abs/1903.05987.

Pichotta, K. and Mooney, R. J. (2016). Using sentence-level lstm language models for script inference. *arXiv preprint arXiv:1604.02993*.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Rajpurkar, P., Jia, R., and Liang, P. (2018). Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016a). SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016b). Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Reddy, S., Chen, D., and Manning, C. D. (2018). Coqa: A conversational question answering challenge. *CoRR*, abs/1808.07042.

Richardson, M., Burges, C. J., and Renshaw, E. (2013a). MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203, Seattle, Washington, USA. Association for Computational Linguistics.

Richardson, M., Burges, C. J., and Renshaw, E. (2013b). Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203.

See, A., Liu, P. J., and Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Seo, M., Kembhavi, A., Farhadi, A., and Hajishirzi, H. (2016). Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Talmor, A. and Berant, J. (2019). Multiqa: An empirical investigation of generalization and transfer in reading comprehension. *arXiv preprint arXiv:1905.13453*.

Tay, Y., Tuan, L. A., Hui, S. C., and Su, J. (2018). Densely connected attention propagation for reading comprehension. *arXiv preprint arXiv:1811.04210*.

Tay, Y., Wang, S., Tuan, L. A., Fu, J., Phan, M. C., Yuan, X., Rao, J., Hui, S. C., and Zhang, A. (2019). Simple and effective curriculum pointer-generator networks for reading comprehension over long narratives. *arXiv preprint arXiv:1905.10847*.

Trischler, A., Wang, T., Yuan, X., Harris, J., Sordoni, A., Bachman, P., and Suleman, K. (2016a). Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830*.

Trischler, A., Ye, Z., Yuan, X., and Suleman, K. (2016b). Natural language comprehension with the epireader. *arXiv preprint arXiv:1606.02270*.

Trotman, A., Puurula, A., and Burgess, B. (2014). Improvements to bm25 and language models examined. In *Proceedings of the 2014 Australasian Document Computing Symposium*, pages 58–65.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Vinyals, O., Fortunato, M., and Jaitly, N. (2015). Pointer networks. *arXiv preprint arXiv:1506.03134*.

Wang, S. and Jiang, J. (2016). Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905*.

Wang, X., Shou, L., Gong, M., Duan, N., and Jiang, D. (2020). No answer is better than wrong answer: A reflection model for document level machine reading comprehension. *arXiv preprint arXiv:2009.12056*.

Welbl, J., Stenetorp, P., and Riedel, S. (2017). Constructing datasets for multi-hop reading comprehension across documents. *CoRR*, abs/1710.06481.

Wixson, K. K. and Lipson, M. Y. (1991). Perspectives on reading disability research.

Xie, Q., Lai, G., Dai, Z., and Hovy, E. (2017). Large-scale cloze test dataset created by teachers. *arXiv preprint arXiv:1711.03225*.

Yang, P., Fang, H., and Lin, J. (2018). Anserini: Reproducible ranking baselines using lucene. *Journal of Data and Information Quality (JDIQ)*, 10(4):1–20.

Zhang, X. and LeCun, Y. (2015). Text understanding from scratch. *arXiv preprint arXiv:1502.01710*.

Zhang, Z., Yang, J., and Zhao, H. (2020). Retrospective reader for machine reading comprehension. *arXiv preprint arXiv:2001.09694*.

Zhao, W., Peyrard, M., Liu, F., Gao, Y., Meyer, C. M., and Eger, S. (2019). Moverscore: Text generation evaluating with contextualized embeddings and earth mover distance. *arXiv preprint arXiv:1909.02622*.