DEEP LEARNING FOR QUESTION ANSWERING

by

Merve Ünlü

B.S., Computer Engineering, Galatasaray University, 2012

M.S., Computer Science, ENS LYON , 2014

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering

Boğaziçi University

2019

DEEP LEARNING FOR QUESTION ANSWERING

APPROVED BY:

Assoc. Prof. Arzucan Özgür .....
(Thesis Co-supervisor)

Assist. Prof. Ebru Arısoy Saraçlar .....
(Thesis Co-supervisor)

Assist. Prof. Şeniz Demir .....

Prof. Tunga Güngör .....

Prof. Murat Saraçlar .....

DATE OF APPROVAL: 12.07.2019

ACKNOWLEDGEMENTS

Foremost, I would like to express my sincere gratitude to Dr. Ebru Arısoy Saraçlar and Prof. Murat Saraçlar for their continuous support of my study and research, for their patience, motivation, enthusiasm, and immense knowledge. Their guidance helped me in all the time of research and writing of this thesis. I would like to thank Dr. Arzucan Özgür for very helpful comments and suggestions on this research.

Besides my advisors, I would like to thank the rest of my thesis committee: Prof. Tunga Güngör and Dr. Şeniz Demir for accepting to be in my thesis committee, and devoting their time to read the dissertation.

The completion of this thesis could not have been accomplished without the support of my classmates from Boğaziçi University and my colleagues from Galatasaray University. I am very thankful for their support and their comments on my work.

Any attempt at any level can't be satifactorily completed without the support and guidance of my family. I am very grateful for their understanding and their support through my difficult times.

I would like to thank the faculty members of MEF University for giving permission for using their flipped learning videos for this research. This work was supported by the TÜBİTAK-ARDEB 3501 Program (Project No: 117E202).

ABSTRACT

DEEP LEARNING FOR QUESTION ANSWERING

This study describes a question answering (QA) system developed with deep learning methods on spoken documents. Question answering on spoken documents is mainly performed by transcribing spoken content with an automatic speech recognition (ASR) system and then applying text-based question answering methods to the ASR transcripts. The questions are presented to the system in written or spoken form and the answers are returned from spoken documents. QA task on spoken documents is more challenging than on text documents. Firstly, spoken documents do not have explicit paragraph boundaries so that end-to-end neural network based systems that utilize question-aware passage representations for retrieving answers may perform poorly. Secondly, ASR transcripts can be erroneous and this effects the performance of QA systems. Therefore we propose two novel approaches to handle these problems in deep learning based end-to-end spoken QA systems. First approach handles the absence of passage boundaries in spoken documents by generating pseudo passages and automatically determining questions related with each pseudo passage. Second approach integrates ASR system output, confusion networks with word confidence scores, into a question answering system. Our proposed approaches are integrated into an end-to-end system and we investigate the capability of these approaches with two newly curated QA datasets. The end-to-end neural network model with the proposed extensions has proven to be effective in spoken QA task and improved the QA performance on spoken documents compared to directly applying the end-to-end model to the ASR transcripts of the spoken documents.

ÖZET

SORU CEVAPLAMA SİSTEMLERİ İÇİN DERİN ÖĞRENME

Bu çalışmada sözlü belgeler üzerinde derin öğrenme yöntemleriyle geliştirilen soru cevaplama sistemleri açıklanmaktadır. Sözlü belgeler üzerindeki soru cevaplama sistemleri ağrılıklı olarak sözlü içeriğin otomatik konuşma tanıma (OKT) sistemi ile yazılandırılması ve ardından metin tabanlı soru-cevaplama tekniklerinin bu OKT çıktılarına uygulanması ile gerçekleştirilir. Sorular sisteme yazılı ya da sözlü olarak sunulur ve cevapları konuşma verilerinden döndürülür. Sözlü belgelerdeki soru cevaplama, metin belgelerinden daha zordur. Birincisi, sözlü belgelerin açıkça paragraf sınırları yoktur; bu nedenle, cevapları almak için soru-bilinçli geçiş gösterimlerini kullanan uçtan uca sinir ağı tabanlı sistemler yetersiz performans gösterebilir. İkinci olarak, OKT çıktıları hatalı olabilir bu durum soru cevaplama sisteminin performansını etkiler. Bu nedenle, konuşma belgeleri üzerindeki soru cevaplama sistemlerinde bu sorunları ele almak için derin öğrenme modeline dayanan iki yeni yaklaşım öneriyoruz. İlk yaklaşım, sözde paragraflar üretip bunlar üzerinden ilgili soruları otomatik tanımlama yaparak konuşulan belgede geçiş sınırlarının bulunmaması sorununu ele almaktadır. İkinci yaklaşım OKT sistem çıktısı olan karışıklık ağını kelime güven skorları ile beraber uçtan uca sinir ağı tabanlı soru cevaplama sistemi ile bütünleştirir. Önerdiğimiz modeller uçtan-uca sinir ağı modeli ile birleştirildi ve bu modellerin performansı oluşturulan iki yeni veri seti ile incelendi. Önerilen uzantıları içeren uçtan uca sinir ağı modelinin konuşma belgeleri üzerinde soru cevaplama üzerinde etkin olduğu kanıtlanmıştır ve soru cevaplama performansını OKT çıktılarını direkt uçtan-uca sinir ağı modelinde kullanmaya göre geliştirmiştir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . iii
ABSTRACT . . . iv
ÖZET . . . v
LIST OF FIGURES . . . viii
LIST OF TABLES . . . xi
LIST OF SYMBOLS . . . xiii
LIST OF ACRONYMS/ABBREVIATIONS . . . xvi
1. INTRODUCTION . . . 1
1.1. Problem Definition . . . 4
1.2. Contributions . . . 5
1.3. Thesis Outline . . . 6
2. BACKGROUND AND RELATED WORK . . . 8
2.1. Question Answering on Textual Documents . . . 8
2.2. Question Answering on Spoken Documents . . . 12
2.3. Question Answering in Turkish . . . 14
2.4. Deep Learning Models . . . 14
2.5. Evaluation . . . 17
3. METHODOLOGY . . . 20
3.1. Problem Formulation . . . 20
3.2. Match LSTM . . . 20
3.2.1. Long Short-Term Memory (LSTM) . . . 21
3.2.2. Pointer Network . . . 26
3.2.3. MatchLSTM . . . 28
3.3. Question-Answering System Extensions . . . 31
3.3.1. Passage-Question Relevance Scoring . . . 32
3.3.2. Integrating ASR Uncertainty into Question Answering . . . 33
4. EXPERIMENTS AND RESULTS . . . 38
4.1. Datasets . . . 38
4.1.1. Stanford Question Answering Dataset - SQuAD . . . 38

4.1.2. Question-Answer Dataset for English Lectures 39
4.1.3. Question-Answer Dataset for Turkish Lectures 41
4.2. Implementation Details 43
4.2.1. QA System 43
4.2.2. ASR System 46
4.3. Results 47
4.3.1. MatchLSTM (Baseline) for QA 47
4.3.2. Passage-Question Relevancy Model 48
4.3.3. ASR uncertainty into QA with Confusion Network 54
4.4. Further Analysis of the Results 56
4.4.1. Examples 60
5. CONCLUSION 66
REFERENCES 67

LIST OF FIGURES

Figure 1.1. A Typical QA system. 2

Figure 2.1. SQuAD evaluation results for version 1.1 (left) and version 2.0 (right). 15

Figure 2.2. General architecture of neural QA systems. 16

Figure 2.3. Timeline for several models tested with SQuAD. 17

Figure 2.4. Confusion matrix with definitions. 18

Figure 3.1. General framework of MatchLSTM model. 22

Figure 3.2. Feed forward neural network. 23

Figure 3.3. Recurrent neural network. 23

Figure 3.4. RNN diagram with unfolded and folded versions. 24

Figure 3.5. LSTM cell. 25

Figure 3.6. Pointer network example. 28

Figure 3.7. Confusion network and ASR one-best example. 34

Figure 3.8. Confusion scores as features in MatchLSTM model. 35

Figure 3.9. Weighted word vector representations with confidence scores. . . . 36

Figure 4.1. Word count distribution for questions/answers in SQuAD training set. 40

Figure 4.2. Word count distribution for questions/answers in SQuAD development set. 40

Figure 4.3. Word count distribution for question/answers in training set of English lectures QA dataset. 41

Figure 4.4. Word count distribution for question/answers in evaluation set of English lectures QA dataset. 41

Figure 4.5. Word count distribution for question/answers in training set of Turkish lectures QA dataset. 43

Figure 4.6. Word count distribution for question/answers development set of Turkish lectures QA dataset. 43

Figure 4.7. F1-Score/Exact match plot and loss plot of training and development sets of Turkish QA dataset. 45

Figure 4.8. F1-Score with different window size on English lectures QA dataset. 51

Figure 4.9. F1-Score by the length of the answer (top) and number of samples by length of answer span (bottom) for Turkish lectures QA dataset. 57

Figure 4.10. F1-Score by the length of the question (top) and number of samples by length of question span (bottom) for Turkish lectures QA dataset. 58

Figure 4.11. F1 score by question type and the number of questions by types in Turkish lectures QA dataset. 59

Figure 4.12. Question-aware passage representation for the sample in Table 4.16. 61

Figure 4.13. Pointer network attentions for the sample in Table 4.16. 61

Figure 4.14. Question-aware passage representation for the sample in Table 4.18. 64

Figure 4.15. Attention vectors from pointer network for the sample in Table 4.18. 65

LIST OF TABLES

Table 1.1. Question-answer pairs with related text from SQuAD [1]. 4

Table 1.2. Question-answer pairs with related text from SQuAD [1]. 5

Table 2.1. Logical form example from semantic parser. 10

Table 2.2. Properties of several recent datasets. 11

Table 4.1. Question-answer pairs with the passage from SQuAD. 39

Table 4.2. SQuAD statistics. 39

Table 4.3. English lectures QA statistics. 41

Table 4.4. Question-answer pairs with related text from Turkish lectures QA dataset. 42

Table 4.5. Turkish lectures QA dataset statistics. 42

Table 4.6. SQuAD dataset results with MatchLSTM. 47

Table 4.7. MatchLSTM (baseline) results for English lectures QA dataset. . . 48

Table 4.8. MatchLSTM (baseline) results for Turkish Lectures QA dataset. . 48

Table 4.9. Results for different test scenarios when passage-question pairs are known for English lectures QA dataset. 50

Table 4.10. Results for different test scenarios with PQ-model for English lectures QA dataset. 52

Table 4.11. Results for different test scenarios when passage-question pairs are known for Turkish lectures QA dataset. 53

Table 4.12. Results for different test scenarios for Turkish lectures QA dataset. 54

Table 4.13. Results when using confidence scores as features. 55

Table 4.14. F1 score of the model when using with different parameters for Turkish lectures QA dataset. 56

Table 4.15. Question classes for Turkish lectures QA dataset. 59

Table 4.16. Question-answer pairs with related text from Turkish lectures QA dataset. 60

Table 4.17. Question-answer pairs with true prediction span from Turkish lectures QA dataset. 62

Table 4.18. Question-answer pairs with incorrect prediction span from Turkish lectures QA dataset. 63

LIST OF SYMBOLS

a	Answer
a^i	Answer of ith sample
a_n	Answer of nth sample
a_{end}	End index of the answer span
a_{start}	Starting index of the answer span
b^a	Bias term of pointer network
b_c	LSTM's cell state bias term
b_f	LSTM's forget gate bias term
b_i	LSTM's input gate bias term
b_o	LSTM's output gate bias term
b_P	MatchLSTM's bias term
b^o_{rel}	Bias term of passage-question relevance model
c_t	Current state vector at time t
$\tilde{C}_t$	LSTM's inner-candidate vector at time t
C_t	LSTM's cell state vector at time t
d	Dimension
d_t	Decoder hidden state at time t
d^l_i	l^{th} vector of d_i
e_t	Encode hidden state at time t
f	Function
F	Cell function of LSTM
$\mathbf{F_k}$	Pointer network inner matrix
f_t	LSTM's forget gate output at time t
$\overrightarrow{G}_i$	MatchLSTM layer inner representation in forward direction
$\overleftarrow{G}_i$	MatchLSTM layer inner representation in backward direction
h^a_k	Hidden state of pointer network
$\overrightarrow{h_i}$	Hidden state vector at time t in forward direction
$\overleftarrow{h_i}$	Hidden state vector at time t in backward direction

h_t	Hidden state vector at time t
$\mathbf{H^P}$	Hidden representation of passage
$\mathbf{H^r}$	Output matrix of MatchLSTM layer
$\overrightarrow{\mathbf{H^r}}$	Output matrix of MatchLSTM layer in forward direction
$\overleftarrow{\mathbf{H^r}}$	Output matrix of MatchLSTM layer in backward direction
$\mathbf{H^Q}$	Hidden representation of question
i_t	LSTM's input gate output at time t
l	Dimension of hidden vectors
$\overrightarrow{LSTM}$	LSTM in forward direction
$\overleftarrow{LSTM}$	LSTM in backward direction
N	Number of samples
o_t	Output of LSTM's output gate
P	The length of the passage
P_n	Passage of nth sample
p	Passage
$\mathbf{P}$	Passage matrix
p^i	Passage of ith sample,set of words
p_i	The jth token of the passage p
Q	The length of the question
Q_n	Question of nth sample
q	Question
$\mathbf{Q}$	Question matrix
q^i	Question of ith sample
q_i	The jth token of the question q
$\mathbb{R}$	Real number class
$Tanh$	Hyperbolic tangent function
u_j^i	Inner-vector representation at time t for sample i
v	Parameter vectors
$\mathbf{V}$	Parameter of pointer network
$\mathbf{W}$	Parameter of pointer network
W_1	Parameters of encoder-decoder models

W_2	Parameters of encoder-decoder models
W_c	LSTM's cell states weights
W_f	LSTM's forget gate weights
W_i	LSTM's input gate weights
W_o	LSTM's output gate weights
$\mathbf{W_p}$	MatchLSTM's weights for passage representation
$\mathbf{W_q}$	MatchLSTM's weights for question representation
$\mathbf{W_r}$	MatchLSTM's weights for hidden representation
$\mathbf{W^h_{rel}}$	Weight of passage-question relevance model
$\mathbf{W^o_{rel}}$	Weight of passage-question relevance model
x	Input
X	Input sequence
x_t	Input at timestamp t
y	Output
Y	Output sequence
y_t	Output at timestamp t
$\overrightarrow{z_t}$	MatchLSTM input at time step t in forward direction
$\overleftarrow{z_t}$	MatchLSTM input at time step t in backward direction
α_i	Attention vector for i^{th} token of the passage
$\overrightarrow{\alpha}_i$	Attention vector for i^{th} token of the passage in forward direction
$\overleftarrow{\alpha}_i$	Attention vector for i^{th} token of the passage in backward direction
β_t	Attention vector in pointer network layer for token t
θ	Parameter set
σ	Sigmoid function
$\otimes e_N$	Operation that repeats the vector N times

LIST OF ACRONYMS/ABBREVIATIONS

2D	Two Dimensional
3D	Three Dimensional
AMRNN	Attention-based Multi-Hop Recurrent Neural Network
ASR	Automatic Speech Recognition
BERT	Bidirectional Encoder Representations from Transformers
BOW	Bag-of-Words
DNN	Deep Neural Network
EM	Exact Match
FN	False Negative
FP	False Positive
FNN	Feedforward Neural Network
GMM	Gaussian Mixture Model
GMM-si	speaker independent GMM
GMM-sa	speaker adaptive GMM
GRU	Gated Recurrent Network
IR	Information Retrieval
LSTM	Long-Short Term Memory
MRC	Machine reading comprehension
NLI	Natural Language Inference
NLP	Natural Language Processing
ODSQA	Open-Domain Spoken Question Answering Dataset
POS	Part-of-speech
PQRelevance	Passage-Question Relevance Model
QA	Question Answering
QAst	Question Answering on Speech Transcripts
RC	Reading Comprehension
RNN	Recurrent Neural Network
SQA	Spoken Question Answering

SQuAD	Stanford Question Answering Dataset
TN	True Negative
TP	True Positive
WER	Word Error Rate

1. INTRODUCTION

Extracting information from structured or unstructured resources is a daily need when we use a search engine or social media platform. *Information Retrieval (IR)*, an academic field of study, has been developed to seek answers to the problem of extracting information from various resources like textual data, web pages, videos, documents, databases etc. The goal of an IR system is to retrieve documents with information that is relevant to the user's need and helps the user complete a task. This task can be ranking relevant pages, answering questions or summarizing documents. Even though the users need different applications, the main aspect is to make the machines to understand natural language.

Question Answering (QA) is an information retrieval task where the user expects an answer to a submitted query. It aims to retrieve an answer by collecting information from one or more resources rather than returning a collection of documents like in most retrieval systems. For example, when a user asks *Where is the capital of Turkey?*, he/she expects an exact answer as *Ankara*, not intends to read documents or related pages.

QA systems can be used for a great variety of purposes and their use cases may vary widely. A use case example would be that of a tourist looking for a vegetarian place the eat. The user would enter the question as: *Where are the best vegetarian restaurants near me?* A QA system would determine the answer type, extract the keywords to search through the available documents, choose and rank relevant pages, extract a list of answers and present the high-rated vegetarian restaurant to the user.

Another example use case of QA system can be a virtual teacher in e-learning systems. These collaborative learning systems are important, since they allow the communication and information flow inside the learning groups. To gain high performance from these systems, an automated QA system can be added as a virtual teacher to respond to the students' questions. While the students can get answers to their questions

in real time, the instructors can understand what kind of problems that the students encountered in learning.

Figure 1.1 shows a general diagram for a QA system. A typical QA system has three modules:

(i) Question Processing: Given a natural language question asked by a human, the question processing module analyzes and creates representations depending on the system. These inner representations can be database queries or logical forms. The questions can be classified, reformulated or mapped into these inner representations. The output of this module can be a set of query terms or keywords that are used for retrieving relevant documents.

(ii) Document Processing: This module is also referred as passage indexing module. Its aim is to return a set of relevant passages that contain a candidate answer to the query from various resources. After extracting query terms from the question, the system searches for the relevant documents, passages, web pages or database tables. From the set of relevant documents, the module filters and ranks them to find the answer.

(iii) Answer Extraction: The module takes the relevant passage/document as input from the document processing module. A candidate answer is extracted from the entries to respond the question by identifying candidates, extracting and validating the answer.

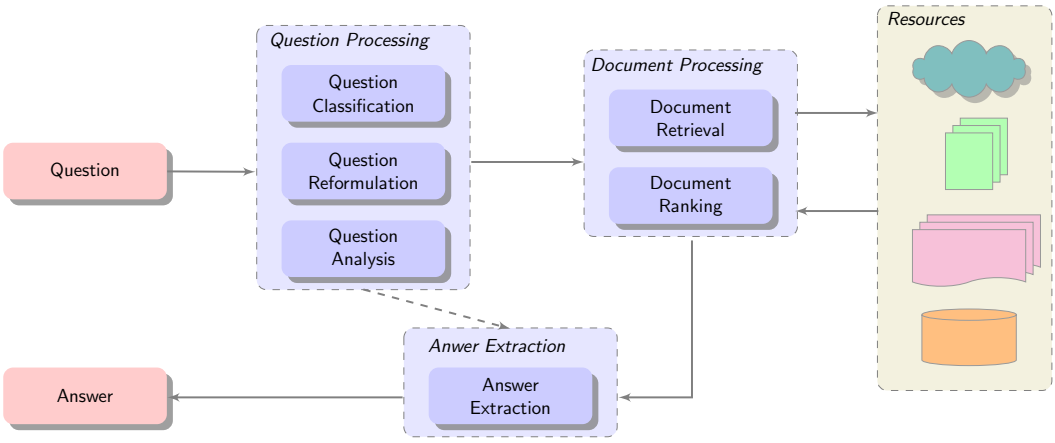

Figure 1.1. A Typical QA system.

QA system that is shown in Figure 1.1 is a traditional IR type system. In recent years, there has been huge developments on QA task with end-to-end systems which represents a complete system, bypassing the intermediate modules usually present in traditional pipeline designs. The idea of using an end-to-end model is to have a system that can specialize to answer questions directly from the texts without any complex language processing. These models emphasize the importance of reasoning and machine comprehension in question answering on text documents.

The state-of-the-art end-to-end QA models developed with recent deep neural network methods have already reached the human performance for QA on text documents. The ease of access to the text has facilitated the collection of QA datasets, which has enabled the development of systems working on textual data. Although spoken documents are becoming widespread, such progress cannot be seen through QA systems on spoken documents.

Main focus of this thesis is to develop a question answering system on spoken documents. Starting from the machine comprehension task which aims to answer a question over a short passage, we investigated the ability of an end-to-end neural network based QA model on spoken data. Different from textual data, spoken content generally does not possess passage boundaries, so that competitive neural network models developed for machine comprehension can not be directly utilized on spoken documents. In order to meet this need, we generated pseudo passages from transcriptions of spoken documents and we extended our neural network model so that it can first predict passage/question pairs before performing the QA task.

QA on spoken documents is mainly performed by transcribing spoken content with an automatic speech recognition (ASR) system and then applying text-based question answering methods to the ASR transcripts. Information extraction from speech requires developing systems that can deal with the uncertainty in the spoken content, because ASR transcripts are prone to recognition errors. Therefore, we propose two approaches to integrate ASR uncertainty into the QA system to improve the system performance.

To develope our proposed spoken QA system, we generated two datasets: one for Turkish and one for English. The datasets consist of short passages and question/answer pairs for each passage. The datasets were generated by annotating the reference transcriptions of video lectures. We first divided the reference texts into short passages and then manually generated questions for each passage and specify the answers as span words from the passages.

1.1. Problem Definition

For a given passage and a question, the task is defined as finding an answer which is a part of the passage. Table 1.1 gives an example for the task. The text is a passage from Wikipedia article about Tesla. To answer the first question, the machine must be able to understand the syntax of the document where the dates have a pattern. For the second question, the machine must establish the relation between the abbreviations and the words.

Table 1.1. Question-answer pairs with related text from SQuAD [1].

Nikola Tesla (Serbian Cyrillic: 10 July 1856 – 7 January 1943) was a Serbian American inventor, electrical engineer, mechanical engineer, physicist, and futurist best known for his contributions to the design of the modern alternating current (AC) electricity supply system.
Question: In what year was Nikola Tesla born? Answer: 1856 Question: What does AC stand for? Answer: alternating current

The problems that we investigate through this work can be separated into two points:

(i) Passage-Question Relevancy Problem: For a given question, the task is to find a passage in a document in which the answer span exists. Table 1.2 shows an example with longer text and two questions. For Question 1, the answer span (1856) can be found in the first passage. For the Question 2, the second passage contains the answer span (cholera). The task is to find which question is more

related to which passage.

Table 1.2. Question-answer pairs with related text from SQuAD [1].

Nikola Tesla (Serbian Cyrillic: 10 July 1856 – 7 January 1943) was a Serbian American inventor, electrical engineer, mechanical engineer, physicist, and futurist best known for his contributions to the design of the modern alternating current (AC) electricity supply system. In 1873, Tesla returned to his birthtown, Smiljan. Shortly after he arrived, Tesla contracted cholera; he was bedridden for nine months and was near death multiple times.
Question: In what year was Nikola Tesla born? Answer: 1856 Question: What disease did Tesla contract in 1873? Answer: cholera

(ii) Dealing with ASR uncertainty: The task is to find an answer in a passage that is a transcription of a spoken document. The task is harder than QA on textual data, because ASR output contains recognition errors. The goal is to improve the QA performance by integrating ASR uncertainty into the QA system.

1.2. Contributions

In this work,

- We develop an end-to-end QA system and investigate its performance on the newly generated lecture QA datasets.
- We test the developed QA system on spoken documents using ASR transcripts. This direction helps us to understand the effect of ASR errors on the QA performance.
- Transcriptions of spoken archives do not have passage boundaries and this makes machine comprehension on spoken documents difficult. Therefore, we propose a model called *Passage-Question Relevance (PQ-Relevance)* that automatically determines if a passage contains an answer to a given question. In our research, we use short pseudo passages obtained from the transcriptions of spoken document.

- To deal with the uncertainty of ASR transcripts, we use confidence scores from confusion network of the words in textual data. Confusion networks are ASR outputs which give the probabilities of words in a timestamp for a spoken document. Two different approaches for integration were investigated. First, word confidence scores are used as neural network features. Second, they are used to generate novel uncertainty-aware word representations.
- To test the model and proposed extensions, we curated two QA datasets for lecture domain. One of them is in Turkish and the other one is in English. Both of them were generated by annotating the reference transcriptions of video lectures. English lectures contain videos of Signals and Systems course from Electrical and Electronics Engineering Departement and Turkish lectures contain video lectures of several courses from Faculty of Law.

1.3. Thesis Outline

Following the brief introduction and the contributions listed in Chapter 1, this thesis has four chapters.

Chapter 2 gives a brief survey on the previous studies in the field. For QA, the history starts around 1960s. From the first developed systems to the recent neural network models, popular QA systems are explained in this chapter. Recent datasets and proposed challenges are given with the models and the evaluation metrics.

Chapter 3 starts with the details of the end-to-end QA system. The detailed explanations of the neural network model used in the system are given along with the technical aspects and formal definitions. The proposed extensions/models (PQ-relevance, using confusion scores etc.) are declared in the chapter.

Chapter 4 contains the experiments and their results for the proposed models. The comparison of the newly generated datasets with the publicly available one are given after the details of the model implementation. The effects of the proposed models to the QA performance and the challenges that we encoutered through this study are

also explained. This chapter ends with an analysis section in which we try to answer why the models can/can't give the true answers to some questions and how the model weights learn.

Chapter 5 gives conclusion and overall review about the study. We give a discussion about our results comparing with different datasets and models. Chapter 5 concludes the thesis by providing a summary of achieved results and future directions.

2. BACKGROUND AND RELATED WORK

This chapter starts with the history of QA on textual documents followed by explaining the developments on spoken documents and recently developed neural network models.

2.1. Question Answering on Textual Documents

Question answering systems were one of the first developments in NLP domain around 1960s. Early QA systems were knowledge based that used a structured database to answer a question [2,3]. Between 1960 and 1970s, the developed systems were mostly based on hand-written rules with narrow knowledge base and they used a limited vocabulary. Finding the answer was handled by a linguistic part that processes the question. The question, posed by a human, was turned into a logical form which depends on the system. Knowledge based systems can be built with rule-based methods or machine learning methods. The common feature of these systems is that they have a knowledge base that was hand-written by experts of the chosen domain. For example LUNAR [2] was built by NASA to answer questions about the geological analysis of rocks returned by the Apollo moon missions.

QA systems can be classified based on different criteria as application domains, questions, data sources, matching functions, and answers [4]. By application domains, there are two groups of QA systems: open domain and closed domain (a.k.a restricted domain).

Closed domain QA systems find the answer for a question in domain specific knowledge database (medicine, baseball etc.). The types of question generally are predefined and limited. The system uses a semantic parser to map question into a logical form that is acceptable by the structured knowledge base. The repository of closed domain QA can be limited, but they satisfy the need of specialized answers of the experts in restricted domain.

Open-domain QA systems are information retrieval based (IR-based) and handle questions on different topics using more general knowledge base (like web). Some IR-based systems convert the question into a substring of possible answers. For example, the question *"Where is Cappadocia?"* can be reformulated as *"Cappadocia is located in"* [5]. Some systems classify the questions by finding the answer types. The answer types are defined with named entities like `PERSON, LOCATION, NUMERIC, DESCRIPTION etc`. For example, *"Who founded Apple Inc.?"* expects an answer of type `PERSON`. Question classification can be built with the hand-written rules using regular expressions or using supervised machine learning methods, where the features can be the words, part-of-speech (POS) tags of the words, named entities, n-grams etc. After the query is reformulated, it is passed to passage/document retrieval module. QA searches the relevant documents or passages in the web or in the document base. Generally, the model can come up with multiple documents. Instead of returning the documents, QA systems can extract relevant passages from the documents. The final step is to extract the answer span from the relevant passage. Answer extraction can be applied with different methods: feature-based methods (using supervised learning with hand-written rules), n-gram tilling (n-gram mining) and neural answer extraction methods [5]. Neural answer extraction methods assume that there is a semantic similarity between the question and the answer, and the similarity is calculated using continous space representations of the question and the answer.

With the developments in Natural Language Processing (NLP), the processing of the question has been improved with further linguistic analysis. The natural language question is translated into logical form, then it is passed to the database as query. Table 2.1 shows examples of logical forms produced by a semantic parser for given questions in the first column. These forms can be simple relations like birth year or more complex calculus definitions. For example, MASQUE [6] translates a question into a database query and searches the query in the related database.

Using the frequently asked questions as knowledge base, FAQFinder maps a question by matching it with the document through statictical and semantic similarity [7]. When training data is small, classifying questions with hand-written rules is promising

in restricted domains. Quarc [8] is an example of rule-based QA system which classifies the question into different types and finds the answers through lexical and semantic rules.

Table 2.1. Logical form example from semantic parser.

Question	Logical Form
When was Tesla born?	`birth-year (Tesla-?x)`
What is the capital of Turkey?	λx.capital(Turkey,x)
What is the newest publisher article?	`argmax(type.article,publicationDate))`

TREC Evaluation Campaign which is arranged every year since 1999 has an impact on the trend of information retrieval. The TREC campaign provides a dataset which is a collection of documents for generating answers. Participants run their own models with the given data and submit their results to the system. The system judges and evaluates the results. In the first TREC dataset, there were 200 questions and a collection of documents. Through the years, TREC competitions progress with increasing size and increasing complexity of document collections, questions and answer evaluation strategies. Several web based QA systems have been developed after TREC campaign release [9], Mulder [10], Webclopedia [11], AnswerBus [12]. Most of these systems are capable to answer factoid questions. More complex questions require different techniques such as rule matching, keyword matching etc.

At the same time with TREC, reading comprehension (RC) tests were used to develop an evaluation method by [13]. They built a corpus with the children's reading book, a pedagogical book in which a child is given a passage to read and answer questions about it. Their proposed baseline system which was based on pattern matching with bag-of-words was then improved by adding rules to analyse the sentences of the question and the passage [8] and by a logistic regression [14]. More recently, MCTest [15], a multiple choice reading comprehension set was proposed. They curated 500 fictional stories with 2000 questions by crowdsourcing. Although most of the questions require deeper analysis with reasoning and the dataset is quite challenging, there are conspicous results [16], [17].

Table 2.2. Properties of several recent datasets.

Dataset	Source	Size	Description
MCTest [15]	Fictional Stories	2640	Multiple-Choice QA Dataset
SQuAD [1, 18]	Wikipedia	100K	Span-based answer selection
CNN/Daily [19]	CNN and Daily Mail News	1.4M	Cloze style questions
bAbI [20]	Synthetic	10K(×20)	A set of 20 QA Tasks

A reading comprehension dataset Stanford Question Answering Dataset (SQuAD), was recently proposed by [1]. SQuAD consists of passages from Wikipedia followed by questions asked by humans. Different from MCTest, SQuAD does not provide a list of answers. Instead, the system must find a substring from the passage as a candidate answer. The dataset contains 107,785 question-answer pairs on 536 articles. They provide a website to test the ability of different models submitted by researchers. The original SQuAD (SQuAD v1.1) dataset was extended around 50000 unanswerable questions and names SQuAD version 2.0 by [18]. Newly added questions do not have any answers in the given passage. The systems developed for SQuAD v2.0 also need to determine unanswerable questions in addition to returning answers to answerable questions in the dataset.

In addition to SQuAD, there are other open datasets proposed for QA task. CNN/Daily News dataset [19] contains around one million training samples obtained by pairing CNN and Daily Mail news articles with their summarized bulletpoints. BAbI [20] is a dataset for testing a QA model's ability to understand the text. The dataset contains a set of 20 tasks. Each task has training and test sets. The sets were generated by a simple text adventure game in which an agent moves around and interacts with objects. A question-answer pair was generated according to the agent's action.

Table 2.2 gives the properties of several recent datasets. The size of the datasets is the number of the question-answer pairs. BAbI contains 10K question-answer pairs for each 20 tasks. Note that among the given datasets, only bAbI is synthetically generated.

2.2. Question Answering on Spoken Documents

The achievements of the state-of-the-art QA models demonstrate that machine has already acquired complex reasoning ability in machine comprehension. Whereas question answering on spoken content is a more difficult task compared to question answering on text. In these systems, the goal is to find an answer to a textual or spoken question from a spoken document.

In 2007, a pilot evaluation system for spoken question answering (SQA) was proposed under CLEF umbrella [21] . There were two different scenarios: lecture and meeting scenarios. For both scenarios, different datasets were proposed along with questions and answers on text. The questions were factoid, which expects a named entity as an answer. QA on Speech Transcriptions (QAst) aimed to encourage research groups which were interested in the topic of question answering on spoken documents. It lasted for three years (2007-2009). Following this campaign, Sibyl a fully functional QA system that works on spoken content was proposed [22].

In 2014, another spoken QA system was proposed by [23]. They used an IR technique based on conditional random fields and tree structures.

Machine comprehension on spoken content task was firstly investigated in [24]. They used TOEFL listening comprehension test as the machine comprehension task. TOEFL is an English examination which tests the knowledge and skills of academic English for English learners whose native languages is not English. In the TOEFL examination, the subject listens an audio and answers questions about the audio. The questions have multiple-choices. The audio of the short stories are first transcribed into text by ASR and the QA system selects an answer from 4 given choices. They

proposed a framework called Attention-based Multi-hop Recurrent Neural Network (AMRNN). An improved framework of AMRNN was proposed with better results in TOEFL dataset by [25].

One of the reasons why QA on spoken documents is less investigated is that the publicly available datasets are limited and the data collection is expensive. Spoken SQuAD [26] was proposed to meet the need of available spoken document QA dataset. Spoken SQuAD was generated from SQuAD by using Google Text-to-Speech system. They examined that ASR errors have a serious impact on the QA system performance and proposed using sub-words to mitigate the effects of ASR errors. Even Spoken SQuAD is a large scale dataset, it is artificially generated.

Recently, a large scale dataset called Open-Domain Spoken Question Answering Dataset (ODSQA) was proposed [27]. The reference texts were from an open domain traditional Chinese machine reading comprehension (MRC) dataset. They recruited 20 speakers who use Chinese as their native language. The textual documents and questions were collected for ODSQA dataset. By testing several competitive models developed for SQuAD on this dataset, they concluded that ASR errors have a serious impact on the performance.

Beside using text-to-speech systems to generate spoken documents dataset, using online available videos were proposed in [28–31]. VideoQA [31] was first known system which utilizes news video collection. VideoQA returns short precise news video summaries as answers. Another available videos are from online lecture videos which can be used to generate datasets as news videos. Even the lecture videos and news videos are the same type of multimedia data, they differ in genre. The scenes in video lectures do not change and are not very informative. Also lecture videos from online learning platforms are generally provided with a presentation or a transcript as a way to help learners engage in the learning process. Two approaches, pattern based and NLP approach were proposed for QA using lecture videos in [29].

2.3. Question Answering in Turkish

Developing QA systems is highly challenging especially when dealing with a morpholocially rich language such as Turkish. One of the first approach was using a formal analysis method in the Natural Language Processing discipline for Turkish [32]. The question received from the user was shown in the vector space model and compared with the answers in the database and the similarity ratio was calculated according to the cosine similarity. The first successful Turkish QA system was presented in [33]. They introduced a novel query expansion approach and the creation of named entity tagged answer patterns. To overcome the language difficulties in Turkish QA system, they [34] preprocessed the questions by performing morphological analysis and dependecy parsing. For the closed QA system, they proposed a Hidden Markov Model (HMM) based model to classify the questions.

2.4. Deep Learning Models

The rise of the neural network based methods and the accessibility of publicly available datasets help the performance of the developed models get close to the human performance. The first attempt of using neural network based methods in reading comprehension was performed by [19]. They proposed two novel methods: one to create around one million training samples by pairing CNN and Daily Mail news articles with their summarized bulletpoints. and one to demonstrate the efficacy of their dataset. Their proposed model, AttentiveReader, an attention-based LSTM, was pioneer of the modern neural network based methods.

With high quality and reliable automatic evaluation, SQuAD has attracted tremendous interest in NLP community and become the benchmark in this field. Since its release, SQuAD has inspired very successful models [35–39]. Figure 2.1 shows recent results taken from SQuAD evaluation web site. The left graph shows the results for version 1.1 and the right one is for version 2.0. X-axis represents the rank of the submitted models and y-axis represents the score of these models. Human performance has been shown with horizontal lines for Exact match (continous/blue) and F1 score

(dashed/orange). Exact match (EM) shows the rate of the correctly predicted answers over all the samples. F1 score calculates the correctness of the model in a word-by-word manner. Further explanation is given about the metrics in Section 2.5. For SQuAD version 1.1, the human performance (EM:82.304,F1:91.221) was beaten by the best model XLNet [40] (F1:95.080,EM:89.898). For SQuAD version 2.0 the best result (F1:89.474,EM:87.147) is very close to the human performance (F1:89.452,EM:86.831).

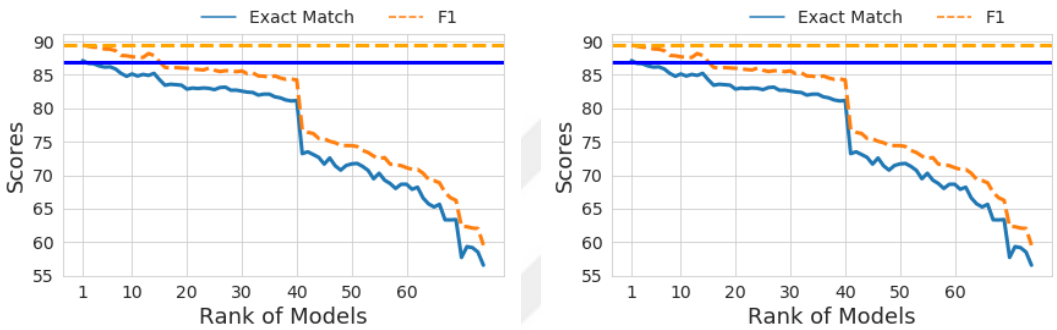

Figure 2.1. SQuAD evaluation results for version 1.1 (left) and version 2.0 (right).

Most of the top-scored models in SQuAD are end-to-end neural networks. The common features of the models are the representation modules for question and passage (embeddings), the matching module (or interaction module) and the output module. These models do not use linguistics rules. Most state-of-the-art neural question answering models have common layers in their architecture. Figure 2.2 shows a simple architecture of these models.

In Figure 2.2, there are 4 layers: input, embedding, interaction and output. The model takes the passage and the question in natural language and translates them into embedding vectors at the input layer and embedding layer. Recurrent neural networks (RNN) are commonly used in the embedding layer. The embedding vectors can be initialized with pretrained vectors using Glove [41] or Word2Vec [42]. These distributed vectors are numerical representations of word features that are learned over multiple documents. The representations of the passage and the question can be combined at interaction layer. This is an important step that yields question-aware passage representation. Then start and the end positions of the answer are predicted

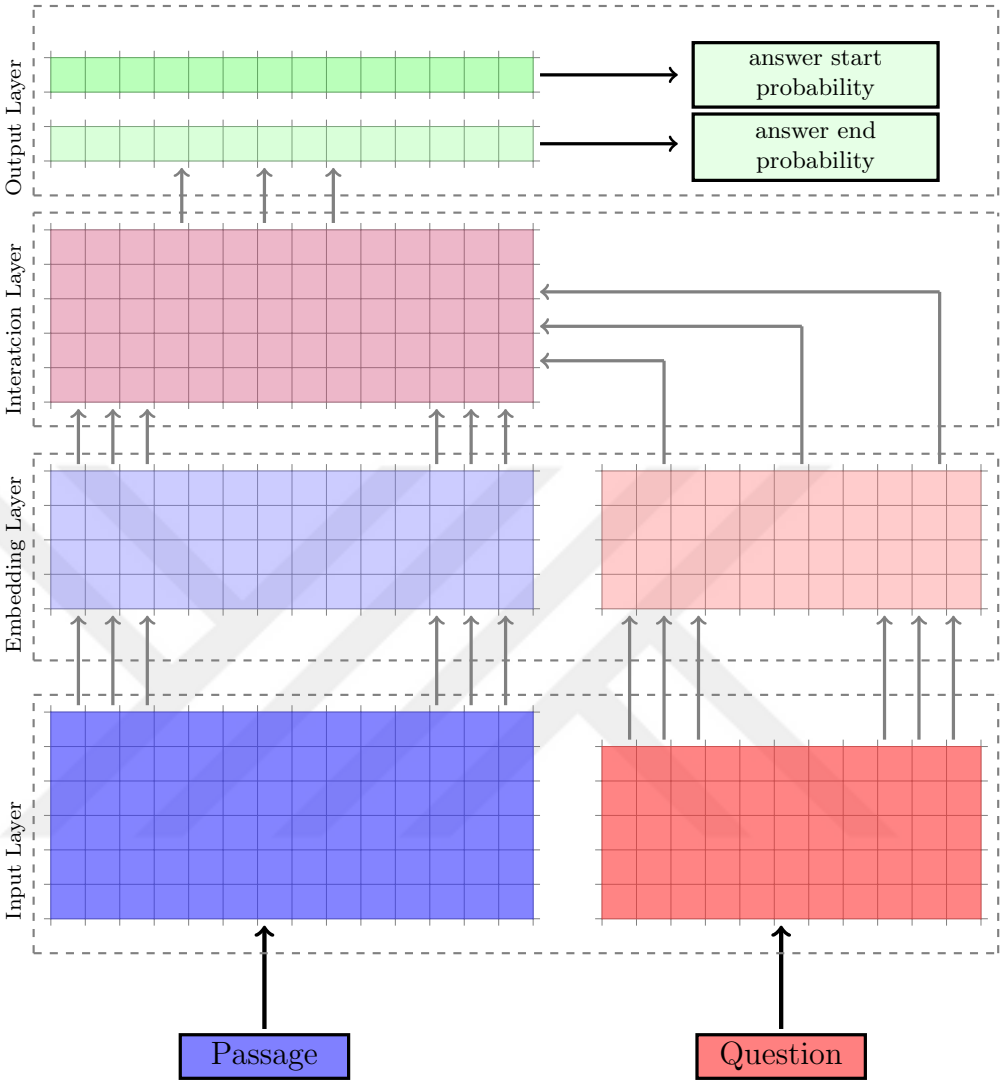

Figure 2.2. General architecture of neural QA systems.

at the output layer.

A proposed neural network based model, R-Net [38], uses a gated attention-based recurrent network after the embedding layer. Then a self-matching layer is added before the output layer to obtain more evidence from the passage words. Another model, Bi-directional Attention Flow (BIDAF) model [36], calculates the attention in two directions: from passage to question and from question to passage. They claimed that their attention vector does not summarize the query and the passage into one attention vector, instead at each time step, the attention vectors are allowed to flow to the subsequent layers along with the embeddings from previous layers.

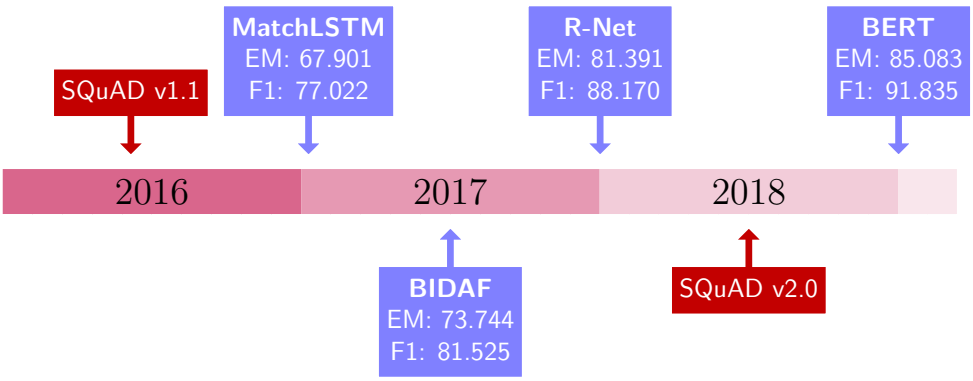

Figure 2.3. Timeline for several models tested with SQuAD.

Figure 2.3 shows a timeline for several models together with their scores. The represented models in Figure 2.3 are chosen for their successful results. SQuAD website gives the rankings for 85 models for version 1.1 and 61 models for version 2.0. Most of the systems in the top rankings are actually based on the models given in Figure 2.3.

As of 2019, the best result with SQuAD version 1.1 is with Bidirectional Encoder Representations from Transformers (BERT) which is a language representation proposed by [35]. It uses transformers, a version of attentions, to learn contextual relations between words in a text by applying bidirectional training. BERT can be used in variety of NLP problems by adding an output layer depending on the task. For SQuAD, an answer pointer layer is enough to use pretrained BERT model.

2.5. Evaluation

For span-prediction taks in QA, the predicted answer string is compared with the ground truth. To compare two spans, F1 score and Exact Match (EM) are used. While calculating F1 score, the prediction and the ground truth are treated as bag-of-words.

- F1 Score is a measure which calculates the balance between precision and recall. It is calculated word-level between the prediction and the ground truth.

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.1}$$

Figure 2.4 shows a confusion matrix with definitions used in classification tasks. The terms positive and negative refer to the classifier's prediction and the terms true and false refer to whether that prediction corresponds to the evaluation.

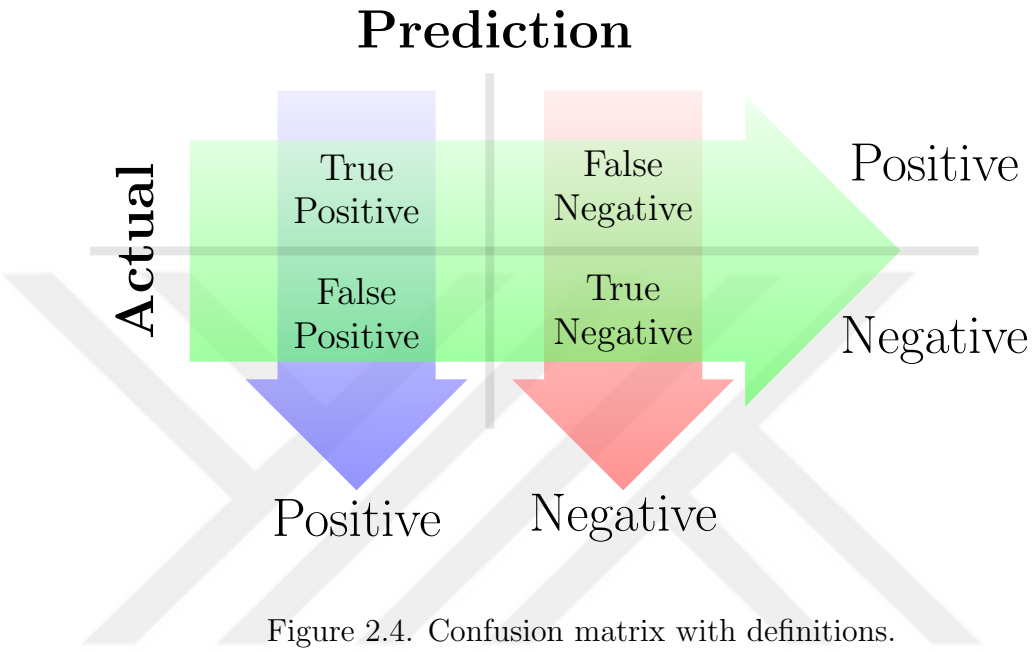

Figure 2.4. Confusion matrix with definitions.

(i) Precision *is the ratio of the number of correctly labeled responses to the total labeled responses.* [5]. In QA, precision is the fraction of the correctly predicted words in the predicted answer span. It is calculated as:

$$\begin{aligned} \text{Precision} &= \frac{\text{Number of correctly predicted words}}{\text{Number of words in prediction}} \\ &= \frac{TP}{TP + FP} \end{aligned} \tag{2.2}$$

(ii) Recall *is the ratio of the number of correctly labeled responses to the total that should have been labeled* [5]. In QA, recall is the fraction of the correctly predicted words in the ground truth. It is calculated as:

$$\begin{aligned} \textit{Recall} &= \frac{\text{Number of correctly predicted words}}{\text{Number of words in ground truth}} \\ &= \frac{TP}{TP + FN} \end{aligned} \tag{2.3}$$

The number of words that exists both in predicted span and ground truth are calculated using bag-of-words. First, unique words in both spans are extracted without any order information, the intersection of these two sets gives the number of words in both spans.

- Exact Match is a binary evaluation that gives 1.0 if two spans are equal and gives 0.0 otherwise. The order of the words in the ground truth and the prediction is taken into account in EM calculation.
- Accuracy describes the closeness of a measurement to the true value. In classification tasks, it gives the model's performance over the whole dataset. It is calculcated as :

$$Accuracy = \frac{\text{Number of sample correctly classified}}{\text{Number of samples in dataset}} \tag{2.4}$$

- Word Error Rate (WER) is a metric used for ASR systems. The number of errors between the predicted text and the reference text is calculated with edit distance algorithm. It is calculated as:

$$WER = \frac{\#D + \#S + \#I}{\text{Mumber of words in reference text}} \tag{2.5}$$

where $\#D, \#S$ and $\#I$ are respectively the minimum number of deletion, substitution and insertion operations required to transform one text into another [43].

3. METHODOLOGY

The aim of this section is to give the details of QA model and proposed extensions. The section starts with the formulation of the QA task and continues with the QA model definition. After a brief explanation of the model framework, basic components of the model are explained. Then, the proposed approaches and their structures are given.

3.1. Problem Formulation

The span-based answer style QA task is a supervised learning problem in which for a given set of training samples as $(p^i, q^i, a^i)_{i=1}^N$ the model learns a function that takes the passage p and the question q, and returns an answer a:

$$f : (p, q) \rightarrow a \tag{3.1}$$

where N is the number of training samples. Each sample p^i and q^i is a set of words denoted as $p^i = \{p_1^i, p_2^i, ...p_P^i\}$ and $q^i = \{q_1^i, q_2^i, ..., q_Q^i\}$ where P and Q denote the length of the passage and the question.

The answer is a span in the passage. It is represented with two numbers which are the start point and the end point of the answer in the passage. Formally, the answer a can be represented as (a_{start}, a_{end}).

3.2. Match LSTM

MatchLSTM is an end-to-end neural network model proposed for Natural Language Inference (NLI) [44]. NLI is the task of determining the relation between two sentences. One sentence is called the premise and the other one is called the hypothesis. The goal is to predict a label that indicates the relationship between two sentences. The label is either entailement, neutral or contradiction. MatchLSTM for NLI takes two

sentences and generates word-by-word attention between them with Long Short-Term Memory (LSTM) neural network. The model matches the premise and the hypothesis by generating an attention vector at each position. Important matching results are remembered while unnecessary ones will be forgotten [44].

Based on the matching mechanism of MatchLSTM, they [39] proposed a model for QA by adding a Pointer Network [45]. Different from NLI task, in QA the model predicts an answer sequence from a given passage. The model takes a passage and a question as input and generates an answer which is a word sequence in the passage.

Figure 4.8 shows the general framework of MatchLSTM model. There are three layers of the model: Preprocessing, MatchLSTM and AnswerPointer. All the layers have LSTM neural network as the main component. The passage and the question are processed at the preprocessing layer with LSTM. Resulting representations are given to the next layer of the model: MatchLSTM layer. In this layer, matching between the passage and the question is calculated with attentional LSTM neural network. The output of this layer ($\mathbf{H^r}$) is given to the answer pointer to find the answer boundary in the passage.

We start with the description of basic components (LSTM, Pointer Network) of MatchLSTM to have better understanding of the model. Then we explain MatchLSTM in detail.

3.2.1. Long Short-Term Memory (LSTM)

LSTM is a recurrent network model proposed by [46]. Recurrent networks are artificial neural networks that can process a temporal sequence with their internal memory. They have feedback connections between their recurrent layers which allow them to store information over time.

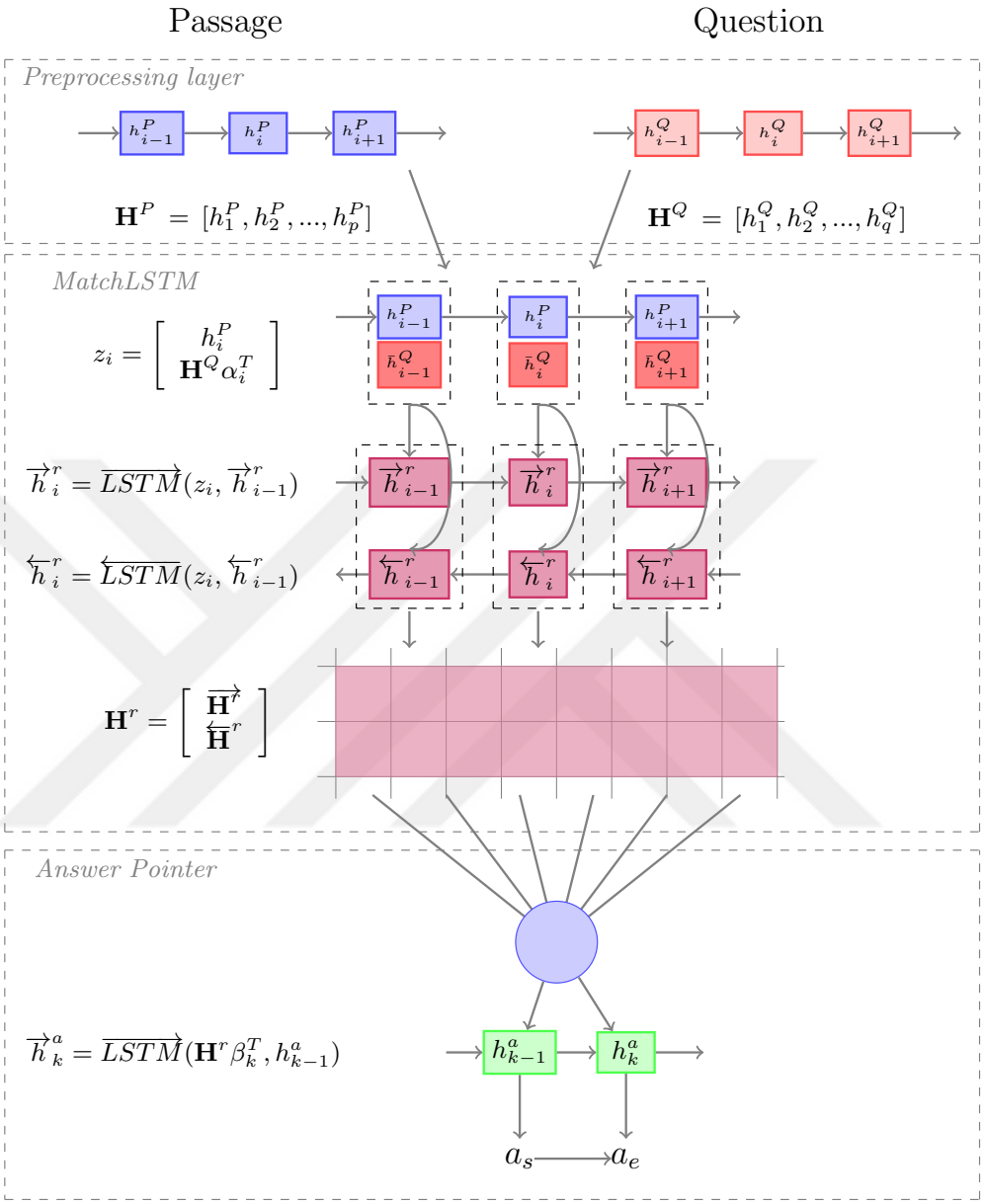

Figure 3.1. General framework of MatchLSTM model.

In traditional neural networks, it is assumed that all inputs are independent from each other. The input samples are fed into the network and processed in the forward direction to the output layer. They are called *feedforward neural network (FNN)* because of this data flow. FNNs don't have any loop mechanism in their hidden layers. Figure 3.2 gives a simple diagram for FNN. There are 3 input, 4 hidden and 1 output neurons of the network. The goal of FNN is to approximate a function

that can map input x to an output y. FNNs are mainly used for supervised problems like classification and regression.

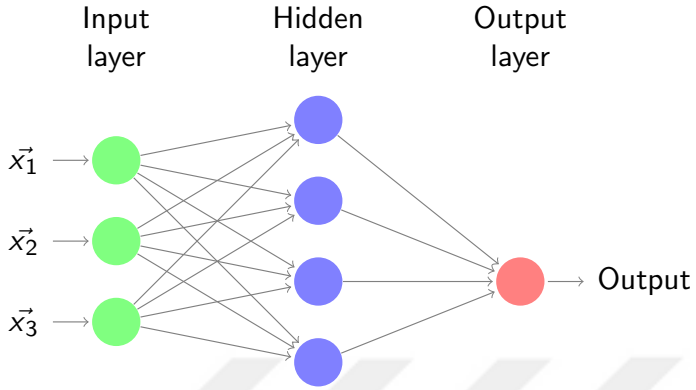

Figure 3.2. Feed forward neural network.

FNNs don't have any loop connections, so they can only remember the formative moments of training. To overcome this problem, recurrent networks were developed [47]. Recurrent neural networks (RNNs) can process information in a loop connecting their internal nodes. Figure 3.3 shows a diagram for RNN. Different from FNN, RNN diagram has self loops in its hidden layers.

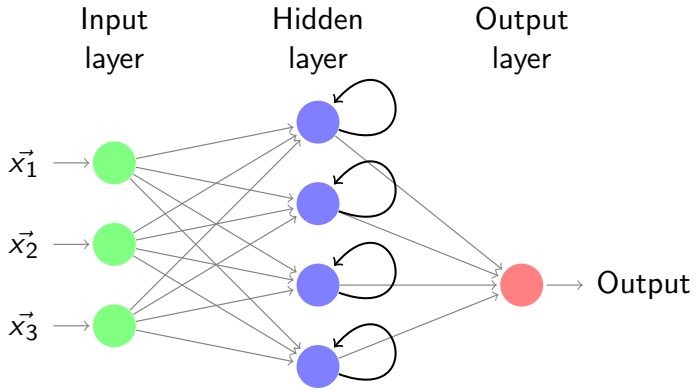

Figure 3.3. Recurrent neural network.

Figure 3.4 shows two versions of RNN. Left image is called the folded version of RNN and right one is called the unfolded version of RNN. RNN can be considered as multiple copies of the same network. x_t represents the input and h_t represents the hidden state at time t. The decision of a time step is affected by the decision from the previous time step, so RNNs take two inputs: the current data and the past data. They can tweak their parameters using these two inputs.In the past years, RNNs have been

applied to variety of problems such as speech recognition, image processing, machine translation and resulted in successful performances.

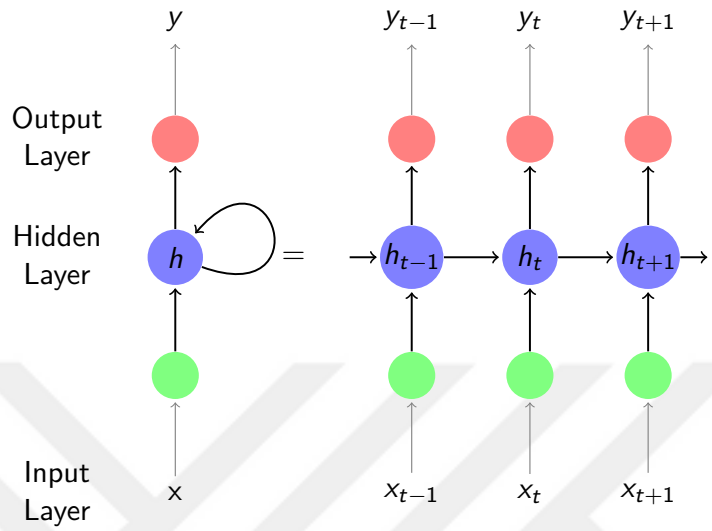

Figure 3.4. RNN diagram with unfolded and folded versions.

RNNs seek to make a connection between the output and the input through the time steps. The layers and time steps of RNNs relate to each other through multiplication. Because the gradients of the error pass through many stages of multiplication while flowing back in network to the inputs, the derivatives sometimes tend to either blow up or vanish. When the gradients become very high, called *exploding gradient problem*, the network gives very high importance to weights without any reason. This situation can be handled by truncating gradients. Whereas *vanishing gradient problem*, when derivatives are very small, is a harder to solve than exploding grandients. LSTM networks was proposed as a solution to vanishing gradient problem of RNNs [46].

LSTM is an extended version of RNNs with their extended memory to deal with the long term dependencies. LSTM can learn what to write to the memory and what to delete from the memory. The information flow is controlled by the gates which are basically the weights that the model learns in time. LSTM has three gates: input gate, forget gate and output gate.

Like all form of recurrent networks, LSTM has a chain structure of repeating cells. Figure 3.5 shows an LSTM cell. For a given sequence data noted as $x_1, x_2, ..x_t$

where x_i denotes the ith time sample and $x_i \in \mathbb{R}^d$, LSTM calculates a function with previous hidden vectors and current time step sample:

$$h_t, c_t = F(h_{t-1}, c_{t-1}, x_t) \tag{3.2}$$

where F is the cell function.

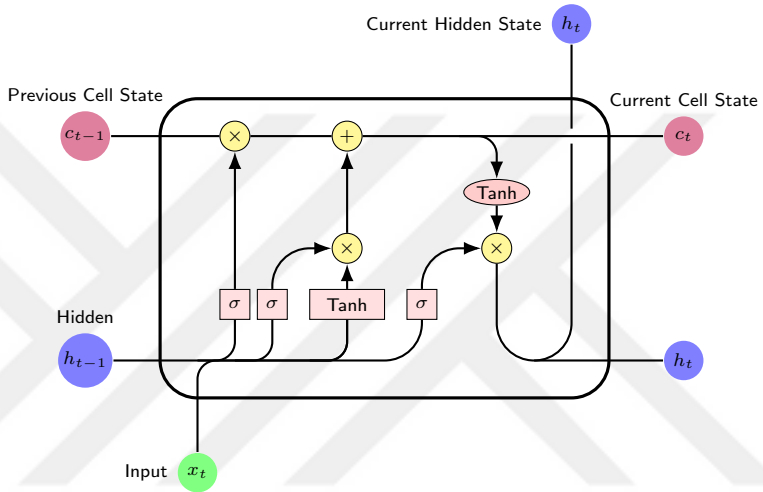

Figure 3.5. LSTM cell.

In LSTM, the first step is to decide what information should be kept or thrown away. The decision is calculated in the forget gate with Equation 3.3. It takes the previous hidden state h_{t-1} and current input x_t and returns a number between 0 and 1 by sigmoid function (σ). If the number is close to 0, it means that the model forgets the previous hidden state. Otherwise, the gate keeps the previous hidden state information.

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \tag{3.3}$$

The second step is to decide the update on the state which is calculated in the input gate with Equation 3.4. The input gate takes previous hidden state and current input and first returns a number i_t between 1 and 0. Then $\tilde{C}_t$ that represents the update on

the state is calculated with hyperbolic tangent ($tanh$) function.

$$\begin{aligned} i_t =& \sigma(W_i.[h_{t-1}, x_t] + b_i) \\ \tilde{C}_t =& tanh(W_C.[h_{t-1}, x_t] + b_C) \end{aligned} \tag{3.4}$$

f_t,i_t and $\tilde{C}_t$ are used to update previous hidden state C_{t-1} as in Equation 3.5.

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \tag{3.5}$$

Finally, the network decides what to output and what the next state should be in the output gate.

$$\begin{aligned} o_t =& \sigma(W_o.[h_{t-1}, x_t] + b_o) \\ h_t =& o_t \times tanh(C_t) \end{aligned} \tag{3.6}$$

In all equations, W matrices are weights and b vectors are bias terms and operation $[,]$ indicates the concatenation between given vectors.

There are different variants of LSTM depending on its equations [48–50]. Gated Recurrent Unit (GRU) is a variant of LSTM that combines forget and input gates into *update* gate [49].

3.2.2. Pointer Network

Pointer Network, proposed by [45], is a neural network architecture that learns the conditional probability of an output sequence whose tokens are related to the positions in the input. The classical encoder-decoder (sequence-to-sequence) models are able to generate sequences by mapping the input to an RNN (encoder) and the output into another RNN (decoder). Encoder-decoder methods usually encode the input sequence into a compressed representation and then decode that representation into the output sequence. It is possible that there is some distortion after multiple time steps while

compressing the information into a smaller representation.

To overcome these problems, Pointer Network uses a neural attention mechanism as a pointer to select a member of the input sequence as the output. Since the output sequence consists of indices (positions) of the input sequence, the attention, which naturally satisfies the restriction of fixed length, can be treated as the confidence of each position being the predicted position.

Given a training pair (X, Y), the encoder-decoder model learns the conditional probability $p(Y|X;\theta)$ with chain rule:

$$p(Y|X;\theta) = \prod_{i=1}^{m} p(Y_i|Y_1, ..., Y_{i-1}, X;\theta) \tag{3.7}$$

where θ is the model parameters, X and Y are sequences ($X = X_1, X_2, ..., X_N$ and $Y = Y_1, Y_2..., Y_m$). For a sequence-to-sequence model with an attention mechanism, the attention vector is calculated at time step i as :

$$\begin{aligned} u_j^i &= v^T tanh(W_1 e_j + W_2 d_i) \\ a_j^i &= softmax(u_j^i) \\ d_i^l &= \sum_{j=1}^{n} a_j^i e_j \end{aligned} \tag{3.8}$$

where v, W_1, W_2 are learnable parameters and e_j and d_i are encoder-decoder hidden states, d_i^l is the attention vector. Lastly, d_i^l is concatenated to decoder states d_i to make prediction from the input. The encoder-decoder model uses a softmax distribution over the fixed size of output dictionary to calculate the attention vector. Pointer Network removes the attention vector calculation to avoid the softmax over the output dictionary and normalizes the vector u_i to be an output distribution over the dictionary of input as shown in Equation 3.9.

$$u_j^i = v^T tanh(W_1 e_j + W_2 d_i)$$
$$p(Y_i|Y_1, ..., Y_{i-1}, X) = softmax(u^i) \tag{3.9}$$

Figure 3.6 shows an example of Pointer Network. The input pairs (x_i, y_i) are marked as green nodes, the hidden nodes h_i are marked as blue nodes. At each time step, the model calculates a distribution over the given input sequence. From this distribution, a node that has the maximum probability is selected as output of this time step and it is returned to the network as input of the next time step. The process stops when an end token is produced as an output with a maximum probability.

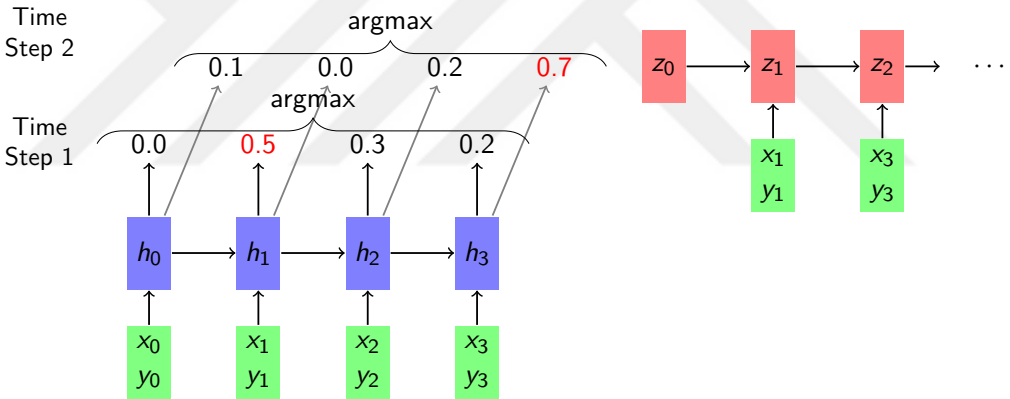

Figure 3.6. Pointer network example.

3.2.3. MatchLSTM

MatchLSTM proposed for QA is an end-to-end neural network which is based on LSTMs. For a given passage and a question, the model generates an answer that is a subsequence from the passage. The passage is denoted as matrix $\mathbf{P} \in \mathbb{R}^{d \times P}$ where P is the length of the passage and d is the dimensionality of embedding dimension. The question is denoted as matrix $\mathbf{Q} \in \mathbb{R}^{d \times Q}$ where Q is the length of the question. Figure 3.1 gives the general framework of the model: preprocessing, matchLSTM and answer pointer layers.

- Preprocess Layer takes the matrix $\mathbf{P}$ and $\mathbf{Q}$ and calculates the hidden representations ($\mathbf{H^p}$,$\mathbf{H^q}$) with uni-directional LSTM.

$$\mathbf{H}^p = \overrightarrow{LSTM}(\mathbf{P}) \quad \mathbf{H}^q = \overrightarrow{LSTM}(\mathbf{Q}) \tag{3.10}$$

where $\mathbf{H^p} \in \mathbb{R}^{l \times P}$ and $\mathbf{H^q} \in \mathbb{R}^{l \times Q}$ and l denotes the dimension of hidden vectors.

- MatchLSTM layer processes the input word-by-word and calculates an attention vector α for each word of the passage. The vector α_i shows the similarity between i^{th} word of the passage and all the words in the question, therefore α_i is a vector of length Q. The attention vector for i^{th} token of the passage, is calculated as follows:

$$\begin{aligned} \overrightarrow{\mathbf{G}}_i &= \tanh(\mathbf{W^q H^q} + (\mathbf{W^p h_i^p} + \mathbf{W^r}\overrightarrow{h}_{\mathbf{i-1}}^{\mathbf{r}} + \mathbf{b^p}) \otimes e_Q) \\ \overrightarrow{\alpha}_i &= \text{softmax}(\mathbf{w^T}\overrightarrow{G}_{\mathbf{i}} + b \otimes e_Q) \end{aligned} \tag{3.11}$$

where $\mathbf{W^q}, \mathbf{W^p}, \mathbf{W^r} \in \mathbb{R}^{\mathbf{l \times l}}, \mathbf{b^p}, \mathbf{w} \in \mathbb{R}, b$ are the parameters to be learned and $\otimes e_Q$ repeats the vector or scalar for Q times to match the matrix or vector dimensions for summation. $\overrightarrow{h}_{i-1}$ is the one dimensional hidden vector for time step $i-1$.

For i^{th} token of the passage and j^{th} token of the question, the similarit is represented as $\alpha_{i,j}$. The hidden representation of token i of the passage is concatenated with the weighted vector representation of the question ($\mathbf{H}^q\overrightarrow{\alpha}_i^T$), then it is fed to a one-directional LSTM to calculate the current hidden vector $\overrightarrow{\mathbf{h}}_i^r$ as in Equation 3.12.

$$\begin{aligned} \overrightarrow{z}_i &= \begin{bmatrix} \mathbf{h}_i^p \\ \mathbf{H}^q\overrightarrow{\alpha}_i^T \end{bmatrix} \\ \overrightarrow{\mathbf{h}}_i^r &= \overrightarrow{LSTM}(\overrightarrow{z}_i, \overrightarrow{\mathbf{h}}_{i-1}^r) \end{aligned} \tag{3.12}$$

Note that $\overrightarrow{\mathbf{H}}^r \in \mathbb{R}^{l \times P}$ contains the hidden vectors $[\overrightarrow{\mathbf{h}}^r_1, .., \overrightarrow{\mathbf{h}}^r_P]$ as columns.

The same calculations are applied in reverse direction to generate the representations of the context in both directions. Equation 3.13 shows all the equations in reverse order.

$$
\begin{aligned}
\overleftarrow{\mathbf{G}}_i &= \tanh(\mathbf{W^q H^q} + (\mathbf{W^p h_i^p} + \mathbf{W^r} \overleftarrow{h}^{\mathbf{r}}_{\mathbf{i-1}} + \mathbf{b^p}) \otimes e_Q) \\
\overleftarrow{\alpha}_i &= \text{softmax}(\mathbf{w^T} \overleftarrow{G}_{\mathbf{i}} + b \otimes e_Q) \\
\overleftarrow{z}_i &= \begin{bmatrix} \mathbf{h}_i^p \\ \mathbf{H}^q \overleftarrow{\alpha}_i^T \end{bmatrix} \\
\overleftarrow{\mathbf{h}}^r_i &= \overleftarrow{LSTM}(\overleftarrow{z}_i, \overleftarrow{\mathbf{h}}^r_{i-1})
\end{aligned}
\tag{3.13}
$$

Note that $\overleftarrow{\mathbf{H}}^r \in \mathbb{R}^{l \times P}$ contains the hidden vectors $[\overleftarrow{\mathbf{h}}^r_1, .., \overleftarrow{\mathbf{h}}^r_P]$ as columns. The concatenation of two matrices $\overleftarrow{\mathbf{H}}^r$ and $\overrightarrow{\mathbf{H}}^r$ gives the representation: $\mathbf{H}^r = \begin{bmatrix} \overrightarrow{\mathbf{H}}^r \\ \overleftarrow{\mathbf{H}}^r \end{bmatrix}$

The matrix $\mathbf{H}^r \in \mathbb{R}^{2l \times P}$ contains the question-aware represention of the passage. The matrix is the output of the matchLSTM layer.

- Answer Pointer layer which is based on Pointer Network [45], takes the inner representation $\mathbf{H}^r$ as input and generates the start and end tokens as the answer. Two models are proposed models in [39]: *boundary* and *sequence* models. In boundary model, the generated answer is a consecutive subsequence of the passage, therefore the model only finds the start and the end tokens of the answer in the passage. In sequence model, the generated answer may not be consecutive words in the original passage. In this work boundary model is used, because the answers are the consecutive subsequences of the given passages in our dataset. The boundary model outputs a sequence of integers which represents the positions of the tokens in the passage. Note that the answer $a = (a_0, a_1)$ contains two numbers: a_0 is the start token's position and a_1 is the end token's position in the passage. Answer Pointer layer generates answer a in a sequential manner. Both a_0 and a_1 can take a value between 1 and P, the length of the passage.

  First an attention vector $\beta_k \in \mathbb{R}^P$ is calculated where $k = 0, 1$. β_{kj} shows the

probability of being start ($k = 0$) or end ($k = 1$) of the jth token of the passage. Equation 3.14 shows the calculation of the attention vector β_k.

$$\begin{aligned}
\mathbf{F}_k =& \tanh(\mathbf{V}\mathbf{H^r} + (\mathbf{W^a}\mathbf{h^a_{k-1}} + \mathbf{b^a}) \otimes e_P) \\
\beta_k =& \text{softmax}(\mathbf{v^T}\mathbf{F_k} + c \otimes e_P) \\
\mathbf{h^a_k} =& \overrightarrow{LSTM}(\mathbf{H}^r\beta_k^T, \mathbf{h^a_{k-1}})
\end{aligned} \tag{3.14}$$

where $\mathbf{V}, \mathbf{W}, \mathbf{v}, \mathbf{b^a}, c$ are the parameters and $\mathbf{h^a_k}$ is the hidden state. The probability of an answer sequence is calculated as follows:

$$\begin{aligned}
p(a_k = j|\mathbf{H^r}) =& \beta_{jk} \\
p(a|\mathbf{H}^r) =& p(a_0|\mathbf{H^r})p(a_1|a_0, \mathbf{H^r})
\end{aligned} \tag{3.15}$$

The model is trained to minimize the negative log likelihood based on training examples $P_n, Q_n, a_n{}_{n=1}^{N}$ where N is the number of examples:

$$-\sum_{n=1}^{N} \log p(a_n|P_n, Q_n) \tag{3.16}$$

3.3. Question-Answering System Extensions

Previously explained MatchLSTM model was developed for machine reading comprehension task for text documents where short passages for each document and questions for each passage are explicitly known. However in spoken documents, there are no explicitly defined passages. The model either can take the ASR transcript of the whole document as a single passage or try to find more relevant parts in the document . To adapt the MatchLSTM model into our spoken QA problem, we propose a passage-question relevance scoring model in Section 3.3.1. To improve the QA performance on erroneous transcripts of spoken documents we propose another model which integrates the output of ASR system into the QA model in Section 3.3.2.

3.3.1. Passage-Question Relevance Scoring

In machine comprehension applications on spoken documents, the passage boundaries are not presented to the model, so that the model must find the most relevant short passage for a given question from ASR transcription. The relevant passage should be the one that matches the question more than other passages. It is also possible that a question may not have an answer in a passage. This questions are named as *unanswerable questions*, presented in SQuAD version 2.0 [18]. One approach is to treat the unanswerable questions with speacial answer spans. For example, one of the best scored model on SQuAD version 2.0 BERT [35] uses $[CLS]$ token as start and end of an unanswerable question. While predicting, they compare two probabilities: having a null answer span and having a non-null answer span with a treshold. Another approach is to define an answer verifier on the passage and the question by matching them. The answer verifier model can decide whether a passage contains an answer to a question. The answer verifier and the answer pointer are trained separately with their own parameters.

Our proposed approach, which is close to an answer verifier, is to solve the problem on the absence of the passage boundaries. We assume that the questions in the dataset has at least one answer, so that they are related to one of the passages. This passage is called the relevant passage to the question and should be the closest one to the question in terms of similarity. This similarity can be matching words, matching stems or matching vector representations. We use inner representations from MatchLSTM model to calculate the matching between a passage and a question.

In our approach, the passages and questions are processed in the input layer of MatchLSTM and question-aware passage representations are generated from Match LSTM layer. Each representation carries the relevance between a question and a passage. A simple one hidden layer neural network is used to obtain a score between the passages and the question using the question-aware passage representations.

Question-aware passage representation $\mathbf{H^r}$ is obtained by concatenating two matrices $\overleftarrow{\mathbf{H}}^r$ and $\overrightarrow{\mathbf{H}}^r$, so $\mathbf{H}^r = \begin{bmatrix} \overrightarrow{\mathbf{H}}^r \\ \overleftarrow{\mathbf{H}}^r \end{bmatrix}$. The last and the first hidden state vectors are extracted from forward and backward representations ($\overrightarrow{\mathbf{h}}^r_P$, $\overleftarrow{\mathbf{h}}^r_P$) respectively and concatenated to form the input vector $\mathbf{h_{rel}} = \begin{bmatrix} \overrightarrow{\mathbf{h}}^r_P \\ \overleftarrow{\mathbf{h}}^r_P \end{bmatrix}$ for the one layer neural network model.

For a passage and a question, the neural network model calculates a score as shown below:

$$\begin{aligned} \mathbf{D} =& \tanh(\mathbf{W^h_{rel}}\mathbf{h_{rel}} + \mathbf{b^h_{rel}}) \\ \theta =& \text{sigmoid}(\mathbf{W^o_{rel}}\mathbf{D} + b^o_{rel}) \end{aligned} \tag{3.17}$$

where $\mathbf{W^h_{rel}}, \mathbf{b^h_{rel}}, \mathbf{W^o_{rel}}, b^o_{rel}$ are the parameters to be learned.

Since the task is a binary classification problem, the network is trained using positive and negative passage-question pairs to optimize binary cross entropy and to predict whether the given passage contains the answer to the given question. *Positive* question means that the question has an answer for given passage, and *negative* means otherwise. The scores from the network are used to select a single passage (with the maximum score) for each question. For a question, the model outputs scores over all paragraphs of a document. After choosing the best scored passage for a question, the pairs are given to MatchLSTM QA model to predict the answer span.

3.3.2. Integrating ASR Uncertainty into Question Answering

The aim of ASR is to generate a transcription for a spoken content. It is the main component in most spoken document retrieval systems. *ASR problem is stated as finding the most probable word string among all possible word strings when the input speech is given* [43]. An ASR system can output representations such as 1-best hypothesis, a word lattice and a confusion network. *1-best hytpothesis* is the most

probable word sequence from ASR for a given spoken document. 1-best hypothesis may not always be the same with the spoken content due to the recognition errors. Another output of ASR system is a *word lattice* which is a directed graph that represents possible word sequences for spoken content. A *confusion network* is a compact representation of the lattice where all the word hypothesis are in order [51].

Confusion network (CN) from ASR output provides a representation of multiple aligned words along with their confidence scores. It contains more information than best hypotheses (1-best) of ASR. The competing words at the same approximate time stamp in ASR are aligned within the same position. A posterior probability is assigned to each word to measure the confidence of the result. Figure 3.7 shows an example of a confusion network and ASR 1-best output. The nodes represent the positions and the edges, called *bin*, represent possible words with their probability with p_{word}. At each position the summation of the words' probabilities is equal to 1. ($\sum_j p_j = 1$).

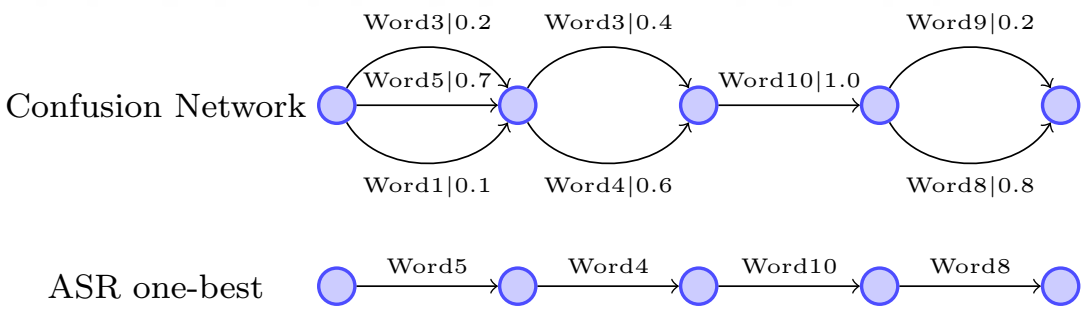

Figure 3.7. Confusion network and ASR one-best example.

As seen in Figure 3.7 there are 3 competing words at position 1: Word3, Word5 and Word1 with their probabilities $0.2, 0.7, 0.1$ respectively. ASR one-best output gives the best result which is Word5 for this position regarding their probability. It is possible that ASR is very certain about its hypothesis at a position and generates only one word. As seen in Figure 3.7 at position 3, there is only one word, Word10, with probability 1.0.

In this work, our goal is to integrate the confusion network into QA system to handle ASR errors. The integration of the confusion network is performed in the

preprocessing layer of the MatchLSTM. We apply two different approaches during integration: (i) using confidence scores as features; (ii) generating uncertainty-aware word representations. Confusion network is only used in passage representations, since the system works with written questions.

(i) Confidence Scores as Features

In the first approach, confidence scores of the words in the best hypothesis are used as constant additional features in the word embedding layer of the LSTM used during preprocessing.

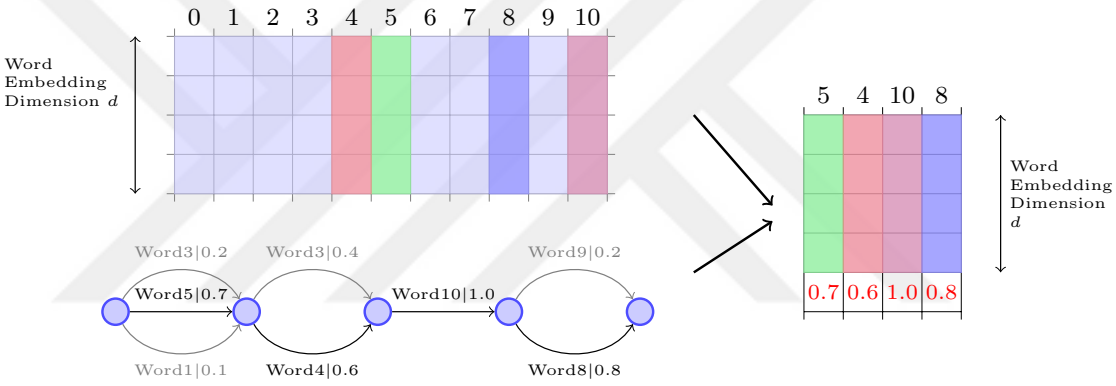

Figure 3.8. Confusion scores as features in MatchLSTM model.

Figure 3.8 shows a representation for the example confusion network. For each bin in the confusion network, a single feature vector is generated with the word embedding and the confidence score of the best scoring word. Word embeddings are updated during training but this additional word confidence feature is kept constant. For a given passage $\mathbf{P}$, the preprocessing layer of the question answering network generates the hidden representations $\mathbf{H}^p$ using also this additional feature. In total, the dimension of $\mathbf{H}^p$ is increased by $1 \times P$ where P is the length of the passage. Note that only the word vector embeddings are updated during training, not confidence scores.

(ii) Uncertainty-aware word representation

In the second approach, we generate a new weighted word embedding vector that represents all the words in the confusion network bin. This weighted embedding vector is called the uncertainty-aware word representation.

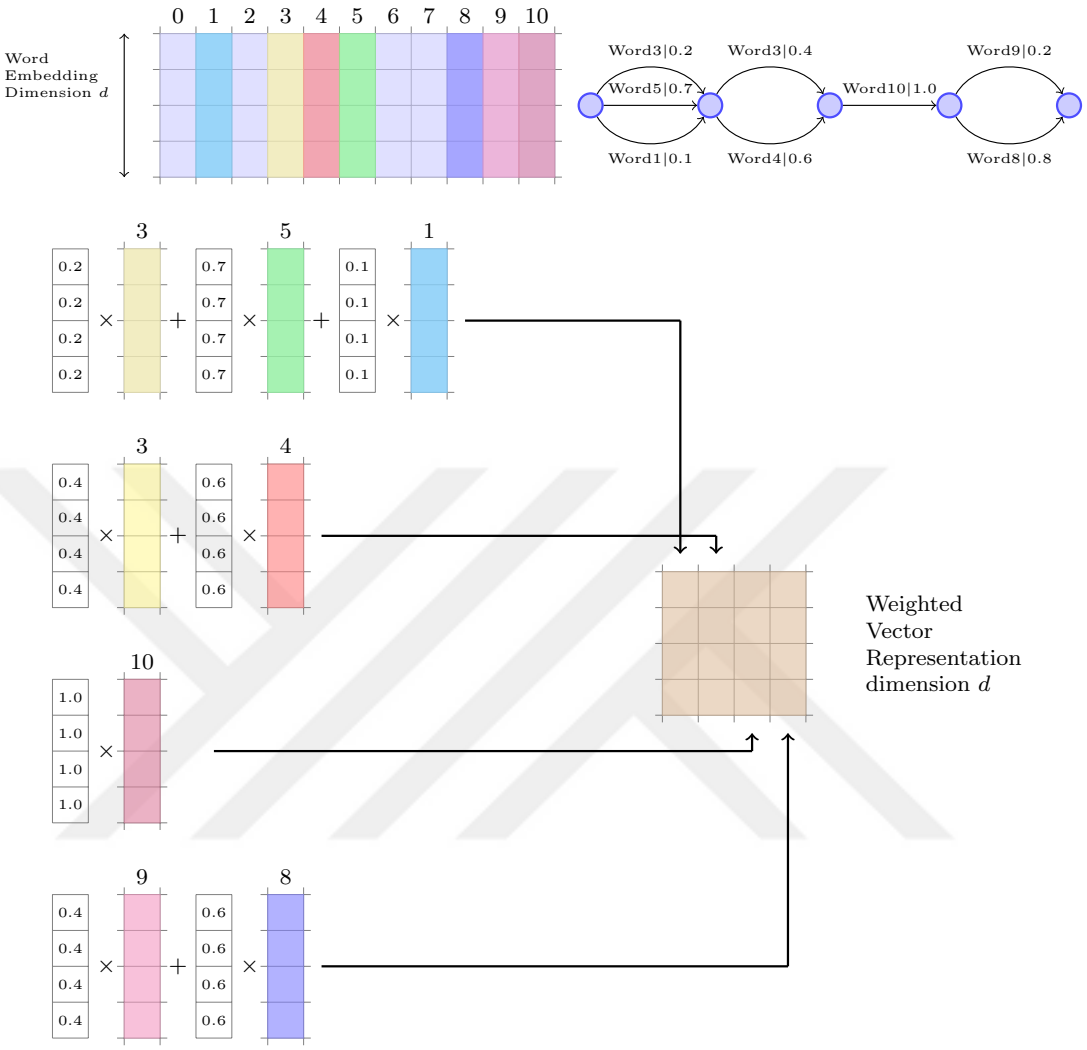

Figure 3.9. Weighted word vector representations with confidence scores.

Figure 3.9 shows how this representation is formed for the example confusion network. The embedding matrix and an example confusion network are given at the first row of the image. The embedding vector dimension is 4 in the example. For each word vector, different color is used to better understanding of the model. For each bin in the confusion network, one vector representation is formed for all the words in the bin by summing the word embedding vectors after weighting them with their confidence scores. For the first bin, the word vectors of Word3, Word5 and Word1 are collected. Each of them is multiplied by its probability from first bin (0.2, 0.7, 0.1 respectively). The multiplication is done element-wise. Then the sum of these vectors are added to the passage vector representation at the first

position. This process is applied to all bins of the confusion network. Then this new embedding vector is used as input to LSTM during preprocessing to generate an uncertainty-aware passage representation. For a given passage $\mathbf{P}$, the hidden representation $\mathbf{H}^p$ is generated using this new embedding representation. Note that during training, only word embedding vectors are updated, not confidence scores.

For both proposed models, the main modification is in the passage representations at the preprocessing layer. The remaining layers (MatchLSTM and answer pointer) are the same with the baseline MatchLSTM model.

4. EXPERIMENTS AND RESULTS

The aim of this section is to explain the details of the experiments and to give the results. Firstly, we explain two curated datasets by comparing them with SQuAD. Then, the experiments and results are given.

4.1. Datasets

4.1.1. Stanford Question Answering Dataset - SQuAD

SQuAD is a freely accessible question-answering dataset released by Standford University [1]. They offer a platform to test the model ability on machine comprehension task. The training and development sets are accessible through SQuAD website, but to preserve the integrity of the results the evaluation set is hidden. The evaluation step is run using a submitted model and the official scores are released on the website.

SQuAD contains a set of Wikipedia articles split into short passages and question-answer pairs for each passage. The answer of a question is a word sequence in the passage and each passage is a part of a Wikipedia article. Table 4.1 gives an example from SQuAD. The passage is in the first row and related three questions are in the second row. The passage is a part of the article titled as *To Kill a Mockingbird*. In Table 4.1, the answers are underlined in the passage and they are superscripted with the number of their related questions.

While generating SQuAD, [1] first Wikipedia articles were collected using Project Nayuki's Wikipedia's internal PageRanks. 536 articles were randomly sampled from 10,000 articles in English. By discarding figures, tables etc. and dividing articles into paragraphs 23,215 paragraphs were obtained for 536 articles. Training set (80%), development set (10%) and test set (10%) were randomly selected from these articles. The questions were proposed by crowdworkers using Daemo platform that has Amazon

Table 4.1. Question-answer pairs with the passage from SQuAD.

To Kill a Mockingbird is a novel by Harper Lee$^{(i)}$ published in 1960$^{(ii)}$. It was immediately successful, winning the Pulitzer Prize, and has become a classic of modern American literature. The plot and characters are loosely based on the author's observations of her family and neighbors$^{(iii)}$, as well as on an event that occurred near her hometown in 1936, when she was 10 years old.
(i) Who wrote To Kill a Mockingbird? (ii) When did To Kill a Mockingbird first get circulated? (iii) Whom did Lee base the characters in To Kill a Mockingbird on?

Mechanical Turk as its backend.

Table 4.2. SQuAD statistics.

	Training Set	**Evaluation Set**
Number of Questions	87599	10570
Number of Passages	18896	2067
Average Question Length (in words)	10	10
Average Answer Length (in words)	3	3
Average Passage Length (in words)	117	123

Table 4.2 shows the size of the dataset interms of the number of the questions and the number of the passages as well as the average number of words in training and development sets for questions, answers and passages. The distributions for the word count in questions and answers for both sets (training, development) are represented in Figure 4.1 and Figure 4.2. The majority of the answers have only one sigle word.

4.1.2. Question-Answer Dataset for English Lectures

QA training dataset for English lectures were prepared using the reference transcriptions of MIT OpenCourseWare *Signals and Systems* course videos. These reference transcripts were first divided into 1309 short passages. 310 question-answer pairs were

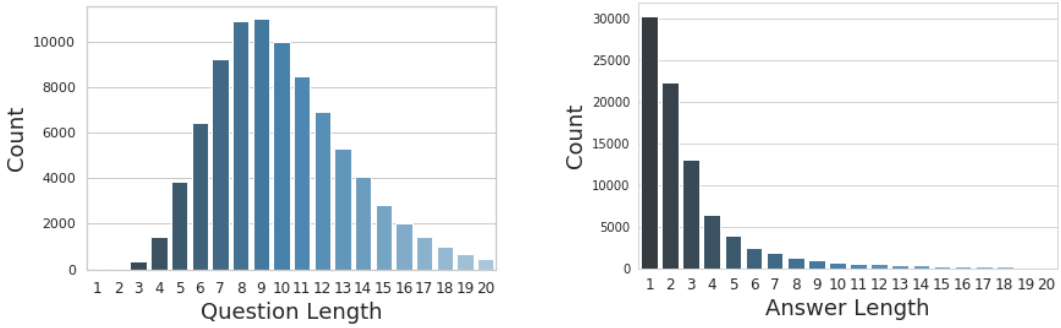

Figure 4.1. Word count distribution for questions/answers in SQuAD training set.

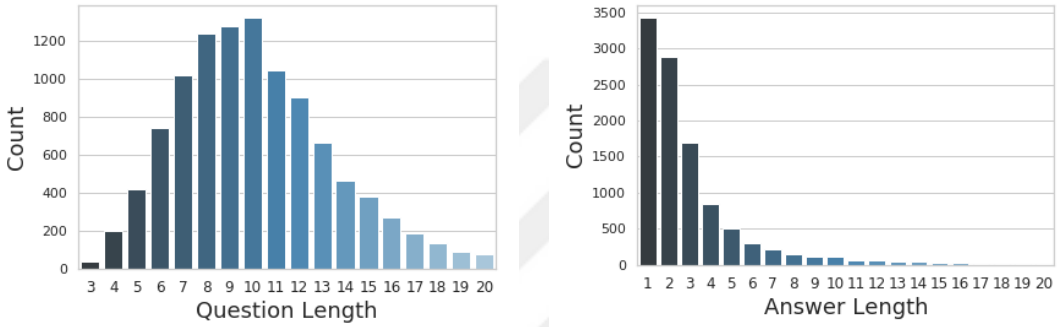

Figure 4.2. Word count distribution for questions/answers in SQuAD development set.

generated for 259 passages. Answers were selected as consecutive word sequences from these passages. The average question length, the average answer length and the average passage length (in words) are given in Table 4.3. QA test set for English lectures were prepared using the reference transcriptions of the Signals and Systems course videos from MEF University. Each video is on average 5 minute long and reference transcription of each video contains on average 652 words. Test data was manually divided into short passages based on the transcriptions corresponding to each lecture slide in the video. Note that the training data and the test data are from the same lecture domain.

Comparing with SQuAD, Table 4.2, our English lectures QA dataset introduces a more difficult task mainly due to the long answers. The difference can also be seen from the distribution graphs for English Lecture QA dataset given in Figure 4.3 and 4.4 and for SQuAD given in Figure 4.1 and 4.2. The distribution of the question length for both datasets looks similar, but the distribution of the answer length shows the

Table 4.3. English lectures QA statistics.

	Training Set	**Evaluation Set**
Number of Questions	309	175
Number of Passages	259	94
Average Question Length (in words)	10	10
Average Answer Length (in words)	24	22
Average Passage Length (in words)	72	81

distinction. The English Lecture QA dataset have longer answer spans than SQuAD.

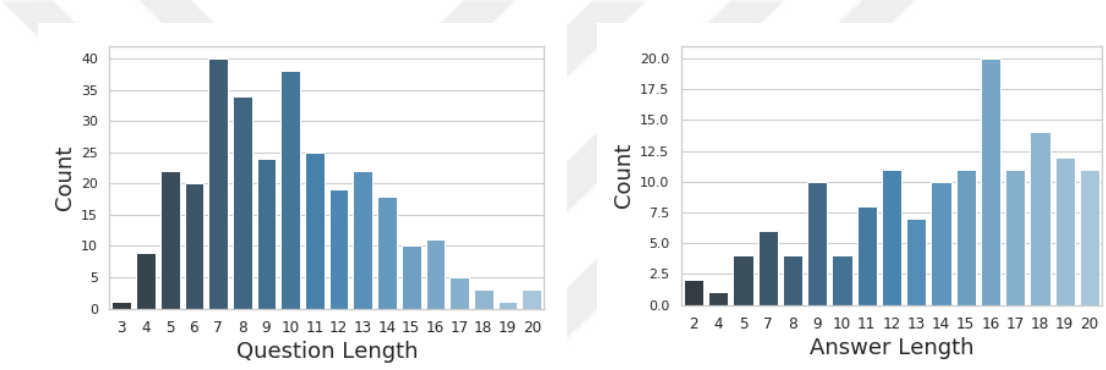

Figure 4.3. Word count distribution for question/answers in training set of English lectures QA dataset.

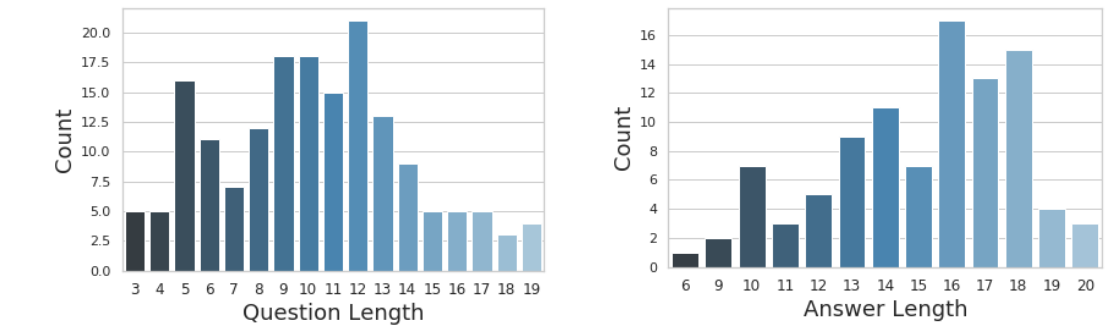

Figure 4.4. Word count distribution for question/answers in evaluation set of English lectures QA dataset.

4.1.3. Question-Answer Dataset for Turkish Lectures

Turkish Lectures QA dataset was prepared using the reference transcriptions of Turkish Law Lecture videos from MEF University. The audio data used for QA contains 18 hours of clean speech. The videos are on average 9-10 minutes long and reference

transcription of each video contains 905 words on average and each video explains a single topic. Table 4.4 contains an example short passage and question/answer pairs from Turkish Lectures QA. The answers are underlined in the passage and they are superscripted with the number of their related questions.

Table 4.4. Question-answer pairs with related text from Turkish lectures QA dataset.

Öncelikle ceza hukukunun kriminolojiyle arasındaki ilişki. Kriminoloji sözcük anlamı olarak suç bilimi demektir$^{(i)}$. Kriminoloji suçun neden ve sonuçlarını araştırır$^{(ii)}$ ve bundan dolayıdır da ceza hukuku kuralları düzenlenirken kriminolojinin elde ettiği sonuçlardan her zaman yararlanılır.$^{(iii)}$ Bundan dolayı da kriminolojiyle ceza hukuku arasında zorunlu bir ilişki bulunmaktadır.
(i) Kriminoloji ne demektir? (ii) Kriminoloji neyi araştırır? (iii) Ceza hukuku ile kriminoloji arasında neden zorunlu bir ilişki vardır?

Table 4.5. Turkish lectures QA dataset statistics.

	Train Set	**Dev Set**	**Eval Set**
Number of Questions	870	279	286
Number of Passages	549	134	208
Average Question Length (in words)	6	6	6
Average Answer Length (in words)	16	17	17
Average Passage Length (in words)	119	90	95

For developing and testing the question answering system, the data was partitioned into training (12 hrs), development (2 hrs) and evaluation (4 hrs) datasets. First, the reference transcription of each lecture recording was manually divided into short passages. Then for each passage, questions related to the passage were created and the answers of that questions were selected as consecutive word sequences from

the passage. The details of the datasets are given in Table 4.5.

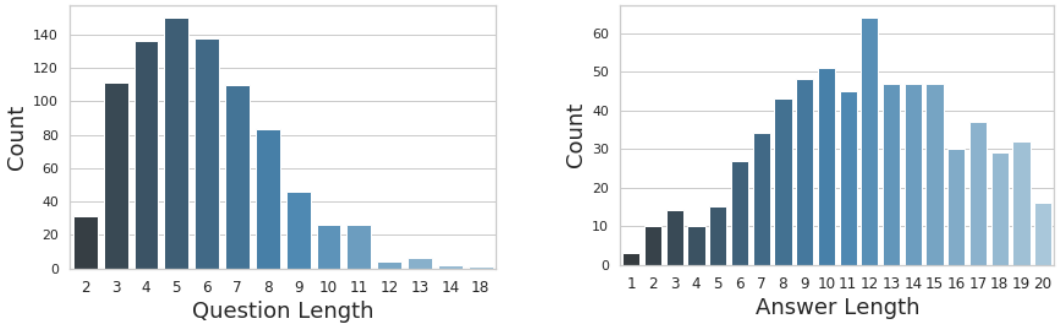

Figure 4.5. Word count distribution for question/answers in training set of Turkish lectures QA dataset.

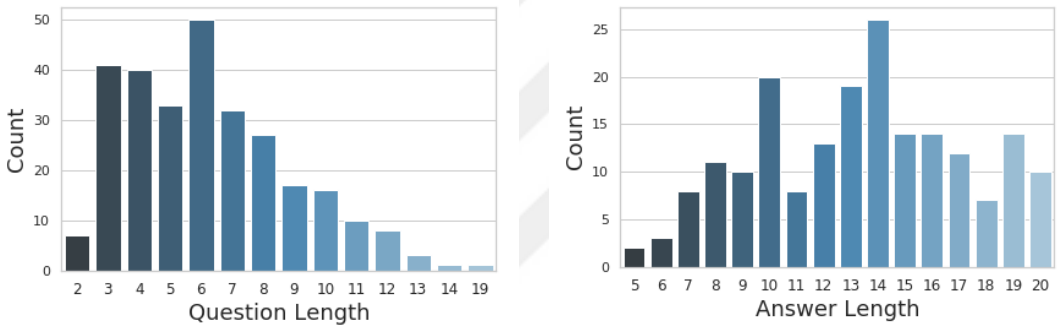

Figure 4.6. Word count distribution for question/answers development set of Turkish lectures QA dataset.

Figure 4.5 and Figure 4.6 give word count distribution in questions and answers in Turkish Lectures QA training and evaluation sets. By comparing the distributions in SQuAD, we can conclude that the answer length in Turkish Lectures QA is longer than the answers in SQuAD.

4.2. Implementation Details

4.2.1. QA System

The QA system takes a passage and a question and returns an answer range that indicates the start and end positions of the answer in the passage. The performance of the model is measured using F1 score. The QA system was implemented using PyTorch 1.0.0 [52]. For the best scored models, the embedding dimension of the word

vectors was set to 50 and the hidden dimension of the model was set to 150. The word embeddings are initialized by randomly sampling between $(0, 1)$.

Along with the word embeddings, character embeddings are used. As additional features for the model, the matching words between the passage and the question are added as binary features. The size of the exact match feature is equal to the number of words in a passage. If a word is common both in the passage and in the question, then the exact match feature value of this word becomes one.

Optimizers update the weight parameters to minimize the loss function. AdaMax algorithm [53] is used for the optimization of the model during training with learning rate 0.002. AdaMax is an adaptive stochastic gradient descent method, and a variant of Adam based on the infinity norm. A scheduler over learning rate is applied to minimize the loss for development set.

A scheduler was used to change the learning rate while training the model. It reduces the learning rate if the loss on development dataset does not change at each epoch. The epoch number is chosen as 20, because it is observed that the model stops learning after 20 epochs. In this setting the model can continue training if the metrics continue to improve. Figure 4.7 shows two plots of this run. The first plot gives the F1-score and exact match (EM) for the epochs. The second plot gives the loss for training and development dataset. As seen from the figures, after 20 epochs the metrics do not change significantly.

To assure the generalization of the model, dropout is added to the LSTM layers with a rate of 0.4 and to the embedding layers with a rate of 0.1. Dropout [54] is a regularization technique that was proposed to solve the over fitting issue in the neural networks. During training, individual nodes are either dropped out (ignored) or kept with a given probability. When testing the model, all the activations and nodes are used.

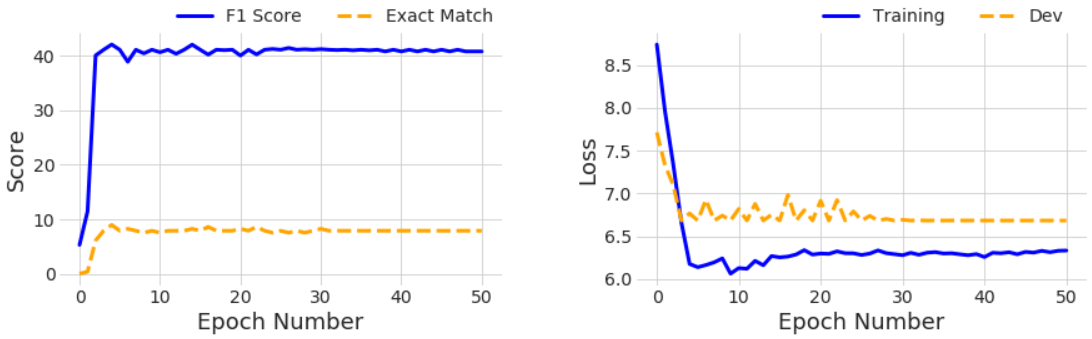

Figure 4.7. F1-Score/Exact match plot and loss plot of training and development sets of Turkish QA dataset.

Passage-question relevancy (PQ) model is also implemented using PyTorch 1.0.0 [52] by modifying matchLSTM model's third layer (answer pointer) with a single layer neural network. To test the effectiveness of PQ-model, a setup with three steps is used:

(i) Obtain short pseudo passages using a fixed length window.
(ii) Finding the passage and the question relevancy using PQ-model
(iii) Finding the answer span for the question in the predicted passage (top-scored passage by PQ-model) using QA system

As the first step, long lectures which does not possess any short passage boundaries are divided into pseudo passages. These pseudo passages contain 200 consecutive words. As the second step, PQ-model is trained with the same parameters with the QA system (embedding dimension, hidden vector dimension etc.), only the epoch number is chosen as 30. For constructing the training set for the PQ model, all the questions in a lecture are collected and for each passage in the lecture, a question, that has an answer in the passage, is marked as 1 (asked), while other questions are marked as 0 (not-asked). For a passage along with its asked questions, non-asked questions are sampled randomly among all the questions. The number of questions in two groups (asked and non-asked) are the same.

For constructing the evaluation set for PQ model, all questions in a lecture are paired with all the passages in that lecture. The trained PQ-model predicts scores

for each passage and each question. From these predicted scores, we choose the top-scored passage-question pairs assuming that each question must have an answer. The predicted passage/question paires are then fed into the QA system to make an answer span prediction. The PQ model is applied both to the Turkish and English QA datasets.

4.2.2. ASR System

Spoken documents used in this thesis contain lecture videos. The English lecture videos are from Electrical and Engineering Department lectures. Each lecture video is a short clip, on average 5 minutes long, explaining basic concepts about the lecture topic. The lecture videos were shot in the recording studio of the university. The synchronized audio was recorded with a high quality close-talking microphone. These videos were prepared by the same instructor so there is a single speaker in the acoustic data. The 15 lecture videos for the *Signals and Systems* course are set apart as the test data (1.2 hours) and the other videos are used as the acoustic training data (2.7 hours) for the ASR system. For building the language model for the ASR system, the reference transcriptions of the *Signals and Systems* course offered in MIT OpenCourseWareis used. These reference transcriptions contain around 100K words. In addition to these transcriptions, the reference transcriptions of the acoustic model training data (22.3K words) and the text coming from the lecture slides of the test data (3.2K words) are also used for language modelling. The acoustic models were trained using the Kaldi toolkit [55]. For English videos, three acoustic models are used: speaker independent Gaussian Mixture Model (GMM-si), speaker dependent Gaussian Mixture Model (GMM-sa) and deep neural network (DNN). The word error rate (WER) for these acoustic models were obtained as 13.0%, 10.5% and 6.7% respectively.

The Turkish lecture videos are from Law Lectures. The audio data contain 30 hours of clean speech coming from 18 law courses. The videos are on average 9-10 minutes long and each video explains a single topic. The audio data was split into four partitions: training set for acoustic model (12 hrs), training set for QA task (12 hrs), development set for QA task (2 hrs) and evalualtion set for QA task (4 hrs). The acoustic model training partition is used together with Turkish broadcast

news data (184 hrs) in training the acoustic model of Turkish ASR system. The ASR transcriptions have 14.4% WER.

4.3. Results

4.3.1. MatchLSTM (Baseline) for QA

First, we tested our baseline model with SQuAD to verify its capability with a well known dataset.

Table 4.6. SQuAD dataset results with MatchLSTM.

Model	**F1 Score**
MatchLSTM	73.45
MatchLSTM from [39]	71.2
MatchLSTM (ensemble) from [39]	76.8

Table 4.6 gives results for SQuAD dataset. All the results were obtained from development set, since the evaluation set is hidden. The score of the first row shows the result of our implemented version of MatchLSTM. Second and the third rows contain results from the original MatchLSTM implementation [39]. One of the reason of the difference between the first and second row is that we use GRU in our implementation instead of vanilla LSTM. They [39] tested the model with several improvements and with GloVe initialization and the best results was with an ensemble model.

Table 4.7 gives the results for English Lectures dataset with MatchLSTM model. In this setup, the reference text (Ref-Text) is the correct transcript of the spoken document and manually divided into short passages. The test sets are obtained from ASR transcripts (1-best output) with different acoustic models. The question-answer pairs for the ASR transcripts in test sets are obtained by aligning the ASR transcripts of the test data with the corresponding reference transcriptions. The details of the acoustic models on English lectures are given in Section 4.2.2. Evaluating a trained model (trained on reference text) on different ASR 1-best transcripts yields different F1

scores depending on ASR WER. The lower WER means that ASR 1-best transcript is closer to the reference text. As a result, the QA model F1 score gets higher while WER gets lower. In Table 4.7, last three rows are F1 results for the QA model evaluated on the ASR transcripts with varying WERs.

Table 4.7. MatchLSTM (baseline) results for English lectures QA dataset.

Train Set	**Eval Set**	**F1**	**WER**
Ref-Text	Ref-Text	58.48	0.0%
Ref-Text	ASR 1-best	56.38	6.7%
		54.38	10.5%
		54.83	13.0%

Table 4.8 gives the results for Turkish lectures QA dataset with MatchLSTM model. The reference text (Ref-Text) is the correct transcription of the spoken document and ASR 1-best is the ASR output. The model uses the evaluation set (Eval Set) only while testing and evaluating the parameters. Training the model on the reference text and evaluating the model on the ASR transcriptions of the evalutaion data yields the lowest F1 score (50.89). This is an expected result due to the mismatch between the reference and ASR transcripts. Note that evalutation set has 14.4% WER. Training the model on the ASR transcriptions (1-best hypothesis) improves the F1 score to 51.45 from 50.89.

Table 4.8. MatchLSTM (baseline) results for Turkish Lectures QA dataset.

Train Set	**Dev Set**	**Eval Set**	**F1**	**WER**
Ref-Text	Ref-Text	Ref-Text	52.44	0.0%
Ref-Text	Ref-Text	ASR 1-best	50.89	14.4%
ASR 1-best	ASR 1-best	ASR 1-best	51.45	14.4%

4.3.2. Passage-Question Relevancy Model

In order to show the effectiveness of the proposed passage-question relevancy approach, we prepared several train-test scenarios. The scenarios were tested with

Turkish and English lectures QA datasets. To be able to compare the results, we first specify a baseline setup for the PQ-model. This baseline setup is same as the setup described in the previous section 4.3.1, so that we use manually divided short passages. Secondly, we prepare another setup by assuming that there is no passage boundaries and for this setup we concatenate all the passages into one passage. Finally, we propose dividing long passages into pseudo-passage. The train-test scenarios are as follows:

(i) short - short: Tranining data and test data were manually divided into short passages. Questions associated with each passage are known explicitly. This is the initial version of our datasets. For English QA dataset, the average number of words in a passage is 81 for 94 test passages. For Turkish QA dataset, the average number of words in a passage is 95 for 286 test passages.

(ii) short - long: The QA model is the same with the one trained in the first scenario. However, each passage in the test data corresponds to the transcription of a lecture video. To obtain the test data, all the passages in a lecture are concatenated and the answer spans of the questions are searched in these long passages. For English QA dataset, the average number of words in a passage is 652 for 15 test passages. For Turkish QA dataset, the average number of words in a passage is 905 for 27 test passages.

(iii) long - long: Short passages in the training data were concatenated to obtain longer passages as in the second scenario. The QA system was trained with this training data. The test data is the same with the one in the second scenario. For English QA dataset, the average number of words in a passage is 1148 for 16 training passages. For Turkish QA dataset, the average number of words in a passage is 988 for 75 training passages.

(iv) window - window: The train and the test data were divided into pseudo passages using around 200 consecutive words per passage by taking sentence boundaries into account.

The QA results with these scenarios are given in Table 4.9 for English lectures QA dataset. The results from Table 4.9 were obtained when the passage and the

questions pairs are known in other words the questions are searched in their related passages. Short-short scenario is the initial version of our datasets in which the test passages were obtained manually. Each column is related to an ASR acoustic model and their WERs are given in the last column of the table. Each model was trained with reference text and evaluated with different ASR transcript from acoustic models. As seen in Table 4.9, the best result was obtained with the reference transcripts with short-short scenario.

Table 4.9. Results for different test scenarios when passage-question pairs are known for English lectures QA dataset.

	GMM-si	**GMM-sa**	**DNN**	**Reference**
short-short	54.83	54.38	56.38	58.48
short-long	23.53	23.62	24.46	25.29
long-long	33.37	33.02	33.13	34.92
WER	13.0%	10.5%	6.7%	0.0%

Note that short-short scenario makes the assumption that we know the passage that contains the answer to each question, which does not hold in a real test scenario. This assumption holds in part for the short-long and long-long scenarios where each test passage contains the transcription of a whole lecture video. However, the QA performances for these scenarios degrade significantly mostly due to the increasing passage length. For each acoustic models, the QA performance is lowest with short-long scenario. Even the long-long scenario's performances are better than short-long, we can't still reach short-short scenario results. In order to reduce the effect of passage length on the QA performance, the simplest method is to divide long passage into pseudo passages.

We tested the matchLSTM model with pseudo-passage settings and obtained F1 score as 43.46 for reference text when window size is 200 consecutive words. For the same settings, F1 scores were obtained as $42.33, 38.88$ and 37.91 for DNN, GMM-sa, GMM-si acoustic models. We also tested the model with different number of words in a window. Figure 4.8 shows F1 scores with various window lengths for English QA

dataset. As seen from the figure, increasing the number of words in the passage window causes the performance to decrease.

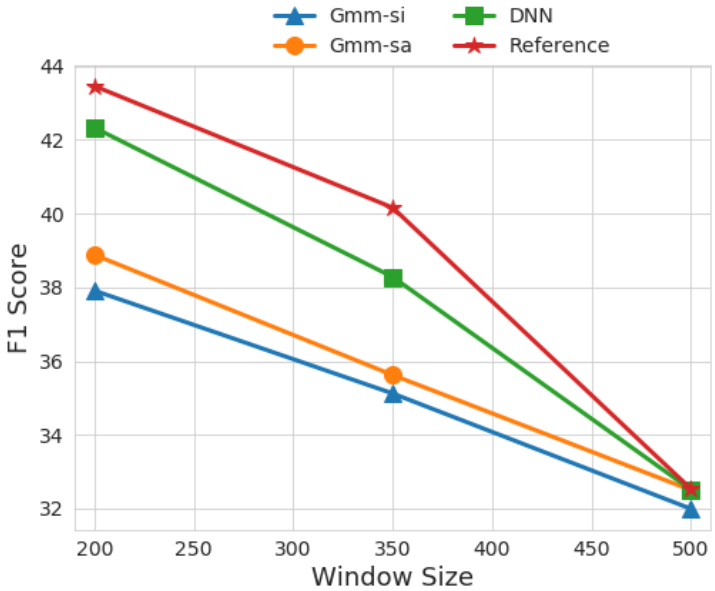

Figure 4.8. F1-Score with different window size on English lectures QA dataset.

Even though using pseudo-passages results better F1 scores in QA task than using no passage boundaries (long-long scenario), the QA task on lecture videos requires more specific approaches. Therefore, we converted the task into a more realistic, but still tractable, scenario by assuming that questions that are related to one of the chapters of the lecture will be searched only in the videos of the same chapter. To make the task tractable, we used the window-window scenario together with the proposed passage-question relevance scoring algorithm.

Table 4.10 gives F1 scores for three different scenarios. The results were obtained on evaluation sets from ASR transcripts with a model trained on reference text. The effect of WER on the QA performance can be seen in Table 4.10. As ASR error gets higher, the QA performance gets lower. The window-window scenario was executed using 200 consecutive words in a pseudo-passage.

The first row of Table 4.10 is from the experiments in which the passage boundaries are known in a lecture, so that each question is asked to its related short passage.

Table 4.10. Results for different test scenarios with PQ-model for English lectures QA dataset.

	GMM-si	**GMM-sa**	**DNN**	**Reference**
short-short	54.83	54.38	56.38	58.48
PQ + short-short	42.62	44.86	45.59	48.61
PQ + window-window	32.28	33.03	35.52	38.64
long-long	33.37	33.02	33.13	34.92
WER	13.0%	10.5%	6.7%	0.0%

This experiment was the result of matchLSTM model and it has the highest F1 score. The *PQ + short-short* experiment was obtained when the passage boundaries are known but it is not known which passage contains the answer to the given question. For this setup, we first run our proposed PQ-model, then the output of PQ-model (top scored passage/question pairs) is used with matchLSTM model on QA task. The *long-long experiment* is done by concatenating all the passages and each question is asked to this long passages. Because of the longer passage length, this experiment yields the lowest F1 score.

From Table 4.10, we can conclude that using longer passages degrades QA performance as does not knowing the passage/question pairs. The *PQ + window-window* experiment covers our proposed approach when no passage boundaries and passage/question paires are presented. For this experiment, PQ-model was applied to pseudo passages (around 200 consecutive words) to find passage/question pairs, then QA model was applied to the top scored pairs. The resulting F1 scores for ASR transcriptions with various WERs are given at the last row of Table 4.10. The F1 score of our proposed model is less than the baseline (short-short) but higher than long-long scenario. Note that short-short scenario is actually not suitable for spoken documents since spoken documents don't possess passage boundaries and any explicitly defined passage/question pairs. Our model is more suitable for spoken documents and results in better scores than asking the question to a long passage as in long-long scenario.

The QA performance on Turkish QA dataset is also tested for prepared scenarios (short-short, long-long, window-window). Table 4.11 shows results for this setup. For Turkish QA dataset, we use a single transcription (ASR 1-best) with 14.4% WER. The columns give the dataset types that the model is trained and evaluated as Train Set/Evaluation Set. The abbreviation *Ref* is used for reference text and *1-best* is used for ASR 1-best transcript.

In short-short scenario, the model knows the passage boundaries and passage-question pairs, whereas in long-long scenario, the model is trained with long passages which correspond to the transcriptions of lecture videos. The best result among short-short scenarios is obtained when we use the reference text for training and evaluation (F1 score 52.44). This is an expected result due to the ASR errors. Training and evaluating the model on ASR 1-best transcripts result in better F1-score (51.45) than training on the reference text and evaluating on ASR 1-best (50.98). The effect of passage length on QA performance can be observed by the decrease of the score between short-short and long-long scenarios.

Table 4.11. Results for different test scenarios when passage-question pairs are known for Turkish lectures QA dataset.

Train-Test Scenarios	**Ref/Ref**	**Ref/1-best**	**1-best/1-best**
short-short	52.44	50.98	51.45
long-long	29.58	26.27	22.45
window-window	34.95	29.52	32.76
WER	0.0%	14.4%	14.4%

In window-window scenario, the long passages are divided into pseudo passages which have 200 consecutive words. In Table 4.11, the results of window-window scenario show that dividing the long passages into pseudo passages yields better performance than using long passages.

The performance of PQ-Relevance model was also tested with Turkish Lectures QA Dataset. Table 4.12 shows the results for different scenarios. First two rows

Table 4.12. Results for different test scenarios for Turkish lectures QA dataset.

Train-Test Scenarios	**Ref/Ref**	**Ref/1-best**	**1-best/1-best**
short-short	52.44	50.98	51.45
PQ + short-short	42.46	40.61	42.47
PQ + window-window	30.86	27.20	29.65
long-long	29.58	26.27	22.45
WER	0.0%	14.4%	14.4%

are from the previous experiments in Table 4.11. As seen, the worst results are obtained when all the lecture's passages are used as a whole document. We applied PQ-model to the datasets when passage boundaries are known, but passage/question pairs are unkown to test the ability of the model (PQ + short-short). The difference between short-short and PQ+short-short experiments show that not knowing the passage/question pairs causes around 20% decrease on the performance of the QA task. On the other hand, using PQ-model with pseudo passages assures the increase on the QA performance, which can be observed by comparing the experiments PQ+window-window and long-long scenario. PQ+window-window covers our proposed model and can handle the problem of no passage boundaries in spoken documents.

4.3.3. ASR uncertainty into QA with Confusion Network

The two proposed approaches for integrating ASR uncertainty into QA are tested only with Turkish lectures QA dataset, because for English lectures QA dataset we don't have ASR confusion output for the training set. Table 4.13 shows the results for two models. The models given with abbreviations as CS (using confusion scores as additional features) and CN (using confusion scores with weighted vectors).

Adding confidence scores of the words in the best hypothesis as additional features improves the F1 score by 2 points compared with the baseline result. Note that the only difference between the ASR best hypothesis and the CN (Confidence score features) experiments is the additional confidence score features. This result shows that we can handle ASR errors better with the QA model. Lastly, we experiment with uncertainty-

aware word representations and this approach yields our best result as 56.17. This experiment's result is even better than using reference text for training and evaluation (F1 score 51.44 in Table 4.12).

Table 4.13. Results when using confidence scores as features.

Model	**Train Set**	**Eval Set**	F1
Baseline	ASR 1-best	ASR 1-best	51.45
Confidence scores as features (CS)	ASR 1-best	ASR 1-best	53.45
Uncertainty-aware word Rep. (CN)	ASR 1-best	ASR 1-best	56.17

The results on Table 4.13 show the importance of dealing with the uncertainty in the spoken question answering performance. Our proposed method can be applied to spoken documents when no reference text is available.

Using the best scored model on Turkish lectures QA dataset, we tested several parameters of the model. Table 4.14 shows F1 score for different setups. The first column gives the name of the parameters. EmbedDim is the dimension of the vector embeddings at the preprocessing layer. HDim is the dimension of the hidden layers of LSTM at the MatchLSTM layer. LR is the starting value of learning rate. A scheduler is used to change learning rate while training by taking into account the loss. If the loss doesn't change over an epoch, the learning rate is reduced. Epoch is the number of epoch for training. BatchSize is the number of sample in one batch. Dropout is the rate of the dropout on weights. The best score is presented on the first row. Each section of table is created to show the change on one parameter. While testing a parameter, other parameters are set as initial.

Increasing embedding dimension and hidden dimension causes a lower F1 score, because the number of learned parameters are increasing too. Also, changing initial learning rate values changes F1 score, because learning rate effects how much parameters change during the learning process.

Table 4.14. F1 score of the model when using with different parameters for Turkish lectures QA dataset.

EmbedDim	HDim	LR	Epoch	BatchSize	Dropout	F1
50	150	0.002	10	32	0.4	56.17
100	150	0.002	10	32	0.4	52.68
200	150	0.002	10	32	0.4	47.18
50	100	0.002	10	32	0.4	53.16
50	200	0.002	10	32	0.4	55.07
50	300	0.002	10	32	0.4	44.45
50	150	0.001	10	32	0.4	53.03
50	150	0.003	10	32	0.4	55.50
50	150	0.002	30	32	0.4	56.17
50	150	0.002	10	16	0.4	55.11
50	150	0.002	10	64	0.4	54.66
50	150	0.002	10	16	0.2	52.48
50	150	0.002	10	64	0.6	53.25

4.4. Further Analysis of the Results

To have a better understanding of the strengths and the weaknesses of the QA model, we perform several analysis on the results. First, we suspect that the length of the answers and the questions affect the performance of QA model. For example, the answers and the questions which possess more words than average are harder to predict. To verify this hypothesis, we analyse the performance in terms of F1 score with respect to the answer and question length on the evaluation set.

Figure 4.9 gives two plots that show the F1 score by the length of the answer (top) and number of samples by the length of the answer span (bottom). First plot shows that the answers that have more words than average (16 in Turkish lectures QA dataset) have lower F1 score and the answers with less words than average don't have better F1 score. Instead the answers which have around 14 words have better F1-score. This case can be explained with the second graph (bottom) in Figure 4.9. The second

graph gives the number of answers by answer length. The graph is similar to the first one and it shows that in Turkish lectures QA dataset the answers that have around 14 words are dominant and as a result they have higher F1 score in total.

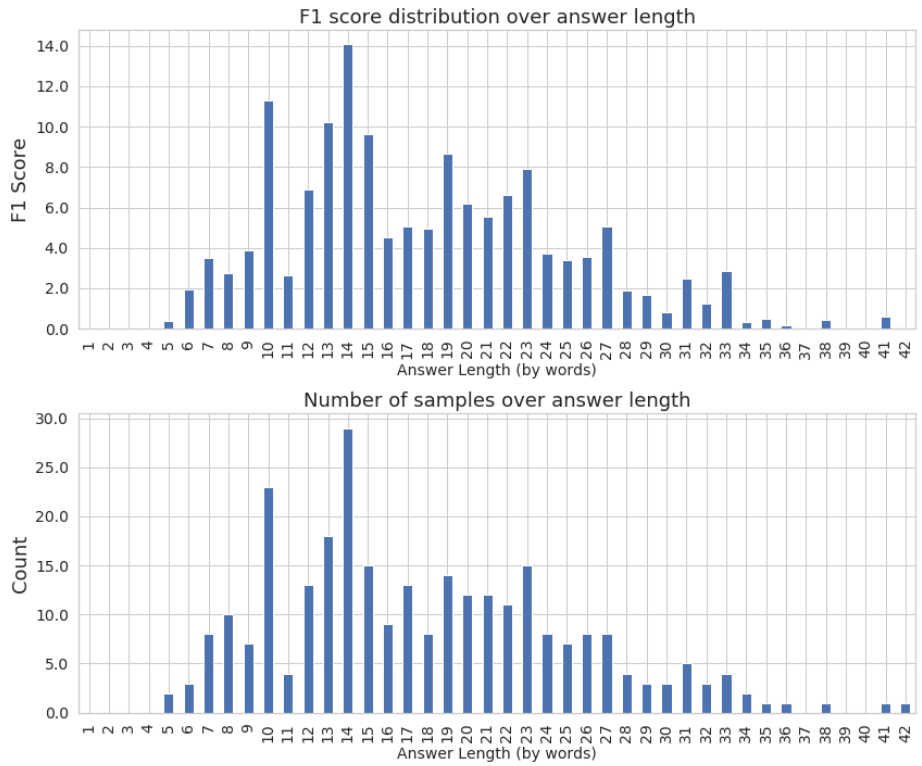

Figure 4.9. F1-Score by the length of the answer (top) and number of samples by length of answer span (bottom) for Turkish lectures QA dataset.

We applied the same analysis as we did with answers to the questions. Figure 4.10 gives two plots that show the F1 score by the length of the question (top) and number of samples by the length of the question. It is observable that longer questions have lower F1 score, but also the number of long questions is quite low. The questions with higher F1 scores have around 6 words, which is the average number of words in questions for Turkish lecture QA dataset.

As a second analysis, we investigate the effect of question types on model's performance. We first divide the questions into different groups manually for Turkish Lectures QA dataset by taking the language characteristics into account. For the evaluation dataset, there are 286 questions. We choose 8 question types to divide the questions into: *what, who, where, which, how many, how, why, when*. Table 4.15 gives

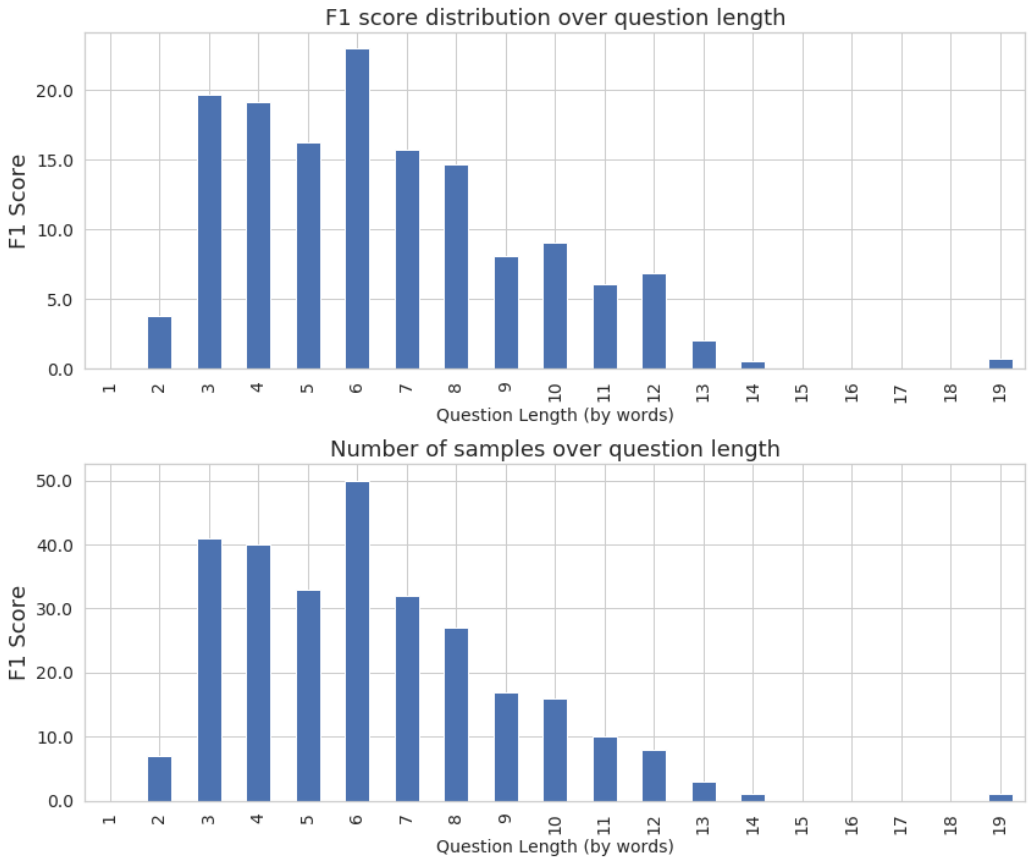

Figure 4.10. F1-Score by the length of the question (top) and number of samples by length of question span (bottom) for Turkish lectures QA dataset.

the question classes and their related Turkish question words.

Figure 4.11 gives two plots by question types. The first one shows the F1 scores by types and the second one shows the number of questions by types. In the first plot, It can be observable that the model can learn the questions of type *where*, however the number of questions in this class is very few. We can't conclude that there is a direct influence of question types on the performance, since there are very few questions in groups. The question class *what* dominates the dataset and have average F1 score.

Table 4.15. Question classes for Turkish lectures QA dataset.

Question Classes	Turkish Question Words
what	ne, nedir, neyi, neler, neye
who	kim, kimler
where	neresi, nere, nerde
which	hangi, hangisi
how many	kaç, ne kadar
how	nasıl
why	neden, neyden
when	ne zaman

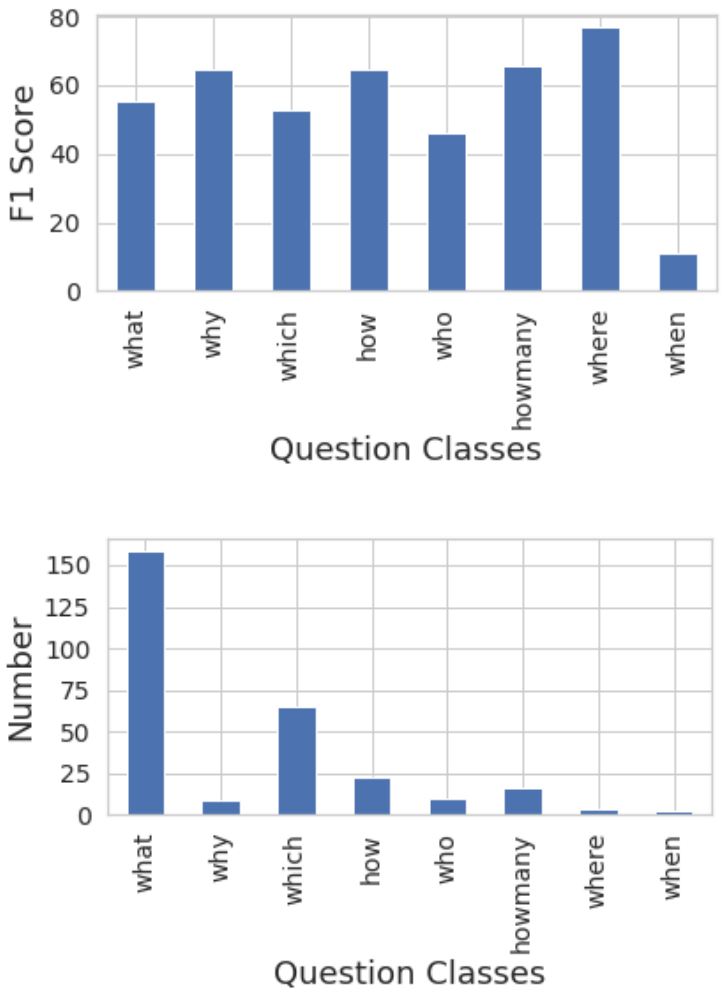

Figure 4.11. F1 score by question type and the number of questions by types in Turkish lectures QA dataset.

4.4.1. Examples

We collected some examples of wrong and correct predictions from top scoring model for analysis. Table 4.16 gives a passage, a question and a correct prediction. Figure 4.12 shows the question-aware passage representation of the sample from Table 4.16. Darker colors represent higher scores. First two words of the question (*müşterek* and *faillik*) have generally higher scores than the third word *nedir*, because they also exist in the passage. However, the meaning of the question is carried by *nedir*, which means *what*, and it has the highest scores with words like *konusudur*, *olmaktır*, *faildir*, *kişilerdir*. These words are used at the end of the sentence as *nedir* and they define the answer for this type of questions.

Table 4.16. Question-answer pairs with related text from Turkish lectures QA dataset.

Fiili birlikte gerçekleştirme veya suçun işlenmesinde başkasını araç olarak kullanma dolaylı fail halinde fail söz konusudur. Müşterek fail lik fiili birlikte gerçekleştirme suçu işlemek suç tanımında yazılı olan hareketi yapmış olmaktır. Suç tanımında yer alan fiili birlikte gerçekleştiren kişilerden her biri faildir. Müşterek failler suçu meydana getirecek hareketleri yapan kişilerdir.
Question: Müşterek faillik nedir? Answer: Müşterek fail lik fiili birlikte gerçekleştirme suçu işlemek suç tanımında yazılı olan hareketi yapmış olmaktır. Prediction: Müşterek fail lik fiili birlikte gerçekleştirme suçu işlemek suç tanımında yazılı olan hareketi yapmış olmaktır.

In Figure 4.13, we can see that the start position of the answer, word *müşterek* and the end position of the answer, word *olmaktır* get the highest values.

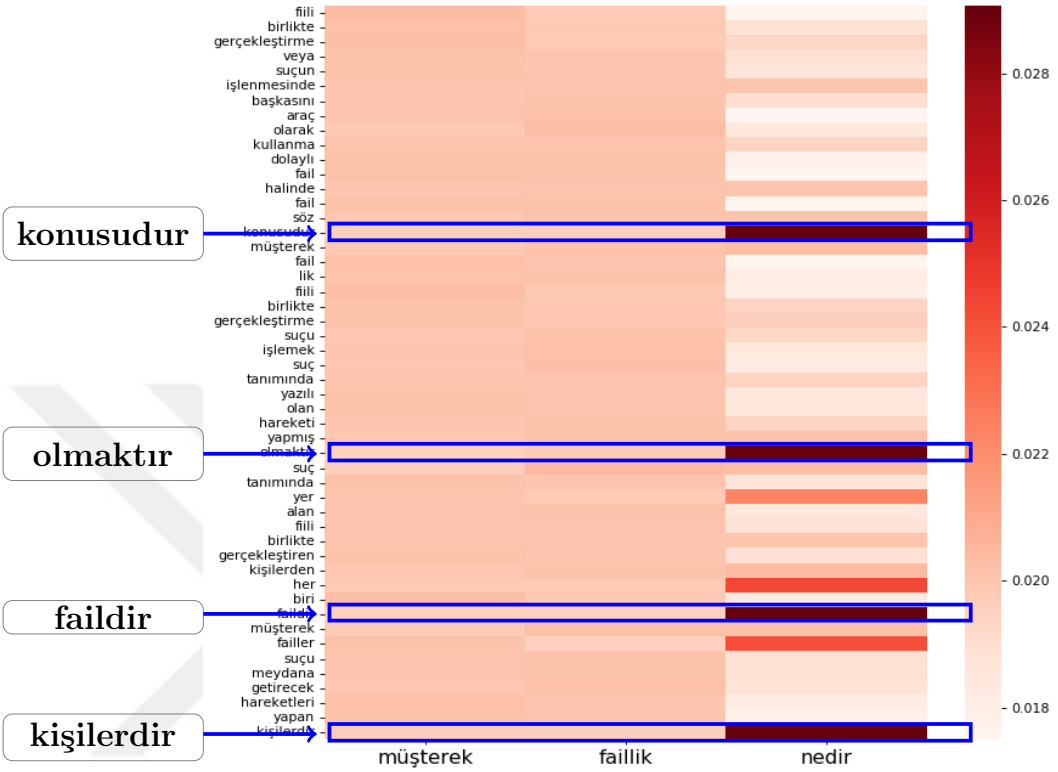

Figure 4.12. Question-aware passage representation for the sample in Table 4.16.

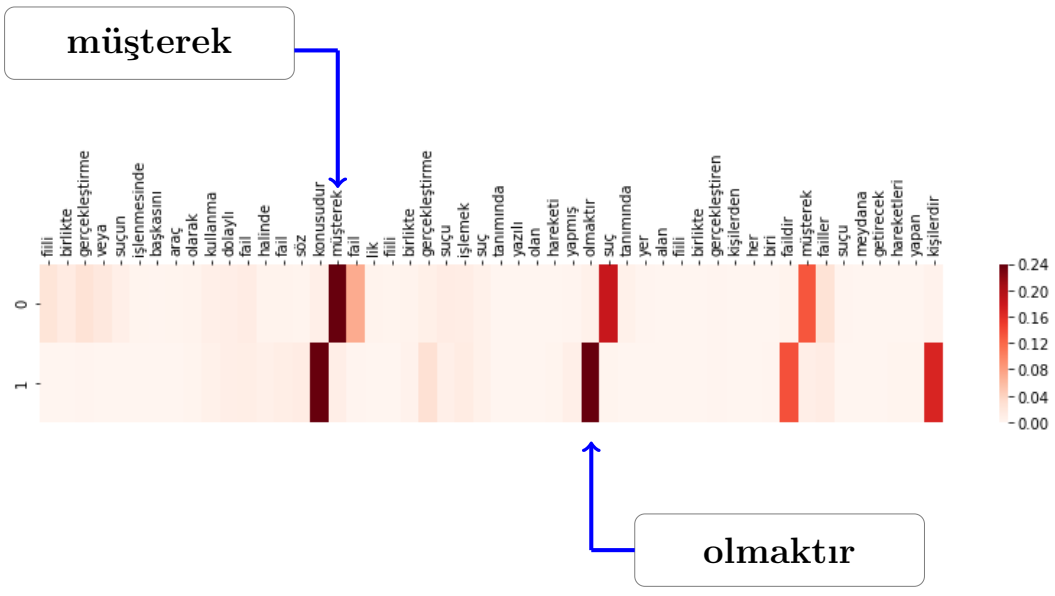

Figure 4.13. Pointer network attentions for the sample in Table 4.16.

Table 4.17 shows another example of true prediction span from Turkish lectures QA dataset. In this example, the model is able to make connection between two words which comes from the same stem (as *örnek* and *örneğin*). Table 4.18 shows an incorrect prediction example. Even though this example has the words derived from the same stem (*örnek* and *örneğin*) the model could not do this connection.

Table 4.17. Question-answer pairs with true prediction span from Turkish lectures QA dataset.

Çok faili suçlarda faillerin birden çok olması yasada kurucu unsur olarak gösterilmiştir. Örneğin ihaleye fesat karıştırma rüşvet gibi suçlar çok faili suçlardır. Bu tür suçlardan bir birden çok fail yoksa bu suçlarda suç oluşmaz. Kamu görevini terk suçunda yine gerek ıı türk ceza yasasının üç yüz altmış altmışıncı maddesinde düzenlenmiştir ıı suç işlemek amacıyla ıı yasada en az iki kişinin bulunması gerekir. Yine türk ceza yasasının iki yüz yirminci maddesindeki örgüt kurma suçunda da yine iştirak halinde en az ıı dört kişinin bu üç kişinin daha doğrusu bu örgüte ıı katılması zorunludur.
Question: Çok failli suçlara örnek ne olabilir? Prediction: örneğin ihaleye fesat karıştırma rüşvet gibi suçlar çok faili suçlardır Answer: örneğin ihaleye fesat karıştırma rüşvet gibi suçlar çok faili suçlardır

Figure 4.14 visualizes the question-aware passage representation from matchLSTM model for the example in Table 4.18. These inner representations show the word-by-word matching between the question and the passage. Darker colors are higher scores. When a question word is also represented in the passage, its weight gets higher than others (as *neticesi, harekete, bitişik, suçlara*). The word *ne* means *what* and carries the question meaning in the sentence. In Figure 4.14, the words *değildir, yapılır* have higher scores with the words of question. They are the verbs and represented at the end of the sentences. Their scores show that the model can understand the verbs and their locations in the sentences. However the words *yapılmaz, olmaz* are also verbs, but the model does not assign high scores to them, which shows that the model is still not able to understand negation in the verbs for this example.

Table 4.18. Question-answer pairs with incorrect prediction span from Turkish lectures QA dataset.

Neticesi harekete bitişik olan suçlarda da teşebbüs yani suça kalkışma söz konusu değildir. Bu suçlar icra hareketleri yapılır yapılmaz tamamlandıklarından yani neticesi harekete bitişik sırf hareket suçları olduklarından bu suçlarda icra hareketlerinin bir tipte neticenin meydana gelmemesi durumu gerçekleşemez. Örneğin yüzde karşı sövme yani hakaret konut dokunulmazlığını bozma yalan yere yemin etme ve fuhuş için çocuk tedarik etme suçlarına suçlarında teşebbüs olmaz.
Question: Neticesi harekete bitişik suçlara örnekler ne olabilir? Prediction: Neticesi harekete bitişik olan suçlarda da teşebbüs yani suça kalkışma söz konusu değildir. Answer: Örneğin yüzde karşı sövme yani hakaret konut dokunulmazlığını bozma yalan yere yemin etme ve fuhuş için çocuk tedarik etme suçlarına suçlarında teşebbüs olmaz

Figure 4.15 shows the pointer network attentions for the passage in Table 4.18. There are two columns indicating the start (column 0) and the end (column 1) probabilities. Note that the model doesn't output the maximum values for the probabilities, instead it tries to find the maximum start and end probability by their multiplication. The true answer start with *örneğin* and ends with *olmaz*, but our model outputs the answer span between *neticesi* and *değildir*. One possible reason of this results it that the question words are similar to the passage words.

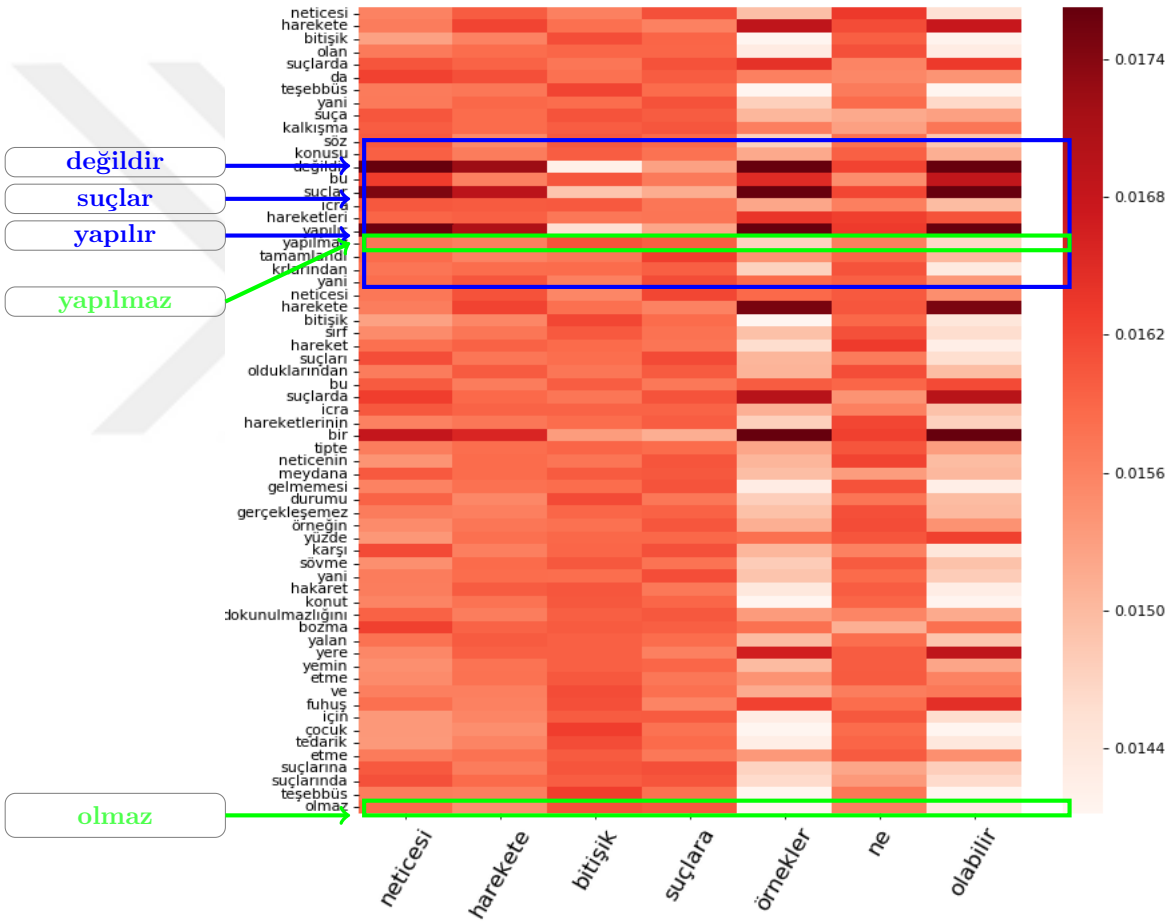

Figure 4.14. Question-aware passage representation for the sample in Table 4.18.

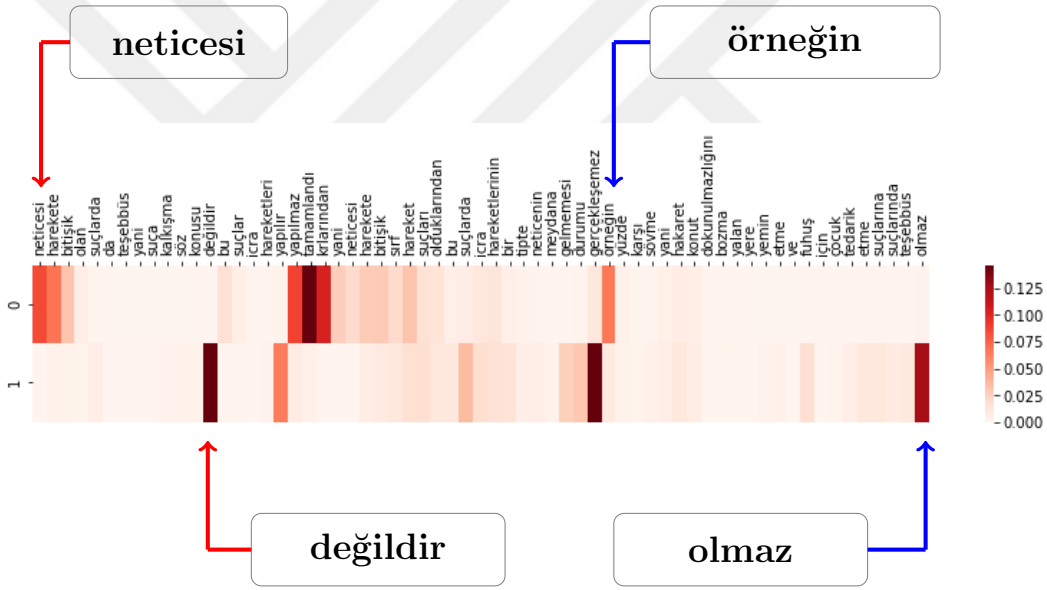

Figure 4.15. Attention vectors from pointer network for the sample in Table 4.18.

5. CONCLUSION

In this study, we presented novel approaches for question answering on spoken documents and proposed two QA datasets: English lectures QA dataset and Turkish lectures QA dataset. Our QA system uses ASR transcripts to answer a question in natural language. We investigated QA on spoken documents task with the methods developed on machine comprehension task.

One of the challenges that we encountered is to adapt an end-to-end neural network model that is developed for reading comprehension task to QA on spoken documents. This is the challenging because the methods in reading comprehension emphasize applying the QA task over short passages while spoken documents do not provide any passage boundaries. We attempted to solve this problem by proposing a passage-question relevance model. We showed that the proposed model has promising QA performance.

While experimenting with ASR transcripts, we observed that the performance of the QA model depends on the errors in ASR transcripts. As the error in ASR transcripts decreases, the performance of the QA model increases. We then proposed two approaches to handle the uncertainity of ASR into the QA model. The confusion networks, an ASR output representations, are integrated into the neural network model and this integration improves the QA performance on the spoken documents.

As a future direction, we will work on handling the unanswerable questions. Beside scoring the passage and questions, the model can also predict if a question doesn't have any answer span in the lecture. The problem can be handled by adding a special character to define *unanswerable* class or using a treshold over the scoring obtained from passage-question relevance model.

REFERENCES

1. Rajpurkar, P., J. Zhang, K. Lopyrev and P. Liang, "SQuAD: 100, 000+ Questions for Machine Comprehension of Text", *CoRR*, Vol. abs/1606.05250, 2016, `http://arxiv.org/abs/1606.05250`.

2. Woods, W. A., "Progress in Natural Language Understanding: An Application to Lunar Geology", *Proceedings of the June 4-8, 1973, national computer conference and exposition*, pp. 441–450, ACM, 1973.

3. Green Jr, B. F., A. K. Wolf, C. Chomsky and K. Laughery, "Baseball: An Automatic Question-Answerer", *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*, pp. 219–224, ACM, 1961.

4. Mishra, A. and S. K. Jain, "A Survey on Question Answering Systems with Classification", *Journal of King Saud University-Computer and Information Sciences*, Vol. 28, No. 3, pp. 345–361, 2016.

5. Jurafsky, D. and J. H. Martin, *Speech and Language Processing*, Vol. 3, chap. 23, Pearson London, 2018.

6. Androutsopoulos, I., G. Ritchie and P. Thanisch, "Masque/SQL An Efficient and Portable Natural Language Query Interface for Relational Databases", *Database technical paper, Department of AI, University of Edinburgh*, 1993.

7. Burke, R. D., K. J. Hammond, V. Kulyukin, S. L. Lytinen, N. Tomuro and S. Schoenberg, "Question Answering From Frequently Asked Question Files: Experiences with the FAQ Finder System", *AI magazine*, Vol. 18, No. 2, pp. 57–57, 1997.

8. Riloff, E. and M. Thelen, "A Rule-based Question Answering System for Reading Comprehension Tests", *Proceedings of the 2000 ANLP/NAACL Workshop on*

Reading comprehension tests as evaluation for computer-based language understanding sytems-Volume 6, pp. 13–19, Association for Computational Linguistics, 2000.

9. Li, X. and D. Roth, "Learning Question Classifiers", *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pp. 1–7, Association for Computational Linguistics, 2002.

10. Kwok, C., O. Etzioni and D. S. Weld, "Scaling Question Answering to the Web", *ACM Transactions on Information Systems (TOIS)*, Vol. 19, No. 3, pp. 242–262, 2001.

11. Hovy, E., L. Gerber, U. Hermjakob, M. Junk and C.-Y. Lin, "Question Answering in Webclopedia", *TREC*, Vol. 52, pp. 53–56, 2000.

12. Zheng, Z., "AnswerBus Question Answering System", *Proceedings of the second international conference on Human Language Technology Research*, pp. 399–404, Morgan Kaufmann Publishers Inc., 2002.

13. Hirschman, L., M. Light, E. Breck and J. D. Burger, "Deep Read: A reading Comprehension System", *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pp. 325–332, Association for Computational Linguistics, 1999.

14. Ng, H. T., L. H. Teo and J. L. P. Kwan, "A Machine Learning Approach to Answering Questions for Reading Comprehension Tests", *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 13*, pp. 124–132, Association for Computational Linguistics, 2000.

15. Richardson, M., C. J. Burges and E. Renshaw, "Mctest: A Challenge dataset for the Open-domain Machine Comprehension of Text", *Proceedings of the 2013*

Conference on Empirical Methods in Natural Language Processing, pp. 193–203, 2013.

16. Narasimhan, K. and R. Barzilay, "Machine Comprehension with Discourse Relations", *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Vol. 1, pp. 1253–1262, 2015.

17. Wang, H., M. Bansal, K. Gimpel and D. McAllester, "Machine Comprehension with Syntax, Frames, and Semantics", *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, Vol. 2, pp. 700–706, 2015.

18. Rajpurkar, P., R. Jia and P. Liang, "Know What You Don't Know: Unanswerable Questions for SQuAD", *arXiv preprint arXiv:1806.03822*, 2018.

19. Hermann, K. M., T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman and P. Blunsom, "Teaching Machines to Read and Comprehend", *Advances in neural information processing systems*, pp. 1693–1701, 2015.

20. Weston, J., A. Bordes, S. Chopra, A. M. Rush, B. van Merriënboer, A. Joulin and T. Mikolov, "Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks", *arXiv preprint arXiv:1502.05698*, 2015.

21. Turmo, J., P. R. Comas, C. Ayache, D. Mostefa, S. Rosset and L. Lamel, "Overview of QAST 2007", *Workshop of the Cross-Language Evaluation Forum for European Languages*, pp. 249–256, Springer, 2007.

22. Comas, P. R., J. Turmo and L. Màrquez, "Sibyl, A Cactoid Question-Answering System for Spoken Documents", *ACM Transactions on Information Systems (TOIS)*, Vol. 30, No. 3, p. 19, 2012.

23. Shiang, S.-R., H.-y. Lee and L.-s. Lee, "Spoken Suestion Answering Using Tree-Structured Conditional Random Fields and Two-layer Random Walk", *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

24. Tseng, B.-H., S.-S. Shen, H.-Y. Lee and L.-S. Lee, "Towards Machine Comprehension of Spoken Content: Initial TOEFL Listening Comprehension Test by Machine", *arXiv preprint arXiv:1608.06378*, 2016.

25. Fang, W., J.-Y. Hsu, H.-y. Lee and L.-S. Lee, "Hierarchical Attention Model for Improved Machine Comprehension of Spoken Content", *2016 IEEE Spoken Language Technology Workshop (SLT)*, pp. 232–238, IEEE, 2016.

26. Li, C.-H., S.-L. Wu, C.-L. Liu and H.-y. Lee, "Spoken SQuAD: A Study of Mitigating the Impact of Speech Recognition Errors on Listening Comprehension", *arXiv preprint arXiv:1804.00320*, 2018.

27. Lee, C.-H., S.-M. Wang, H.-C. Chang and H.-Y. Lee, "ODSQA: Open-Domain Spoken Question Answering Dataset", *2018 IEEE Spoken Language Technology Workshop (SLT)*, pp. 949–956, IEEE, 2018.

28. Cao, J., J. A. Robles-Flores, D. Roussinov and J. F. Nunamaker, "Automated Question Answering From Lecture Videos: NLP vs. Pattern Matching", *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, pp. 43b–43b, IEEE, 2005.

29. Cao, J. and J. F. Nunamaker, "Question Answering on Lecture Videos: A Multifaceted Approach", *Proceedings of the 2004 Joint ACM/IEEE Conference on Digital Libraries, 2004.*, pp. 214–215, IEEE, 2004.

30. Li, G., Z. Ming, H. Li and T.-S. Chua, "Video Reference: Question Answering on YouTube", *Proceedings of the 17th ACM international conference on Multimedia*, pp. 773–776, ACM, 2009.

31. Yang, H., L. Chaisorn, Y. Zhao, S.-Y. Neo and T.-S. Chua, "VideoQA: Question Answering on News Video", *Proceedings of the eleventh ACM international conference on Multimedia*, pp. 632–641, ACM, 2003.

32. İlhan, S., N. Duru, Ş. Karagöz and M. Sağır, "Metin Madenciliği ile Soru Cevaplama Sistemi", *Elektronik ve Bilgisayar Mühendisliği Sempozyumu (ELECO), Bursa*, pp. 26–30, 2008.

33. Er, N. P. and I. Cicekli, "A Factoid Question Answering System Using Answer Pattern Matching", *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pp. 854–858, 2013.

34. Derici, C., K. Celik, E. Kutbay, Y. Aydın, T. Güngör, A. Özgür and G. Kartal, "Question Analysis for A Closed Domain Question Answering System", *International Conference on Intelligent Text Processing and Computational Linguistics*, pp. 468–482, Springer, 2015.

35. Devlin, J., M.-W. Chang, K. Lee and K. Toutanova, "Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding", *arXiv preprint arXiv:1810.04805*, 2018.

36. Seo, M., A. Kembhavi, A. Farhadi and H. Hajishirzi, "Bidirectional Attention Flow for Machine Comprehension", *arXiv preprint arXiv:1611.01603*, 2016.

37. Xiong, C., V. Zhong and R. Socher, "Dynamic Coattention Networks for Question Answering", *arXiv preprint arXiv:1611.01604*, 2016.

38. Wang, W., N. Yang, F. Wei, B. Chang and M. Zhou, "R-NET: Machine Reading Comprehension with Self-Matching Networks", *Natural Lang. Comput. Group, Microsoft Res. Asia, Beijing, China, Tech. Rep*, Vol. 5, 2017.

39. Wang, S. and J. Jiang, "Machine Comprehension Using Match-lstm and Answer Pointer", *arXiv preprint arXiv:1608.07905*, 2016.

40. Dai, Z., Z. Yang, Y. Yang, W. W. Cohen, J. Carbonell, Q. V. Le and R. Salakhutdinov, "Transformer-XL: Attentive Language Models Beyond A Fixed-length Context", *arXiv preprint arXiv:1901.02860*, 2019.

41. Pennington, J., R. Socher and C. Manning, "Glove: Global Cectors for Word Representation", *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

42. Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado and J. Dean, "Distributed Representations of Words and Phrases and Their Compositionality", *Advances in neural information processing systems*, pp. 3111–3119, 2013.

43. Arısoy, E., *Statistical and Discriminative Language Modeling for Turkish Large Vocabulary Continuous Speech Recognition*, Ph.D. Thesis, Citeseer, 2009.

44. Wang, S. and J. Jiang, "Learning Natural Language Inference with LSTM", *arXiv preprint arXiv:1512.08849*, 2015.

45. Vinyals, O., M. Fortunato and N. Jaitly, "Pointer Networks", *Advances in Neural Information Processing Systems*, pp. 2692–2700, 2015.

46. Hochreiter, S. and J. Schmidhuber, "Long Short-Term Memory", *Neural computation*, Vol. 9, No. 8, pp. 1735–1780, 1997.

47. Elman, J. L., "Finding Structure in Time", *Cognitive science*, Vol. 14, No. 2, pp. 179–211, 1990.

48. Gers, F. A. and J. Schmidhuber, "Recurrent Nets that Time and Count", *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, Vol. 3, pp. 189–194, IEEE, 2000.

49. Cho, K., B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk

and Y. Bengio, "Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation", *arXiv preprint arXiv:1406.1078*, 2014.

50. Yao, K., T. Cohn, K. Vylomova, K. Duh and C. Dyer, "Depth-gated Recurrent Neural Networks", *arXiv preprint arXiv:1508.03790*, Vol. 9, 2015.

51. Mangu, L., E. Brill and A. Stolcke, "Finding Consensus in Speech Recognition: Word Error Minimization and Other Applications of Confusion Networks", *Computer Speech & Language*, Vol. 14, No. 4, pp. 373–400, 2000.

52. Paszke, A., S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga and A. Lerer, "Automatic Differentiation in PyTorch", *NIPS-W*, 2017.

53. Kingma, D. P. and J. Ba, "Adam: A Method for Stochastic Optimization", *arXiv preprint arXiv:1412.6980*, 2014.

54. Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks From Overfitting", *The Journal of Machine Learning Research*, Vol. 15, No. 1, pp. 1929–1958, 2014.

55. Povey, D. *et al.*, "The Kaldi Speech Recognition Toolkit", *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, IEEE Signal Processing Society, Dec. 2011.