University of Tunis El Manar

National Engineering School of Tunis

Industrial Engineering Department

Modeling for the Industry and Services Section

Graduation Project Report

Project Topic:

Building a Language Model for Tunisian Dialect Using NLP Based Techniques

Host Company:

InstaDeep™

Student :
Ms. Nourchene FERCHICHI

Supervisors :
Mr. Amine KERKENI
Ms. Eya RHOUMA
Mr. Azmi MAKHLOUF

Academic Year: 2018-2019

Summary — The Tunisian dialect is significantly different from the formal Arabic language. However, despite its complicated written forms, it is the language used in the everyday communications and in most Tunisian social media.
From here came the challenge of creating a model that can understand this dialect by testing it on a specific Natural Language Processing (NLP) task. In our use case, we propose to train our model on a ”Multi-Turn Context Response Selection” NLP task. We propose to use the ”Deep Attention Matching” (DAM) [14] model, a recently introduced model based entirely on ”Attention”.

Key Words : Deep Learning, Natural Language Processing (NLP), Multi-Turn Context Response Selection, Word Embedding, Deep Attention Matching(DAM), Attention, Tunisian Dialect.

Résumé — Le dialecte Tunisien est très différent de la langue arabe enseignée dans les écoles et utilisée dans les écrits formels. Cependant, malgré ses formes écrites compliquées, il est la langue utilisée dans les communications quotidiennes et dans la plupart des médias sociaux Tunisiennes.
D’où le défi de créer un modèle capable de comprendre ce dialecte en le testant sur une tâche spécifique de Traitement du Langage Naturel (NLP). Dans notre cas, nous proposons de former notre modèle pour la tâche de NLP ”Sélection de réponse de contexte à plusieurs tours”. Nous proposons d’utiliser le modèle du Mécanisme d’Attention Approfondie (DAM) [14], un modèle récemment introduit, entièrement basé sur l’ ”Attention”.

Mots Clés : Apprentissage en profondeur, Traitement du Langage Naturel (NLP), Sélection de réponse de contexte à plusieurs tours, Incorporation de mots, Mécanisme d’Attention Approfondie (DAM), Attention, dialecte Tunisien.

ملخص -- تختلف اللهجة التونسية اختلافاً كبيراً عن اللغة العربية التي يتم تدريسها في المدارس وتستخدم في الكتابات الرسمية. ومع ذلك ، على الرغم من أشكالها المكتوبة المعقدة ، فهي اللغة المستخدمة في الاتصالات اليومية وفي معظم وسائل الإعلام التونسية الاجتماعية.

من هنا جاء التحدي المتمثل في إنشاء نموذج يمكنه فهم هذه اللهجة من خلال اختباره في مهمة محددة لمعالجة اللغة الطبيعية. في حالتنا ، نقترح تدريب نموذجنا على مهمة "تحديد استجابة السياق المتعددة". نقترح استخدام "مطابقة الاهتمام العميق" ، وهو نموذج تم تقديمه مؤخراً يعتمد بالكامل على "لاهتمام".

الكلمات الدالة : التعلم العميق ، معالجة اللغة الطبيعية ، اختيار استجابة السياق متعدد الأدوار ، تضمين الكلمات ، مطابقة الاهتمام العميق ، الاهتمام ، اللهجة التونسية.

Company Supervisor Appreciation

Academic Supervisor Appreciation

Acknowledgement

First of all, I would like to thank the whole pedagogical team of the National Engineering School of Tunis (ENIT), and especially the professors of the Modeling for the Industry and Services (MIndS) section for ensuring the needed theoretical part of this work during my internship period.

Then, I would like to express my gratitude to the following persons for the interesting experience they gave me during my internship: Mr. Amine KERKENI, Head of AI Product at InstaDeep, for the trust he gave me since my arrival to the company, ensuring the smooth running of the internship, Ms. Eya RHOUMA, AI Engineer at InstaDeep, for the time she devoted to me throughout this period and for her precious help, ensuring the completion of my required tasks. I would also like to thank Mr. Azmi MAKHLOUF Associate Professor at ENIT, for the valuable advice he offered me, always knowing how to answer my questions.

Finally, I would like to thank the entire InstaDeep's AI Team for their generosity and group-spirit. I was deeply touched by the sense of collaboration and the love of knowledge sharing that marks the work atmosphere. I also appreciated the professional climate prevailing between the different units' member, as well as between the different departments of the company.

Contents

List of Figures **9**

List of Tables **10**

Introduction **11**

1 Project Framework **12**

Introduction 12

1.1 Company Presentation 12

1.1.1 About InstaDeep 12

1.1.2 Main Services 13

1.1.3 Main Values 14

1.2 Project Setting 14

1.2.1 Natural Language Processing Challenges 14

1.2.2 Tunisian Dialect Challenges 15

1.2.3 Project Final Goals 16

Conclusion 17

2 state-of-the-art **18**

Introduction 18

2.1 Multi-Layer Perceptrons 18

2.1.1 Network Architecture 18

2.1.2 Network Strong Points 19

2.1.3 Network Drawbacks 19

2.2 Recurrent Neural Networks 20
2.2.1 Network Architecture 20
2.2.2 Network Strong Points 21
2.2.3 Network Drawbacks 21
2.3 Long Short-Term Memory 22
2.3.1 Network Architecture 22
2.3.2 Network Strong Points 23
2.3.3 Network Drawbacks 24
2.4 Attention Based Networks 24
2.4.1 Network Architecture 24
2.4.2 Attention Module 25
2.4.3 Network Strong Points 26
2.5 Summary 26
Conclusion 28

3 Proposed Solution **29**
Introduction 29
3.1 Multi-turn Context Response Selection Task 29
3.1.1 Automatic Response Selection 29
3.1.2 Problem Formalization 30
3.2 Deep Attention Matching Model 31
3.2.1 Model Overview 31
3.2.2 Word Embedding Module 32
3.2.3 Representation Module 33
3.2.4 Matching Module 34
3.2.5 Aggregation Module 35
Conclusion 37

4 Results and Analysis **38**
Introduction 38
4.1 Tunisian Dialect Dataset 38

4.1.1 Dataset Collection 38

4.1.2 Dataset Pre-processing 40

4.1.3 Dataset Statistics 42

4.2 Deep Attention Matching Model 44

4.2.1 Word Embedding 44

4.2.2 Model Training and Validation 53

4.2.3 Model Test 55

Conclusion 59

Bibliography **63**

List of Figures

1.1 The location of the four InstaDeep offices. 13

2.1 The Multilayer Perceptrons architecture. 19

2.2 The Recurrent Neural Networks architecture. 20

2.3 The Recurrent Neural Networks unrolled architecture. 20

2.4 The Long Short-Term Memory network architecture. 22

2.5 The Long Short-Term Memory hole network architecture. 23

2.6 An example for a response-answer with Key, Value and Query inputs. . . . 24

2.7 The Attention architecture. 25

2.8 The Attention module 26

3.1 Example of multi-turn context conversation. 30

3.2 The Deep Attention Matching model. 31

3.3 The Word2Vec model architecture. 32

3.4 The Representation module. 33

3.5 The Matching module. 35

3.6 The Aggregation module. 36

4.1 Example from the row collected OoredooTn data. 39

4.2 Train example from OoredooTn dataset. 40

4.3 Validation example from OoredooTn dataset. 41

4.4 Test example from OoredooTn dataset. 41

4.5 OoredooTn dataset statistics 43

4.6 Example from the created vocabulary document. 44

4.7 A 200-d vector representation of a Tunisian word. 45
4.8 A 3-d vector representation of the whole vocabulary words using PCA. . . 45
4.9 A 2-d vectors representation of the whole vocabulary words. 46
4.10 Example of translated words. 46
4.11 Extracted and annotated vocabulary words 2-d representation 47
4.12 20 chosen vocabulary words clusters. 48
4.13 T-SNE first iterations. 49
4.14 Clusters of similar words using T-SNE. 49
4.15 Four extracted word clusters. 50
4.16 Unfixed word spelling. 50
4.17 Latin and Arabic alphabet words. 50
4.18 Technical information related to words. 50
4.19 Misspelled words. 51
4.20 Misspelled words. 51
4.21 Four extracted clusters from the final iteration. 52
4.22 The Recalls with respect to the number of stacked Attention layers. 54
4.23 The loss function with respect to the number of steps. 54
4.24 The Learning Rate with respect to the number of steps. 55
4.25 The Recalls with respect to the maximum number of turns per context. . . 56
4.26 The Recalls with respect to the maximum number of words per turn. . . . 57
4.27 Different experiments. 58

List of Tables

1.1 Publically available Tunisian datasets. 16

2.1 Comparing MLP, RNN, LSTM and Attention networks 27

4.1 A selection of open-source datasets for dialogue systems. The last entry is our contribution. 42
4.2 Statistics of open-source datasets for dialogue systems. 42
4.3 Answer-response system dataset different Recall experimental results. . . . 58
4.4 Answer-response system dataset best Recall results. 59

Introduction

The internship is an essential element in the curriculum of an engineering student. It offers them the opportunity to face the everyday problems, an engineer could face. Adding to that, it helps them establish a link between what they must learn at school and what they need to apply in their future professional career.

As part of my studies in the third year of the Modeling for the Industry and Services (MIndS) section, at the National Engineering School of Tunis (ENIT), I realized my graduation internship project at InstaDeep. This internship started on February 1^{st} 2019 and ended on Mai 31^{st} 2019. It was held in the AI Team department and was supervised by Mr. Amine KERKENI, Head of AI Product at InstaDeep and by Mr. Azmi MAKHLOUF Associate Professor at ENIT.

It seemed to me very opportune to do my internship at InstaDeep, a leader in the field of Artificial Intelligence. Its world renown was built thanks to the diversity of its in-house products and the specialization in its areas of application. This explains the great importance that the company attaches to the continuous improvement of its work. In this context, the goal of my internship at InstaDeep was to build a language model for Tunisian dialect.

This report starts with a presentation of the framework. In this chapter, we introduce the host company. A part of the chapter will be devoted to the introduction of the project's role. In the next chapter, we present the state-of-the-art neural network commonly used for building automated language models. Then, we introduce the proposed solution from a theoretical perspective. In the final chapter, we presented the collected, cleaned and organised Tunisian corpus. We illustrate then the results of the proposed method and validate it by training and testing it over the collected Tunisian dialect dataset. A conclusion at the end will summarize the work.

Chapter 1

Project Framework

Introduction

As a Modeling for the Industry and Services (MIndS) student, Artificial Intelligence was the attractive field of study that kept my attention and that aroused my curiosity. Having studied applied mathematics in the industry field, the AI project proposed by the host company, InstaDeep was a targeted one for me. The objective of the proposed graduation project was to build a language model for the Tunisian dialect and to test it over a chosen NLP task based on a collected Tunisian dataset.

1.1 Company Presentation

1.1.1 About InstaDeep

InstaDeep is an African AI start up that was first launched in Tunisia. In fact, it was founded in October, 2014 by Mr. Karim BEGUIR and Ms. Zohra SLIM. The start-up first began with few but very enthusiastic engineers. Very quickly the company reached a number of more than 70 employees with four new offices in London-UK, Paris-France, Nairobi-Kenya and most recently Lagos-Nigeria, as shown in Figure 1.1.

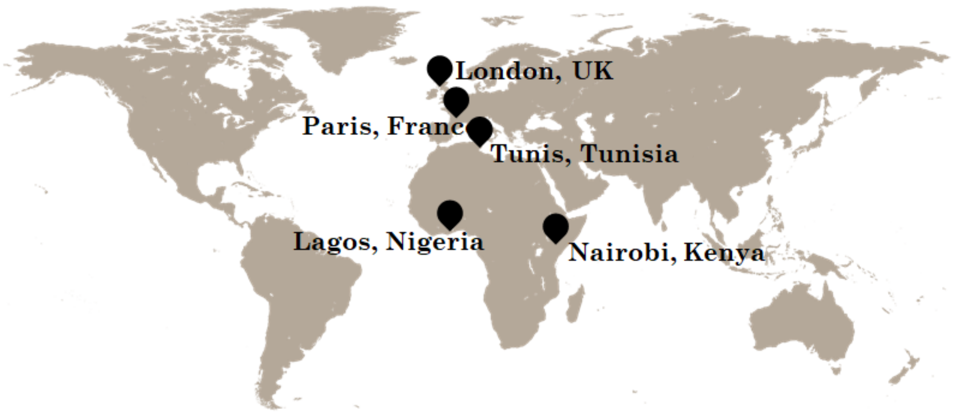


Figure 1.1: The location of the four InstaDeep offices.

1.1.2 Main Services

InstaDeep works on the research and development in the AI field. It develops in-house products and solutions for enterprises. It offers many AI solutions, ranging from self-learning decision making systems, GPU-accelerated insights, to optimized pattern-recognition :

- Decision-making systems: InstaDeep uses the power of reinforcement learning to create systems that can make decisions on their own, based on their own autonomous training. Many fields can benefit greatly from this technology such as : robotics, mobility, logistics, finance and healthcare.

- GPU-accelerated insights: InstaDeep helps companies achieve computing power to solve intensive AI problems. It offers the use of one of the most powerful AI machines on the market: Nvidia's DGX1.

- Optimized Deep Learning: InstaDeep boosts the deep learning process to save companies time and money. The optimized deep learning can be applied on computer vision, natural language processing and predictive analytical projects.

1.1.3 Main Values

InstaDeep aims to build and enlarge its research community. In fact, it has published two research papers which were presented at the Neural Information Processing Systems (NeurIPS) conference [1] last year, in 2018.

The company believes in the democratization of AI by making it accessible to the public. This is shown through its participation and sponsorship of many AI fruitful events all over the word. We can site the example of IndabaX [2] whose first edition in Tunisia was held with great success on April 13^{th} 2019.

The AI start-up pushes the creativity and self-development of its employees. In fact, it encourages their participation in worldwide competitions. We can site the examples of winning the First Prize Global Facebook Hackathon Competition [3]and the Prediction Competition and Uber Movement Prize by two Tunisian InstaDeepers [4].

1.2 Project Setting

1.2.1 Natural Language Processing Challenges

Natural language processing (NLP) is a sub-field of Artificial Intelligence. It aims at raising the intelligence level of computers in terms of understanding the human natural languages. Important applications of NLP include two major categories: the recognition tasks and the generation tasks. Examples of the recognition tasks include: sentiment analysis, paraphrase detection, and question answer selection. The generation tasks include: machine translation, next word prediction, and text summarising. The solution to those problems lies in the way to properly represent sequences (words, phrases and eventually context paragraphs) and the ability to robustly represent the interactions between them.

[1]The Conference on Neural Information Processing Systems (NIPS) is a machine learning and computational neuroscience conference that is held every December. The conference includes invited talks from all over the world, workshops, paper submissions as well as oral and poster presentations.

[2]The IndabaX is a locally-organised, one-day event that helps spread knowledge and builds capacity in deep learning. It is an opportunity for the AI community to meet, discover the most recent advances in the field and interact with world-leading AI researchers. [3]

[3]Hackathons are a big tradition at Facebook. They serve as the foundation for some great (and not so great) ideas. It gives the employees the opportunity to try out new ideas and collaborate with other people in a fun environment.

[4]The competition, sponsored by Uber and Mobiticket, tasked the contestants with creating a predictive model that could forecast the number of bus ticket sold for buses into Nairobi from cities in up-country Kenya.

The problem of mapping from one sequence to another is an important challenge of Natural Language Processing. Traditionally, NLP problems have been tackled by a combination of hand-crafted features and external linguistic resources. While these models have achieved encouraging results, they suffer from three major drawbacks: First, such models rely on a significant amount of feature engineering and each model needs to be tuned separately. Second, the external linguistic resources are very expensive to build, particularly for resource-low languages. Third, it is difficult to adapt such models to new domains, so we need to separate feature extraction and resource development for each domain.

At the same time, neural networks have been available for decades, but not until recently explored. Their introduction is being more and more used in large sequence mapping problems as they are achieving promising results in many fields. Different types of neural networks were introduced to tackle the challenge of mapping sequences such as Multi-Layer Perceptron (MLP), Recurrent Neural Network (RNN), Long Short-term Memory (LSTM) and Attention based models. Each one was introduced in order to solve some discovered issues related to the networks structure. Deeper information about those neural networks will be introduced in the next chapter.

1.2.2 Tunisian Dialect Challenges

Language Complexity

Tunisian dialect is significantly different from the Arabic language taught in schools and used in formal writings. It is very complicated and challenging when it comes to understanding its written forms. What distinguishes this dialect from any other written formal language is the use of both Latin and Arabic alphabet. We add to that the mixed words with number characters coding some letters. Another point to add is the unfixed spelling in most of the words, as no spelling rules are applied. We also noticed the presence of foreign languages (such as French and English). Not to forget the huge amount of misspelling when it comes to foreign languages.

Despite the complex aspect of the Tunisian dialect, it remains the language used in the everyday communications and in most social media in Tunisia. In fact, in recent years, we noticed increasing use of Tunisian Arabic in interviews, news and debate programs, as well as in tweets, posts and Facebook comments.

dataset Availability

The increasing need to create automatic processing models for languages requires a huge amount of well-designed language datasets. We observe that in several NLP tasks, breakthroughs were achieved with formal written languages corpora (English, French ...). Similar progress has not yet been observed in the development of NLP results with informal languages (Dialects). We hypothesize that this is due to the lack of sufficiently large datasets.

For a more concrete image, let's briefly review existing Tunisian corpora. A list of freely available datasets is provided in Table 1.1 which are: The Tunisian Sentiment Analysis Corpus (TSAC) [5] [12], the Tunisian Arabic Corpora (TAC)[6] [6], the Tunisian Dialect and Modern Standard Arabic Dataset (TDMSAD) [8].

Corpus	Size	Purpose	Source
TSAC	17K comments	Sentiment analysis for linguistic experiments	Facebook pages
TAC	4M words	Linguistic experiments	SMS and Facebook pages
TDMSAD	5K tweets	Sentiment analysis for election context	Tweeter

Table 1.1: Publically available Tunisian datasets.

From the table, we notice the weak availability of Tunisian corpora. In fact, very few freely available data can be found. Moreover, the already existing datasets are designed for sentiment analysis and linguistic experiments. So there is a lack of data for different NLP tasks. This makes the Tunisian dialect a challenging topic to apply an automatic processing model on.

1.2.3 Project Final Goals

From the NLP challenges and the Tunisian dialect challenges came the idea of building a model, based on neural networks, that can understand our dialect. Then, this model will be tested on a chosen Natural Language Processing task.

In this graduation project we were required to:

[5]This corpus is freely available for research purpose on: https://github.com/fbougares/TSAC
[6]This corpus is freely available for research purpose on: http://www.tunisiya.org/

- Collect and annotate a Tunisian dialect corpus. The corpus will be valuable to researchers working in the field of Tunisian language processing models and similar areas. The corpus will also be used by InstaDeep engineers in their future AI projects.
- Deliver a model, inspired by the recent advances in NLP, capable of understanding the Tunisian dialect.
- Train the model on a chosen NLP task.
- Deliver test results, revealing how far the model can understand the Tunisian dialect.

Conclusion

In this chapter, we presented the framework of the project. We first had a look at the host company and focused on its main services and values. Then, we introduced the proposed problematic by showing the challenges of both Natural Languages Processing and the Tunisian dialect. We finally pointed the deliverable of the graduation project.

In order to achieve the required tasks of the project, an overview of the state-of-the-art neural networks is essential. This will provide as with a general understanding of the used models for sequential data mapping problems. A comparison between the proposed models is very important to help us choose the suitable solution for our project. This will be handled in the next chapter.

Chapter 2

state-of-the-art

Introduction

In this chapter, we investigate state-of-the-art neural networks commonly used for sequential data. Neural networks can be viewed as a general class of parametric nonlinear functions F that approximate a true function F^*. Our function F is defined with a parameter θ, from input variables x to output variables y such as:

$$y = F(x, \theta) \tag{2.1}$$

We aim to determine this parameter θ that best approximates the true function F^* over all the input-output pairs, via a numerical optimisation method.

2.1 Multi-Layer Perceptrons

2.1.1 Network Architecture

Multi-Layer Perceptrons (MLPs) are the fundamental types of artificial neural networks. They are called feed-forward neural networks as they are built with no feedback connections.

The structure of MLPs is characterized by a directed acyclic graph, as shown in Figure 2.1. The bottom layer is called the *Input Layer*. The topmost layer is known as the *Output Layer*. The layers between the Input and Output Layers are named the *Hidden Layers*. The arrows between layers represent connections known as *Activations* and are passed from lower layers to their adjacent higher layers.

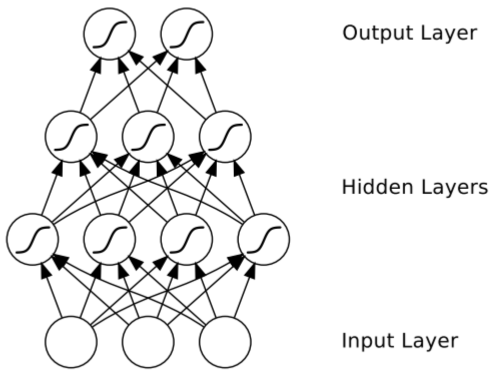


Figure 2.1: The Multilayer Perceptrons architecture [4].

The input x is presented to the input layer. Its outputs are successively activated through the hidden layers until the output y is generated from the output layer. The output value of each neuron is calculated by applying the activation function f to the weighted sum of the outputs of the neurons in the previous layer.

2.1.2 Network Strong Points

One dominant feature of MLPs is their ability to learn hierarchical representations, which models based on hand-crafted features and external linguistic resources can not learn well. This ability is due to neural networks' structure represented by a directed acyclic graph.

In addition to that, such neural network models can be trained with minimal domain knowledge, which makes their adaptation to new domains easily applied.

2.1.3 Network Drawbacks

The MLP network cannot capture orderings efficiently. For many NLP tasks, what distinguishes such problems from the other tasks is that the individual data points cannot be assumed to be independent. Instead, both the inputs and the outputs are strongly correlated sequences. So, in practice, it is necessary to consider orderings and dependencies.

For example, in language modelling, the prediction of the next word task extensively relies on the words before it. Another example is the selection of correct replies to specific questions, which depends heavily on the hole context.

In fact, traditional MLPs can only map from current input to current output vectors. Therefore, it is impossible for such models to memorise previous inputs in order to generate dependent outputs.

2.2 Recurrent Neural Networks

2.2.1 Network Architecture

Recurrent Neural Networks (RNNs) are a class of artificial neural networks that allows cyclical connections between their layers [4]. The RNNs can be seen as neural networks with loops in them. The RNNs' compact model is illustrated in the Figure 2.2.

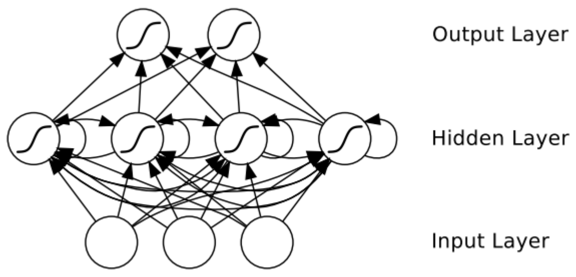


Figure 2.2: The Recurrent Neural Networks architecture [4].

To better understand the network, we can look at its unrolled version, illustrated in the Figure 2.3.

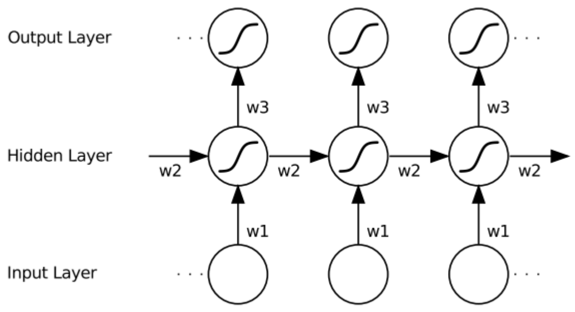


Figure 2.3: The Recurrent Neural Networks unrolled architecture [4].

The model architecture of RNNs is the same as MLPs' with a single hidden layer, except that activations arrive at the hidden layer from both the current external input and the hidden layer activations from the previous time-step.

First, the model takes the input at time-step t, x_t from the sequence and then it outputs h_t. This previous output h_t together with input at the next time-step $t+1$, x_{t+1} are the new input for the next step. Similarly, the output of the previous time-step h_{t+1} and the new input at the step $t+2$, x_{t+2} are the inputs for the next step and so on.

2.2.2 Network Strong Points

RNNs are simple and powerful models for sequence modelling thanks to their cyclical connections, as this property has very enriching implications for sequence learning. In fact, an RNN can map from the entire history of previous inputs to each output, which MLPs networks are not able to do.

This fundamental propriety is due to the recurrent connections. Those specific connections allow the information of previous inputs to persist in the network's internal state, and thereby influence the network output.

Besides, RNNs have several properties that make them an attractive choice for tasks using sequences. First, they are flexible in their use of context information. Second, they accept many different types and representations of data and third, they can recognise sequential patterns.

2.2.3 Network Drawbacks

While the cyclical connections of recurrent neural networks made them a powerful model for sequence modelling tasks, in practice it is challenging to train them well. This is largely due to the difficulty of learning long-range dependencies. As a consequence, it is not preferable to get them to store information for long periods of time.

Two of the major reasons for this specific difficulty are the problems of: exploding and vanishing gradients. Exploding gradients refers to the situation where gradients increase exponentially during back-propagation, making learning diverge. Conversely, vanishing gradients happens when gradients decrease exponentially fast towards zero, making learning extremely slow or even stop.

Those two problems happen when we back-propagate errors across the network layers many time steps. As a result, the influence of a given input on the hidden layer, and therefore on the network output, either decays or blows up exponentially while it cycles around the network's recurrent connections. This limits the range of context RNNs can access, which is of critical importance to multiple NLP tasks.

2.3 Long Short-Term Memory

2.3.1 Network Architecture

Long Short-Term Memory networks (LSTMs), are a special class of RNNs [5]. Their architecture consists of a set of recurrently connected sub-nets, known as memory blocks. Each block contains one or more self-connected memory cells and three multiplicative units: the Input Gate, Output Gate, and Forget Gate, as shown in Figure 2.4.

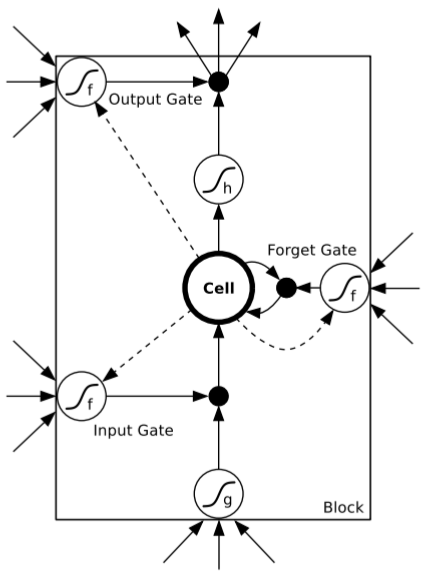


Figure 2.4: The Long Short-Term Memory network architecture [5].

The three gates are nonlinear summation units that collect activations from inside and outside the block, and control the activation of the cell via multiplications. The Input and Output Gates multiply the input and output of the Cell while the Forget Gate multiplies the Cell's previous state. No activation function is applied within the cell. The gate activation function f is a sigmoïd function, while the gate activation functions g and h are tanh functions.

The only outputs from the block to the rest of the network emanate from the Output Gate multiplication. An over view of the hole LSTMs' network can be illustrated in the Figure 2.5 below.

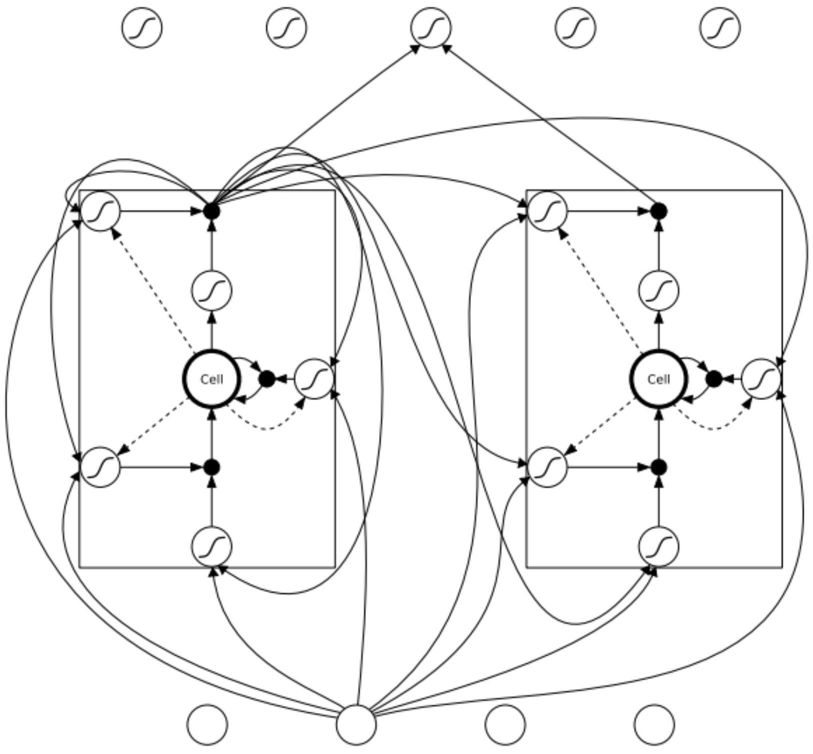


Figure 2.5: The Long Short-Term Memory hole network architecture [5].

2.3.2 Network Strong Points

LSTMs came to tackle one of most frequent RNNs' problems, previously explained: the vanishing or exploding gradient problems. This network is one of the most widely used and effective architecture for such problems.

The key idea of the LSTMs is to replace each ordinary node in the hidden layer of a standard RNN with a memory cell. Each memory cell contains a node with a self-recurrent connection, ensuring that gradients flow smoothly through time. In addition, there are gating units controlling the reading, writing and resetting of the cells.

In various NLP tasks, LSTMs have shown capacity of storing and accessing information over long sequences which is very important when it comes to sequence learning tasks.

2.3.3 Network Drawbacks

LSTMs have been firmly established, for a long time, as state of the art approaches in sequence modeling. Unfortunately, as the demand on networks' memory has increased for many NLP tasks, even those models struggle as very long sequences are required.

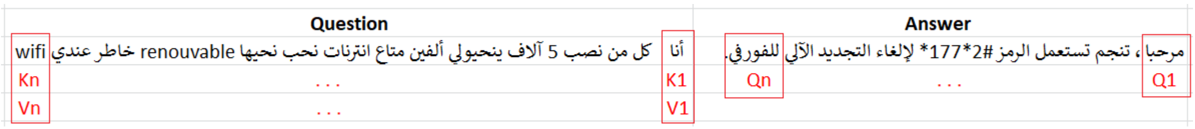
In fact, LSTMs typically factor time computation: The long the input sequences is, the more the model consumes time while computing. And so, with very long data sequences, solving complex NLP problems can be prohibitively time-consuming to process with LSTM networks.

Moreover, the quality of LSTMs deteriorates dramatically as the lengths of the source sentence increases. To tackle this problem, a big network is needed to achieve good performances. However, in practice, this requires a large amount of computational resources.

2.4 Attention Based Networks

2.4.1 Network Architecture

An Attention function can be described as mapping a set of (Key, Value) pairs and a Query to an output. More concretely, for a given source sentence, each input word is represented by pairs of (Key, Value). The Key is the address of the word and the Value is the actual element. The output word in the target sentence is the Query. For a response-answer task an example is illustrated in Figure 2.6.



Question			Answer		
wifi	كل من نصب 5 آلاف ينحيولي ألفين متاع انترنات نحب نحيها renouvable خاطر عندي	أنا	للفورقي.	تنجم تستعمل الرمز #2*177* لإلغاء التجديد الآلي	مرحبا،
Kn	. . .	K1	Qn	. . .	Q1
Vn	. . .	V1			

Figure 2.6: An example for a response-answer with Key, Value and Query inputs.

Through the calculation of similarity between Query and each Key, we can get the attention score of the Value, corresponding to the Key. Attention score represents the importance of an input word. We then multiply each value vector by the attention score and sum up the weighted value vector, which is the Attention vector, as shown in Figure 2.7.

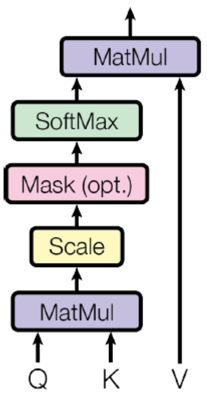


Figure 2.7: The Attention Architecture [1].

2.4.2 Attention Module

We introduce the network equations [1]. The input consists of Queries and Keys of dimension d_k, and Values of dimension d_v. In practice, we compute the Attention function on a set of queries simultaneously, packed together into a matrix of sentences Q. The keys and values are also packed together into matrices of sentences K and V, respectively.

We compute the matrix of outputs as defined in equation 2.2: The Attention function first takes each word in the query sentence Q to attend to words in the key sentence K via a dot product. We then, divide each by $\sqrt{d_k}$, and apply a Softmax function to obtain the weights on the value sentence V.

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.2}$$

A layer normalization operation is then applied, which prevents vanishing or exploding of gradients. A feed-forward network (FFN) with Rectified Linear Units (ReLU) activation is then applied upon the normalization result, in order to further process the fused embeddings, defined in equation (2.3):

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{2.3}$$

where, x is a 2D-tensor in the same shape of query sentence Q and W_1, b_1, W_2, b_2 are learnt parameters. The result $FFN(x)$ is a 2D-tensor that has the same shape as x.

FFN(x) is then residually added to x, and the fusion result is then normalized as the final outputs. We refer to the whole Attentive Module as: $AttentionModule(Q, K, V)$ represented in Figure 2.8 below.

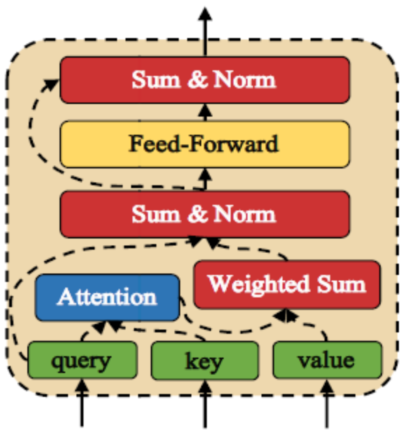


Figure 2.8: The Attention module [15].

2.4.3 Network Strong Points

Attention based networks have recently shown good results in compelling sequence modeling at various NLP tasks. This is thanks to their superior ability to capture semantic dependencies. With this advantage, Attention allows modeling of dependencies without regard to their distance in the input or output sequences.

One other dominant feature of attention based networks is that they are very fast in training and predicting. Along with the sequential nature of many NLP tasks, if we align the positions to steps, those tasks would be very computationally time-consuming. So Attention based networks can be a solution thanks to the possibility of fully parallelizing their computation.

Another thing is that Attention is more space-efficient in practice, since it can be implemented using highly optimized matrix multiplication code. Those superior advantages have inspired us to use the Attention mechanism for the sequence mapping NLP task.

2.5 Summary

The Table 2.1 represents a comparison between the previously cited networks.

	Strong Points	Drawbacks
MLP	- Directed acyclic-graph structure: Ability to learn hierarchical representations. - Ability to be trained with minimal domain knowledge: Easy adaptation to new domains.	- Built with no feedback connections: Can only map from current input to current output. - Inability to capture orderings efficiently: Impossible to memorize previous inputs in order to generate dependent outputs.
RNN	- Cyclic-graph structure: Ability to learn from the entire history of previous inputs. - Flexibility in their use of context information. - Acceptance of many different types of data.	- Difficulty to learn long-range dependencies: Not adequate for storing information for long sequences. - Exploding gradients: Learning diverge. - Vanishing gradients: Learning extremely slows or even stop.
LSTM	- Containing a memory cell: Gradients flow smoothly through time. - Capacity of storing information over long sequences: Avoiding Exploding and Vanishing gradient.	- Factoring computation along the input and output size: Very large data are time consuming to process. - Quality deteriorates dramatically as the lengths of sequences increase. - Big network is needed to achieve good performance: Need to large computational resources
Attention	- Ability to capture semantic dependencies: Adequate for sequence related tasks. - Possibility to fully parallelize the computation: Less time consumption. - Ability to be implemented using highly optimized matrix multiplication code: Space-efficient in practice.	- With very large paragraphs, the quality can deteriorate

Table 2.1: Comparing MLP, RNN, LSTM and Attention networks

Based on the previously done comparison between MLP, RNN, LSTM, and Attention, we came to the conclusion that for our future task, the most adequate model on top of which the model will depend is the Attention model.

Conclusion

In this part of the report, we provided an overview of Artificial Neural Networks commonly used for sequence mapping tasks. We began by describing the architecture of MPLs, one of the most basic fully connected feed-forward neural networks. We then introduced RNNs, a cyclic type of neural networks. We also presented the LSTMs, a specific version of recurrent networks. And then, we introduced the Attention based neural networks.

Finally, based on a comparison between the stated networks, we will choose the Attention based network on which our final model will fully depend. This model, known as, Deep Attention Mechanism (DAM), will be our proposed solution. We will introduce it for solving a specific chosen NLP task which is the ”Multi-turn Context Response Selection”. More details about this task will be introduced in the next chapter.

Chapter 3

Proposed Solution

Introduction

Humans understand languages relying on semantic and functional dependencies, and the relationship between dialogue elements and their context. Attention has been proven to be very effective in understanding humans' languages. In fact, recent works have shown the ability of the Attention module to capture semantic dependencies.

In this chapter, we investigate matching a response with its multi-turn context, using dependency information based entirely on Attention. This task is known as "Multi-Turn Response Selection" NLP task.

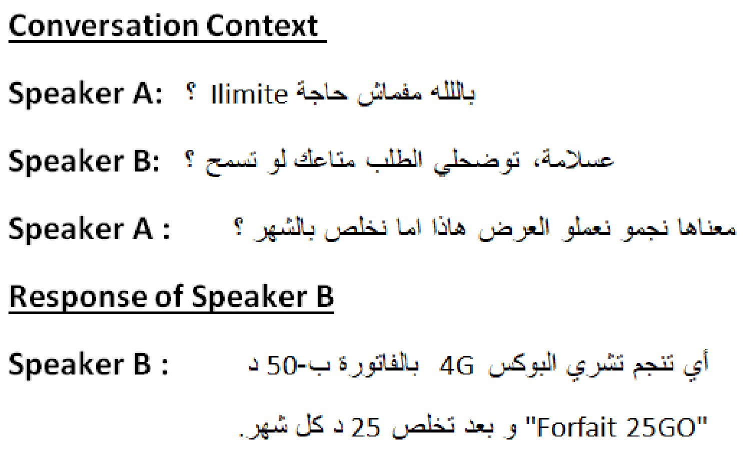
The solution was inspired by the recently proposed Deep Attention Mechanism (DAM) model [14]. It is a fully Attention-dependant model inspired by the Transformer [1], which addresses the issue of sequence-to-sequence generation only using Attention.

3.1 Multi-turn Context Response Selection Task

3.1.1 Automatic Response Selection

The ability for a computer to converse in a natural and coherent manner with a human has long been held as one of the primary objectives of artificial intelligence (AI).

One important task in natural conversations is response selection, which aims to select the best matched response from a set of candidates given the context of a conversation. Response selection can be generally categorized into ”single-turn” and ”multi-turn”. Most early studies are single-turn that only consider the last utterance for matching a response. In our case, we focus on multi-turn, where we consider all the previous utterances for selecting the response. An example of a multi-turn context conversation is shown in 3.1. The example was extracted from comments and replies on the Ooreedoo Tunisie official Facebook page.



Conversation Context

Speaker A: باللله مفماش حاجة Ilimite ؟

Speaker B: عسلامة، توضحلي الطلب متاعك لو تسمح ؟

Speaker A : معناها نجمو نعملو العرض هاذا اما نخلص بالشهر ؟

Response of Speaker B

Speaker B : أي تنجم تشري البوكس 4G بالفاتورة ب-50 د
"Forfait 25GO" و بعد تخلص 25 د كل شهر.

Figure 3.1: Example of multi-turn context conversation.

3.1.2 Problem Formalization

Let D be our dataset such as, $D = (c, r, y)_{Z=1..N}$. Here, $c = (u_i)_{i=0..n-1}$ is a conversation context, composed of n utterances (or phrases) u_i, which are composed of n_{u_i} words $(w_{u_i,k})_{k=0..n_{u_i}-1}$, And, $r = (w_{r,t})_{t=0..n_r-1}$ is a response candidate, composed of n_r words $w_{r,t}$. Then, $y = (0, 1)$ is a label, showing if r is a good response for c ($y = 1$) or a bad one ($y = 0$).

The DAM model is expected to learn a function $g(c, r)$, named the Matching Model function, that has the capacity to measure the relevance between a context c and a candidate response r, from our dataset D.

3.2 Deep Attention Matching Model

3.2.1 Model Overview

We introduce the neural matching network, namely the Deep Attention Matching network (DAM), for multi-turn response selection. Figure 3.2 gives an overview of the model. It is composed of four modules:

- Word Embedding Module
- Representation Module
- Matching Module
- Aggregation Module

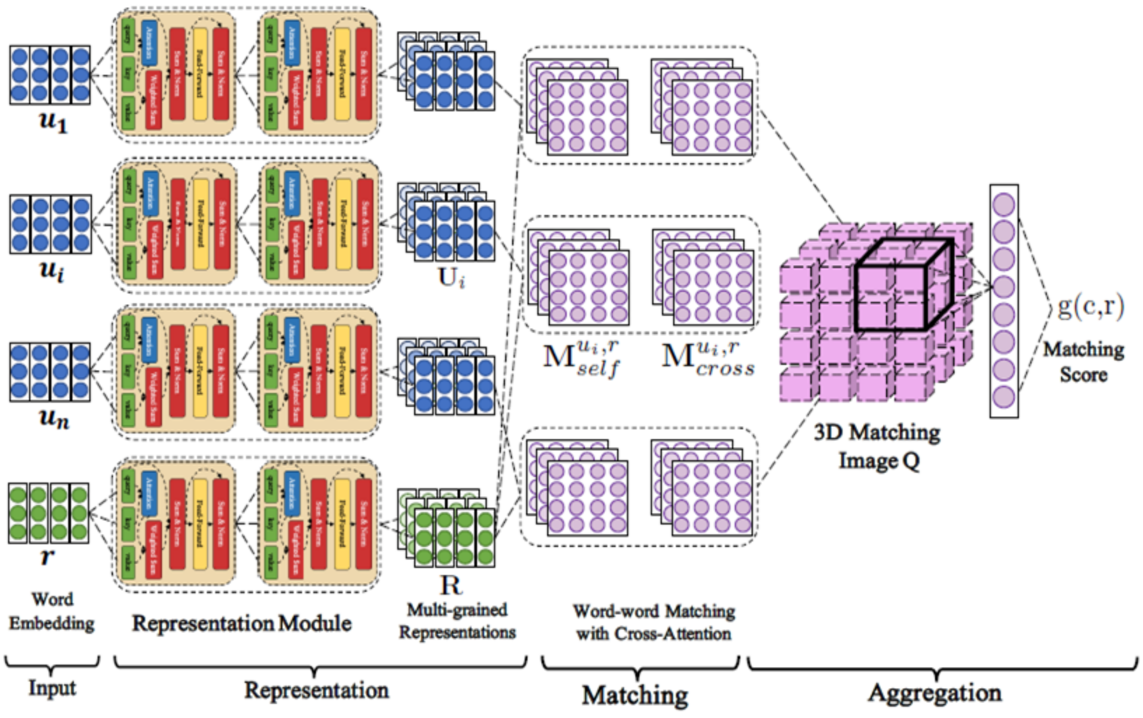


Figure 3.2: The Deep Attention Matching model [14].

In general, DAM takes each word of an utterance in context or response and hierarchically enriches its representation with successive levels of Attention modules. This gradually produces sophisticated segment representations surrounding the word from one level to another. In this way, each utterance in context and response are matched based on segment pairs at different levels. So, DAM captures matching information between the context and the response from word-level to sentence-level, until arriving to the context-response-level.

3.2.2 Word Embedding Module

As we can not feed a word just as a text string to a neural network, we need a way to represent the words to the network. To do that, we start with a pre-processed input text dataset D, as introduced previously. Then, each utterance u_i from a context c and its response candidate r are represented with sequences of Word Embedding vectors.

To do this, we first build a vocabulary of words from our training documents. Then, the Word Embedding vectors are constructed through a specifically designed model for that purpose, known as Word2Vec Model.

Word2Vec is a neural network with a single hidden layer, as shown in Figure 3.3. Each vocabulary word is initialized as a 1-hot encoded vector. The model then reconstructs the context of words by estimating the probability that a word is "close" to another word given as input.

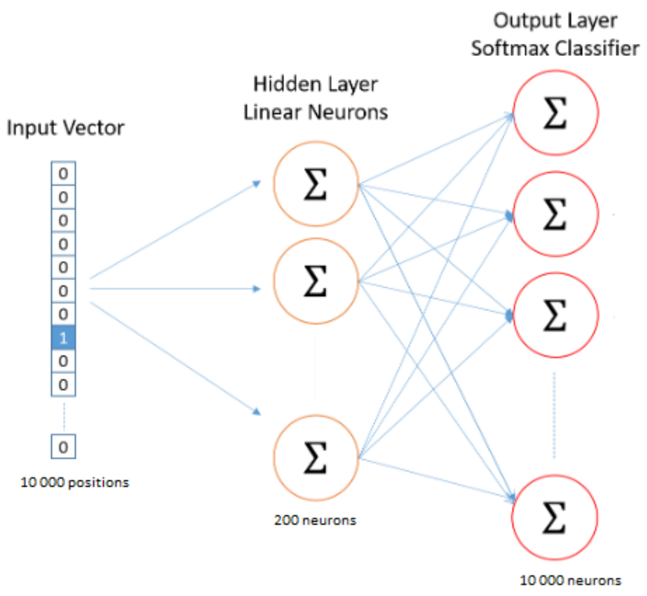


Figure 3.3: The Word2Vec model architecture [9].

More concretely, for each input word in the middle of a sentence, the Word2Vec model looks at the words nearby and picks one at random. The output of the network is a single vector containing, for every word in our vocabulary, the probability that a randomly selected nearby word is that vocabulary word.

The goal of this process is to extract the weights that have been learned by the neurons of the model's hidden layer. It is these weights that form the word vector. For example, if you have a 200 neurons in the hidden layer, the model will create a 200-dimension word vector for each word in the corpus. The output of this process, therefore, is a word-vector mapping of size $n_{Vocabulary\,size} \times n_{Embedding\,size}$.

The final context c and the response candidate r representations are respectively named $U_i^0 = (e^0_{u_i,k})_{k=0..n_{u_i}-1}$ and $R^0 = (e^0_{u_i,t})_{t=0..n_r-1}$, where $e \in R^d$ with d is the dimension of the Word Embedding vectors.

3.2.3 Representation Module

The data sequences, obtained from the Word Embedding Module are then introduced into a Representation Module Figure 3.4, totally based on Attention, previously presented in equation 2.2. This step tends to capture the semantic dependencies. Those dependencies are constructed at many hierarchical levels $l_{l=0..L}$ for each context utterance u_i and candidate response r.

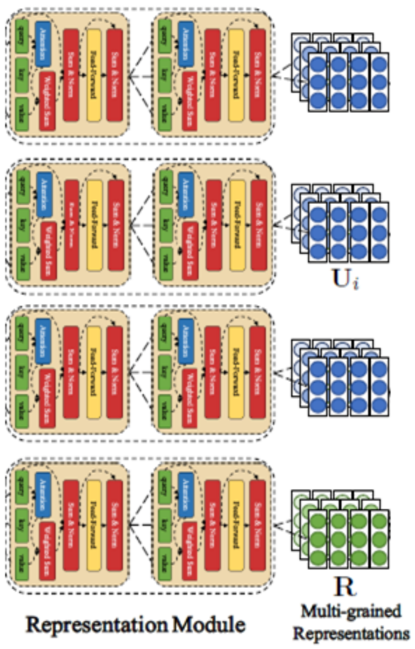


Figure 3.4: The Representation module [14].

Concretely, for each introduced U_i^0 and R^0 (the Word Embedding representations for the utterances u_i and their candidate response r, respectively), the Representation Module stacks the Attentive Module hierarchically, from level $l = 0$ to level $l = L$ Figure 2.7. To do that, each l^{th} attention layer (representing the l^{th} level) takes the output of the $(l-1)^{th}$ layer as its input. As a result, two types of attention representations are constructed: The self-Attention representation and the cross-Attention representation.

The self-Attention representation captures the intra-word-level dependencies. We talk about semantic representations by making a sentence attend to itself. This procedure is formulated as equations (3.1) and (3.2), where $(U_i^l)_{l=0..L}$ and $(R^l)_{l=0..L}$ denotes the self-Attention representation for the utterances u_i and their candidate response r, respectively:

$$U_i^{l+1} = AttentionModule(U_i^l, U_i^l, U_i^l) \qquad \forall i = 0..n-1\,, \forall l = 0..L \tag{3.1}$$

$$R_i^{l+1} = AttentionModule(R^l, R^l, R^l) \qquad \forall i = 0..n-1\,, \forall l = 0..L \tag{3.2}$$

The cross-Attention representation captures the context-response-level dependencies. We talk about semantic representation by making a context and its candidate response match each other. This procedure is formulated as equations (3.3) and (3.4), where $(\tilde{U_i^l})_{l=0..L}$ and $(\tilde{R_i^l})_{l=0..L}$ denotes the cross-Attention representation for the utterances u_i and their candidate response r, respectively:

$$\tilde{U_i^l} = AttentionModule(U_i^l, R^l, R^l) \qquad \forall i = 0..n-1\,, \forall l = 0..L \tag{3.3}$$

$$\tilde{R_i^l} = AttentionModule(R^l, U_i^l, U_i^l) \qquad \forall i = 0..n-1\,, \forall l = 0..L \tag{3.4}$$

3.2.4 Matching Module

Given the Attention representations U_i^{l+1}, R^{l+1}, $\tilde{U_i^l}$ and $\tilde{R^l}$, obtained from the Representation Module, we tend in the Matching Module Figure 3.5 to match the utterances u_i and their candidate responses r with each other in similarity matrices.

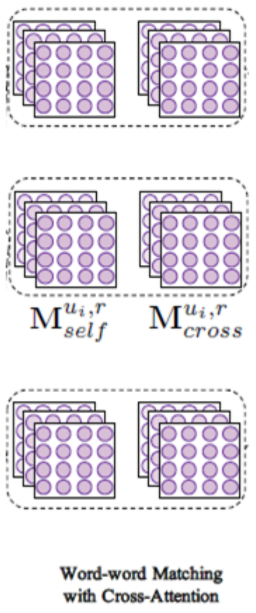


Figure 3.5: The Matching module [1].

To do that, we construct two types of Attention matrices: a self-Attention-match matrix and a cross-Attention-match matrix. Those matrices tend to measure the relevance between utterances and responses, resulting in matching scores.

The self-Attention-matching matrix is based on self-Attention. More concretely, it is the dot-product of the k^{th} embedding in U_i^l and the t^{th} embedding in R^l, as presented in equation (3.5), where $(M_{self}^{u_i,r,l})_{l=0..L}$ denotes the self-Attention-match at different levels l:

$$M_{self}^{u_i,r,l} = (U_i^l[k]^T.R^l[t])_{n_{u_i}\times n_r} \qquad \forall l = 0..L \tag{3.5}$$

And the cross-Attention-matching matrix is based on cross-Attention. It is the dot-product of the k^{th} embedding in $\tilde{U_i^l}$ and the t^{th} embedding in $\tilde{R^l}$, as presented in equation (3.6), where $(M_{cross}^{u_i,r,l})_{l=0..L}$ denotes the cross-Attention-match at different levels l:

$$M_{cross}^{u_i,r,l} = (\tilde{U_i^l}[k]^T.\tilde{R^l}[t])_{n_{u_i}\times n_r} \qquad \forall l = 0..L \tag{3.6}$$

3.2.5 Aggregation Module

The matching scores $(M_{self}^{u_i,r,l})_{l=0..L}$ and $(M_{cross}^{u_i,r,l})_{l=0..L}$, obtained from the Representation Module, are finally merged into the Aggregation Module Figure 3.6, in order to get a combined $3D$ representation.

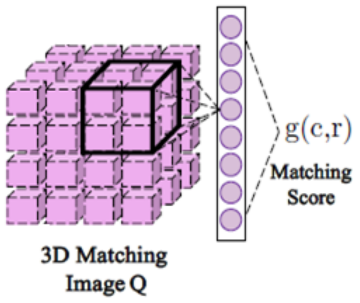


Figure 3.6: The Aggregation module [1].

This is done by creating a $3D$ matching image, or a cube $Q = (Q_{i,k,t})_{n \times n_{u_i} \times n_r}$. The three dimensions of Q represents: Each utterance u_i in context c, each word $w_{u_i,k}$ in utterance u_i and each word $w_{r,t}$ in response r respectively. The aggregated cube is formulated as in equation (3.7):

$$Q_{i,k,t} = (M_{self}^{u_i,r,l}[k,t])_{l=0..L} \oplus (M_{cross}^{u_i,r,l}[k,t])_{l=0..L} \tag{3.7}$$

In equation (3.7), the $\oplus$ symbol represents the concatenation operation, so each element of the cube, also named pixel, has $2(L+1)$ channels.

In order to extract the most important matching features from the whole image, DAM applies a $3D$ convolution with max-pooling. As a result, the most important matching features $f_{match}(c,r)$ between segment pairs of context c and candidate response r are extracted.

The result is finally fused into a single-layer perceptron, to obtain the matching score $g(c,r)$ between the whole context and the response candidate. The matching score is formulated in equation (3.8):

$$g(c,r) = \sigma(W_3 f_{match}(c,r) + b_3) \tag{3.8}$$

In the equation (3.8), W_3 and b_3 represent the learnt parameters from the single-layer perceptron neural network. And σ is the Sigmoid function that gives the probability if a response r is a proper candidate to the context c. The loss function L of DAM is the negative log likelihood, defined in equation (3.9):

$$L = -\sum_{(c,r,y) \in D} log(p(y|c,r)) \tag{3.9}$$

where,

$$p(y|c, r) = g(c, r)y + (1 - g(c, r))(1 - y) \tag{3.10}$$

Conclusion

In this chapter, we introduced the chosen NLP task which is the "Multi-Turn Response Selection" task and formalized its problematic. We then, explained the proposed neural network solution; the DAM model, an entirely Attention based model. We separately defined each one of its modules ranging from the Word Embedding module, to the Representation module, then the Matching module, and finally the Aggregation module.

Now that the theoretical part is explained we will move to the application. We will start by collecting a Tunisian dialect data cleaning and organising it in a new corpus named the OoredooTn corpus. Then we will train and validate our model based on this corpus. We will finally test the model with different parameters to investigate its performances on choosing the correct response based on a context conversation.

Chapter 4

Results and Analysis

Introduction

This chapter represents the achieved results. We will start by presenting the collected data. Then, we will train and validate the DAM model. We will extract the most relevant obtained results at the validation step. Finally, we will end up testing our model by varying the dataset statistics parameters.

4.1 Tunisian Dialect Dataset

4.1.1 Dataset Collection

We hypothesized that the lack of progress in NLP with informal Tunisian dialect is due to the lack of sufficiently large datasets. We also notified the complete absence of Tunisian corpora related to the problem of question-answer systems. We aim to overcome this barrier by providing a new large corpus for researchers working on Tunisian language processing models. For this purpose, we decided to collect and pre-process our own dataset.

We focused on collecting a dataset corresponding to our chosen task the ”Multi-tun Context Response Selection”. So, we sought a large corpus for research in response-selection systems with the following properties:

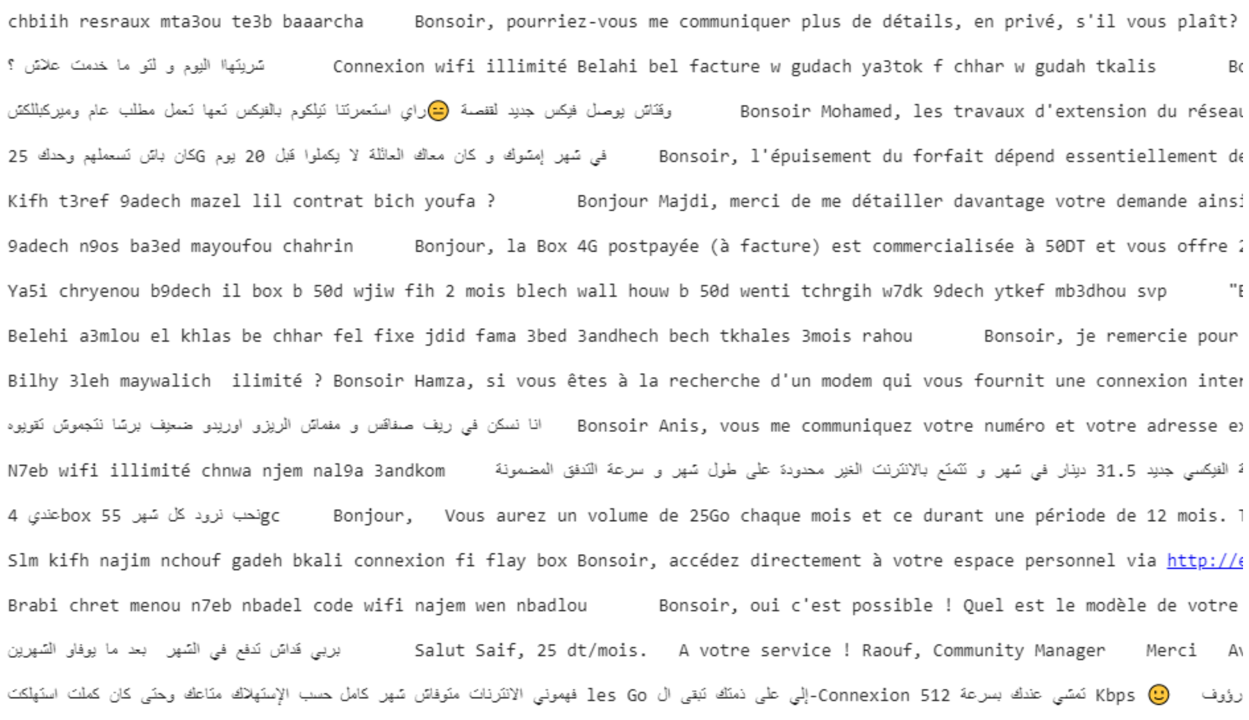
- Written Tunisian dialect.
- Multi-participant human-human chat.
- Large number of conversations (more than $10K$ conversations).

- Conversations with one or several turns (1 or more).
- Task-specific domain (commercial/technical domains ...).

The dataset was collected using the open source application ”Facepager”[1] [7]. The application is made for fetching publicly available data from Facebook, Twitter and other JSON-based APIs. All data is stored in a SQLite database and may be exported to a .csv file.

The collected data, the OoredooTn corpus, was extracted from the *OoredooTunisie* official Facebook page comments and replies. The collection period was from July 2018 until March 2019. Those conversations were comments posted by Ooredoo clients (complaints, suggestions, questions ...) in order to receive support from Ooredoo managers replies.

An example from the raw collected data is illustrated in the Figure 4.1. As can be seen, the raw data is noisy: It is full of emojis, it contains signs of punctuation and successively repeated characters. It is also not organized to satisfy the chosen NLP task representation.



```
chbiih resraux mta3ou te3b baaarcha     Bonsoir, pourriez-vous me communiquer plus de détails, en privé, s'il vous plaît?
شريتها اليوم و لتو ما خدمت علاش ؟       Connexion wifi illimité Belahi bel facture w gudach ya3tok f chhar w gudah tkalis       Bo
وقتاش يوصل فيكس جديد لقفصة 😑راي استعمرتنا تيلكوم بالفيكس تعها تعمل مطلب عام وميركبللكش        Bonsoir Mohamed, les travaux d'extension du réseau
في شهر إمشوك و كان معاك العائلة لا يكملوا قبل 20 يوم Gكان باش تسعملهم وحدك 25       Bonsoir, l'épuisement du forfait dépend essentiellement de
Kifh t3ref 9adech mazel lil contrat bich youfa ?        Bonjour Majdi, merci de me détailler davantage votre demande ainsi
9adech n9os ba3ed mayoufou chahrin      Bonjour, la Box 4G postpayée (à facture) est commercialisée à 50DT et vous offre 1
Ya5i chryenou b9dech il box b 50d wjiw fih 2 mois blech wall houw b 50d wenti tchrgih w7dk 9dech ytkef mb3dhou svp      "E
Belehi a3mlou el khlas be chhar fel fixe jdid fama 3bed 3andhech bech tkhales 3mois rahou       Bonsoir, je remercie pour
Bilhy 3leh maywalich  ilimité ? Bonsoir Hamza, si vous êtes à la recherche d'un modem qui vous fournit une connexion inter
انا نسكن في ريف صفاقس و مفماش الريزو اوريدو ضعيف برشا نتجموش تقويوه    Bonsoir Anis, vous me communiquez votre numéro et votre adresse ex
N7eb wifi illimité chnwa njem nal9a 3andkom     الفيكسي جديد 31.5 دينار في شهر و تتمتع بالانترنت الغير محدودة على طول شهر و سرعة التدفق المضمونة
نحب نزود كل شهر 55 box عندي 4gc      Bonjour,   Vous aurez un volume de 25Go chaque mois et ce durant une période de 12 mois. T
Slm kifh najim nchouf gadeh bkali connexion fi flay box Bonsoir, accédez directement à votre espace personnel via http://e
Brabi chret menou n7eb nbadel code wifi najem wen nbadlou       Bonsoir, oui c'est possible ! Quel est le modèle de votre
بربي قداش تدفع في الشهر  بعد ما يوفاو الشهرين        Salut Saif, 25 dt/mois.   A votre service ! Raouf, Community Manager    Merci   Av
فهموني الانترنات متوفاش شهر كامل حسب الإستهلاك متاعك وحتى كان كملت استهلكت les Go ال تبقى على ذمتك إلي-Connexion 512 بسرعة عندك تمشي Kbps 🙂   رؤوف
```

Figure 4.1: Example from the row collected OoredooTn data.

[1]Facepager application’s source code and releases are available at https://github.com/strohne/Facepager.

4.1.2 Dataset Pre-processing

Cleaning

Given the nature of the raw noisy collected data, we did some cleaning before the organization step. In order to do that, we automatically [2]:

- Removed specific signs of punctuation.
- Deleted all emojis.
- Eliminated successively repeated characters.
- Removed short words (less than 3 characters).

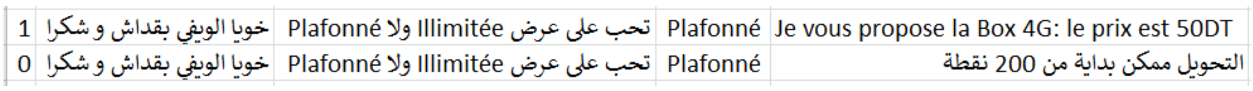
Organization

To adjust our collected and cleaned data to the chosen NLP task, we automatically organized [3] it in such way:

- Separated data into labels, contexts and responses.
- Associated label 1 to the original conversations.
- Generated random false responses and associate label 0 to them.

Then, the OoredooTn corpus was divided into training, validation, and test sets. It was done such as:

- 68 % of the cleaned data for training. Then, each context will have one positive response (the original data response) and one negative response (randomly created). So that the final training file has 1:1 positive-negative ratios. An example from the train data is illustrated in the Figure 4.2.

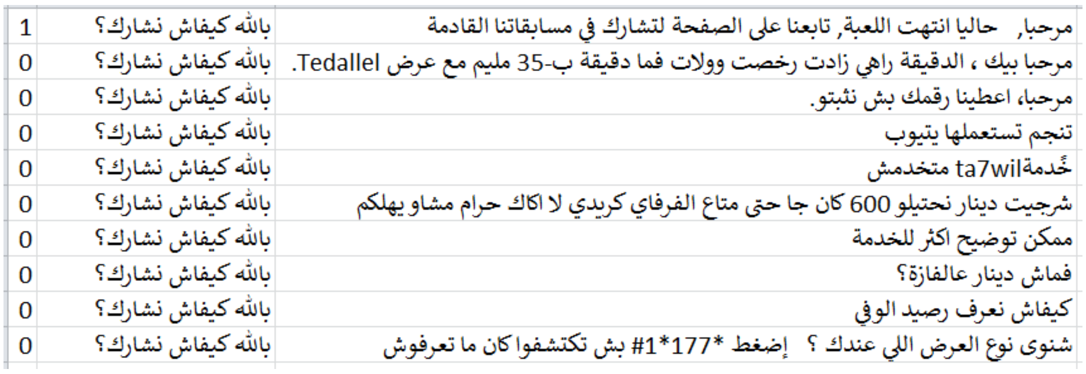


1	خويا الويفي بقداش و شكرا	تحب على عرض Illimitée ولا Plafonné	Plafonné	Je vous propose la Box 4G: le prix est 50DT
0	خويا الويفي بقداش و شكرا	تحب على عرض Illimitée ولا Plafonné	Plafonné	التحويل ممكن بداية من 200 نقطة

Figure 4.2: Train example from OoredooTn dataset.

[2]Using Python language
[3]Using Python language

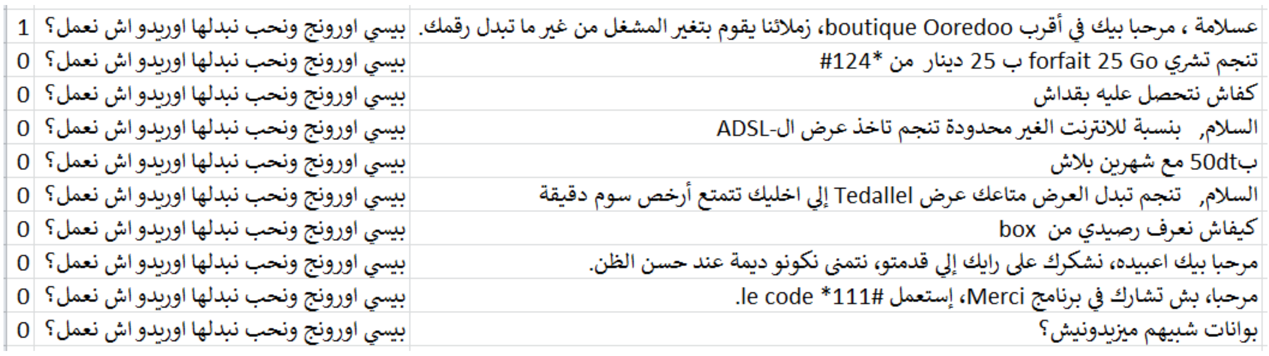
- 14.6 % of the cleaned data for validation. Then, each context will have one positive response (the original data response) and nine negative replies (randomly created). So that the final validation file has 1:9 positive-negative ratios. An example from the validation data is illustrated in the Figure 4.3.



1	بالله كيفاش نشارك؟	مرحبا, حاليا انتهت اللعبة, تابعنا على الصفحة لتشارك في مسابقاتنا القادمة
0	بالله كيفاش نشارك؟	مرحبا بيك ، الدقيقة راهي زادت رخصت وولات فما دقيقة ب-35 مليم مع عرض Tedallel.
0	بالله كيفاش نشارك؟	مرحبا، اعطينا رقمك بش نثبتو.
0	بالله كيفاش نشارك؟	تنجم تستعملها يتيوب
0	بالله كيفاش نشارك؟	خُدمةta7wil متخدمش
0	بالله كيفاش نشارك؟	شرجيت دينار نحتيلو 600 كان جا حتى متاع الفرفاي كريدي لا اكاك حرام مشاو يهلكم
0	بالله كيفاش نشارك؟	ممكن توضيح اكثر للخدمة
0	بالله كيفاش نشارك؟	فماش دينار عالفازة؟
0	بالله كيفاش نشارك؟	كيفاش نعرف رصيد الوفي
0	بالله كيفاش نشارك؟	شنوى نوع العرض اللي عندك ؟ إضغط *177*1# بش تكتشفوا كان ما تعرفوش

Figure 4.3: Validation example from OoredooTn dataset.

- 16.9 % of the cleaned data for test. Then, each context will have one positive response (the original data response) and nine negative replies (randomly created). So that the final test file has 1:9 positive-negative ratios. An example from the test data is illustrated in the Figure 4.4.



1	بيسي اورونج ونحب نبدلها اوريدو اش نعمل؟	عسلامة ، مرحبا بيك في أقرب boutique Ooredoo، زملائنا يقوم بتغير المشغل من غير ما تبدل رقمك.
0	بيسي اورونج ونحب نبدلها اوريدو اش نعمل؟	تنجم تشري forfait 25 Go ب 25 دينار من *124#
0	بيسي اورونج ونحب نبدلها اوريدو اش نعمل؟	كفاش نتحصل عليه بقداش
0	بيسي اورونج ونحب نبدلها اوريدو اش نعمل؟	السلام, بنسبة للانترنت الغير محدودة تنجم تاخذ عرض ال-ADSL
0	بيسي اورونج ونحب نبدلها اوريدو اش نعمل؟	ب50dt مع شهرين بلاش
0	بيسي اورونج ونحب نبدلها اوريدو اش نعمل؟	السلام, تنجم تبدل العرض متاعك عرض Tedallel إلي اخليك تتمتع أرخص سوم دقيقة
0	بيسي اورونج ونحب نبدلها اوريدو اش نعمل؟	كيفاش نعرف رصيدي من box
0	بيسي اورونج ونحب نبدلها اوريدو اش نعمل؟	مرحبا بيك اعبيده، نشكرك على رايك إلي قدمتو، نتمنى نكونو ديمة عند حسن الظن.
0	بيسي اورونج ونحب نبدلها اوريدو اش نعمل؟	مرحبا، بش تشارك في برنامج Merci، إستعمل *111# le code.
0	بيسي اورونج ونحب نبدلها اوريدو اش نعمل؟	بوانات شبيهم ميزيدونيش؟

Figure 4.4: Test example from OoredooTn dataset.

4.1.3 Dataset Statistics

We compared our dataset with the freely available datasets for response-selection systems: The Ubuntu Corpus V1 [4] [11] and the Douban Conversation Corpus [5] [17]. We present those datasets in the Table 4.1.

Dataset	Language	Description
Ubuntu	English	Extracted from Ubuntu chat logs.
Douban	Chinese	Extracted from Douban groupe.
OoredooTn	Tunisian dialect	Extracted from Ooredoo Tunisie Facebook page comments and replies.

Table 4.1: A selection of open-source datasets for dialogue systems. The last entry is our contribution.

The OoredooTn data contains more than 42K conversations such as all conversations are carried out in text form. It is divided into Train, Valid and Test files as shown in Table 4.2 . It is on a less scale than formal language datasets for solving dialogue problems. However, it has an average of more than 2 turns each as conceived earlier. Adding to that, each conversation in our dataset includes long utterances.

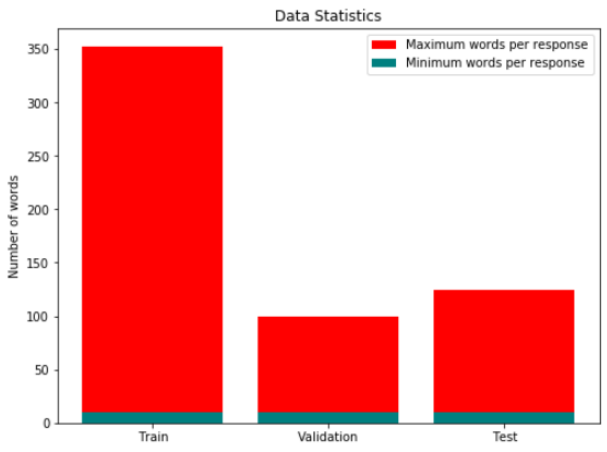
	Ubuntu			Douban			Ooredoo		
Parameters	Train	Valid	Test	Train	Valid	Test	Train	Valid	Test
Context-response pairs	1M	500K	500K	1M	50K	10K	20K	11.3K	11.29K
Candidates per context	2	10	10	2	2	10	2	10	10
Avg turns per context	10.13	10.11	10.11	6.69	6.75	6.45	2.25	2.10	2.06
Avg words per utterance	11.35	11.34	11.37	18.56	18.50	20.74	16.73	17.05	15.77

Table 4.2: Statistics of open-source datasets for dialogue systems [18]. The last entry is our contribution.

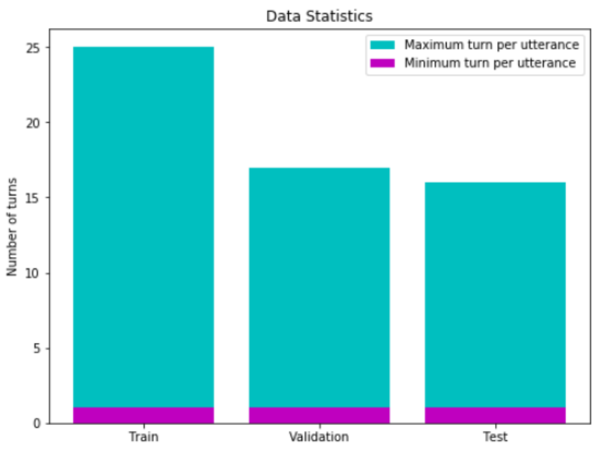
[4]You can find Ubuntu Corups V1 at: https://github.com/rkadlec/ ubuntu-ranking-dataset-creator

[5]You can find Douban Conversation Corups at: https://github.com/MarkWuNLP/ MultiTurnResponseSelection

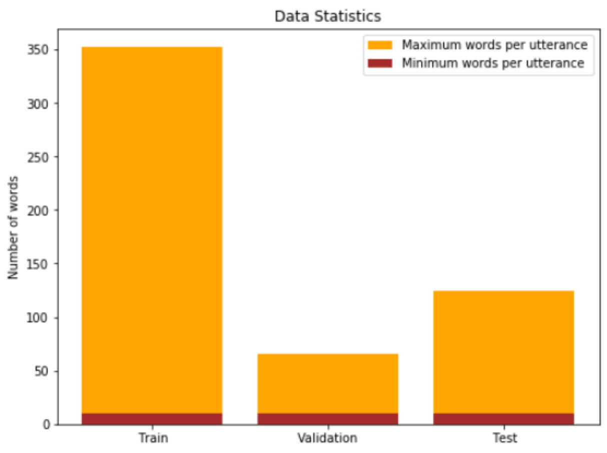
More details about the OoredooTn data is illustrated on Figure 4.5: The corpus contains a 350 maximum number of turns per response and a minimum number of 1 as shown in the sub-figure (a). It is also characterised by a maximum turn of 25 per context and a minimum of 1 as shown in the sub-figure (b). The data has a number of words per utterance varying from 1 to 350 as shown in the sub-figure (c).



(a) Maximum and minimum words per response.



(b) Maximum and minimum turns per utterance.



(c) Maximum and minimum words per utterance.

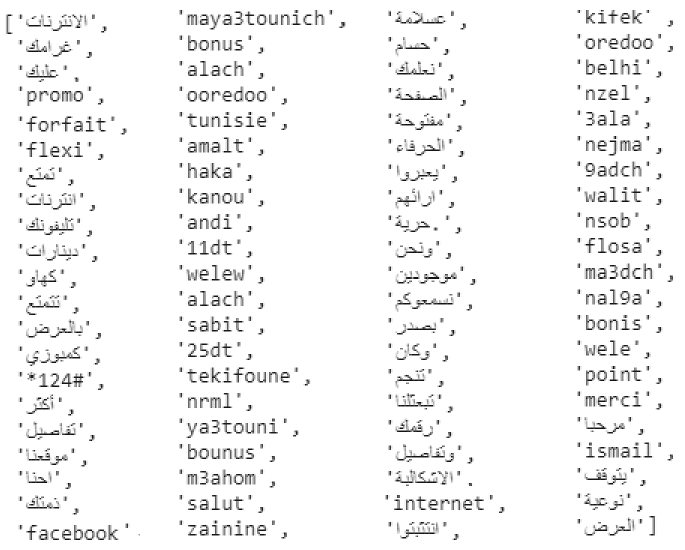
Figure 4.5: OoredooTn dataset statistics

4.2 Deep Attention Matching Model

4.2.1 Word Embedding

Word Embedding Module was done using the Gensim Python package[6]. The training algorithms were originally converted from Google C source code [7]. Then, it was extended with additional functionalities and optimizations over the years to finally be included in the Gensim package.

We started by creating a set of vocabulary extracted from the collected data. It is a document containing all the unique words, as illustrated in the Figure 4.6. The final vocabulary document contains 36506 different words.



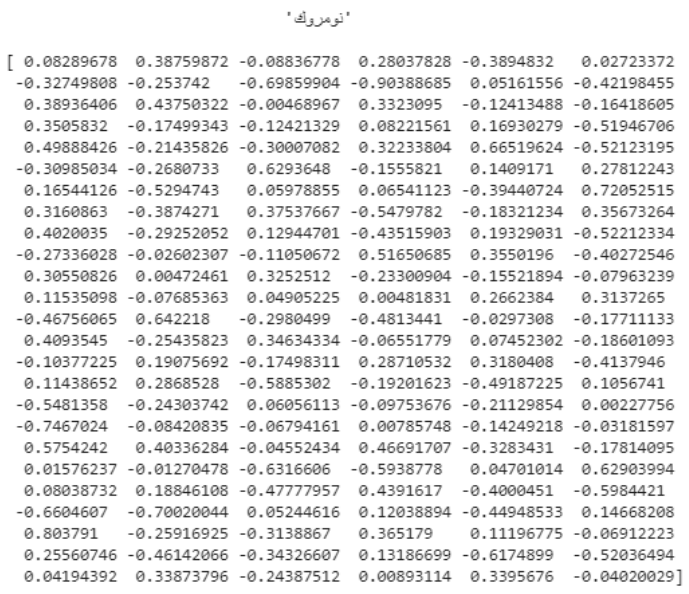
```
['الانترنات',   'maya3tounich',  'عسلامة',      'kifek' ,
 'غرامك',       'bonus',         'حسام',        'oredoo',
 'عليك',        'alach',         'نعلمك',       'belhi',
 'promo',       'ooredoo',       'الصفحة',      'nzel',
 'forfait',     'tunisie',       'مفتوحة',      '3ala',
 'flexi',       'amalt',         'الحرفاء',     'nejma',
 'تمتع',        'haka',          'يعبروا',      '9adch',
 'انترنات',     'kanou',         'ارائهم',      'walit',
 'تليفونك',     'andi',          'حرية.',       'nsob',
 'دينارات',     '11dt',          'ونحن',        'flosa',
 'كهاو',        'welew',         'موجودين',     'ma3dch',
 'تتمتع',       'alach',         'نسمعوكم',     'nal9a',
 'بالعرض',      'sabit',         'بصدر',        'bonis',
 'كمبوزي',      '25dt',          'وكان',        'wele',
 '*124#',       'tekifoune',     'تتجم',        'point',
 'أكثر',        'nrml',          'تبعثلنا',     'merci',
 'تفاصيل',      'ya3touni',      'رقمك',        'مرحبا',
 'موقعنا',      'bounus',        'وتفاصيل',     'ismail',
 'احنا',        'm3ahom',        'الاشكالية'.   'يتوقف',
 'نمتك',        'salut',         'internet',    'نوعية',
 'facebook'     'zainine',       'لاتتثبتوا',   'العرض']
```

Figure 4.6: Example from the created vocabulary document.

We set the embedding size to 200 (refers to the dimension of the word vector representation) and fed the vocabulary input into the Word2Vec neural network. We then extracted the optimized weights from the hidden layer which are the words vectors representation. Here after, in Figure 4.7, one example of a represented Tunisian word by a 200-d vector.

[6]All Gensim package functionalities can be found in: https://radimrehurek.com/gensim/ models/word2vec.html

[7]The original Google C source code can be found in: https://code.google.com/p/word2vec/



```
'نومروك'

[ 0.08289678  0.38759872 -0.08836778  0.28037828 -0.3894832   0.02723372
 -0.32749808 -0.253742   -0.69859904 -0.90388685  0.05161556 -0.42198455
  0.38936406  0.43750322 -0.00468967  0.3323095  -0.12413488 -0.16418605
  0.3505832  -0.17499343 -0.12421329  0.08221561  0.16930279 -0.51946706
  0.49888426 -0.21435826 -0.30007082  0.32233804  0.66519624 -0.52123195
 -0.30985034 -0.2680733   0.6293648  -0.1555821   0.1409171   0.27812243
  0.16544126 -0.5294743   0.05978855  0.06541123 -0.39440724  0.72052515
  0.3160863  -0.3874271   0.37537667 -0.5479782  -0.18321234  0.35673264
  0.4020035  -0.29252052  0.12944701 -0.43515903  0.19329031 -0.52212334
 -0.27336028 -0.02602307 -0.11050672  0.51650685  0.3550196  -0.40272546
  0.30550826  0.00472461  0.3252512  -0.23300904 -0.15521894 -0.07963239
  0.11535098 -0.07685363  0.04905225  0.00481831  0.2662384   0.3137265
 -0.46756065  0.642218   -0.2980499  -0.4813441  -0.0297308  -0.17711133
  0.4093545  -0.25435823  0.34634334 -0.06551779  0.07452302 -0.18601093
 -0.10377225  0.19075692 -0.17498311  0.28710532  0.3180408  -0.4137946
  0.11438652  0.2868528  -0.5885302  -0.19201623 -0.49187225  0.1056741
 -0.5481358  -0.24303742  0.06056113 -0.09753676 -0.21129854  0.00227756
 -0.7467024  -0.08420835 -0.06794161  0.00785748 -0.14249218 -0.03181597
  0.5754242   0.40336284 -0.04552434  0.46691707 -0.3283431  -0.17814095
  0.01576237 -0.01270478 -0.6316606  -0.5938778   0.04701014  0.62903994
  0.08038732  0.18846108 -0.47777957  0.4391617  -0.4000451  -0.5984421
 -0.6604607  -0.70020044  0.05244616  0.12038894 -0.44948533  0.14668208
  0.803791   -0.25916925 -0.3138867   0.365179    0.11196775 -0.06912223
  0.25560746 -0.46142066 -0.34326607  0.13186699 -0.6174899  -0.52036494
  0.04194392  0.33873796 -0.24387512  0.00893114  0.3395676  -0.04020029]
```

Figure 4.7: A 200-d vector representation of a Tunisian word.

We wanted to have an overview of the obtained vocabulary representation. So, we plotted all vocabulary vectors in a 3-d representation using the Principle Component Analysis (PCA)[8] method. The obtained plot is illustrated in the Figure 4.8.

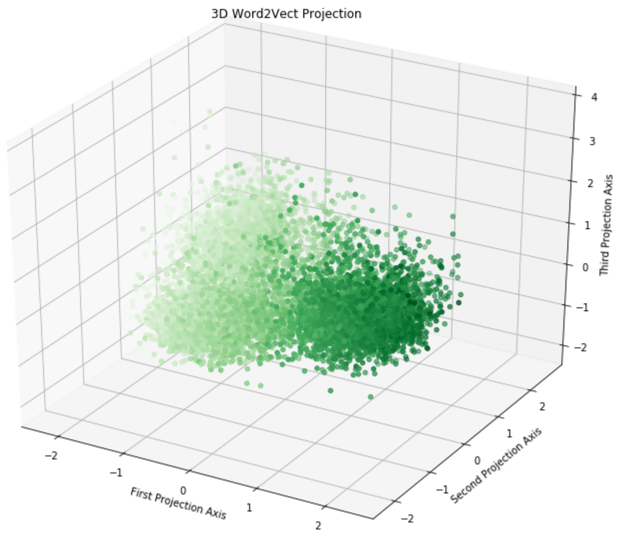


Figure 4.8: A 3-d vector representation of the whole vocabulary words using PCA.

[8]Principal Component Analysis (PCA) is a dimensionality-reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set. [13].

As the plot shows, we can notice the presence of two different clusters. We re-projected the obtained representation into a two 2-d plot and we got the next Figure 4.9, where the two clusters are obviously clearer.

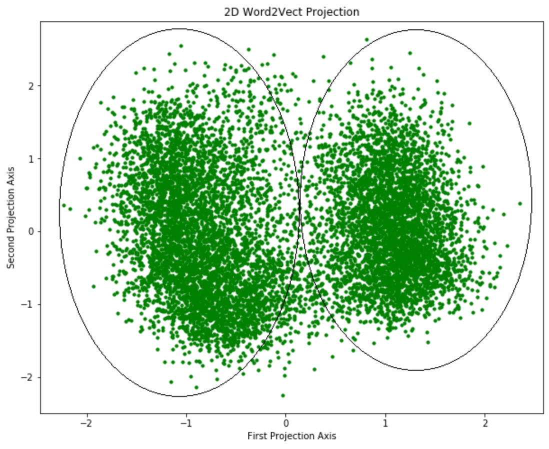
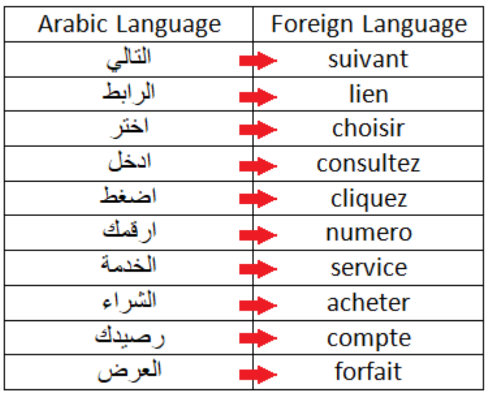


Figure 4.9: A 2-d vectors representation of the whole vocabulary words.

In order to understand the meaning behind the two clusters, we extracted 300 vocabulary words, represented and annotated them. We obtain the next page Figure 4.11. If we take an overview into the plot, we can easily figure out that the left side words are written in Arabic and the right side words are written in foreign languages. So the two clusters represent words in two different languages. This result should be predictable as the Tunisian language is a mixture of both Arabic and foreign languages.

Another important thing to notice is the translation between the two plot sides. So, the model has captured the translation meaning of the words and separated them as so. We can extract some highlighted examples from the Figure 4.11 in the Table 4.10:



Arabic Language		Foreign Language
التالي	➡	suivant
الرابط	➡	lien
اختر	➡	choisir
ادخل	➡	consultez
اضغط	➡	cliquez
ارقمك	➡	numero
الخدمة	➡	service
الشراء	➡	acheter
رصيدك	➡	compte
العرض	➡	forfait

Figure 4.10: Example of translated words.

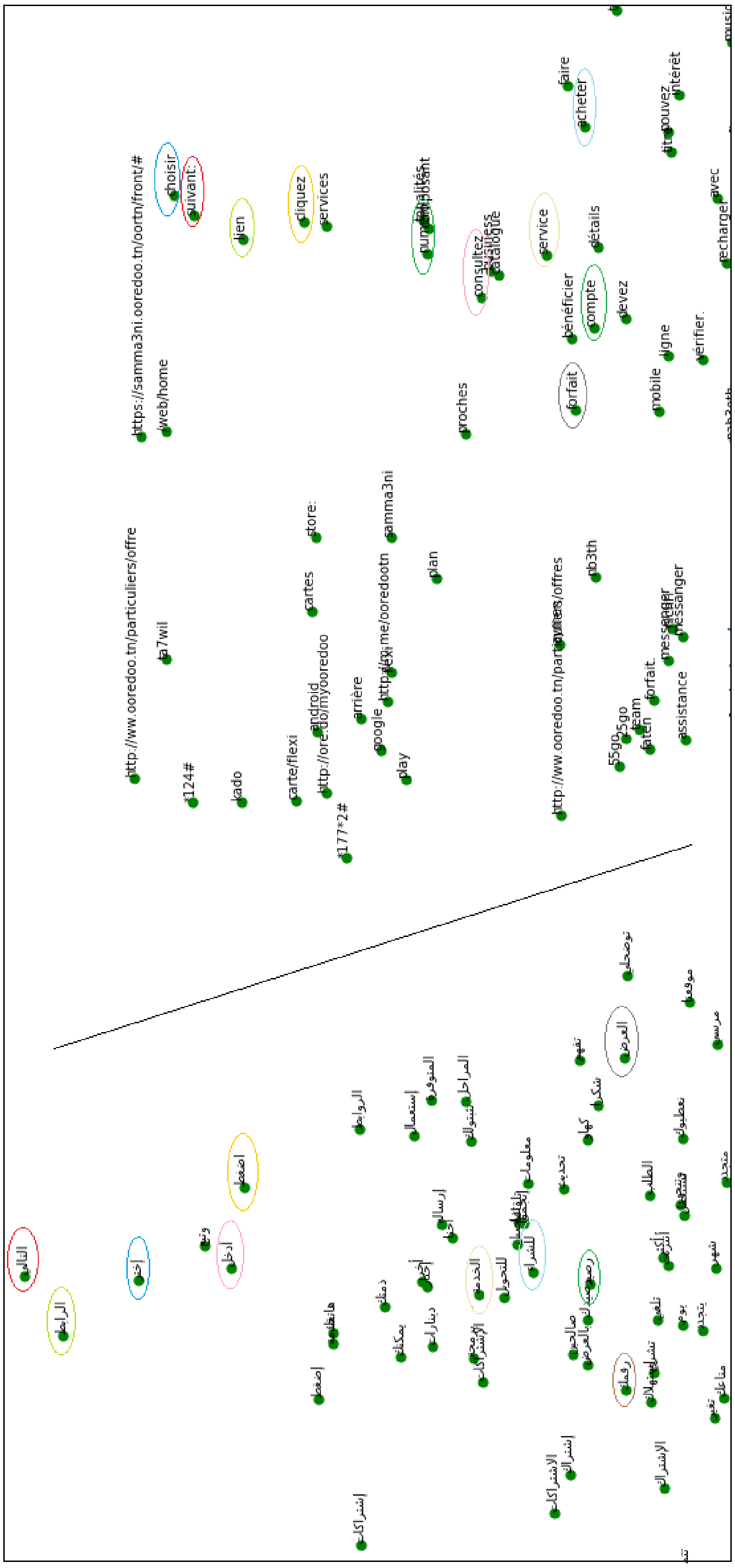
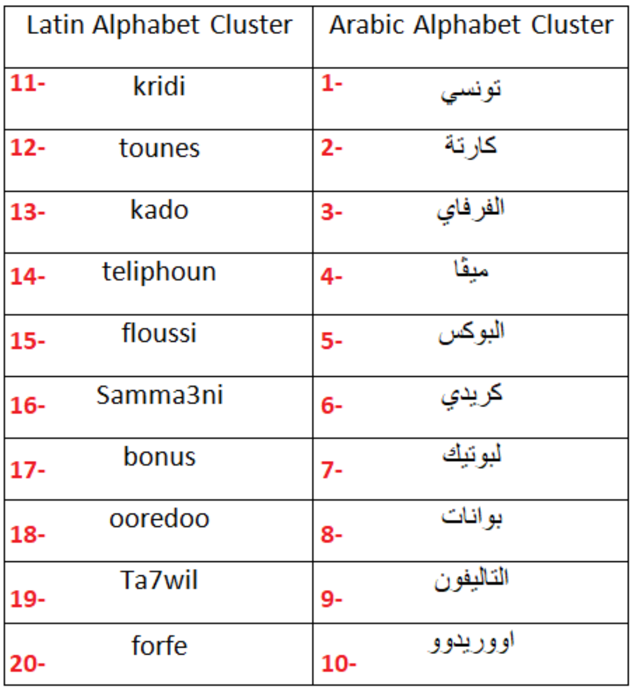


Figure 4.11: Extracted and annotated vocabulary words 2-d representation

Now we try to figure out if we can extract some valuable information, when looking from top to bottom. To do that we plotted 20 randomly chosen vocabulary words (a mixture of Arabic and foreign languages). The chosen words are illustrated in the Figure 4.12. We then plotted their top 200 most similar words using the t-distributed Stochastic Neighbor Embedding (t-SNE)[9].



Latin Alphabet Cluster		Arabic Alphabet Cluster	
11-	kridi	1-	تونسي
12-	tounes	2-	كارتة
13-	kado	3-	الفرفاي
14-	teliphoun	4-	ميڨا
15-	floussi	5-	البوكس
16-	Samma3ni	6-	كريدي
17-	bonus	7-	لبوتيك
18-	ooredoo	8-	بوانات
19-	Ta7wil	9-	التاليفون
20-	forfe	10-	اووريدوو

Figure 4.12: 20 chosen vocabulary words clusters.

The Figure 4.13 demonstrates the arrangement of the 20 words and their 200 most similar word representations in a 2D plot in the initial iteration. As we can see, the instances cannot be separated into different classes as they are too mixed.

As the algorithm operates, the word representations became more and more easily separable. The clusters are updated from an iteration to another as shown in the first three sub-Figure 4.14. Finally, the algorithm approaches convergence, after 30 iterations, in the last sub-Figure 4.14.

[9]T-distributed Stochastic Neighbor Embedding (T-SNE) is a non-linear "feature extraction" algorithm that constructs a new representation of the data so that close data in the original space has a high probability of having close representations in the new space. [2]

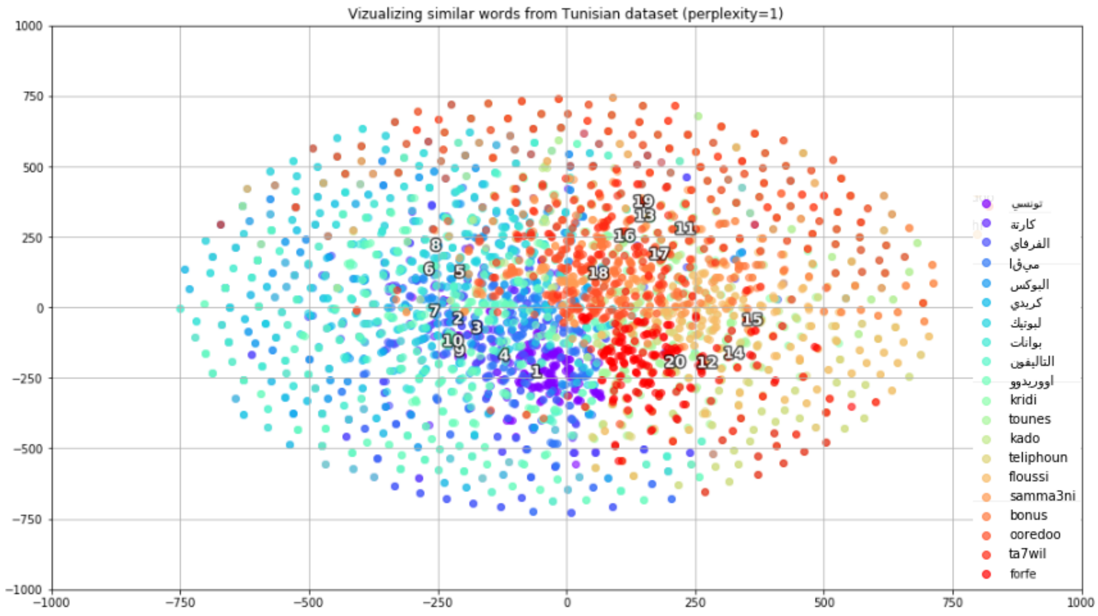
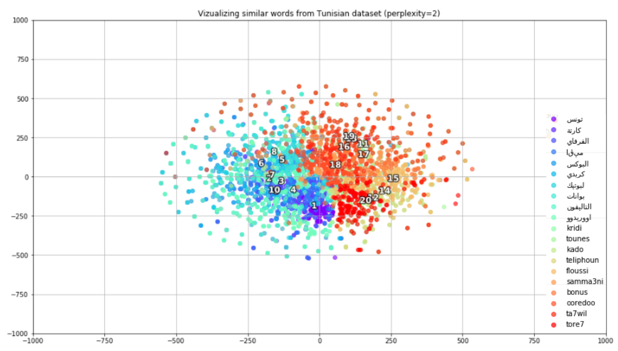
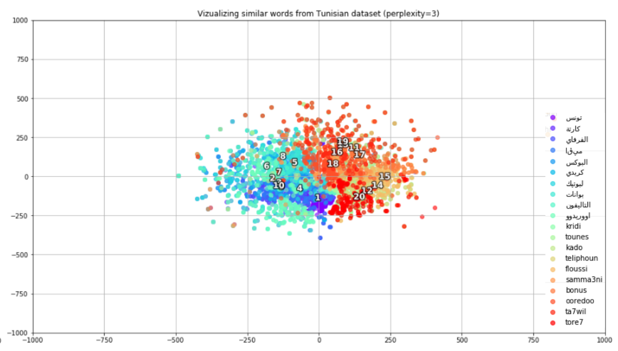


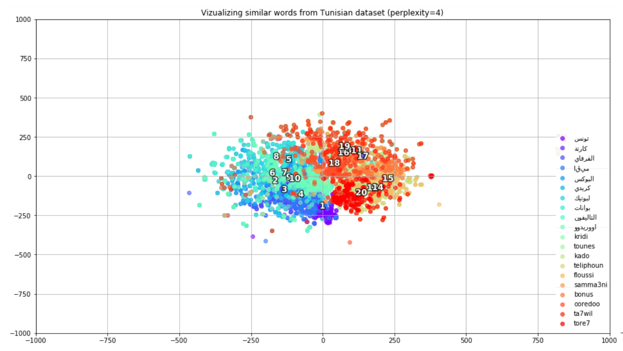
Figure 4.13: T-SNE first iterations.



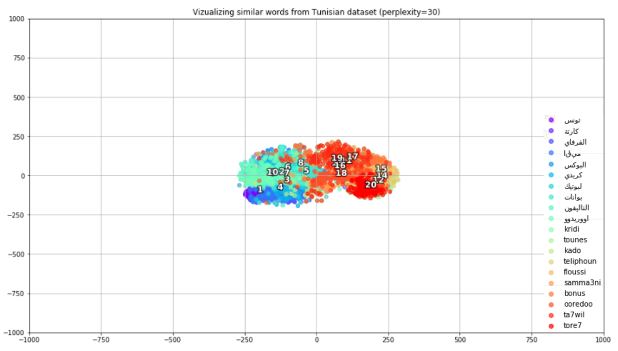
(a) Second iteration.



(b) Third iteration.



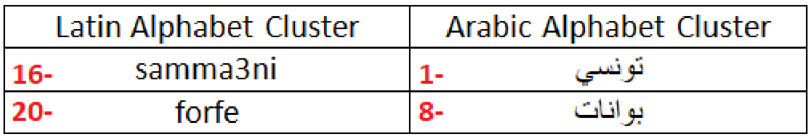
(c) Fourth iteration.



(d) Thirty-th iteration.

Figure 4.14: Clusters of similar words using T-SNE.

To get a closer look, we extracted few clusters from the last t-SNE iteration and dispalyed them in the next page Figure 4.21. We chose the following four clusters in order to analyse the results:

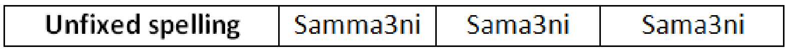


Latin Alphabet Cluster	Arabic Alphabet Cluster
16- samma3ni	1- تونسي
20- forfe	8- بوانات

Figure 4.15: Four extracted word clusters.

If we focus on the two cluster ”samma3ni” [10] and ”forfe” in the next page Figure 4.21, we can notice that:

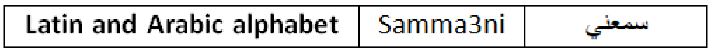
The model catches the unfixed spelling in the words as it represents the differently spelled ones close to each other.



Unfixed spelling	Samma3ni	Sama3ni	Sama3ni

Figure 4.16: Unfixed word spelling.

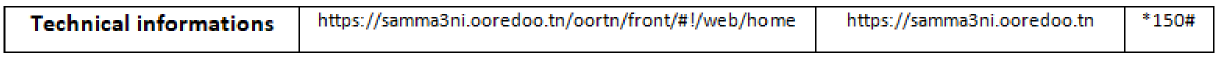
The model captures the Latin and Arabic alphabet writing of the words as it groups Latin and Arabic written ones in the same cluster.



Latin and Arabic alphabet	Samma3ni	سمعني

Figure 4.17: Latin and Arabic alphabet words.

The model detects technical information related to the words by extracting websites and and USSD commands [11] related to a specific operation.

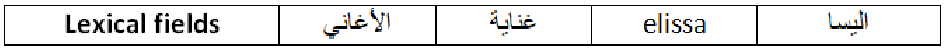


Technical informations	https://samma3ni.ooredoo.tn/oortn/front/#!/web/home	https://samma3ni.ooredoo.tn	*150#

Figure 4.18: Technical information related to words.

The model groups lexical fields related to the words such as name of singers.

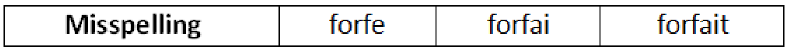
[10]Samma3ni is an offer from Ooredoo where you can customize your hold tone or record your own waiting message.
[11]USSD refers to Unstructured Supplementary Service Data



Lexical fields	الأغاني	غناية	elissa	اليسا

Figure 4.19: Misspelled words.

The model captures the presence of miss-spelling when it comes to foreign languages.



Misspelling	forfe	forfai	forfait

Figure 4.20: Misspelled words.

From the Word Embedding Module, we can summarize that the model solved one side from the Tunisian dialect challenges, introduced in the first chapter, which is: The language complexity. In fact, this module detected the presence of foreign languages and added a translation meaning to them. It also extracted unfixed spelling to the same words, and detected the use of both Latin and Arabic alphabet. Finally it pointed the misspelling when it came to foreign languages. We can also add that the Word Embedding Module solved the second final goal, mentioned in the same first chapter, by understanding the Tunisian dialect.

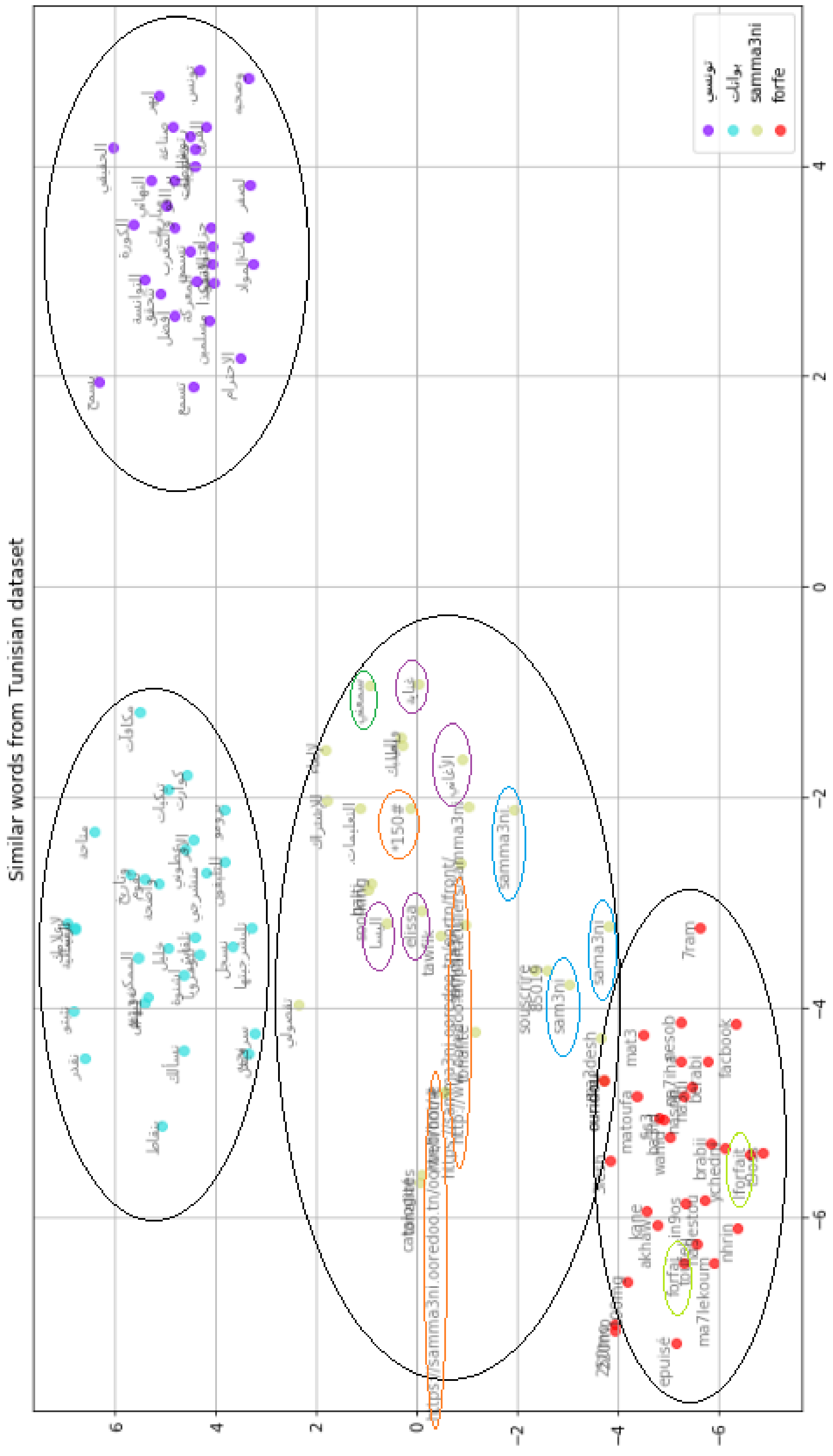


Figure 4.21: Four extracted clusters from the final iteration.

4.2.2 Model Training and Validation

DAM model next three modules (Representation Module, Matching Module and Aggregation Module) were implemented in TensorFlow, following the source code[12] from the original paper [14].

Evaluation Metrics

We want to train and validate the model performances on OoredooTn data. For our task, the model is asked to select the k best-matched responses from n available candidate responses r for the given context c. So, we calculated the recall $R_n@k$ of the true positive replies among the k selected ones. The Recall is expressed in the equation below (4.1), where y_i are the labels.

$$R_n@k = \frac{\sum_{i=1}^{k} y_i}{\sum_{i=1}^{n} y_i} \tag{4.1}$$

More specifically, we choose to work with $R_2@1$, $R_{10}@1$, $R_{10}@2$ and $R_{10}@5$, as in previous works [14].

Number of Stacked Attention Layers

We started by determining the number of stacked Attention layers. To do that, we arrange successive Attentive Modules, introduced in Figure 2.8, in the Representation Module. The purpose is to determine the best number of hierarchically added Attentive Modules for our final model. To do that, we trained the model stacking from 1 to 10 Attention layers. We used validation set to select the best model, by calculating the different Recalls. The model performances from the validation results are shown in the Figure 4.22.

As the figure shows, the best Recalls are obtained for 6 and 9 layers. So, we chose to work with 6 Attention layers because it gains the best scores on validation set, and it is less time consuming when running (The more we add layers, the more complicated the model becomes so it takes more time to compile).

[12]The source code is available at https://github.com/baidu/Dialogue/DAM

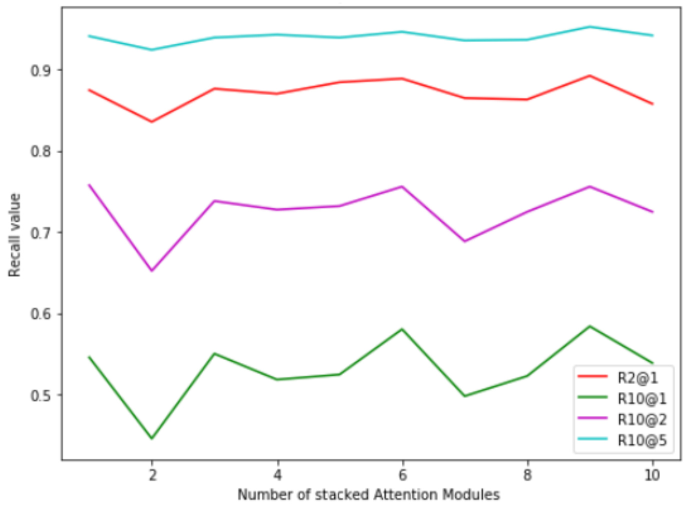


Figure 4.22: The Recalls with respect to the number of stacked Attention layers.

Loss Function

We tuned the model with Adam Optimizer [10] to minimize the loss function, introduced in the previous chapter equation (3.9). We choose this method as it is straightforward to implement, is computationally efficient and has little memory requirements. Moreover, Adam algorithm is well suited for problems that are large in terms of data, which is needed in our case. As shown in the Figure, the loss started with a value of 2.87 then it decreased gradually with the number of epochs. As the algorithm approached convergence the loss reached its minimum value of 0.70 . This demonstrates the effectiveness of the Adam Optimizer for minimizing the loss function with our OoredooTn dataset.

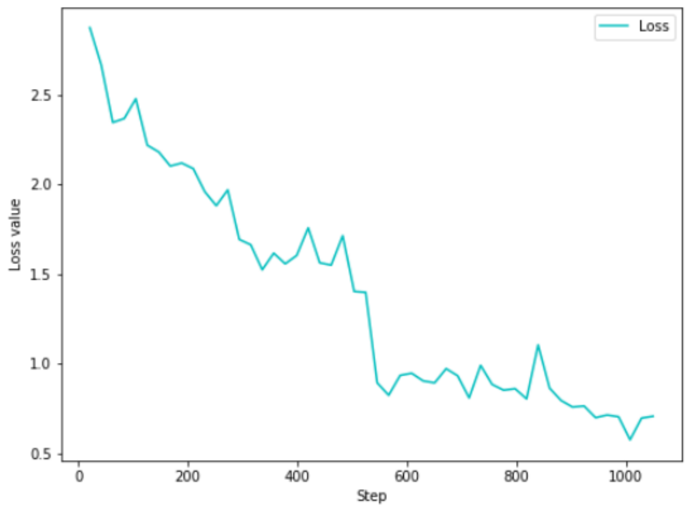


Figure 4.23: The loss function with respect to the number of steps.

Learning Rate

In this part, we want to adjust the weights of our network with respect the loss. To do that, we variate the hyper-parameter Learning rate. We should know that the lower the Learning rate value is, the slower but precise we are while searching the local minima.

In our case, the Learning rate was initialized as 0.001 and is gradually decreased at specific points during training. We choose to decrease the learning rate value as we get close to the loss convergence (see Figure 4.23). Here, we decreased twice the learning rate value with $0,0001$ at the 400^{th} and 800^{th} epochs. It reached the value of 0.0008 at the last epoch as illustrated in Figure 4.24.

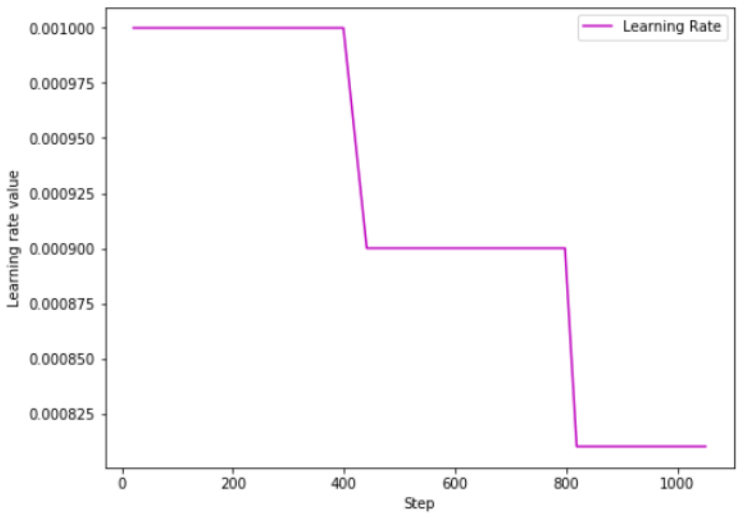


Figure 4.24: The Learning Rate with respect to the number of steps.

Through the three modules (Representation Module, Matching Module and Aggregation Module), we trained and validated the DAM model by selecting the optimal number of stacked Attention layer. We investigated the minimization of the Loss function with the Adam Optimizer and evaluated the variation of the Learning rate with respect to the epochs. We can summarize that the DAM model solved the third final goal, mentioned in the first chapter, by preparing a pre-trained model ready to test on the ”Multi-turn context response selection” NLP task.

4.2.3 Model Test

We now study the model performance by testing it over the unseen data test set. We varied parameters related to the data and investigated how the model performs. We based our experiments on the OoredooTn data statistics from the Figure 4.4.

Maximum Number of Turns per Context for Test

We started by evaluating the model with different utterance number. In this experiment, we fixed the number of stacked Attention layers to 6 and tested different maximum number of turns per context. Figure 4.25 illustrates the changes of the Recalls, introduced in equation4.1, on OoredooTn across contexts with different number of utterances.

As demonstrated, DAM shows good performances at matching response with short context that has only 2 utterances or less. It can still stably deal with long context length with more than 6 turns.

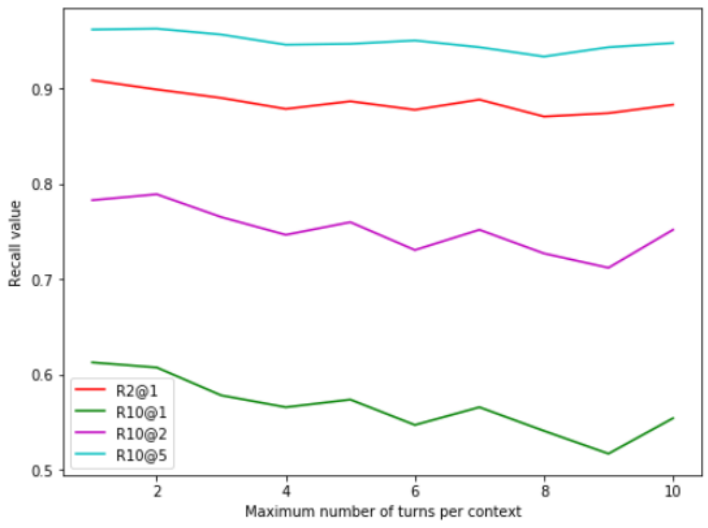


Figure 4.25: The Recalls with respect to the maximum number of turns per context.

Maximum Words per Context Test

We then evaluated the model on different maximum length of turns (word number per utterance) per context. We fixed the number of stacked Attention layers to 6, and the maximum number of turns to 2 then tried different maximum length of turns. The next figure illustrates the obtained results.

As can be seen, DAM reacts well when it comes to long turns (more then 10 words per turn). Unfortunately, the performance of matching short utterances, that have less than 10 words, is obviously lower. This is because the shorter the utterance text is, the fewer information it contains, and the more difficult for selecting the next utterance.

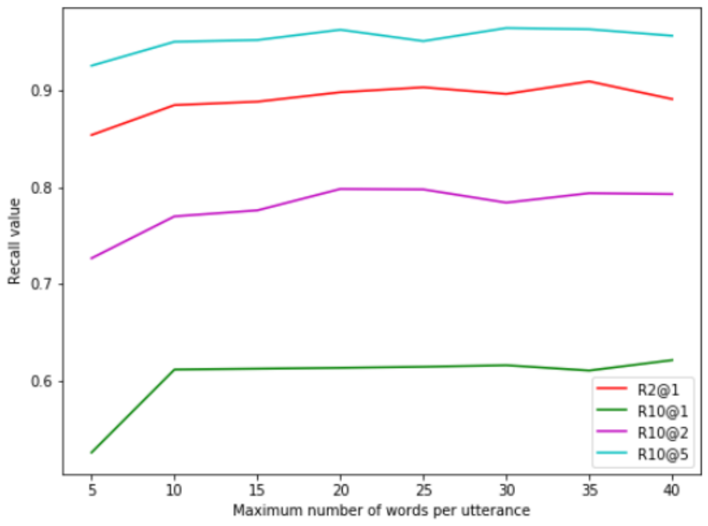


Figure 4.26: The Recalls with respect to the maximum number of words per turn.

Experimental Tests

We perform final 5 experiments in order to obtain the best model Recalls. We set parameters of the word-embedding size to 200 and the batch size to 32 [13]. We fixed the number of stacked Attention layers to 6, the maximum number of turns to 2, and the maximum length of turns to 35.

The experiment consists of extracting, 5 times, new train, validation and test sets, from the hole data. The different sets are extracted with the same percentages at each experiment as shown in Figure 4.27. We then re-perform training, validation and test experiments. Those experiences are meant to give as representative sets of the overall distribution of the data, by getting different sets of training, validation, and test data at each experience.

[13]The batch size refers to the number of training examples in one iteration

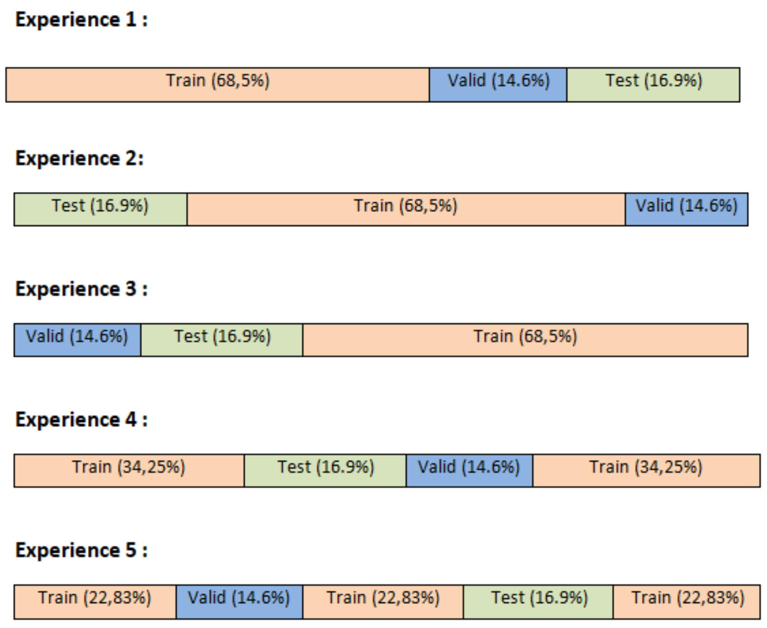


Figure 4.27: Different experiments.

In Table 4.3 are illustrated the obtained results. As the table shows, the 5 experiments have very close results with the OoredooTn data. But, experiment 5 obtained the best test Recalls .

- - -	R_2@1	R_{10}@1	R_{10}@2	R_{10}@5
experiment 1	0.882	0.563	0.760	0.951
experiment 2	0.880	0.544	0.749	0.947
experiment 3	0.896	0.565	0.761	0.948
experiment 4	0.894	0.584	0.765	0.955
experiment 5	**0.892**	**0.588**	**0.770**	**0.952**

Table 4.3: Answer-response system dataset different Recall experimental results.

We extract the best obtained test Recalls and compare our results with the previous ones on Ubuntu and Douban datasets, introduced in Table 4.1. We show the final test results on the table below.

Dataset	R_2@1	R_{10}@1	R_{10}@2	R_{10}@5
Ubuntu	0.938	0.767	0.874	0.969
Douban	- - -	0.254	0.410	0.757
OoredooTn	**0.892**	**0.588**	**0.770**	**0.952**

Table 4.4: Answer-response system dataset best Recall results.

Our final results are far behind Douban Chineese data results as they outperform them in all the tested Recalls. They are still under but close to the Ubuntu English dataset. This shows the performance of DAM model in different language dataset even with informal ones. We can summarize that through testing the pre-trained DAM model, we solved the fourth and last goal, mentioned in the first chapter. In fact, we delivered test results that revealed the model performances on the ”Multi-turn context response selection” NLP task.

Conclusion

In this chapter, we introduced the collected Tunisian dataset. We explained the way it was collected, cleaned and organized. We exposed some statistics about the data. Our data is comparable with the freely available dataset for answer-response systems, as it shows close statistics.

In a second part, we investigated matching a response with its multi-turn context using dependency information based entirely on Attention with the DAM model. Different test done with different parameters showed the good performance of our model. Empirical results on our dataset OoredooTn demonstrated the effectiveness of the Attention in multi-turn response selection.

Conclusion

To conclude, in this graduation project we worked on the problematic of "Building a Language Model for Tunisian Dialect". We tackled the project from both research and industrial aspects. We started by presenting the project challenges ranging from the diversity of the NLP used model to the complexity of our Tunisian dialect and the lack of the available dataset.

In order to guarantee the understanding of the required theoretical knowledge, we introduced the state of the art Neural Networks commonly used in the NLP field. This phase allowed us to get a well-founded understanding of the different Neural Networks. In the next phase, we introduced our chosen NLP task: the Multi-turn context response selection task. After that we got an overview of a successful model DAM fully based on Attention, one of the recent advanced models in NLP.

Finally, we described the several steps taken to collect, clean and organize our Tunisian Dialect corpus OoredooTn. Statistics demonstrated that our data is comparable to freely available ones for response-answer systems. We then introduced the implementation of the proposed approach. During the implementation, we faced a lot of challenges, that ranged from theoretical to technical ones. Finally, we described the experimental setup as well as the experimental results where we interpreted every experiment and ended up comparing our results against other dataset results.

The proposed solution was successful. In fact, the approach has proven very exciting results in the Word Embedding module as it succeeded in tackling all the Tunisian dialect challenges that we described in the first chapter. Adding to that, the overall test results outperformed Douban dataset's result and are close to the Ubuntu dataset, the two answer-response comparable datasets. Moreover, the pre-trained and tested model can be used for future InstaDeep up-coming project in the field of Tunisian language processing models and similar areas. One other promising result is that the model can still be used in any new Tunisian dataset for even more exciting results. It can also be tune for different wordwide dialects.

However, our solution is considered as a first version. Various improvements can be added. On the first hand, the collected OoredooTn dataset can still be enlarged. And additional dataset can be added to our repository. On the other hand, DAM model that is only one example of a solution that addresses the problem proposed by the host company. This later was able to solve the problem of Multi-Turn Context Response Selection task, not in its integrity, but in a large part. In fact, the model results can still be improved by choosing the best hyper-parameters and by reducing the running time.

I am pleased to express my satisfaction with this graduation project internship, which was for me a very rewarding experience. During the internship period, I discovered the field of Artificial Intelligence from a theoretical but also from an industrial side. I was lead to understanding more deeply this science and to applying it in an industrial problem. Adding to this, I had the opportunity to consolidate my knowledge with the Python programming thanks to sharing the know-how of the whole team of the AI department. I have been enrolled to work within a skilled team, and I learned a lot from this great experience.

Bibliography

[1] Google Brain. Attention is all you need. page 4, Decembre 2017.

[2] dataanalyticspost.com. T-ENS. (Visited on May 27, 2019).

[3] deeplearningindaba.com. Deep learning indaba. (Visited on June 03, 2019).

[4] Alex Graves. Supervised sequence labelling with recurrent neural networks. pages 18–20, 2012.

[5] Alex Graves. Supervised sequence labelling with recurrent neural networks. pages 31–38, 2012.

[6] Hadhemi Achour Jihene Younes and Emna Souissi. Constructing linguistic resources for the Tunisian dialect using textual user-generated contents on the social web. 2015.

[7] Till. Jünger, Jakob / Keyling. Facepager. 2018.

[8] Rolf Ingold Marc Bui Karim Sayadi, Marcus Liwicki. Tunisian dialect and modern standard arabic dataset for sentiment analysis : Tunisian election context. 2015.

[9] Mccormickml. Word2vec tutorial the skip gram model. (Visited on May 19, 2019).

[10] Adam Coates Abhik Lahiri Bobby Prochnow Quoc V Le, Jiquan Ngiam and Andrew Y Ng. In proceedings of the 28th international conference on international conference on machine learning. 2011.

[11] Iulian V.Serban† Ryan Lowe, Nissan Pow and Joelle Pineau. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. 2015.

[12] Yannick Estève Salima Mdhaffar, Fethi Bougares and Lamia Hadrich-Belguith. Sentiment analysis of tunisian dialect: Linguistic resources and experiments. 2017.

[13] towardsdatascience.com. A step by step explanation of principal component analysis. (Visited on May 27, 2019).

[14] Daxiang Dong Yi Liu Ying Chen Wayne Xin Zhaoy Dianhai Yu Xiangyang Zhou, Lu Li and Hua Wu. Multi-turn response selection for chatbots with deep attention matching network. pages 1120–1122, July 2018.

[15] Daxiang Dong Yi Liu Ying Chen Wayne Xin Zhaoy Dianhai Yu Xiangyang Zhou, Lu Li and Hua Wu. Multi-turn response selection for chatbots with deep attention matching network. page 1121, July 2018.

[16] Lei Yu. Tackling sequence to sequence mapping problems with neural networks. pages 09–12, Octobre 2018.

[17] Chen Xing Zhoujun Li Ming Zhou Yu Wu, Wei Wu. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. 2017.

[18] Pengfei Zhu Hai Zhao Gongshen Liu Zhuosheng Zhang, Jiangtong Li. Modeling multi-turn conversation with deep utterance aggregation. 2018.

Appendix

In this appendix, a description of the related background is presented which will provide the essential knowledge on the discussed topic all over the report.

A. Activation Functions

The activation function can be the logistic Sigmoïd function σ, expressed in equation (4.2) and shown in graph, the Hyperbolic tangent function tanh, expressed in equation (4.3) and shown in graph, or the Rectified Linear Unit function *ReLu*, expressed in equation (4.4) and shown in graph.

$$\sigma(x) = \frac{1}{1 + \exp(x)} \tag{4.2}$$

$$\tanh(x) = \frac{\exp(2x) - 1}{\exp(2x) + 1} \tag{4.3}$$

$$ReLu(x) = \max(0, x) \tag{4.4}$$

B. Neural Networks' Equations

B.1 Multil-Layer Perceptron

Here after we formalize the MLPs equations [16]. We consider an MLP with L layers, J neurons on the $l - th$ layer and I neurons on the $(l + 1) - th$ layer. We set w_{ij}^{l} the weight from neuron j in layer l to neuron i in layer $l + 1$.

B.1.1 Forward Propagation

The forward pass can be represented by the equations (4.5) and (4.6). Let w_{ij} be the weight of the connection from unit i to unit j. The term a_i^l is the network input to the i^{th} neuron of layer l, b_j^l is the activation from the j^{th} neuron of layer l, and f represents the activation function. Some examples of activation functions are presented in the Appendix.

$$a_i^{l+1} = \sum_{j=1}^{J} w_{ij}^l b_j^l \qquad \forall l = 1...L\,,\, i = 1...I \tag{4.5}$$

$$b_i^{l+1} = f(a_i^{l+1}) \qquad \forall l = 1...L\,,\, i = 1...I \tag{4.6}$$

The predicted output vector $\hat{y}$ of an MLP is given by the activation of the neurons in the output layer. The network input a_k to each output neuron k is calculated by summing over the neurons connected to it at the final layer L, as expressed in the equation (4.7).

$$a_k = \sum_{h=1}^{H_L} w_{hk} b_h \qquad \forall k = 1...K \tag{4.7}$$

For classification problems with $K > 2$ classes, the convention is to have K output units. To normalise the activation of the K output neurons, we apply the Softmax function to obtain the predicted output $\hat{y}$, which represents the class probabilities, as expressed in the following equation (4.8).

$$\phi(a_k) = \hat{y_k} = \frac{\exp(a_k)}{\sum_{k'=1}^{K} \exp a_{k'}} \qquad \forall k = 1...K \tag{4.8}$$

B.1.2 The Objective Function

In most cases, the objective function L follows the principle of *Maximum Likelihood*. In this principle, we aim to minimise the cross-entropy error between the predicted outputs $\hat{y}$ and the true targets y.

For problems with multiple classes, we have:

$$L(y, \hat{y}) = -\ln p(\hat{y}|y) \tag{4.9}$$

known that:

$$p(\hat{y}|y) = \prod_{k=1}^{K} y_k^{\hat{y}} \tag{4.10}$$

We get,

$$L(y, \hat{y}) = -\sum_{k=1}^{K} \hat{y} \ln y_k \tag{4.11}$$

Neural networks are trained using numerical optimisation methods such as *Gradient Descent* (GD). The GD algorithm iteratively reduces the loss function L, previously defined, using the following update equation (**??**). Here, η is the learning rate, $\nabla_\theta L$ is the gradient of the objective function L with respect to the parameters θ.

$$\theta \leftarrow \theta - \eta \nabla_\theta L \tag{4.12}$$

B.1.3 Backward Propagation

The backward propagation algorithm provides an efficient way of calculating the gradient of the loss function with respect to the parameters. The algorithm essentially applies the chain rule repeatedly for the partial derivatives at each layer.

The backward pass starts by calculating the derivatives of the loss function with respect to the output nodes δ_k^L, as shown in equation (4.13):

$$\delta_k^L := \frac{\partial L}{\partial a_k^L} = \frac{\partial L}{\partial \hat{y_k}} \frac{\partial \hat{y_k}}{\partial a_k^L} = \frac{\partial L}{\partial \hat{y_k}} \phi'(a_k^L) \qquad \forall k = 1...K \tag{4.13}$$

For the derivative of the parameters in the hidden layers, we introduce the error term δ_j^l that measures how much the $j - th$ node at the $l - th$ layer is responsible for the error in the output. The expression of δ_j^l is calculated based on the chain rule, as follows (4.14). The use of the summation is due to the fact that the activation of every node in the $l - th$ layer contributes to the activation of any node in the $(l + 1) - th$ layer.

$$\delta_j^l := \frac{\partial L}{\partial a_j^l} = \sum_{i=1}^{I} \frac{\partial L}{\partial a_i^{l+1}} \frac{\partial a_i^{l+1}}{\partial b_j^l} \frac{\partial b_j^l}{\partial a_j^l} = \sum_{i=1}^{I} \delta_j^{l+1} w_{ij}^l f'(a_j^l) \qquad \forall l = 1...L\,,\ j = 1...J \tag{4.14}$$

Therefore, the error term of the nodes in the hidden layers can be calculated recursively. Having the values b_j^l calculated from the forward pass, and δ_j^l obtained from the backward pass, we arrive at the derivatives of the loss function with respect to the network weights:

$$\frac{\partial L}{\partial w_{ij}^l} = \frac{\partial L}{\partial a_i^{l+1}} \frac{\partial a_i^{l+1}}{\partial w_{ij}^l} = \delta_i^{l+1} b_j^l \qquad \forall l = 1...L\,,\ i = 1...I\,,\ j = 1...J \tag{4.15}$$

B.2. Recurrent Neural Networks

The RNNs' equations [4] are given for a single block only. For multiple blocks, the calculations are simply repeated for each block, in any order. The network is composed of I hidden units, and K output units.

B.2.1. Forward Pass

The forward pass can be represented by the equations (4.16) and (4.17). Let w_{ij} be the weight of the connection from unit i to unit j. The term x_j^t is the value of the j^{th} input at the time-step t, and a_i^t is the network input to the i^{th} unit at the time-step t and b_i^t is the activation of the i^{th} unit at time-step t.f is a nonlinear, differentiable activation function.

$$a_i^t = \sum_{j=1}^{J} w_{ij} x_j^t + \sum_{i'=1}^{I} w_{i'i} b_{i'}^{t-1} \qquad \forall i = 1...I\,,\, t = 1...T \tag{4.16}$$

$$b_i^t = f(a_h^t) \qquad \forall i = 1...I\,,\, t = 1...T \tag{4.17}$$

The network inputs to the output units can be calculated at the same time as the hidden activations. The network's input a_k^t to each output unit k is calculated by summing over the units connected to it, exactly as for a hidden unit.

$$a_k^t = \sum_{i=1}^{I} w_{ik} b_i^t \qquad \forall i = 1...I\,,\, t = 1...T \tag{4.18}$$

The output vector $\hat{y}$ of an RNN is given by the activation of the units in the output layer.

$$\phi(a_k) = \hat{y_k} = \frac{\exp(a_k)}{\sum_{k'=1}^{K} \exp a_{k'}} \qquad \forall k = 1...K \tag{4.19}$$

B.2.2. Backward Pass

We use the same loss function as for the MLPs, derived from the principle of $Maximum\ Likelihood.$ Given the partial derivatives of the differentiable loss function L with respect to the network outputs, the next step is to determine the derivatives with respect to the weights.

The back-propagation through time-step consists of a repeated application of the chain rule of derivation. We start by calculating the derivatives of the loss function with respect to the output nodes δ_k^L, as shown in equation (4.20):

$$\delta_k^L := \frac{\partial L}{\partial a_k^L} = \frac{\partial L}{\partial \hat{y}_k}\frac{\partial \hat{y}_k}{\partial a_k^L} = \frac{\partial L}{\partial \hat{y}_k}\phi'(a_k^L) \qquad \forall k = 1...K \tag{4.20}$$

For the hidden layers, the error δ_j^t for nodes at time t is not only propagated from the output layer but also from the hidden layer at time $t+1$. Therefore, we have:

$$\delta_j^t := \frac{\partial L}{\partial a_j^t} = \sum_{i=1}^{I} \frac{\partial L}{\partial a_i^{t+1}}\frac{\partial a_i^{t+1}}{\partial b_j^t}\frac{\partial b_j^t}{\partial a_j^t} = f'(a_j^t)(\sum_{i=1}^{I}\delta_i^t w_{ij}^l + \sum_{i'=1}^{I}\delta_i^{t+1\prime} w_{ii}^l) \qquad \forall t = 1...T\,,\ j = 1...J \tag{4.21}$$

Finally, we have the derivatives of the loss function with respect to the parameters:

$$\frac{\partial L}{\partial w_{ij}} = \sum_{t=1}^{T}\frac{\partial L}{\partial a_j^t}\frac{\partial a_j^t}{\partial w_{ij}} = \sum_{t=1}^{T}\delta_j^t b_i^t \qquad \forall i = 1...I\,,\ j = 1...J \tag{4.22}$$

B.3. Long Short-term Memory

We present here the network equations [5]. In the next subsections, the forward pass is calculated for a length T input sequence x by starting at $t=1$ and recursively applying the update equations while incrementing t, and the backward pass is calculated by starting at $t=T$, and recursively calculating the unit derivatives while decrementing t to one. The final weight derivatives are found by summing over the derivatives at each time-step.
The network is composed of I input units, H hidden units, K output units, C cell memory, and G total input to the hidden layer (including cells and gates).

B.3.1 Forward Pass

We use the same notation as for the RNN network. Let w_{ij} be the weight of the connection from unit i to unit j, x_i^t be the value of the i^{th} input at the time-step t, and let a_j^t be the network input to the j^{th} unit at the time-step t and b_j^t be the activation of the j^{th} unit at time-step t.

The subscripts ι, ϕ and ω refer respectively to the Input Gate, Forget Gate, and Output Gate of the block.

For the Input Gate,

$$a_\iota^t = \sum_{i=1}^{I} w_{i\iota} x_i^t + \sum_{h=1}^{H} w_{h\iota} b_h^{t-1} + \sum_{c=1}^{C} w_{c\iota} s_c^{t-1} \qquad \forall t = 1...T \tag{4.23}$$

$$b_\iota^t = f(a_\iota^t) \qquad \forall t = 1...T \tag{4.24}$$

For the Forget Gate,

$$a_\phi^t = \sum_{i=1}^{I} w_{i\phi} x_i^t + \sum_{h=1}^{H} w_{h\phi} b_h^{t-1} + \sum_{c=1}^{C} w_{c\phi} s_c^{t-1} \qquad \forall t = 1...T \tag{4.25}$$

$$b_\phi^t = f(a_\phi^t) \qquad \forall t = 1...T \tag{4.26}$$

For the Cell Gate,

$$a_c^t = \sum_{i=1}^{I} w_{ic} x_i^t + \sum_{h=1}^{H} w_{hc} b_h^{t-1} \qquad \forall t = 1...T \tag{4.27}$$

$$s_c^t = b_\phi^t s_c^{t-1} + b_c^t g(a_c^t) \qquad \forall t = 1...T \tag{4.28}$$

For the Output Gate,

$$a_\omega^t = \sum_{i=1}^{I} w_{i\omega} x_i^t + \sum_{h=1}^{H} w_{h\omega} b_h^{t-1} + \sum_{c=1}^{C} w_{c\omega} s_c^{t} \qquad \forall t = 1...T \tag{4.29}$$

$$b_\omega^t = f(a_\omega^t) \qquad \forall t = 1...T \tag{4.30}$$

And for the Cell Output,

$$b_c^t = b_w^t h(s_c^t) \qquad \forall t = 1...T \tag{4.31}$$

B.3.3 Backward Pass

The back-propagation through time-step consists of a repeated application of the chain rule of derivation. We use the same loss function as for the RNN model.

We set:

$$\delta_j^t := \frac{\partial L}{\partial a_j^t} \qquad \forall t = 1...T \tag{4.32}$$

$$\epsilon_c^t := \frac{\partial L}{\partial b_c^t} \qquad \forall t = 1...T \tag{4.33}$$

$$\epsilon_s^t := \frac{\partial L}{\partial s_c^t} \qquad \forall t = 1...T \tag{4.34}$$

For the Cell output,

$$\epsilon_c^t = \sum_{k=1}^{K} w_{ck}\delta_k^t + \sum_{g=1}^{G} w_{cg}\delta_g^{t+1} \qquad \forall t = 1...T \tag{4.35}$$

For the Output Gate,

$$\delta_\omega^t = \frac{\partial L}{\partial a_\omega^t} = \frac{\partial L}{\partial b_c^t}\frac{\partial b_c^t}{\partial b_\omega^t}\frac{b_\omega^t}{\partial a_\omega^t} = \epsilon_c^t h(S_c^t)f'(a_\omega^t) \qquad \forall t = 1...T \tag{4.36}$$

For the States,

$$\epsilon_s^t = b_\omega^t h'(s_c^t)\epsilon_c^t + b_\phi^{t+1}\epsilon_s^{t+1} + w_{c\iota}\epsilon_\iota^{t+1} + w_{c\phi}\delta_\phi^{t+1} + w_{c\omega}\delta_\omega^t \qquad \forall t = 1...T \tag{4.37}$$

For the Cells,

$$\delta_c^t = \frac{\partial L}{\partial a_c^t} = \frac{\partial L}{\partial s_c^t}\frac{\partial s_c^t}{\partial b_\omega^t}\frac{b_\omega^t}{\partial a_c^t} = \epsilon_s^t b_c^t g'(a_c^t) \qquad \forall t = 1...T \tag{4.38}$$

For the Forget Gate,

$$\delta_\phi^t = \frac{\partial L}{\partial a_\phi^t} = \frac{\partial L}{\partial s_c^t}\frac{\partial s_c^t}{\partial b_\phi^t}\frac{b_\phi^t}{\partial a_\phi^t} = \epsilon_s^t s_c^{t-1} f'(a_\phi^t) \qquad \forall t = 1...T \tag{4.39}$$

And for the Input Gate,

$$\delta_\iota^t = \frac{\partial L}{\partial a_\iota^t} = \frac{\partial L}{\partial s_c^t}\frac{\partial s_c^t}{\partial b_\iota^t}\frac{b_\iota^t}{\partial a_\iota^t} = \epsilon_\iota^t s_c^{t-1} g(a_\iota^t)f'(a_\iota^t) \qquad \forall t = 1...T \tag{4.40}$$