الجمهورية الجزائرية الديمقراطية الشعبية

Ministry of Higher Education and Scientific Research

YAHIA FARES UNIVERSITY OF MEDEA

Faculty of Science

Department of Mathematics and Computer Science



Graduation Project

Submitted for graduation in

Computer science master

Specialty: Systems Engineering and Web Technology

THEME :

Design and Implementation of a deep learning model for an automatic translation sequences between English and Arabic

Proposed and directed by

Mr. HADJIR ISMAIL

Prepared and Presented by

RAHMOUNI AMIRA

BOUCHERIT KHAOULA

Academic year : 2020 – 2021

بِسْمِ اللهِ الرَّحْمنِ الرَّحِيمِ

"وَقُلِ اعْمَلُوا فَسَيَرَى اللَّهُ عَمَلَكُمْ وَرَسُولُهُ وَالْمُؤْمِنُونَ وَسَتُرَدُّونَ إِلَىٰ عَالِمِ الْغَيْبِ وَالشَّهَادَةِ فَيُنَبِّئُكُم بِمَا كُنتُمْ تَعْمَلُونَ"

صدق الله العظيم

(التوبة 105)

اللهم اجعل هذا العمل خالصا لوجهك الكريم؛ ولا تجعلنا من الذين ضل سعيهم فى الحياة الدنيا وهم يحسبون انهم يحسنون صنعا

Thanks

At the beginning, we thank God Almighty who provided us with the strength necessary to accomplish this work successfully.

We would like to express our deep gratitude to our adviser, Mr. Hadjir, for his constant guidance, challenged discussions, tips, enthusiasm and knowledge. We also made a great effort to build up the structure and improve every detail of our work. We are grateful to him for working with us. It was an honor for us.

We express our deep appreciation to the members of the jury for their interest in accepting and reviewing our work.

Our eternal gratitude to our parents who encouraged us to be the best we can be and we have high expectations and constant prayer for our success .

Also, we are grateful to our entire family and friends for their encouragement and support.

We would also like to thank all the professors of Medea who helped us achieve what we are now.

We also sincerely thank the Information Department for facilitating the means to accomplish this work.

Finally, we would like to thank all those who participated in or near this work.

“We hope you like it our memory.”

Medéa, September 2021

Khaoula , Amira .

Dedication

Praise God, the one and only".

That this modest work, a little towards others bigger,

But he will always remain modest before your mercy.

I dedicated this humble work

To my dear mother and dear father,

Who always supported me during my academic

career in particular and life in general.

To my seven sisters and brothers Mohammed and Ahmed.

To all my dear friends "Anfal, Maryam, Aya, Shaima, Karima".

To my partner in the project (Amira).

To everyone who helped or supported me, even if it was a word

To all my teachers throughout my educational career.

For everyone who knows me .

«B.Khaoula»

Dedication

"Praise God, the one and only".

That this modest work, a little towards others bigger,

But he will always remain modest before your mercy.

I dedicated this modest work

To my dear mother and father,

Who always supported during my life, my studies, and

encourage moving forward.

To My sister "Amani" and

My brothers "Mohammed, Mounir, Marouane and Sami Yousef".

To all My dear friends "Mouna, Aya, Chayma, Karima, Sara, Mbarka, Ahlem, Sara ".

To my partner (Khaoula).

To all the promotion of 2nd year master's degree in

Computer Science (2020/2021).

To everyone who knows me.

«R. Amira»

Abstract

Abstract:

Since the turn of the century, machine translation has experienced considerable growth on the Web, with several online systems capable of automatically translating increasingly long Web pages or texts in a matter of seconds. It is a much appreciated help from the general public. Globalization has had considerable effects on the development of the language industry, and more particularly in machine translation where demand continues to grow. Thus, the need for more reliable machine translation systems is increasing . That is why we are interested in designing a system for machine translation (translating text from one language to another, in our case from English to Arabic), and in order to implement this process, we used a modern adaptation of machine translation: neural machine translation. This approach depends largely extent on the neurons in our brain and how we learn as humans. We proposed an approach to create a translation system between English and Arabic , and then test it on a set of sentences, paragraphs and short texts. In order to achieve this, we built a model that processes and translates some English sequence as a sequence of words , producing sentences that are also made up of a sequence of Arabic words . More advanced MT models usually process sentences word for by word . This method used is based on the encoding / decoding model in which both the encoder and decoder units are Long Short Term Memory models, which in turn are considered as type of Recurrent Neural Networks and the latter is also classified as a type of Neural Networks Neural Machine Translation relies on Neural Networks that are part of Natural Language Processing and implicit in Deep Learning .According to the results we evaluated by Bleu score, the translation was close to the average.

Keywords : machine translation , encoding / decoding structure , Long Short Term Memory , Recurrent Neural Networks , Natural Language Processing , Deep Learning .

ملخص

ملخص :

منذ مطلع القرن ، شهدت الترجمة الآلية نموًا كبيرًا عبر الشبكة العنكبوتية ، مع وجود العديد من الأنظمة عبر الإنترنت القادرة تلقائيًا على ترجمة أكبر صفحات الشبكة العنكبوتية أو النصوص في غضون ثوانٍ معدودة. هذه المساعدة كانت محل تقدير كبير من عامة الناس .كان للعولمة تأثيرات كبيرة على تطور صناعة اللغة ، وخاصة في الترجمة الآلية حيث يستمر الطلب في النمو. وبالتالي ، تزداد الحاجة إلى أنظمة ترجمة آلية أكثر موثوقية . لهذا نحن مهتمون بتصميم نظام للترجمة الآلية ,ومن أجل تنفيذ هذه العملية، استخدمنا تكيفًا حديثًا للترجمة الآلية: الترجمة الآلية العصبية. يعتمد هذا النهج إلى حد كبيرعلى الخلايا العصبية الدماغية و كيفية تعلم البشر.اقترحنا طريقة منهجية لإنشاء نظام ترجمة من الإنجليزية إلى العربية، ثم اختباره على مجموعة من الجمل والفقرات والنصوص القصيرة. من أجل تحقيق ذلك ، قمنا ببناء نموذج يعالج ويترجم بعض التسلسلات الإنجليزية كسلسلة من الأحرف الكلمات ، وينتج جملًا تتكون أيضًا من سلسلة من الكلمات العربية الموافقة لها. تتمثل الطريقة المتبناة في استعمال بنية التشفير/ فك التشفير، بحيث أن كل من وحدة التشفير ووحدة فك التشفير هي عبارة عن نماذج الذاكرة طويلة المدى و التي تعتبر بدورها كنوع من الشبكات العصبية المتكررة. إن الترجمة الآلية العصبية تعتمد على الشبكات العصبية والتي تدخل في إطار معالجة اللغات الطبيعية والمنطوية تحت غطاء التعلم المعمق . وفقًا للنتائج التي قمنا بتقييمها من خلال تقييم الطالب ثنائي اللغة ، كانت الترجمة قريبة من المتوسط .

الكلمات الرئيسية : الترجمة الآلية ، بنية التشفير / فك التشفير ، الذاكرة طويلة المدى ، الشبكات العصبية المتكررة ، معالجة اللغة الطبيعية ، التعلم العميق.

Résumé :

Depuis le début du siècle, la traduction automatique a connu une croissance considérable sur le Web, avec plusieurs systèmes en ligne capables de traduire automatiquement des pages Web ou des textes de plus en plus longs en quelques secondes. C'est une aide très appréciée du grand public.La mondialisation a eu des effets considérables sur le développement de l'industrie de la langue, et plus particulièrement dans la traduction automatique où la demande ne cesse de croître. Ainsi, le besoin de systèmes de traduction automatique plus fiables augmente .C'est pourquoi nous nous intéressons à la conception d'un système de traduction automatique (traduire du texte d'une langue à une autre, dans notre cas de l'Anglais vers l'Arabe), et afin de mettre en œuvre ce processus, nous avons utilisé une adaptation moderne de la traduction automatique : la machine neuronale Traduction. Cette approche dépend dans une large mesure des neurones de notre cerveau et de la façon dont nous apprenons en tant qu'êtres humains.Nous avons proposé une approche permettant de créer un système de traduction de l'anglais vers l'arabe, puis de le tester sur un ensemble de phrases, de paragraphes et de textes courts. Pour ce faire, nous avons construit un modèle qui traite et traduit une séquence en anglais sous forme de séquence de mots , produisant des phrases qui sont également constituées d'une séquence de mots en Arabe . Les modèles de TA plus avancés traitent généralement les phrases mot par mot .La méthode adoptée consiste à utiliser la structure de codage/décodage dans laquelle les unités d'encodeur et de décodeur sont des modèles de mémoire à long court terme, qui à leur tour sont considérés comme un type de réseaux de neurones récurrents et ce dernier est également classé comme un type de réseaux de neurones. .La traduction automatique neuronale repose sur des réseaux neuronaux qui font partie du traitement de la langue naturelle et implicite dans l'apprentissage approfondie. D'après les résultats que nous avons évalués par score Bleu, la traduction était proche de la moyenne.

Mots-clés : traduction automatique, structure codage / décodage, mémoire à long terme, réseaux de neurones récurrents, réseaux de neurones , apprentissage en profondeur.

Table of content :

Chapitre I : State Of The Art 1
1. Introduction : 2
2. Natural language processing (NLP) : 2
2.1 Natural language processing definition : 2
2.2 A brief history of Natural language processing : 3
2.3 Steps of Natural language processing : 4
2.4 Major tasks in NLP : 5
2.5 Approaches to Natural language processing : 6
2.6 Natural Language Processing Applications : 8
2.7 Future of Natural language processing : 8
3. Machine Translation : 9
3.1 Machine Translation Definition : 9
3.2 A brief history of machine translation : 9
3.3 The translation process : 11
3.4 The different approaches to machine translation : 11
4. Neural Network : 16
4.1 Neural Network definition : 16
4.2 A Brief History of Neural Networks : 16
4.3 Architecture of Neural Network : 16
4.4 Components of Neural Network : 17
4.5 Algorithms of Neural Network : 18
5. Neural Machine Translation : 19
5.1 Neural Machine Translation definition : 19
5.2 A brief history of Neural Machine Translation : 19
5.3 Properties : 20
5.4 How does NMT Work? : 20
6. Deep Learning : 21
6.1 Deep Learning Definition : 21
6.2 What Is the Difference between Deep Learning and Machine Learning?: 21
6.3 Deep Learning type : 23
6.4 Difference between Neural Network and Deep Learning Neural Network : 26
6.5 Deep learning application and examples: 27

Table of content

7. Conclusion :.....29
Chapter II : Deep learning and Machine translation. 30
1. Introduction : 31
2. Recurrent Neural Networks :.....31
2.1 Recurrent Neural Networks Definition : 31
2.2 Why Recurrent Neural Networks ? : 31
2.3 Advantages & Disadvantages of Recurrent Neural Network : 32
2.4 Types of recurrent neural networks : 33
2.4 How a Recurrent Neural Network Works ? : 36
2.6 Applications of Recurrent Neural Networks :.....39
2.7 Variant RNN architectures : 40
3. Long Short Term Memory (LSTM) :.....41
3.1 LSTM Definition : 41
3.2 Workings of LSTM : 41
3.4 Architecture of LSTM : 48
4. Encoder-Decoder Sequence to Sequence Model : 49
4.1 Sequence-to-Sequence Model : 49
4.2 Encoder-Decoder system : 51
5. Word Embeddings : 52
5.1 What Are Word Embeddings ? : 52
5.2 Word Embedding Algorithms : 53
5.3 Using Word Embeddings : 55
6. BLEU scores :.....56
6.1 What it BLEU :.....56
6.2 How it works : 57
6.3 BLEU score algorithm : 59
7. Conclusion :.....65
Chapter III : Proposed methodology. 66
1. Introduction : 67
2. Global Architecture :.....67
3. Process steps :.....69
3.1 Data Preparation : 69
3.2 Training the Model : 78
3.3 Testing the Machine Translation : 80
4. Conclusion :.....81

Table of content

Chapitre VI : Implementation , Results and discussions. 82
1. Introduction : 83
2. Development environment : 83
2.1 Python : 83
2.2 Python Features : 83
2.3 Python module used : 85
2.4 Google Colab : 86
2.5 Google Drive : 87
3. Explanation of our approach : 88
3.1 Dataset : 88
3.2 Data download : 89
3.3 Dataset preprocessing : 90
3.4 Model Development : 93
3.5 Create the model : 96
3.6 training the model : 98
3.7 Testing and Result : 100
4. Conclusion : 121
General conclusion : 122
Web references : 125
Books : 125
Articles references : 129

List of figures and tables:

List of Figures :

Figure 1 : SWAC type computers in 1950.....10
Figure 2: Triangle de Vauquois, representation of the different architectures 12
Figure 3 : Example of translation of a nominal group in English into a sentence in Arabic...13
Figure 4 : Example of the architecture of a statistical machine translation system 14
Figure 5 : Components of Neural Network 17
Figure 6 : Difference between Machine Learning and Deep Learning. 23
Figure 7: Architecture of Simple Neural Network and Deep Learning Neural Network. 27
Figure 8: Tupe of RNN : One-to-one 33
Figure 9: Tupe of RNN : One-to-many 34
Figure 10: Tupe of RNN : Many-to-one 34
Figure 11: Tupe of RNN : Many-to-many [Tx=Ty].....35
Figure 12: Tupe of RNN : Many-to-many [Tx!=Ty] 36
Figure 13: Processing sequence one by one 36
Figure 14: Passing hidden state to next time step 37
Figure 15: RNN Cell 37
Figure 16: Tanh squishes values to be between -1 and 1 38
Figure 17: Vector transformations without tanh 38
Figure 18: Vector transformations with tanh 38
Figure 19: LSTM Gates 42
Figure 20: Read the review and then decide if the product is good or bad. 43
Figure 21: The brain subconsciously remembers only the important keywords. 43
Figure 22: LSTM Cell and It's Operations 44
Figure 23: Sigmoid squishes values to be between 0 and 1 45
Figure 24: Forget gate operations 46
Figure 25: Input gate operations 47
Figure 26: Calculating cell state 47
Figure 27: Output gate operations 48
Figure 28: Sequence-to-Sequence Model 49
Figure 29: Neural Machine Translation "Sequence-to-Sequence Model". 49
Figure 30: Encoder-Decoder Model 50
Figure 31: Neural Machine Translation "Encoder-Decoder Model". 50
Figure 32: The context is a vector of floats. 51
Figure 33: Encoder-Decoder Architecture 52
Figure 34: Word2Vec Training Models 54
Figure 35: Example of Bleu score Work 58
Figure 36: General architecture of our proposed approach 68
Figure 37: Steps in the neural network model life cycle 71
Figure 38: Example of Method of Teacher forcing 72
Figure 39: input layer 74
Figure 40: Google Drive applications and functions 88
Figure 41: Dataset example from the Dataset 89
Figure 42: Load and divide dataset 89
Figure 43: Split dataset to Train / Test 90
Figure 44: Shape Train / Test 90
Figure 45: Filtering dataset 91

List of figures and tables:

Figure 46: Split dataset to input and target sequence 91
Figure 47: Determine the beginning and end of target sequence 91
Figure 48: Example of Arabic and English vocabulary 92
Figure 49: Creation of dictionaries 92
Figure 50: Creation Input Encoder layer 94
Figure 51: Creation encoder embedding layer 94
Figure 52: Creation encoder LSTM layer 94
Figure 53: Creation encoder states 95
Figure 54: Creation decoder layers 95
Figure 55: Creation of model 96
Figure 56: Compile and summary the model 96
Figure 57: Show Model Summary 97
Figure 58: Graph Drawing of model 97
Figure 59: Plot of Model Graph for NMT 98
Figure 60: Fit model 99
Figure 61: Result of fit model 99
Figure 62: Exemple of dataset 'ara.txt' 101
Figure 63: Show example of train Neural translation english-arabic 101
Figure 64: Show reslt of BLEU train 102
Figure 65: Show example of test Neural translation english-arabic 102
Figure 66: Show reslt of BLEU test 103
Figure 67: Exemple of dataset 'ara.txt' 103
Figure 68: Show example of train Neural translation arabic -english 104
Figure 69: Show reslt of BLEU train 104
Figure 70: Show example of test Neural translation arabic -english 105
Figure 71: Show reslt of BLEU test 105
Figure 72: Exemple of dataset 'fra.txt' 105
Figure 73: Show example of train Neural translation English-French 106
Figure 74: Show reslt of BLEU train 107
Figure 75: Show example of test Neural translation English-French 107
Figure 76: Show reslt of BLEU test 108
Figure 77: Exemple of dataset 'deu.txt' 108
Figure 78: Show example of train Neural translation English-German 109
Figure 79: Show reslt of BLEU train 109
Figure 80: Show example of test Neural translation English-German 110
Figure 81: Show reslt of BLEU test 110
Figure 82: Exemple of dataset 'ita.txt' 111
Figure 83: Show example of train Neural translation English-Italian 111
Figure 84: Show reslt of BLEU train 112
Figure 85: Show example of test Neural translation English-Italian 112
Figure 86: Show reslt of BLEU test 113
Figure 87: Exemple of dataset 'spa.txt' 113
Figure 88: Show example of train Neural translation English-Spanish 114
Figure 89: Show reslt of BLEU train 114
Figure 90: Show example of test Neural translation English-Spanish 115
Figure 91: Show reslt of BLEU train 115
Figure 92: Exemple of dataset 'tur.txt' 116

List of figures and tables:

Figure 93: Show example of train Neural translation English-Turkish 116
Figure 94: Show reslt of BLEU train 117
Figure 95: Show example of test Neural translation English-Turkish 117
Figure 96: Show reslt of BLEU test 118
Figure 97: Show example of train Neural translation English-Arabic with other model(English ang German) 119
Figure 98: Show reslt of BLEU train 120
Figure 99: Show example of test Neural translation English-Arabic with other model(English ang German) 120
Figure 100: Show reslt of BLEU test 121
Figure 101: Show result of test new sentence english to arabic 121

List of Tables :

Table 1: Statistical methods 15
Table 2: Diversity between ML and DL 22
Table 3: Clip counts for unigram 61
Table 4: Clip counts for bigrams 62
Table 5: Information in both English and Arabic sequence 92
Table 6 : Traing parameters 99
Table 7 : Vocabulary in each language in five language pairs 100
Table 8 : Results BLEU Score of train for all languges 118
Table 9 : Results BLEU Score of test for all languges 118

List of Acronyms

AES : Advanced Encryption Standard .

AI : Artificial Intelligence .

ALPAC : Automatic Language Processing Advisory Committee .

API : Application Programming Interface .

AROS: Research Operating System .

AS/400: Application System/400 .

BLEU :BiLingual Evaluation Understudy .

BP : Brevity Penalty .

BRNN : Bidirectional Recurrent Neural Networks .

CBOW : Continuous Bag-of-Words .

CLIR: Cross-Language Information Retrieval .

CNN : Convolutional Neural Network .

CNNs: Convolutional Neural Networks .

CPU : Central Processing Unit .

D : Dimensions .

DBN : Deep Belief Networks .

DBSCAN : Density-Based Spatial Clustering of Applications with Noise .

DL : Deep Learning .

FreeBSD: free Berkeley Software Distribution .

GB: gigabyte .

GloVe : Global Vectors .

GPU : Graphical Processing Unit .

GRU's : Gated Recurrent Units .

IBM : International Business Machine Corporation .

LSA : Latent Semantic Analysis .

LSTM : Long Short Term Memory .

LSTM's : Long Short Term Memory networks .

MIT : Massachusetts Institute of Technology .

MT : Machine Learning .

NER : Named entity recognition .

NLG: Natural Language Generation .

NLP : Natural Language Processing .

NLTK : Natural Language Toolkit .

List of Acronyms

NMT: Neural Machine Translation .

OCR : Optical character recognition .

OS/2 : Operating System/2 .

QA : Question-Answering .

RAM : Random Access Memory .

RBM : restricted Boltzmann machine .

RegEx : Regular Expression .

Relu : Rectified Linear Unit .

RNN : Recurrent Neural Network .

RNNs: Recurrent Neural Networks .

RvNN : Recursive Neural Network .

RvNNs: Recursive Neural Networks .

Seq2seq : Sequence to Sequence .

SMT : Statistical Machine Translation .

SVMs : Support Vector Machines .

SWAC : Standards Western Automatic Computer .

TAB : tabulation .

Tanh : Tangent Hyperbolic .

TPU : Tensor Processing Unit .

US : United States .

WMT: Workshop on Statistical Machine Translation .

General Introduction :

Translation, a word borrowed from Latin (traductio), comes from the verb (tradūcĕre): “to pass”, composed of (trans) “through” and (dūcō) “to lead”. (Traducere) in Latin meant ‘to transport’, a meaning widely attested in medieval Latin. Until the 15th century, the word most used to translate was (translatare); however, the extension of (traducere) with the new meaning was so strong that it spread throughout Western Europe as well as Romanian. The most common meaning is: "to switch a text from one language to another". In English (translate) and German (übersetzen), it is to the notion of displacement that the etymology refers.[53]

Translating a text presupposes understanding the meaning for a restitution in the target language. This seemingly simple process is actually complex. Translation is not limited to simple word-for-word substitution. The translator must analyze and interpret the text and understand the relationships between words that can influence its meaning. This requires knowledge of grammar, syntax and semantics, both in the source language and in the target language, in our case English and Arabic. Arabic is a very rich literary language: it has a very extensive vocabulary and a rare flexibility of forms. There would be 80 different terms to express honey, 200 for the serpent, 500 for the lion, 1000 for the camel, as many for the sword, and up to 4000 for the idea of doom. It is because a multitude of nuances of ideas, the subtlety of which means that they are translated by special terms, and that, in the great number of expressions used for the same idea, there is a multitude of figures and tropes. Arabic has over 12,300,000 words, while the Oxford dictionary has 1,000,000 words, and the French dictionary 100,000 words.

Machine translation is perhaps one of the most challenging AI tasks due to the fluidity of human language. Traditionally, rule-based systems have used this task, which in the 1990s was replaced by statistical methods. More recently, deep neural network models have achieved recent results in the field aptly called neural machine translation. Neural machine translation models are suitable for a single model rather than for refining a number of models, and they are currently yielding advanced results.

In this thesis we are interested in the neural machine translation approach. The latter builds a translation model that is satisfactory in terms of quality, because it depends on artificial intelligence, so that this system encodes the sentence to be translated into a fixed-length vector, and then the encoder decodes it to produce the translation.

General Introduction

More precisely, we use RNN so that the data is entered into the network, the network processes the data, and then the network uses a set of parameters called "state" to record the characteristics of the current data, and when the next set of data is in the network, the network's data processing will be combined with the previous residence, processed The “state” parameter sets the data together, and after each time the data is processed, the corresponding “state” parameter set will be updated accordingly, and then participate in the next network processing of the new input data. The latter, in turn, adopts long-term memory (LSTM), which has a key role in obtaining the correct and desired results. This is because it makes it easier to remember the previous data in memory. It also solves the problem of RNN's vanishing gradient . Thus, with the utmost brevity, we perform the process of machine translation from the source language to the target language

Problems and Objectives

Problems :

In the past, translation was not of great importance, but with the opening of the countries of the world to each other's civilizations and languages, it has become of great importance. This is what prompted researchers to conduct many researches on machine translation and worked to facilitate this process of all kinds (neurological, statistical...), but in the way of this research they faced many problems in particular the use of word-for-word translation (sic by letter) so that The main problem is the translator's inability to convey the exact meaning of the text that he wants to transfer into the other language.

Neural machine translation systems provide output with high fluency. However, a weakness of NMT systems is that they sometimes lose the original meaning of the source words during translation. This is because she cannot understand the underlying meanings such as sarcasm.

We do not forget to mention the fact that only a few works have been done on Arabic translation in this field.

Objectives :

This project introduces the use of a neural machine translation approach in an automatic machine translation system under the cover of artificial intelligence , and the goal is to develop an approach or system capable of improving the quality of translation between English and Arabic .

Neural machine translation is the use of neural network models to learn a statistical model of machine translation. The main benefit of this approach is that a single system can be trained directly on the source and target text, as this no longer requires the sum of the operations of the specialized systems.

Neural Machine Translation attempts to build and train a single large neural network that reads the sentence and produces a correct translation.

As such, it can be argued that NMT systems are end-to-end systems . The strength of Neural Machine Translation systems is their ability to learn directly, in an end-to-end manner, by representations from the input text to the output text associated with it .

Dissertation organization

Dissertation organization :

In our work, we are interested in creating a translation system between English and Arabic, and then testing it on a set of words, sentences and short paragraphs.

Depending on the nature of our project, we have organized our report into 4 chapters as follows:

The first chapter is: the state of the art, in which we explain : we are talking in a short path of Natural language processing ,machine translation (neural machine translation) and deep learning .

In the second chapter, we explain in detail Recurrent Neural Networks , point after point, how it works, etc.Then we will explain the way of Long Short Term Memory in detail .Then we talked about the Encoder-Decoder Sequence to Sequence Model, Word Embeddings and BLEU scores .

The third chapter is devoted to explaining our approach in applying the neural machine translation system to improve the translation process between Arabic and English languages.

In Chapter Four we explain all the techniques used to succeed in our approach. We also present the method of implementing our system, we also explain the development tools, and the steps of operating the system are explained by several tests that are explained step by step, as well as its results.



Chapitre I : State Of The Art .



1. Introduction :

With the development of high-speed information technology in various fields, especially informatics, and the human need to adapt to this speed, it was necessary to develop and innovate new ways, and this evolution appears in many of the ideas and research in the field of informatics, for today in this chapter some of them will be introduced.

The purpose of this chapter is to give a comprehensive and detailed idea on many concepts, so it is based on two concepts: Deep Learning, Neural Machine Translation .

For deep learning, some information about the concept of deep learning is provided in the form of simplified definitions, types and how it works, as well as areas of application. Second, the machine translation system, his definition, his history , process and his different approaches.Third, the Neural Machine Translation system, his definition, his history and how it works are dealt with.

2. Natural language processing (NLP) :

2.1 Natural language processing definition :

Natural language processing (NLP) is the study of mathematical and computational modeling of various aspects of language and the development of a wide range of systems. These include spoken language systems that integrate speech and natural language; cooperative interfaces to databases and knowledge bases that model aspects of human-human interaction; multilingual interfaces; machine translation; and message-understanding systems, among others. Research in NLP is highly interdisciplinary, involving concepts in computer science, linguistics, logic, and psychology. NLP has a special role in computer science because many aspects of the field deal with linguistic features of computation and NLP seeks to model language computationally.[58]

Natural language processing (NLP) seeks to model language computationally. So that NLP researchers aim to collect knowledge about how humans understand and use language so that appropriate tools and techniques can be developed to make computer systems understand natural languages and process them to perform the required tasks. The foundations of NLP lie in a number of disciplines, namely computer and information science, linguistics, mathematics, electrical and electronic engineering, artificial intelligence and robotics, and psychology. NLP

applications include a number of areas of study, such as machine translation, natural language text processing and summarization, user interfaces, multilingual and cross-language information retrieval (CLIR), speech recognition, artificial intelligence, and expert systems.[59]

2.2 A brief history of Natural language processing :

The idea of NLP had emerged from the need for Machine Translation in the 1940s. Then the original language was English and Russian. But the use of other words such as Chinese also came into existence in the initial period of the 1960s. Then a lousy era came for MT/NLP during 1966, this fact was supported by a report of ALPAC, according to which MT/NLP almost died because the research in this area did not have the pace at that time. This condition became better again in the 1980s when the product related to MT/NLP started providing some results to customers.

After reaching in dying state in the 1960s, the NLP/MT got a new life when the idea and need of Artificial Intelligence emerged. LUNAR is developed in 1978 by W.A woods; it could analyze, compare and evaluate the chemical data on a lunar rock and soil composition that was accumulating as a result of Apollo moon missions and can answer the related question.

In the 1980s the area of computational grammar became a very active field of research which was linked with the science of reasoning for meaning and considering the user ‘s beliefs and intentions.

In the period of 1990s, the pace of growth of NLP/MT increased. Grammars, tools and Practical resources related to NLP/MT became available with the parsers. The research on the core and futuristic topics such as word sense disambiguation and statistically colored NLP, the work on the lexicon got a direction of research. This quest of the emergence of NLP was joined by other essential topics such as statistical language processing, Information Extraction and automatic summarising.

The discussion on the history of NLP cannot be considered complete without the mention of the ELIZA, a chatbot program which was developed from 1964 to 1966 at the Artificial Intelligence Laboratory of MIT. It was created by Joseph Weizenbaum. It was a program which was based on script named as DOCTOR which was arranged to Rogerian Psychotherapist and used rules, to response the questions of the users which were psychometric-based. It was one of the chatbots which were capable of taking the Turing test at that time.[61]

Currently, a shift has been made to deep learning. This was due to the dominance of deep learning to perform tasks that are difficult to perform by simply using rules and fixed criteria. When it comes to NLP, there is no way to solve the ambiguity of the language. Single word can have multiple meanings according to its neighboring situation and it is not possible to write a rule or to use a decision tree to represent every possible meaning. Deep learning solves this problem efficiently as it does not require a programmer to provide the rules for deciding rather an algorithm itself deduces the process of mapping an input to an output.[60]

2.3 Steps of Natural language processing :

There are 5 phases involved in natural language processing :

2.3.1 Morphological and Lexical Analysis :

The lexicon of a language is its vocabulary that includes its words and expressions. Morphology depicts analysing, identifying and description of structure of words.

- **Lexical Analysis :**

It involves dividing a text into paragraphs,words and the sentences .

2.3.2 Syntactic Analysis :

This involves analysation of the words in a sentence to depict the grammatical structure of the sentence. The words are transformed into structure that shows how the words are related to each other Eg. “the girl the go to the school”. This would definitely be rejected by the English syntactic analyser.

2.3.3 Semantic Analysis :

This abstracts the dictionary meaning or the exact meaning from context. The structures which are created by the syntactic analyser are assigned meaning. There is a mapping between the syntactic structures and the objects in task domain.Eg. “colorless blue idea” .This would be rejected by the analyser as colorless blue do not make any sense together.

2.3.4 Discourse Integration :

The meaning of any single sentence depends upon the sentences that preceeds it and also invokes the meaning of the sentences that follow it .Eg the word “it” in the sentence “she wanted it” depends upon the prior discourse context.

2.3.5 Pragmatic Analysis :

It means abstracting or deriving the purposeful use of the language in situations importantly those aspects of language which require world knowledge the main focus is on what was said is reinterpreted on what it actuallymeans. Eg “closethewindow?” should have been interpreted as a request rather than an order.[62]

2.4 Major tasks in NLP :

This lists some of the different researches done in NLP.

2.4.1 Automatic summarization :

It produces s an understandable summary of a set of text. It is used to provide summaries or detailed information of text of a known type .

2.4.2 Coreference resolution :

It refers to a sentence or larger set of text that determines which words refer to the same objects ,example of this is concerned with matching up pronouns with the nouns or names that they link to.

2.4.3 Discourse analysis :

The task is identifying the discourse structure of connected text, i.e. the nature of the discourse relationships between sentences e.g. elaboration, explanation, contrast. Another possible task is recognizing and classifying the speech acts in a large set of text e.g. yes and no questions, content question, statements, assertion etc.

2.4.4 Machine translation :

Automatically translates text from one human language to another.

- **Morphological segmentation :**

Separate words into individual morphemes and identify the class of the morphemes. The difficulty of this task depends greatly on the complexity of the morphology i.e. the structure of words of the language being considered.

2.4.5 Named entity recognition (NER) :

It describes a stream of text, determine which items in the text relates to proper names, such as people or places, and what the type of each such name or place we are referring to is.

- **Other tasks includes**

a) Optical character recognition (OCR):Given an image representing printed text, helps in determining the corresponding or related text.
b) Part-of-speech tagging:It describes a sentence, determines the part of speech for each word.
c) Parsing:It refers to the parse tree (grammatical analysis or evaluation) of a given sentence
d) Question answering:It answers a given human language question and determines its answer.[22]

2.5 Approaches to Natural language processing :

This section examines NLP methods in terms of foundations, typical techniques, differences in processing and system aspects, strength, flexibility, and suitability for various tasks.

2.5.1 Symbolic Approach :

Symbolic approaches perform deep analysis of linguistic phenomena and are based on explicit representation of facts about language through well-understood knowledge representation schemes and associated algorithms . The primary source of evidence in symbolic systems comes from human-developed rules and lexicons.

Symbolic approaches have been used for a few decades in a variety of research areas and applications such as information extraction, text categorization, ambiguity resolution, and lexical acquisition. Typical techniques include: explanation-based learning, rule-based learning, inductive logic programming, decision trees, conceptual clustering, and K nearest neighbor algorithms .

2.5.2 Statistical Approach :

Statistical approaches employ various mathematical techniques and often use large text corpora to develop approximate generalized models of linguistic phenomena based on actual examples of these phenomena provided by the text corpora without adding significant linguistic or world knowledge. In contrast to symbolic approaches, statistical approaches use observable data as the primary source of evidence.

Statistical approaches have typically been used in tasks such as speech recognition, lexical acquisition, parsing, part-of-speech tagging, collocations, statistical machine translation, statistical grammar learning, and so on.

2.5.3 Connectionist Approach :

Similar to the statistical approaches, connectionist approaches also develop generalized models from examples of linguistic phenomena. What separates connectionism from other statistical methods is that connectionist models combine statistical learning with various theories of representation - thus the connectionist representations allow transformation, inference, and manipulation of logic formulae. In addition, in connectionist systems, linguistic models are harder to observe due to the fact that connectionist architectures are less constrained than statistical ones.

Generally speaking, a connectionist model is a network of interconnected simple processing units with knowledge stored in the weights of the connections between units. For example, one unit might represent the concept “mammal” while another unit might represent the concept “whale”. Relations between concepts are encoded by the weights of connections between those concepts. Knowledge in such models is spread across the network, and the connectivity between units reflects their structural relationship.

2.5.4 Neural networks Approach :

A major drawback of statistical methods is that they require elaborate feature engineering. Since 2015, the field has thus largely abandoned statistical methods and shifted to neural networks for machine learning. Popular techniques include the use of word embeddings to capture semantic properties of words, and an increase in end-to-end learning of a higher-level task (e.g., question answering) instead of relying on a pipeline of separate intermediate tasks (e.g., part-of-speech tagging and dependency parsing). In some areas, this shift has entailed substantial changes in how NLP systems are designed, such that deep neural network-

based approaches may be viewed as a new paradigm distinct from statistical natural language processing. For instance, the term neural machine translation (NMT) emphasizes the fact that deep learning-based approaches to machine translation directly learn sequence-to-sequence transformations, obviating the need for intermediate steps such as word alignment and language modeling that was used in statistical machine translation (SMT). Latest works tend to use non-technical structure of a given task to build proper neural network.[63]

2.6 Natural Language Processing Applications :

2.6.1 Automatic summarization :

Automatic summarization is especially relevant when used to provide an overview of a news item or blog posts, while avoiding redundancy from multiple sources and maximizing the diversity of content obtained.

2.6.2 Sentiment analysis :

The goal of sentiment analysis is to identify sentiment among several posts or even in the same post where emotion is not always explicitly expressed. Companies use natural language processing applications, such as sentiment analysis, to identify opinons and sentiment online to help them understand what customers think about their products and services .sentiment analysis understands sentiment in context to help you better understand what's behind an expressed opinion, which can be extremely relevant in undersanding and driving purchasing decisions.

2.6.3 Text classification :

Text classification makes it possible to assign predefined categories to a document and organize it to help you find the information you need or simplify some activities. For example, an application of text categorization is spam filtering in email.

2.6.4 Question Answering :

As speech-understanding technology and voice-input applications improve, the need for NLP will only increase. Question-Answering (QA) is becoming more and more popular thanks to applications such as Siri, OK Google, chat boxes and virtual assistants .[12]

2.7 Future of Natural language processing :

Human level or human readable natural language processing is an AI-complete problem.It is equivalent to solving the central artificial intelligence problem and making computers as intelligent as people so that they can solve problems like humans and think like humans as well as perform activities that humans cant perform and making it more efficient than humans .NLP's future is closely linked to the growthof Artificial intelligence.As natural language understanding or readability improves, computers or machines or devices will be able to learn from the information online and apply what they learned in the real world. Combined with natural language generation,computers will become more and more capable of receiving and giving useful and resourceful information or data.[22]

3. Machine Translation :

3.1 Machine Translation Definition :

Machine Translation means automatic translation, It is a field of Artificial Intelligence. Machine translation is computer program which is designed to translate text from one language (source language) to another language (target language) without the help of human. The aim of Machine Translation is to provide a system that translate text of source language into target language and translation express the same meaning as it in source language.[8]

Machine translation (MT) is automated translation or “translation carried out by a computer”, as defined in the Oxford English dictionary. It is a process, sometimes referred to as Natural Language Processing which uses a bilingual data set and other language assets to build language and phrase models used to translate text.[9]

3.2 A brief history of machine translation :

The idea of mechanizing the translation of texts was put forward with the release of the first computers in 1947 by Warren Weaver in a conversation with Andrew Booth [Hutchins, 2004]. In 1948, Booth collaborated with Richard Richens on the first experiments using mechanical translation of punch cards in 1948. It was during World War II that the first attempts to develop automatic translating machines messages emerge when the US military tries to develop computers capable of deciphering the encoded messages of the Japanese military.

In 1949, it was reported in a newspaper that Harry Huskey was considering translating with the computers at SWAC in Los Angeles. Then, in July of the same year, Warren Weaver wrote the famous memorandum which stimulated the beginnings of research in machine translation: at the University of Washington (Erwin Reifler), University of California at Los Angeles (Victor Oswald, Stuart Fletcher) as well as the Massachusetts Institute of Technology (MIT). Then IBM, in collaboration with Georgetown University, carried out in 1954 in New York, the first public demonstration of a Russian-English machine translation system. It was a small-scale experiment of only 250 words and six "grammatical" rules according to [Hutchins, 2004]. It was not until the end of the 1990s and the advent of the information society that real opportunities appeared for machine translation software, for example: on-the-fly translation of WEB pages, translation of news dispatches. . .



Figure 1 : SWAC type computers in 1950.

Machine translation was subsequently enriched by ten additional language pairs and statistical analysis. Although these programs have benefited from considerable theoretical and technical progress, the results obtained have remained somewhat disappointing. No machine was able to translate, on its own and in a fully satisfactory manner, any text, even if machine translation was of great service to human translators, and made it possible to achieve fairly significant productivity gains.

In recent years, machine translation has experienced considerable growth thanks to the development of the Internet, since various online systems make it possible to automatically translate web pages shorter or shorter texts. Today, this technology is appreciated by the general public, because it allows to roughly decipher the theme of a text in a completely unknown language, and the main facts or pieces of information it contains. However, most of the available tools are limited to an engine without a "trade"…….. vocabulary or to dictionaries translating

word-for-word; they only process a limited number of words, and offer general services, unsuitable for documents that are too technical or written in a strong language.

Such tools have several other advantages for users: they are free (for the most part), "instant", accept inputs of different kinds (words, texts, mail, web pages, etc.) and offer the translation of several languages (example Google translation: which manages 145 languages).

On the other hand, for companies developing these online tools (Reverso, Systran, etc.), they represent a demo version that encourages users to buy more advanced paid software.[55]

3.3 The translation process :

The translation process (in the human sense) can be divided into three successive phases:

- **Comprehension :** assimilation of the meaning conveyed by a text, of the meaning of an author ...
- **Deverbalization :** forgetting words and retaining meaning; "Operation by which a subject becomes aware of the meaning of a message by losing consciousness of the words and sentences which gave it body".
- **Re-expression :** reformulation of meaning in the target language.[48]

3.4 The different approaches to machine translation :

A. Linguistic approaches :

The linguistic MT process takes place in three basic phases:

- **Analysis :** analyze the source text into intermediate representations in the source language .
- **Transfer :** transfer these intermediate representations to intermediate representations in the target language .
- **Generation :** generate the new text in the target language from the intermediate representations in the target language .

The "Vauquois triangle" proposed in describes the different possible linguistic architectures of a MT system. Each path in the triangle corresponds to a linguistic architecture “Figure 2” .

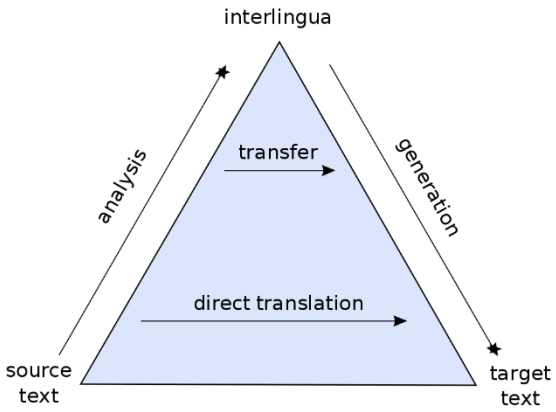


Figure 2: Triangle de Vauquois, representation of the different architectures

[56]

1) Direct translation system :

The source sentence is segmented into words and the system performs the translation verbatim. The transfer step uses a bilingual table which associates for each word or sequence of source words a set of translation and rearrangement rules which allow to translate and reorder the words in the target sentence.

2) Transfer rule-based MT system :

The source sentence is analyzed using a parser and grammar. This analysis phase gives rise to a tree representation which is then converted into the target language. To ensure the transition from the source intermediate representation to the target representation, a bilingual table, containing the transfer rules between the source and target representations, is required.“Figure 3” below, illustrates an example of transfer-based translation .

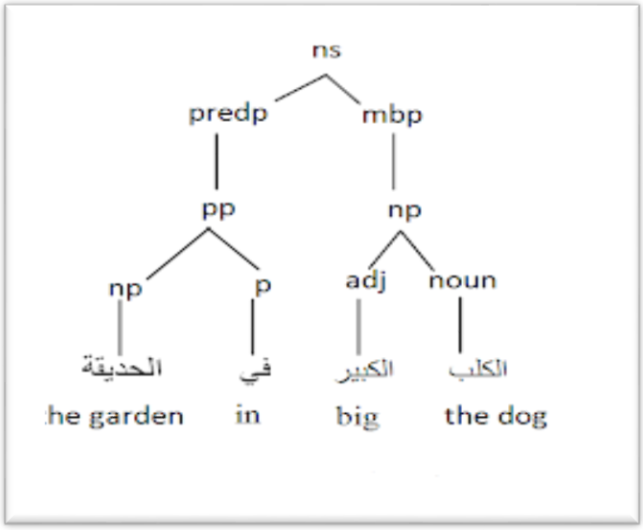


Figure 3 : Example of translation of a nominal group in English into a sentence in Arabic.

3) Interlingual translation system :

Translating from any source language to any target language is the process of "converting" the source sentence to the pivot representation and then "de-converting" the target sentence from that pivot representation. The transfer of representations from the source language to those of the target language is no longer necessary. The advantage of this method is the possibility of applying it in a multilingual environment. To cover all translation directions between n languages, we only need n conversion modules and n deconverting modules .

However, the complexity of this method lies in the obligation to construct a pivot vocabulary to represent all the possible concepts of all languages and the links of the concepts between two languages. The construction can be based on a "logical" artificial language, on a "natural" auxiliary language (such as English or Esperanto), on a set of primitive concepts common to all languages, or on a "universal" vocabulary. . UNL (Universal Networking Language) is an example of a pivot language .

B. Statistical approaches :

Statistical Machine Translation enables text to be automatically translated from a source language to a target language by machine learning the system by applying statistical mathematical calculations to aligned bilingual data. A text is translated according to the probability law P (e | f) which calculates the probability that the target language sentence e is a translation of the sentence f.

Statistical MT requires two models: a language model, and a translation model. These models are based on the mathematical theory of distribution and probabilistic estimation of Frederick Jelinek, developed at IBM T.J. Watson Research Center.

The goal of statistical translation is to find the sentence Ê, which maximizes P (e | f) given a source sentence f.

Mathematically: Ê = argmaxE P (e | f).

According to Bayes' theorem: P (e | f) = P (f | e) .P (e) / P (f)

Since the denominator P (f) is independent of P (e) the maximization then becomes:

Ê = argmaxE P (e | f) = argmaxE P (e) * P (f | e).

P (e) is called the target language model, while P (f | e) is called a translation model. Thus, statistical translation is based on two models :

- Translation model that offers candidate translations in the target language for each segment of the source sentence .
- Language model which is responsible for capturing the constraints imposed by the syntax of the target language by one or more probabilistic functions .

“Figure 4” below illustrates the architecture of a statistical translation system .

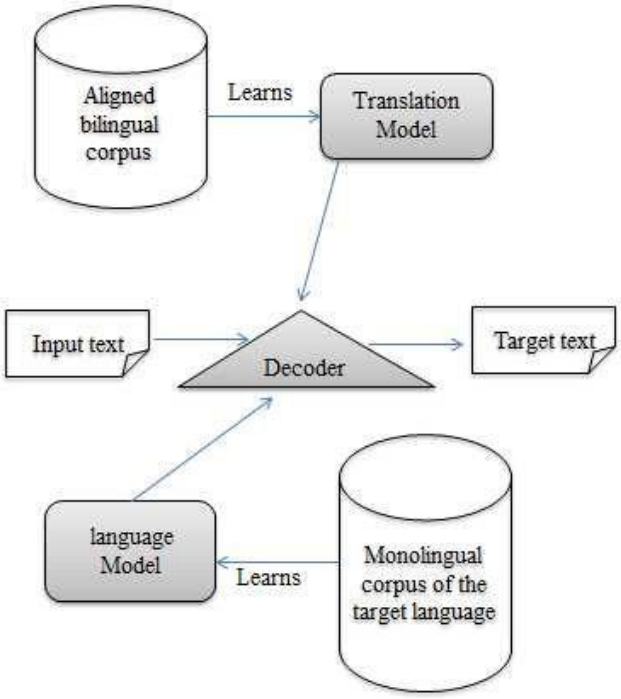


Figure 4 : Example of the architecture of a statistical machine translation system .

Statistical methods can be based either on corpora or on examples. Table.1 defines the two types of statistical approaches .

Statistical methods	
Corpus-based	Based on examples
Machine learning of translation "rules" from a bilingual corpus	**Reuse the examples of existing translations as the basis for the new translation**

Table 1: Statistical methods .[14]

Statistical machine translation models were word-based where sentences are translated word by word. Then, Koehn, Och, and Marcu improved these models and founded word sequence-based MT systems.

These models represent, today, the state of the art of statistical machine translation.

C. Hybrid machine translation (mixed) :

In recent years, researchers have become increasingly interested in mixed approaches that could take advantage of the strengths of each of the statistical and linguistic approaches. For example, hypotheses from an empirical translation system can be sorted using source or target linguistic information .[14]

D. Neural Machine Translation (NMT):

While both statistical and neural MT use huge datasets of translated sentences to teach software to find the best translation, the models themselves are different. Statistical MT translates sentences by breaking them up into phrases, translating the pieces, then trying to stitch those translations back together. Neural MT, on the other hand, uses neural networks to consider whole sentences when predicting translations, which allows it to take into account the context in which each word and phrase is used.

4. Neural Network :

4.1 Neural Network definition :

A neural network is a computing model whose layered structure resembles the networked structure of neurons in the brain. It features interconnected processing elements called neurons that work together to produce an output function. Neural networks are made of input and output layers/dimensions, and in most cases, they also have a hidden layer consisting of units that transform the input into something that the output layer can use.[13]

4.2 A Brief History of Neural Networks :

Neural networks date back to the early 1940s when mathematicians Warren McCulloch and Walter Pitts built a simple algorithm-based system designed to emulate human brain function. Work in the field accelerated in 1957 when Cornell University's Frank Rosenblatt conceived of the perceptron, the groundbreaking algorithm developed to perform complex recognition tasks. During the four decades that followed, the lack of computing power necessary to process large amounts of data put the brakes on advances. In the 2000s, thanks to the advent of greater computing power and more sophisticated hardware, as well as to the existence of vast data sets to draw from, computer scientists finally had what they needed, and neural networks and AI took off, with no end in sight. To understand how much the field has expanded in the new millennium, consider that ninety percent of internet data has been created since 2016. That pace will continue to accelerate, thanks to the growth of the Internet of Things (IoT).[15]

4.3 Architecture of Neural Network :

Neural Networks are complex structures made of artificial neurons that can take in multiple inputs to produce a single output. This is the primary job of a Neural Network – to transform input into a meaningful output. Usually, a Neural Network consists of an input and output layer with one or multiple hidden layers within.

In a Neural Network, all the neurons influence each other, and hence, they are all connected. The network can acknowledge and observe every aspect of the dataset at hand and how the different parts of data may or may not relate to each other. This is how Neural Networks are capable of finding extremely complex patterns in vast volumes of data . [14]

4.4 Components of Neural Network :

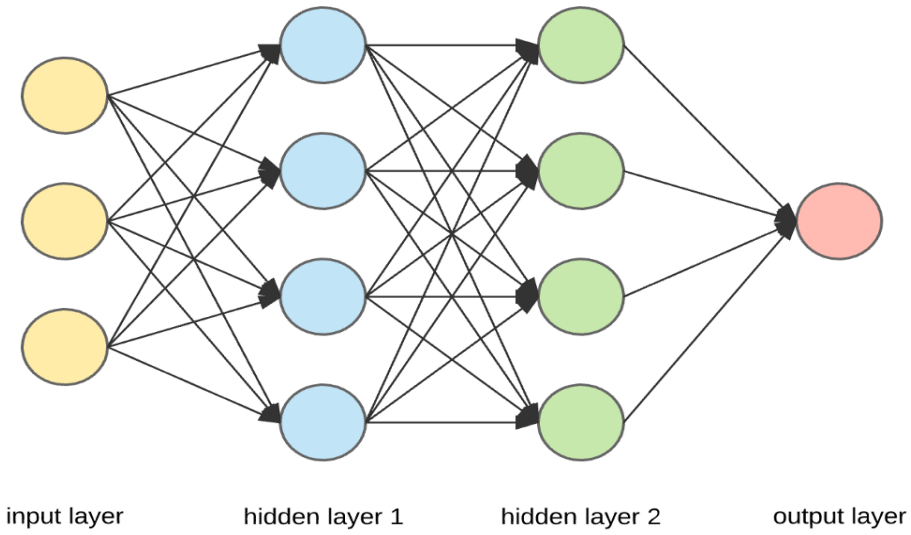


Figure 5 : Components of Neural Network .

[14]

- **Input Layers, Neurons, and Weights :**

In the picture given above "Figure 7", the outermost yellow layer is the input layer. A neuron is the basic unit of a neural network. They receive input from an external source or other nodes. Each node is connected with another node from the next layer, and each such connection has a particular weight. Weights are assigned to a neuron based on its relative importance against other inputs.

When all the node values from the yellow layer are multiplied (along with their weight) and summarized, it generates a value for the first hidden layer. Based on the summarized value, the blue layer has a predefined “activation” function that determines whether or not this node will be “activated” and how “active” it will be.

For example making tea. In the tea making process, the ingredients used to make tea (water, tea leaves, milk, sugar, and spices) are the “neurons” since they make up the starting points of the process. The amount of each ingredient represents the “weight.” Once you put in the tea leaves in the water and add the sugar, spices, and milk in the pan, all the ingredients will

mix and transform into another state. This transformation process represents the “activation function.”

- **Hidden Layers and Output Layer :**

The layer or layers hidden between the input and output layer is known as the hidden layer. It is called the hidden layer since it is always hidden from the external world. The main computation of a Neural Network takes place in the hidden layers. So, the hidden layer takes all the inputs from the input layer and performs the necessary calculation to generate a result. This result is then forwarded to the output layer so that the user can view the result of the computation .

In the example of making tea, when all ingredients are mixed, the formulation changes its state and color on heating.The ingredients represent the hidden layers. Here heating represents the activation process that finally delivers the result ""tea"".[14]

4.5 Algorithms of Neural Network :

In a Neural Network, the learning (or training) process is initiated by dividing the data into three different sets:

i. **Training dataset :** This dataset allows the Neural Network to understand the weights between nodes .
ii. **Validation dataset :** This dataset is used for fine-tuning the performance of the Neural Network .
iii. **Test dataset :** This dataset is used to determine the accuracy and margin of error of the Neural Network .

Once the data is segmented into these three parts, Neural Network algorithms are applied to them for training the Neural Network. The procedure used for facilitating the training process in a Neural Network is known as the optimization, and the algorithm used is called the optimizer. There are different types of optimization algorithms, each with their unique characteristics and aspects such as memory requirements, numerical precision, and processing speed .[14]

5. Neural Machine Translation :

5.1 Neural Machine Translation definition :

Neural Machine Translation is a machine translation approach that applies a large artificial neural network toward predicting the likelihood of a sequence of words, often in the form of whole sentences. Unlike statistical machine translation, which consumes more memory and time, neural machine translation, NMT, trains its parts end-to-end to maximize performance. NMT systems are quickly moving to the forefront of machine translation, recently outcompeting traditional forms of translation systems.[10]

5.2 A brief history of Neural Machine Translation :

Already during the last wave of neural network research in the 1980s and 1990s, machine translation was in the sight of researchers exploring these methods (Waibel et al., 1991). In fact, the models proposed by Forcada and Ñeco (1997) and Castaño et al. (1997) are striking similar to the current dominant neural machine translation approaches. However, none of these models were trained on data sizes large enough to produce reasonable results for anything but toy examples. The computational complexity involved by far exceeded the computational resources of that era, and hence the idea was abandoned for almost two decades.

During this hibernation period, data-driven approaches such as phrase-based statistical machine translation rose from obscurity to dominance and made machine translation a useful tool for many applications, from information gisting to increasing the productivity of professional translators.

The modern resurrection of neural methods in machine translation started with the integration of neural language models into traditional statistical machine translation systems. The pioneering work by Schwenk (2007) showed large improvements in public evaluation campaigns.

However, these ideas were only slowly adopted, mainly due to computational concerns. The use of GPUs for training also posed a challenge for many research groups that simply lacked such hardware or the experience to exploit it.

Moving beyond the use in language models, neural network methods crept into other components of traditional statistical machine translation, such as providing additional scores or extending translation tables (Schwenk, 2012; Lu et al., 2014), reordering (Kanouchi et al., 2016; Liet al., 2014) and pre-ordering models (de Gispert et al., 2015), and so on. For instance, the joint translation and language model by Devlin et al. (2014) was influential since it showed large quality improvements on top of a very competitive statistical machine translation system.

More ambitious efforts aimed at pure neural machine translation, abandoning existing statistical approaches completely. Early steps were the use of convolutional models (Kalchbrenner and Blunsom, 2013) and sequence-to-sequence models (Sutskever et al., 2014; Cho et al., 2014).

These were able to produce reasonable translations for short sentences, but fell apart with increasing sentence length. The addition of the attention mechanism finally yielded competitive results (Bahdanau et al., 2015; Jean et al., 2015b). With a few more refinements, such as byte pair encoding and back-translation of target-side monolingual data, neural machine translation became the new state of the art.

Within a year or two, the entire research field of machine translation went neural. To give some indication of the speed of change: At the shared task for machine translation organized by the Conference on Machine Translation (WMT), only one pure neural machine translation system was submitted in 2015. It was competitive, but outperformed by traditional statistical systems. A year later, in 2016, a neural machine translation system won in almost all language pairs. In 2017, almost all submissions were neural machine translation systems.[57]

5.3 Properties :

They require only a fraction of the memory needed by traditional statistical machine translation (SMT) models. Furthermore, unlike conventional translation systems, all parts of the neural translation model are trained jointly (end-to-end) to maximize the translation performance.

5.4 How does NMT Work? :

The goal of any neural MT model is to take an input in one language and output it into another language. The first thing to understand is how a language input is transformed into data that can be used by the NMT model.

NMT models use a translation method more commonly called the Encoder-Decoder structure. This structure takes the content in its original source language, assigns each word a number, finds the corresponding word in the target language, then spits out a translation into the new language using the numerical representation that each word has.[11]

6. Deep Learning :

6.1 Deep Learning Definition :

Deep learning is a form of machine learning that enables computers to learn from experience and understand the world in terms of a hierarchy of concepts. Because the computer gathers knowledge from experience, there is no need for a human computer operator formally to specify all of the knowledge needed by the computer. The hierarchy of concepts allows the computer to learn complicated concepts by building them out of simpler ones .[47]

Deep learning is an artificial intelligence (AI) function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial intelligence that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as deep neural learning or deep neural network.[1]

6.2 What Is the Difference between Deep Learning and Machine Learning?:

Machine Learning (ML) is basically used to implement Artificial Intelligence. Machine Learning works efficiently when we have some moderate amount of sorted data which we train and get the desired results according to the problem we need to solve.

Machine Learning working does not depend on the type of system or hardware we are using, it works efficiently on every system. Machine Learning takes less time to train data and get ready for analyzing results. It works by converting the complexity of data to a lower level so that analysis of results get more efficiency and produce more .

accurate result. It starts by matching pattern by doing so it breaks a problem into sub-problems as it is easy to get an output from a short problem then at the end it cumulate the results to get the final result as in other words we can say it uses divide-n-conquer technique to analyze the data and get results from it .

Diversity between machine learning (ML) and deep learning (DL):

Deep Learning (DL):	Machine Learning (ML)
Requires a lot of unlabeled training data to make concise conclusions .	Can use small data amounts provided by users.
Needs high-performance hardware.	Not needs high-performance hardware.
Creates new features by itself.	Requires features to be accurately identified by users
Solves the problem on the end-to-end basis.	Divides tasks into small pieces and then combine received results into one conclusion
Needs much more time to train.	Not needs much more time to train.
Can not provide enough transparency for its decisions.	Can provide enough transparency for its decisions.

Table 2: Diversity between ML and DL .

The concept of deep learning implies that the machine creates its functionality by itself as long as it is possible at the current time. To infer, deep learning applications use a hierarchical approach involving determining the most important characteristics to compare. [54] .

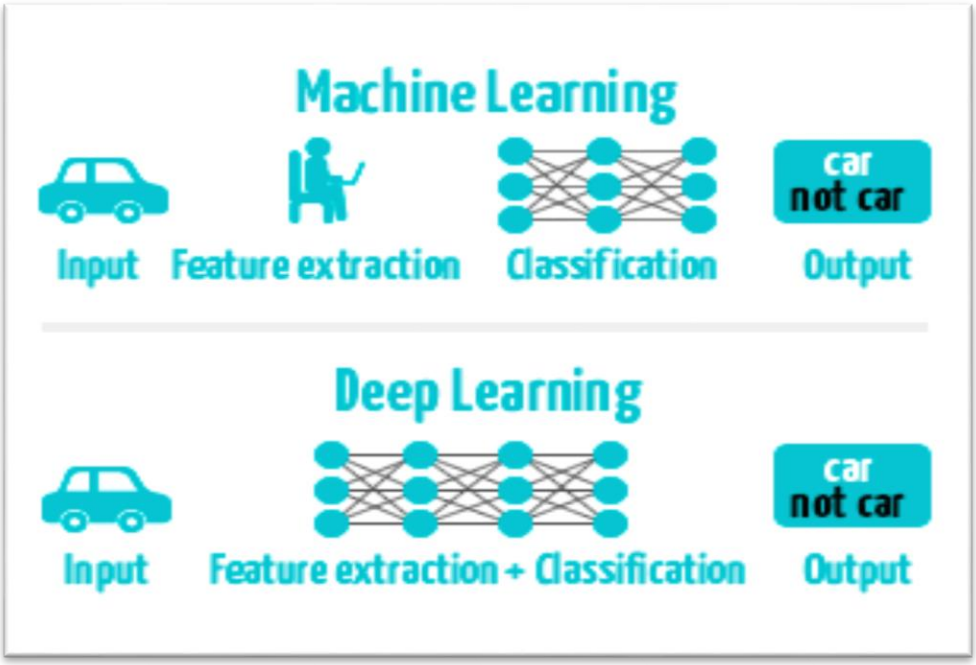


Figure 6 : Difference between Machine Learning and Deep Learning.[4]

6.3 Deep Learning type :

In this section, some types of Deep Learning are presented :

6.3.1 Feedforward Neural Network :

This type of neural network is the very basic neural network where the flow control occurs from the input layer and goes towards the output layer.

These kinds of networks are only having single layers or only 1 hidden layer .

Since the data moves only in 1 direction there is no backpropagation technique in this network .

In this network, the sum of the weights present in the input is fed into the input layer .

These kinds of networks are used in the facial recognition algorithm using computer vision .[5]

6.3.2 Convolutional Neural Networks :

A convolutional neural network (CNN) is another variant of the feedforward multilayer perceptron. It is a type of feedforward neural network, where the individual neurons are ordered in a way that they respond to all overlapping regions in the visual area.

This is a type of network that constitutes of learning weight and biases. Every input layer is composed of a set of neurons where at every input a dot product is performed and move further with the concept of non-linearity. It is a kind of fully-connected type of network that uses SVM/Softmax function as a loss function.

CNNs form the core of most sophisticated fancy computer vision applications , facial security features, and automatic number plate recognition.[6]

6.3.3 Recurrent Neural Networks :

The convolutional model works on a fixed number of inputs, generates a fix-sized vector as output with a predefined number of steps. The recurrent networks allow us to operate over sequences of vectors in input and output. In the case of recurrent neural network, the connection between units forms a directed cycle. Unlike the traditional neural network, the recurrent neural network input and output are not independent but related. Further, the recurrent neural network shares the standard parameters at every layer. One can train the recurrent network in a way that is like the traditional neural network using the backpropagation method.

Here, calculation of gradient depends not on the current step but previous steps also. A variant called a bidirectional recurrent neural network is also used for many applications. The bidirectional neural network considers not only the previous but also the expected future output. In two-way and straightforward recurrent neural networks, deep learning can be achieved by introducing multiple hidden layers. Such deep networks provide higher learning capacity with lots of learning data. Speech, image processing, and natural language processing are some of the candidate areas where recurrent neural networks can be used.[6]

6.3.4 Autoencoders :

An autoencoder is an artificial neural network that is capable of learning various coding patterns. The simple form of the autoencoder is just like the multilayer perceptron, containing an input layer or one or more hidden layers, or an output layer. The significant difference between the typical multilayer perceptron and feedforward neural network and autoencoder is in the number of nodes at the output layer. In the case of the autoencoder, the output layer contains the same amount of nodes as in the input layer. Instead of predicting target values as per the output vector, the autoencoder has to predict its inputs.[6]

6.3.5 Restricted Boltzmann Machine :

A restricted Boltzmann machine (RBM) is a generative stochastic artificial neural network that can learn a probability distribution over its set of inputs.

It is a kind of stochastic neural network that consists of one layer of visible units, one layer of hidden units and a bias unit. The architecture is developed in such a way that each visible unit is connected to all hidden units and bias units are connected to all visible and hidden units. During the learning process, the restriction is developed so that no visible unit is connected with any visible units and no hidden unit is connected with any hidden unit.

6.3.6 Deep Belief Networks (DBN):

A deep belief network is a solution to the problem of handling non-convex objective functions and local minima while using the typical multilayer perceptron. It is an alternative type of deep learning consisting of multiple layers of latent variables with connection between the layers. The deep belief network can be viewed as restricted Boltzmann machines (RBM), where each subnetwork’s hidden layer acts as the visible input layer for the adjacent layer of the network. It makes the lowest visible layer a training set for the adjacent layer of the network. This way, each layer of the network is trained independently and greedily. The hidden variables are used as the observed variables to train each layer of the deep structure.[6]

6.3.7 Recursive Neural Network :

A recursive neural network is a kind of deep neural network created by applying the same set of weights recursively over a structured input, to produce a structured prediction over variable-size input structures, or a scalar prediction on it, by traversing a given structure in topological order. Recursive neural networks, sometimes abbreviated as RvNNs, have been successful, for instance, in learning sequence and tree structures in natural language processing, mainly phrase and sentence continuous representations based on word embedding.

6.4 Difference between Neural Network and Deep Learning Neural Network :

Since Deep Learning and Neural Networks are so deeply intertwined, it is difficult to tell them apart from each other on the surface level. However, there's a significant difference between Deep Learning and Neural Networks.

While Neural Networks use neurons to transmit data in the form of input values and output values through connections, Deep Learning is associated with the transformation and extraction of feature which attempts to establish a relationship between stimuli and associated neural responses present in the brain.[2]

Neural Network is less complex and requires more information about features for performing feature selection and feature engineering method. On the other hand, Deep Learning Neural Network does not require any information about features rather they perform optimum model tuning and model selection on their own.

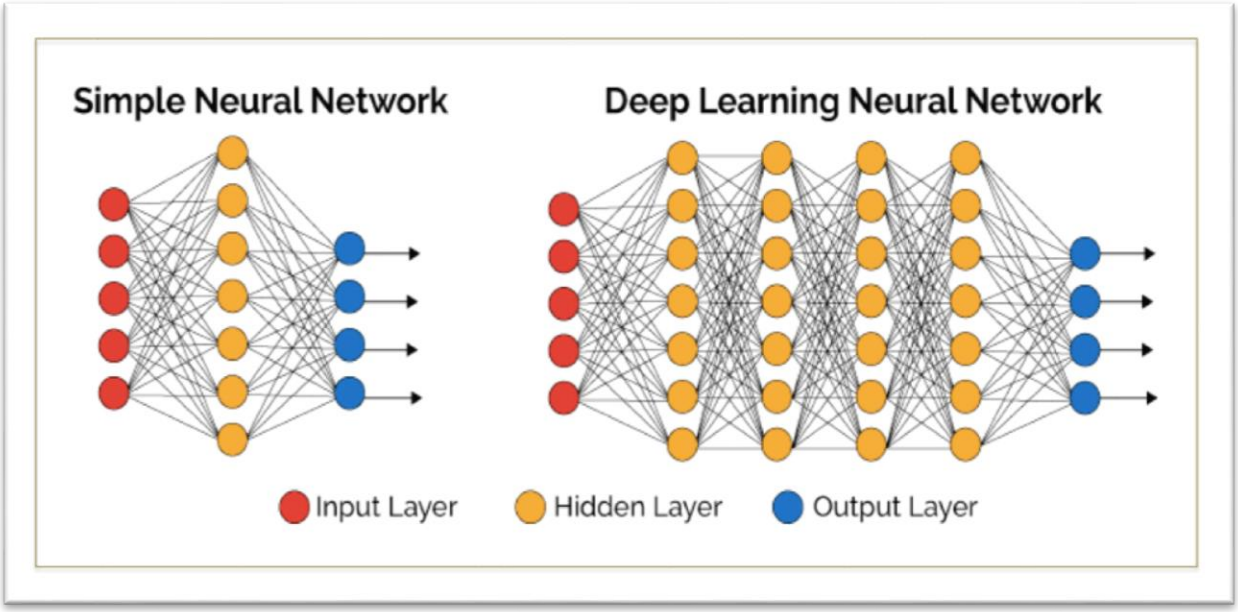


Figure 7: Architecture of Simple Neural Network and Deep Learning Neural Network. [3]

6.5 Deep learning application and examples:

Deep learning is currently being utilized in almost every industry ,Here are some of the examples :

- **Machine Translation :**Thanks to deep learning, we have access to different translation services. One of the most popular one, Google Translate helps its user to easily translate a language. No need for complicated steps, deep learning has helped this application improve tremendously. From just typing a word, to pronouncing a word, it sure is a big improvement from what is started out to be.
- **Virtual Assistants :** The most popular application of deep learning is virtual assistants. From the likes Siri, Alexa and Google Assistant, these digital assistants are heavily reliant on deep learning to understand its user and at the same time give the appropriate response in a natural manner.Virtual assistants uses deep learning to know more about their subjects ranging from your favorite places to your favorite songs.
- **Customer Service :** "Chatbots " . Deep learning has played a major role in helping businesses by improving customer service and making it more accessible to its customers.Trained with large amount of data, chatbots can understand customer requests and also guide the customer and help them resolve their problem in a human-like manner. Plus, it saves up customers time and brings down the costs of business.

- **Self-Driving Cars :** Because of deep learning, self-driving cars do exist and is just going to keep on improving over time. Although it hasn't been made available to public yet, The Uber Artificial Intelligence Labs at Pittsburgh is not only working on making driverless cars, but also integrating food delivery option with the use of this new invention.
- **News Aggregation :** Deep learning helps develop classifiers that can detect fake or biased news and remove it from your feed.
- **Digital Marketing :** The application of deep learning in digital marketing helps marketing professionals gauge the effectiveness of their campaigns.
- **Natural Language Processing :** One of the hardest task that human can learn is understanding the complexities associated with language. Whether its semantics, syntax, tonal nuances, expressions or even sarcasms, humans find it hard to easily grasps learning a language.In deep learning, machines are trained to achieve the same thing and develop human-like response and personalized expressions. It is also trying to catch linguistic nuances and answer questions. It is also training machines to build phrases and sentences and capture local word semantics with word embedding.
- **Colorizing Videos and Images :** Thanks to deep learning frameworks, machines can flaunt their creativity by adding color to old black and white photos and videos.
- **Personalization :** Have you ever felt that Spotify, Amazon, E-bay, Alibaba and Netflix recommends you exactly the things you like? Deep learning is the main reason for that. It plays a major role in understanding its consumers' behavior and generating recommendations to help them make choices for product and services.
- **Healthcare :** Deep learning has been playing an important role in medical diagnosis and research. It helps with diagnosis of life-threatening diseases, pathology results and treatment cause standardization and understanding genetics to predict future risks of diseases. Readmissions are a huge problem in the healthcare industry and deep learning is helping to combat this.
- **Fraud Detection :** The banking and financial sector also benefit from deep learning application especially money transaction are going digital. Furthermore, there are applications under development that will help detect fraudulent credit cards saving billions of dollars of in recovery and insurance of financial institutions.[7]

7. Conclusion :

In this first chapter, the basic concept of: deep learning, machine translation , natural language processing and neural machine translation, were introduced in order to simplify the understanding of their basic concepts .

For deep learning, we talked about different types of deep learning networks; How they work and their areas of application. Therefore, the definition of the machine translation system, its history, and its operation are discussed. Then an additional specialization is conducted and neural machine translation is addressed, its definition, history, characteristics and how it works. The details of natural language processing from all its angles are also explained. Finally, the neural network is addressed and defined .



Chapter II : Deep learning and Machine translation.



1. Introduction :

Machine translation systems are useful in many fields today, and many researchers have tested several machine translation systems that differ in terms of the technology or approach followed, starting from rule-based technology to statistical to neural machine translation technology, and This last technique is considered the most successful and accurate according to the findings of the researchers.

The purpose of this chapter is to provide an idea of natural language processing, the sequence to sequence system, neural networks, recurrent neural networks, and long-term memory, all of which are implicit in neural machine translation.

2. Recurrent Neural Networks :

2.1 Recurrent Neural Networks Definition :

A recurrent neural network (RNN) is a type of artificial neural network which uses sequential data . These deep learning algorithms are commonly used for ordinal or temporal problems, such as language translation, natural language processing (nlp), speech recognition, and image captioning; they are incorporated into popular applications such voice search, and Google Translate. recurrent neural networks utilize training data to learn. They are distinguished by their “memory” as they take information from prior inputs to influence the current input and output. While traditional deep neural networks assume that inputs and outputs are independent of each other, the output of recurrent neural networks depend on the prior elements within the sequence. While future events would also be helpful in determining the output of a given sequence .[16]

2.2 Why Recurrent Neural Networks ? :

Recurrent neural networks were created because there were a few issues in the feed-forward neural network:

- Cannot handle sequential data.
- Considers only the current input.
- Cannot memorize previous inputs.

The solution to these issues is the Recurrent Neural Network (RNN). An RNN can handle sequential data, accepting the current input data, and previously received inputs. RNNs can memorize previous inputs due to their internal memory.[18]

2.3 Advantages & Disadvantages of Recurrent Neural Network :

2.3.1 Advantages :

1. RNN can process inputs of any length.
2. An RNN model is modeled to remember each information throughout the time which is very helpful in any time series predictor.
3. Even if the input size is larger, the model size does not increase.
4. The weights can be shared across the time steps.
5. RNN can use their internal memory for processing the arbitrary series of inputs which is not the case with feedforward neural networks.

2.3.2 Disadvantages :

1. Due to its recurrent nature, the computation is slow.
2. Training of RNN models can be difficult.
3. If we are using Relu [1] or tanh[2] as activation functions, it becomes very difficult to process sequences that are very long.
4. Prone to problems such as exploding and gradient vanishing.[17]

[1] Relu or Rectified Linear Activation Function is the most common choice of activation function in the world of deep learning. Relu provides state of the art results and is computationally very efficient at the same time.[34]

[2] Python tanh() is an inbuilt method that is defined under the math module, which is used to find the hyperbolic tangent of the given parameter in radians.[35]

2.4 Types of recurrent neural networks :

Feednets map one input to one output, and although we conceptualize recurrent neural networks in this way as an initial concept, they do not actually have this limitation. Instead, their inputs and outputs can vary in length, and different types of RNNs are used for different use cases, such as music generation, sentiment classification, and machine translation[16]. Different types of RNNs are usually expressed using the following diagrams:

- ✓ **One-to-one :**

One to One RNN (Tx=Ty=1) is the most basic and traditional type of Neural network giving a single output for a single input, as can be seen in the “figure 8”. [20]

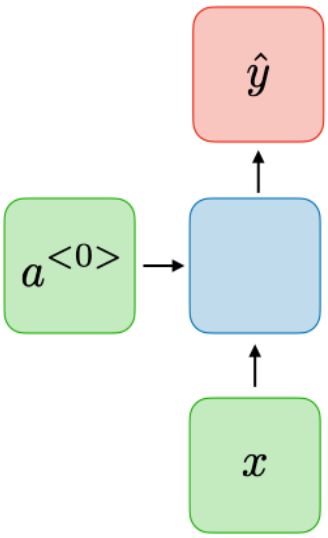


Figure 8: Tupe of RNN : One-to-one .[19]

- ✓ **One-to-many :**

One to Many (Tx=1,Ty>1) is a kind of RNN architecture is applied in situations that give multiple output for a single input , as can be seen in the “figure 9”. A basic example of its application would be Music generation. In Music generation models, RNN models are used to generate a music piece(multiple output) from a single musical note(single input). [20]

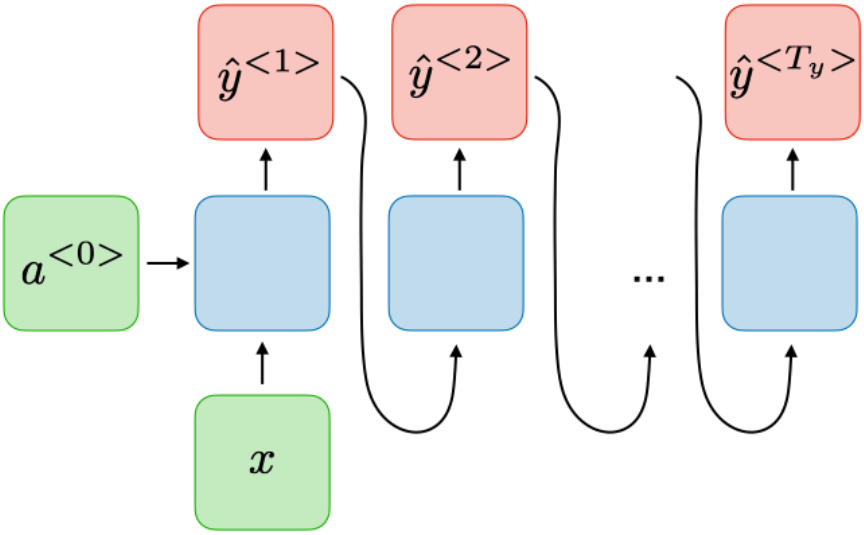


Figure 9: Tupe of RNN : One-to-many .[19]

✓ **Many-to-one:**

Many-to-one RNN architecture (Tx>1,Ty=1) is usually seen for sentiment analysis model as a common example. As the name suggests, this kind of model is used when multiple inputs are required to give a single output , as can be seen in the "figure 10".

Take for example The Twitter sentiment analysis model. In that model, a text input (words as multiple inputs) gives its fixed sentiment (single output). [20]

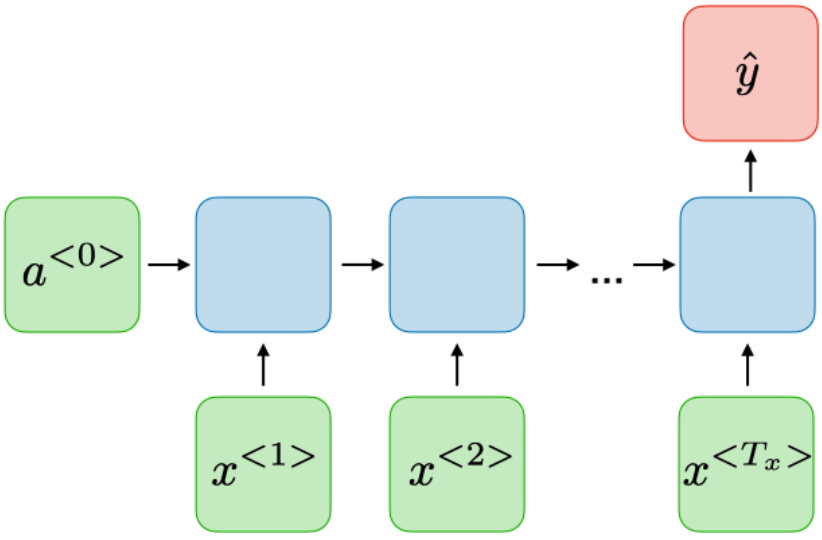


Figure 10: Tupe of RNN : Many-to-one . [19]

✓ **Many-to-many :**

As is pretty evident, Many-to-Many RNN (Tx>1,Ty>1) Architecture takes multiple input and gives multiple output, but Many-to-Many models can be two kinds :

1. Tx=Ty :

This refers to the case when input and output layers have the same size , as can be seen in the “figure 11”. This can be also understood as every input having a output, and a common application can be found in Named-entity Recognition. [20]

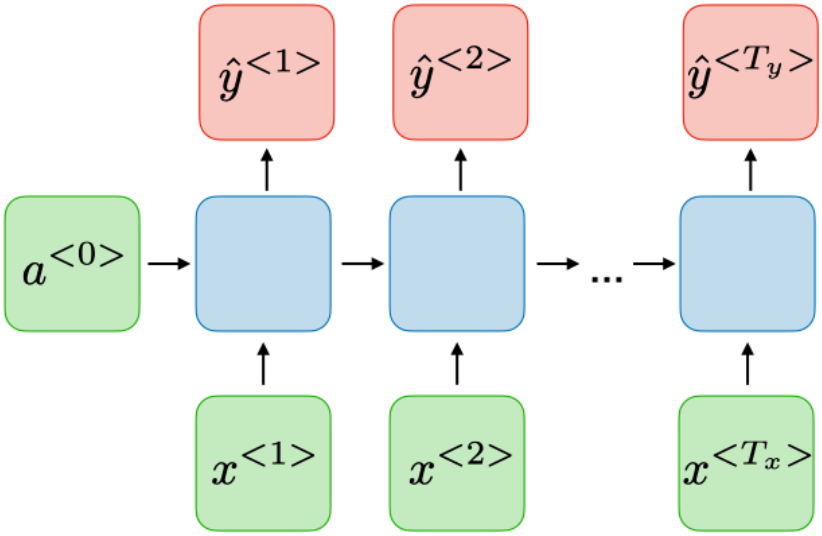


Figure 11: Tupe of RNN : Many-to-many [Tx=Ty]. [19]

2. Tx!=Ty :

Many-to-Many architecture can also be represented in models where input and output layers are of different size, as can be seen in the “figure 12”. and the most common application of this kind of RNN architecture is seen in Machine Translation. For example, "I Believe", the 2 words of the English language translates to only 1 in Arabic , "أعتقد". Thus, machine translation models are capable of returning words more or less than the input string because of a non-equal Many-to-Many RNN architecture works in the background. [20]

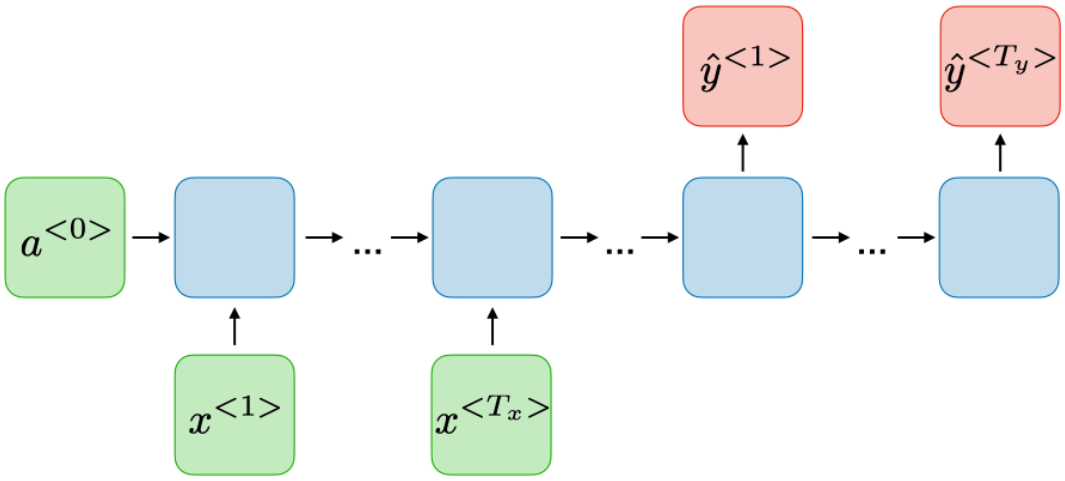


Figure 12: Tupe of RNN : Many-to-many [Tx!=Ty] . [19]

2.4 How a Recurrent Neural Network Works ? :

An RNN works as follows ; First words get transformed into machine-readable vectors. Then the RNN processes the sequence of vectors one by one.

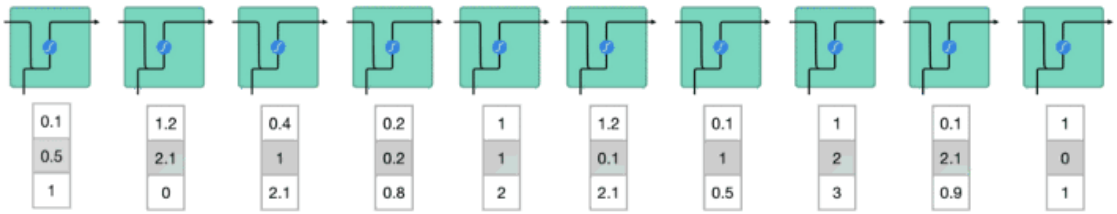


Figure 13: Processing sequence one by one .

While processing, it passes the previous hidden state to the next step of the sequence. The hidden state acts as the neural networks memory. It holds information on previous data the network has seen before .

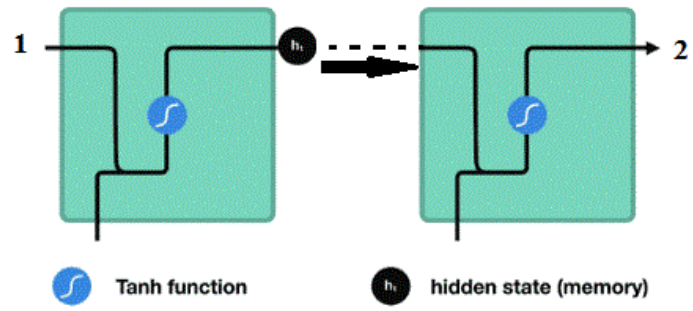


Figure 14: Passing hidden state to next time step .

Let's look at a cell of the RNN to see how we would calculate the hidden state. First, the input and previous hidden state are combined to form a vector. That vector now has information on the current input and previous inputs. The vector goes through the tanh activation, and the output is the new hidden state, or the memory of the network.

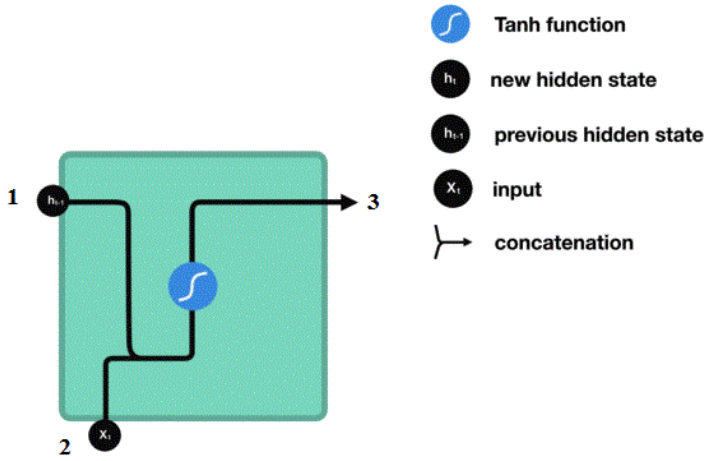


Figure 15: RNN Cell .

- **Tanh activation :**

  The tanh activation is used to help regulate the values flowing through the network. The tanh function squishes values to always be between -1 and 1 .

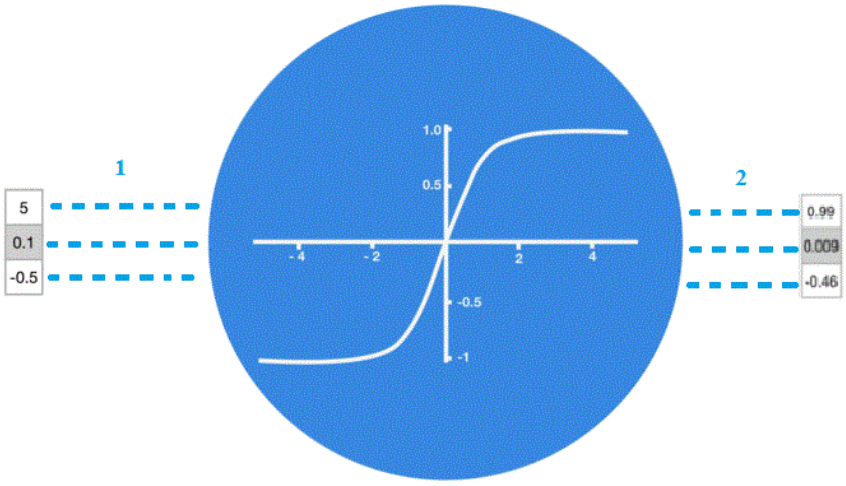


Figure 16: Tanh squishes values to be between -1 and 1 .

When vectors are flowing through a neural network, it undergoes many transformations due to various math operations. So imagine a value that continues to be multiplied by let's say 3. You can see how some values can explode and become astronomical, causing other values to seem insignificant.

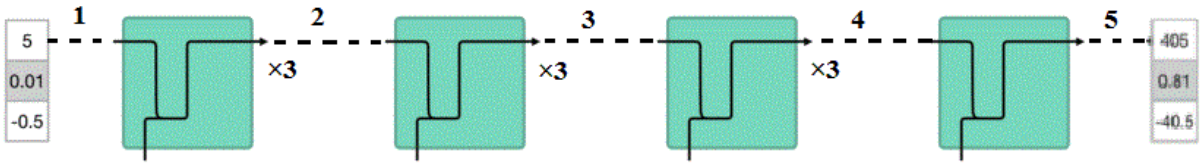


Figure 17: Vector transformations without tanh .

A tanh function ensures that the values stay between -1 and 1, thus regulating the output of the neural network. We can see how the same values from above remain between the boundaries allowed by the tanh function.

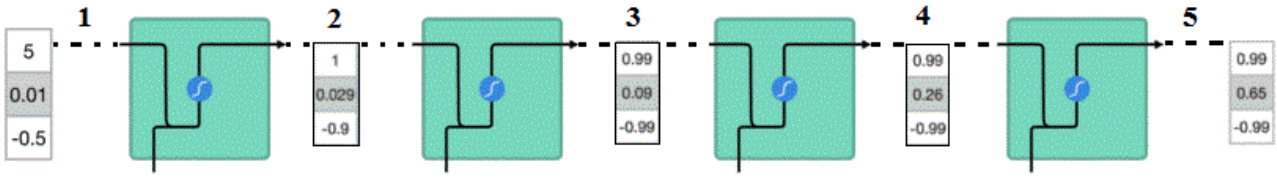


Figure 18: Vector transformations with tanh .

So that's an RNN. It has very few operations internally but works pretty well given the right circumstances (like short sequences). RNN's uses a lot less computational resources than it's evolved variants, LSTM's and GRU's.[24]

2.6 Applications of Recurrent Neural Networks :

2.6.1 Language Modelling and Generating Text :

Taking a sequence of words as input, we try to predict the possibility of the next word. This can be considered to be one of the most useful approaches for translation since the most likely sentence would be the one that is correct.In this method, the probability of the output of a particular time-step is used to sample the words in the next iteration.

2.6.2 Machine Translation :

RNNs in one form or the other can be used for translating text from one language to other . Almost all of the Translation systems being used today use some advanced version of a RNN. The input can be the source language and the output will be in the target language which the user wants.

Currently one of the most popular and prominent machine translation application is Google Translate.There are even numerous custom recurrent neural network applications used to refine and confine content by various platforms.

2.6.3 Speech Recognition :

RNNs can be used for predicting phonetic segments considering sound waves from a medium as an input source .The set of inputs consists of phoneme or acoustic signals from an audio which are processed in a proper manner and taken as inputs. The RNN network will compute the phonemes and then produce a phonetic segment along with the likelihood of output.The steps used in speech recognition are as follows:

- The input data is first processed and recognized through a neural network. The result consists of a varied collection of input sound waves.
- The information contained in the sound wave is further classified by intent and through keuwords related to the query.

- Then input sound waves are classified into phonetic segments and are pieced together into cohesive words using a RNN application. The output consists of a pattern of phonetic segments put together into a singular whole in a logical manner.

2.6.4 Generating Image Descriptions :

A combination of CNNs and RNNs are used to provide a description of what exactly is happening inside an image. CNN does the segmentation part and RNN then uses the segmented data to recreate the description.

2.6.5 Video Tagging :

RNNs can be used for video search where we can do image description of a video divided into numerous frames.

2.6.6 Text Summarization :

This application can provide major help in summarizing content from literatures and customising them for delivery within applications which cannot support large volumes of text. For example, if a publisher wants to display the summary of one of his books on its backpage to help the readers get an idea of the content present within, Text Summarization would be helpful. [22]

2.7 Variant RNN architectures :

2.7.1 Bidirectional recurrent neural networks (BRNN):

These are a variant network architecture of RNNs. While unidirectional RNNs can only drawn from previous inputs to make predictions about the current state, bidirectional RNNs pull in future data to improve the accuracy of it, More precisely "the output layer can obtain information from the past (backward) and future (forward) states simultaneously".

2.7.2 Long short-term memory (LSTM) :

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. It can not only process single data points (such as images), but also entire sequences of data (such as speech or video). For example, LSTM is applicable to tasks such as unsegmented, connected handwriting recognition,speech recognition and anomaly detection in network traffic or IDSs (intrusion detection systems).

A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.

LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs.

2.7.3 Gated recurrent units (GRUs) :

This RNN variant is similar the LSTMs as it also works to address the short-term memory problem of RNN models. Instead of using a "cell state" regulate information, it uses hidden states, and instead of three gates, it has two—a reset gate and an update gate. Similar to the gates within LSTMs, the reset and update gates control how much and which information to retain.[16]

3. Long Short Term Memory (LSTM) :

3.1 LSTM Definition :

Long Short-Term Memory (LSTM) networks are a modified version of recurrent neural networks, which makes it easier to remember past data in memory. The vanishing gradient problem of RNN is resolved here. LSTM is well-suited to classify, process and predict time series given time lags of unknown duration. It trains the model by using back-propagation. In an LSTM network, three gates are present: Input gate , Forget gate , Output gate .[23]

3.2 Workings of LSTM :

LSTM was created as a solution for short-term memory. It has internal mechanisms called gates that can regulate the flow of information.

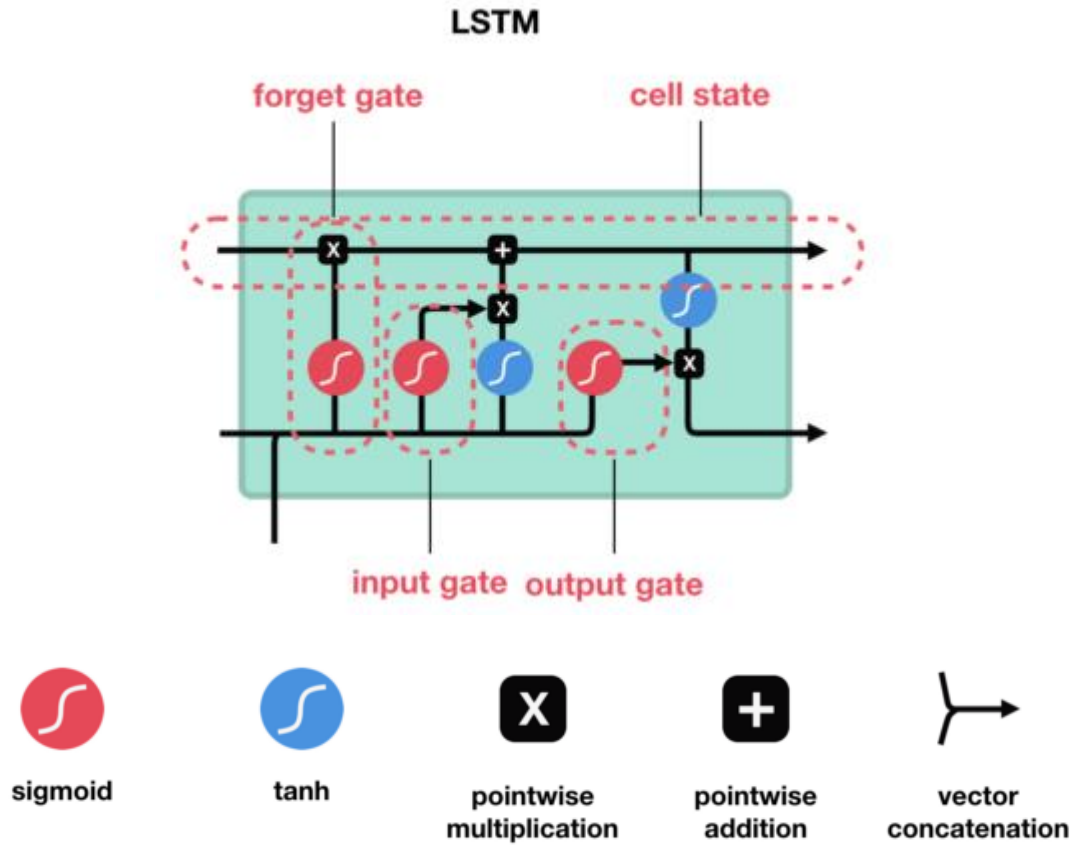


Figure 19: LSTM Gates .

These gates can learn which data in a sequence is important to keep or throw away. By doing that, it can pass relevant information down the long chain of sequences to make predictions. Almost all state of the art results based on recurrent neural networks are achieved with these network . LSTM's can be found in speech recognition, speech synthesis, and text generation. You can even use them to generate captions for videos.

Let's start with a thought experiment. Let's say you're looking at reviews online to determine if you want to buy Morning cereal . You'll first read the review then determine if someone thought it was good or if it was bad.

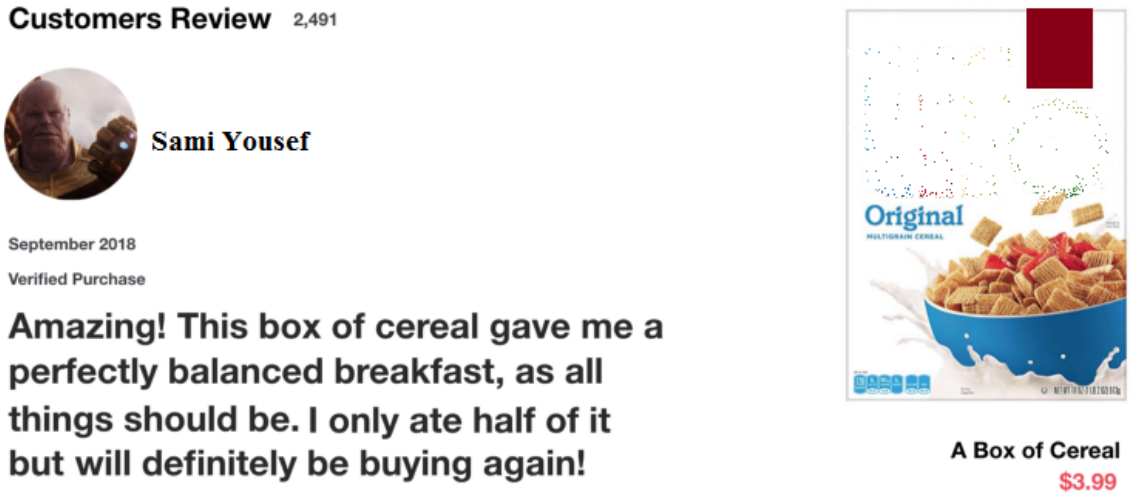


Figure 20: Read the review and then decide if the product is good or bad.

When you read the review, your brain subconsciously only remembers important keywords. You pick up words like “amazing” and “perfectly balanced breakfast”. You don’t care much for words like “this”, “gave“, “all”, “should”, etc. If a friend asks you the next day what the review said, you probably wouldn’t remember it word for word. You might remember the main points though like “will definitely be buying again”. the other words will fade away from memory.

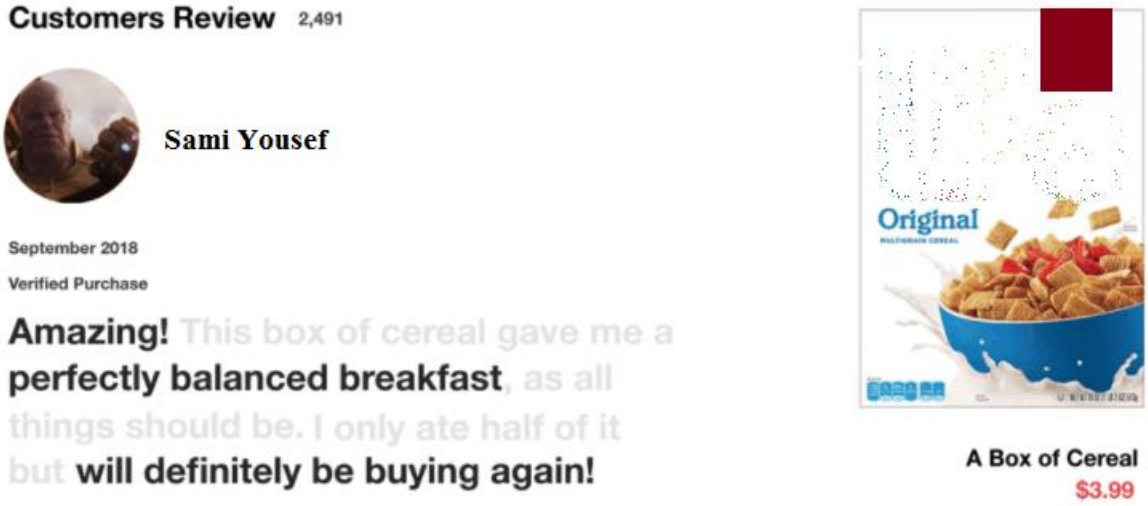


Figure 21: The brain subconsciously remembers only the important keywords.

And that is essentially what an LSTM does. It can learn to keep only relevant information to make predictions, and forget non relevant data. In this case, the words you remembered made you judge that it was good .

An LSTM has a similar control flow as a recurrent neural network(As already mentioned in the Recurrent Neural Network section , exactly in "4.5 How a Recurrent Neural Network Works?"). It processes data passing on information as it propagates forward. The differences are the operations within the LSTM's cells.

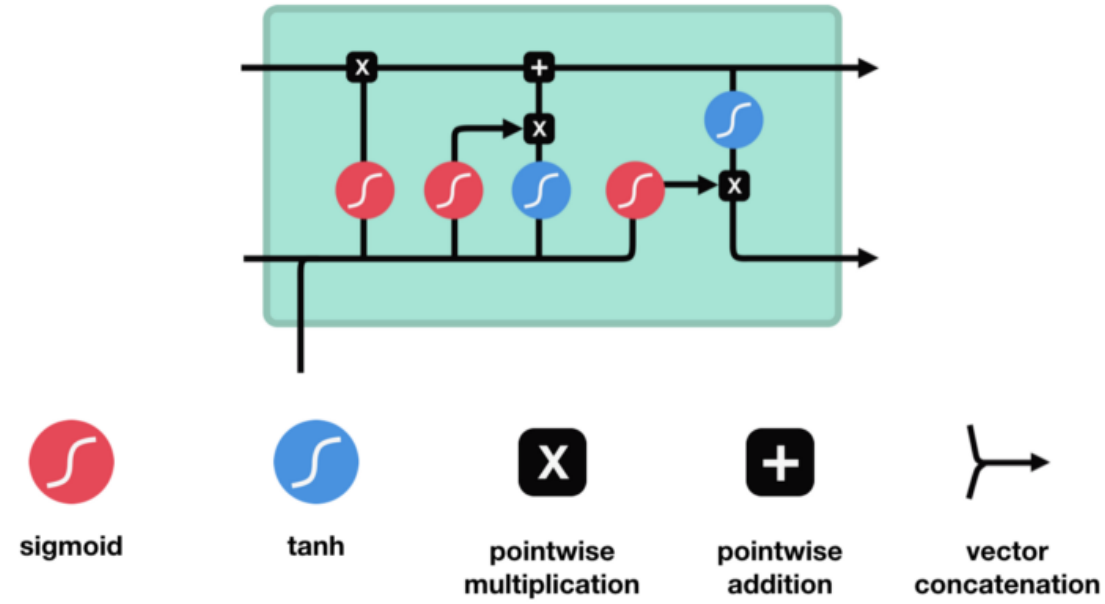


Figure 22: LSTM Cell and It's Operations .

These operations are used to allow the LSTM to keep or forget information.

- **Core Concept :**

The core concept of LSTM's are the cell state, and it's various gates. The cell state act as a transport highway that transfers relative information all the way down the sequence chain. We can think of it as the "memory" of the network. The cell state, in theory, can carry relevant information throughout the processing of the sequence. So even information from the earlier time steps can make it's way to later time steps, reducing the effects of short-term memory. As the cell state goes on its journey, information get's added or removed to the cell state via gates.

The gates are different neural networks that decide which information is allowed on the cell state. The gates can learn what information is relevant to keep or forget during training.

- **Sigmoid :**
- Gates contains sigmoid activations. A sigmoid activation is similar to the tanh activation. Instead of squishing values between -1 and 1, it squishes values between 0 and 1. That is helpful to update or forget data because any number getting multiplied by 0 is 0, causing values to disappears or be “forgotten.” Any number multiplied by 1 is the same value therefore that value stay’s the same or is “kept.” The network can learn which data is not important therefore can be forgotten or which data is important to keep.

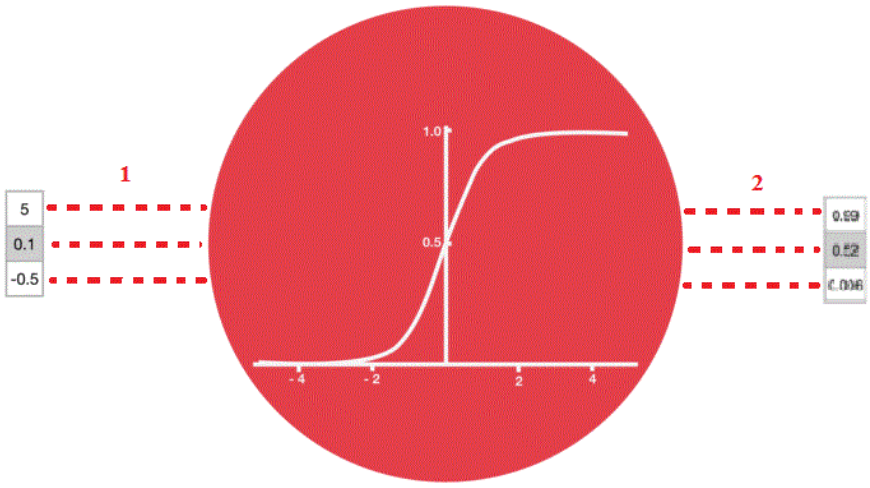


Figure 23: Sigmoid squishes values to be between 0 and 1 .

Let’s dig a little deeper into what the various gates are doing , So we have three different gates that regulate information flow in an LSTM cell. A forget gate, input gate, and output gate.

1) Forget gate :

First, we have the forget gate. This gate decides what information should be thrown away or kept. Information from the previous hidden state and information from the current input is passed through the sigmoid function. Values come out between 0 and 1. The closer to 0 means to forget, and the closer to 1 means to keep.

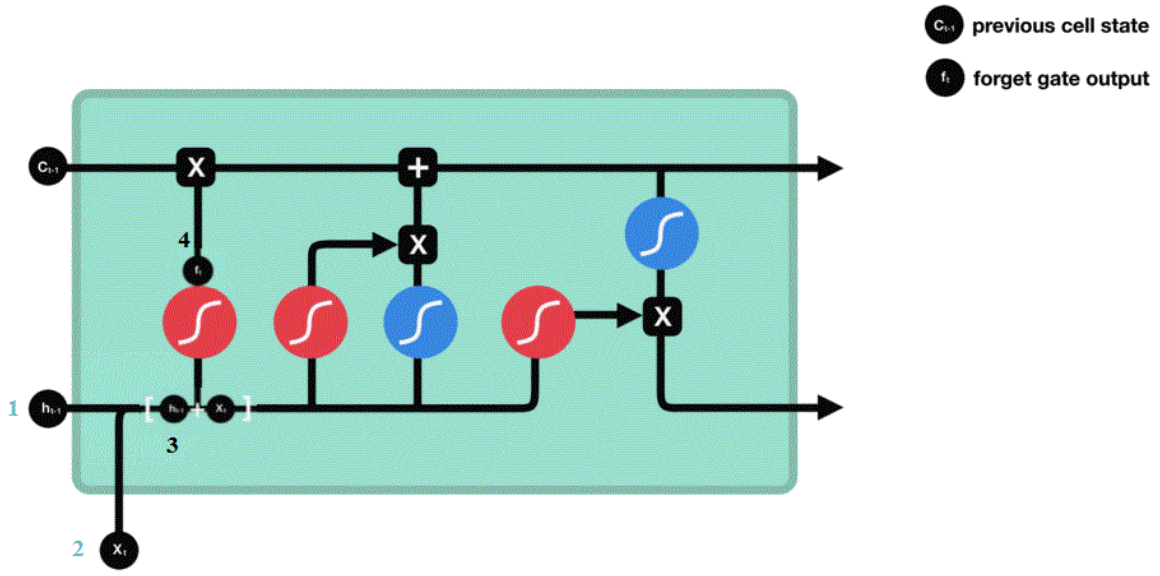


Figure 24: Forget gate operations .

2) Input Gate :

To update the cell state, we have the input gate. First, we pass the previous hidden state and current input into a sigmoid function. That decides which values will be updated by transforming the values to be between 0 and 1. 0 means not important, and 1 means important. You also pass the hidden state and current input into the tanh function to squish values between -1 and 1 to help regulate the network. Then you multiply the tanh output with the sigmoid output. The sigmoid output will decide which information is important to keep from the tanh output .

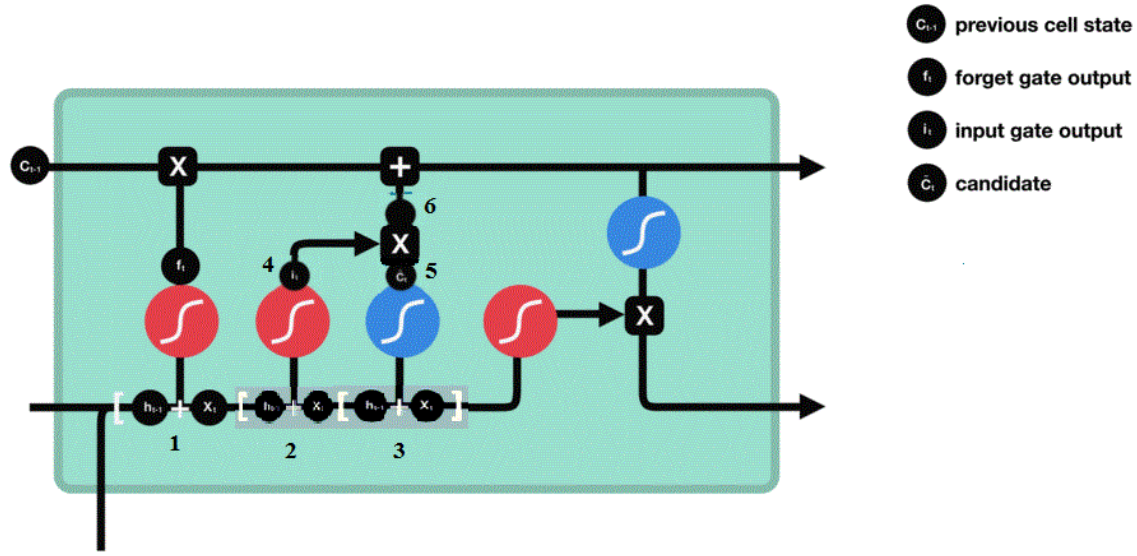


Figure 25: Input gate operations .

3) Cell State :

Now we should have enough information to calculate the cell state. First, the cell state gets pointwise multiplied by the forget vector. This has a possibility of dropping values in the cell state if it gets multiplied by values near 0. Then we take the output from the input gate and do a pointwise addition which updates the cell state to new values that the neural network finds relevant. That gives us our new cell state.

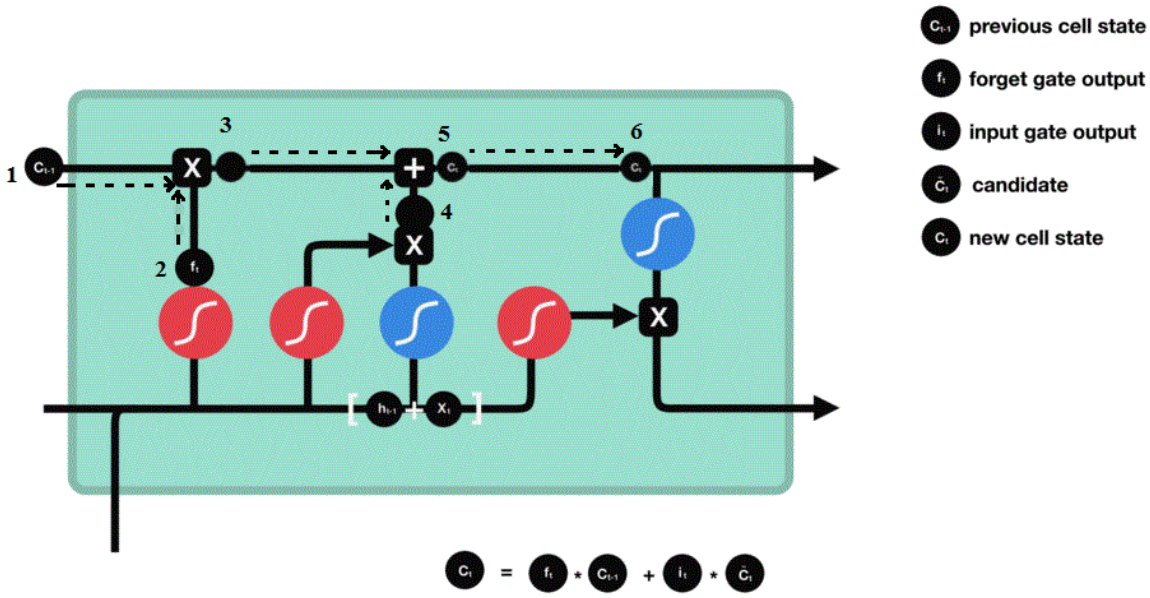


Figure 26: Calculating cell state .

4) Output Gate :

The output gate decides what the next hidden state should be. don't forget that the hidden state contains information on previous inputs. The hidden state is also used for predictions. First, we pass the previous hidden state and the current input into a sigmoid function. Then we pass the newly modified cell state to the tanh function. We multiply the tanh output with the sigmoid output to decide what information the hidden state should carry. The output is the hidden state. The new cell state and the new hidden is then carried over to the next time step.[24]

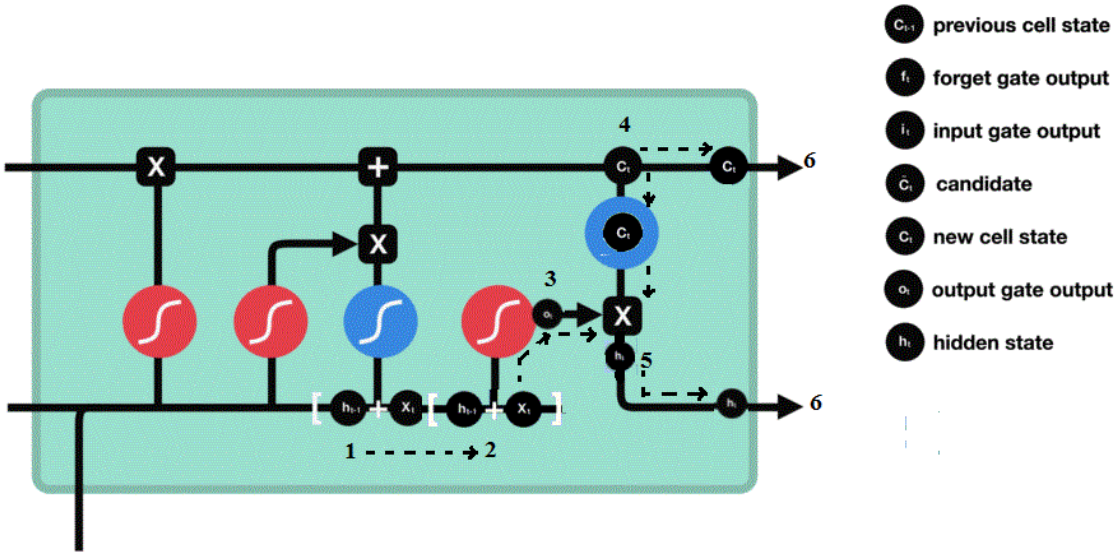


Figure 27: Output gate operations .

3.4 Architecture of LSTM :

- **Vanilla LSTM :** Memory cells of a single LSTM layer are used in a simple network structure.
- **Stacked LSTM :** LSTM layers are stacked one on top of another into deep networks.
- **CNN LSTM :** A convolutional neural network is used to learn features in spatial input like images and the LSTM can be used to support a sequence of images as input or generate a sequence in response to an image.
- **Encoder-Decoder LSTM :** One LSTM network encodes input sequences and a separate LSTM network decodes the encoding into an output sequence.

- **Bidirectional LSTM :** Input sequences are presented and learned both forward and backward.
- **Generative LSTM :** LSTMs learn the structure relationship in input sequences so well that they can generate new plausible sequences.[50]

4. Encoder-Decoder Sequence to Sequence Model :

4.1 Sequence-to-Sequence Model :

A sequence-to-sequence model is a model that takes a sequence of items (words, letters, features of an images…etc) and outputs another sequence of items.

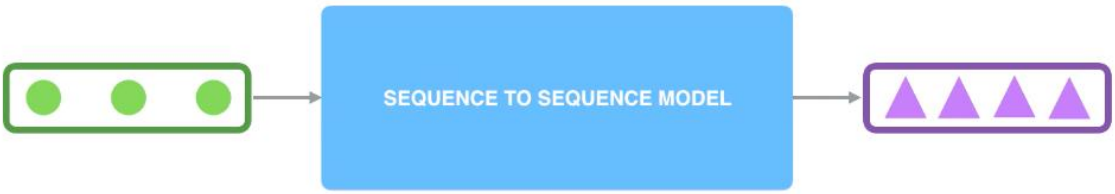


Figure 28: Sequence-to-Sequence Model .

In neural machine translation, a sequence is a series of words, processed one after another. The output is, likewise, a series of words :

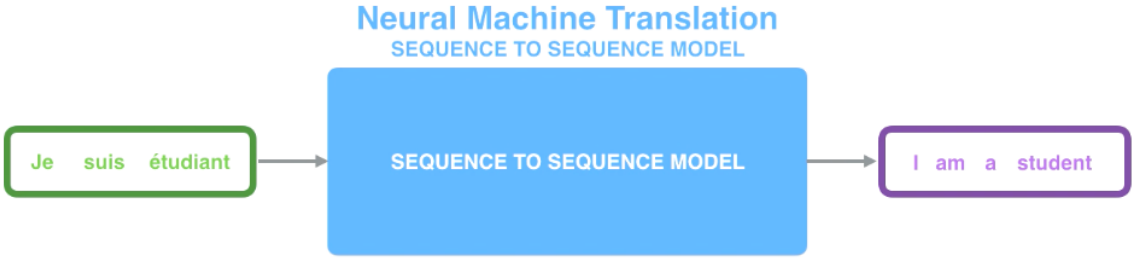


Figure 29: Neural Machine Translation "Sequence-to-Sequence Model".

- **Looking under the hood :**

Under the hood, the model is composed of an encoder and a decoder.

The encoder processes each item in the input sequence, it compiles the information it captures into a vector (called the context). After processing the entire input sequence, the encoder sends the context over to the decoder, which begins producing the output sequence item by item.

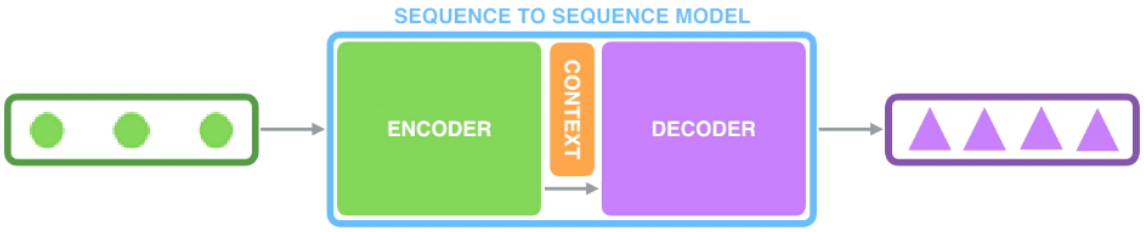


Figure 30: Encoder-Decoder Model .

The same applies in the case of machine translation .

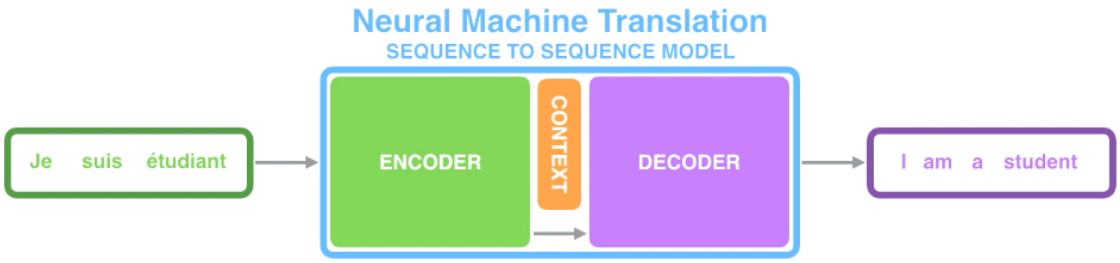


Figure 31: Neural Machine Translation “Encoder-Decoder Model”.

The context is a vector (an array of numbers, basically) in the case of machine translation. The encoder and decoder tend to both be recurrent neural networks .

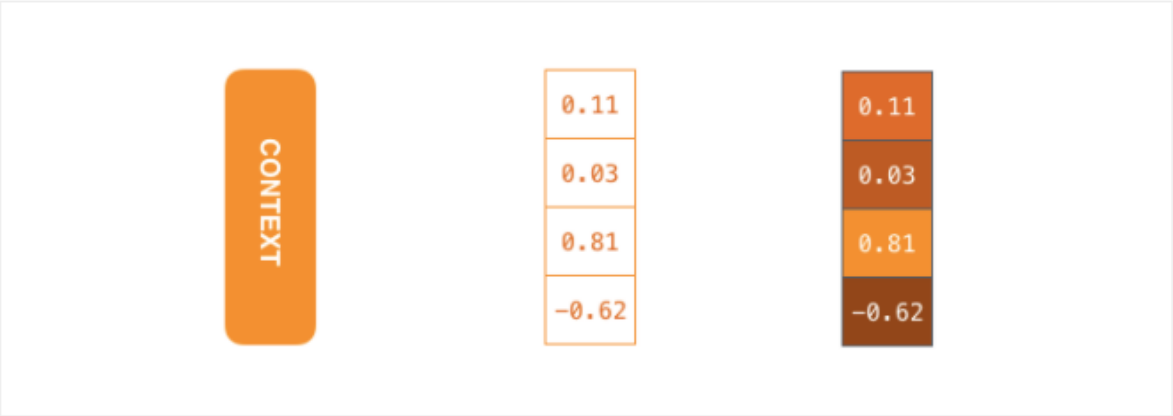


Figure 32: The context is a vector of floats.[25]

4.2 Encoder-Decoder system :

An encoder/decoder is a hardware tool that interprets information and converts it into a code, while also possessing the ability to convert that code back to its original source. In computing, an encoder takes either a sequence of characters or an analog signal and formats it for efficient transmission and/or storage.

The most common architecture used to build Seq2Seq models is the Encoder Decoder architecture.

- **Points to note :**

a. Both encoder and the decoder are typically LSTM models (or sometimes GRU models).
b. Encoder reads the input sequence and summarizes the information in something called as the internal state vectors (in case of LSTM these are called as the hidden state and cell state vectors). We discard the outputs of the encoder and only preserve the internal states.
c. Decoder is an LSTM whose initial states are initialized to the final states of the Encoder LSTM. Using these initial states, decoder starts generating the output sequence.
d. The decoder behaves a bit differently during the training and inference procedure. During the training, we use a technique call teacher forcing which helps to train the decoder faster. During inference, the input to the decoder at each time step is the output from the previous time step.

e. Intuitively, the encoder summarizes the input sequence into state vectors (sometimes also called as Thought vectors), which are then fed to the decoder which starts generating the output sequence given the Thought vectors. The decoder is just a language model conditioned on the initial states.[26]

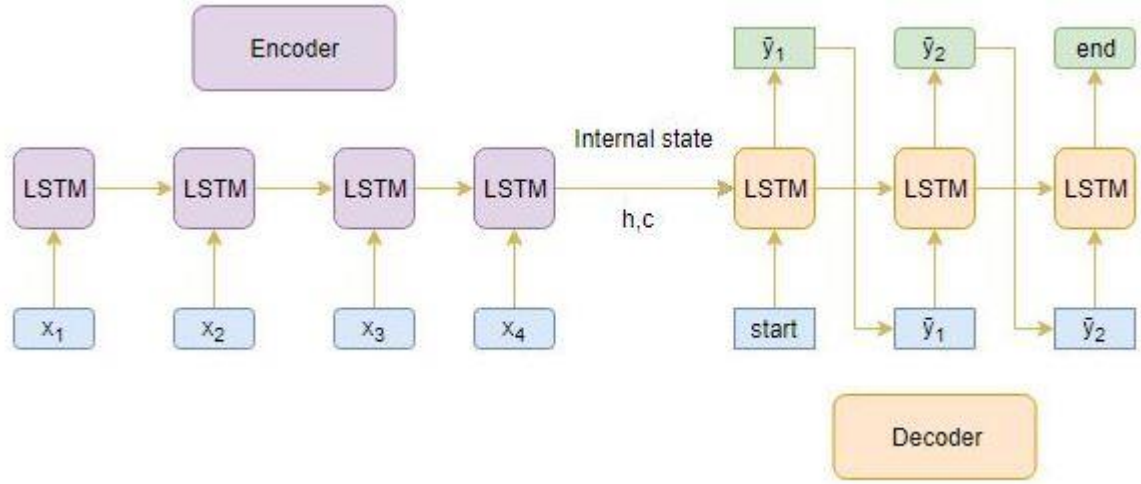


Figure 33: Encoder-Decoder Architecture .[27]

5. Word Embeddings :

- **The Word Embedding Model :**

Word embeddings are a type of word representation that allows words with similar meaning to have a similar representation. They are a distributed representation for text that is perhaps one of the key breakthroughs for the impressive performance of deep learning methods on challenging natural language processing problems.[50]

5.1 What Are Word Embeddings ? :

A word embedding is a learned representation for text where words that have the same meaning have a similar representation. It is this approach to representing words and documents that may be considered one of the key breakthroughs of deep learning on challenging natural language processing problems.

Word embeddings are in fact a class of techniques where individual words are represented as real-valued vectors in a predefined vector space. Each word is mapped to one vector and the vector values are learned in a way that resembles a neural network, and hence the technique is often lumped into the field of deep learning .

Key to the approach is the idea of using a dense distributed representation for each word.

Each word is represented by a real-valued vector, often tens or hundreds of dimensions. This is contrasted to the thousands or millions of dimensions required for sparse word representations, such as a one hot encoding .

The distributed representation is learned based on the usage of words. This allows words that are used in similar ways to result in having similar representations, naturally capturing their meaning. This can be contrasted with the crisp but fragile representation in a bag-of words model where, unless explicitly managed, different words have different representations, regardless of how they are used . [50]

5.2 Word Embedding Algorithms :

Word embedding methods learn a real-valued vector representation for a predefined fixed sized vocabulary from a corpus of text. The learning process is either joint with the neural network model on some task, such as document classification, or is an unsupervised process, using document statistics. This section reviews three techniques that can be used to learn a word embedding from text data.

5.2.1 Embedding Layer :

An embedding layer, for lack of a better name, is a word embedding that is learned jointly with a neural network model on a specific natural language processing task, such as language modeling or document classification. It requires that document text be cleaned and prepared such that each word is one hot encoded. The size of the vector space is specified as part of the model, such as 50, 100, or 300 dimensions. The vectors are initialized with small random numbers. The embedding layer is used on the front end of a neural network and is fit in a supervised way using the Backpropagation algorithm.

This approach of learning an embedding layer requires a lot of training data and can be slow, but will learn an embedding both targeted to the specific text data and the NLP task.

5.2.2 Word2Vec :

Word2Vec is a statistical method for efficiently learning a standalone word embedding from a text corpus. It was developed by Tomas Mikolov, et al. at Google in 2013 as a response o make the neural-network-based training of the embedding more efficient and since then has become the de facto standard for developing pre-trained word embedding .

Two different learning models were introduced that can be used as part of the Word2Vec approach to learn the word embedding; they are :

- Continuous Bag-of-Words, or CBOW model .
- Continuous Skip-Gram Model .

The CBOW model learns the embedding by predicting the current word based on its context . The continuous skip-gram model learns by predicting the surrounding words given a current word .

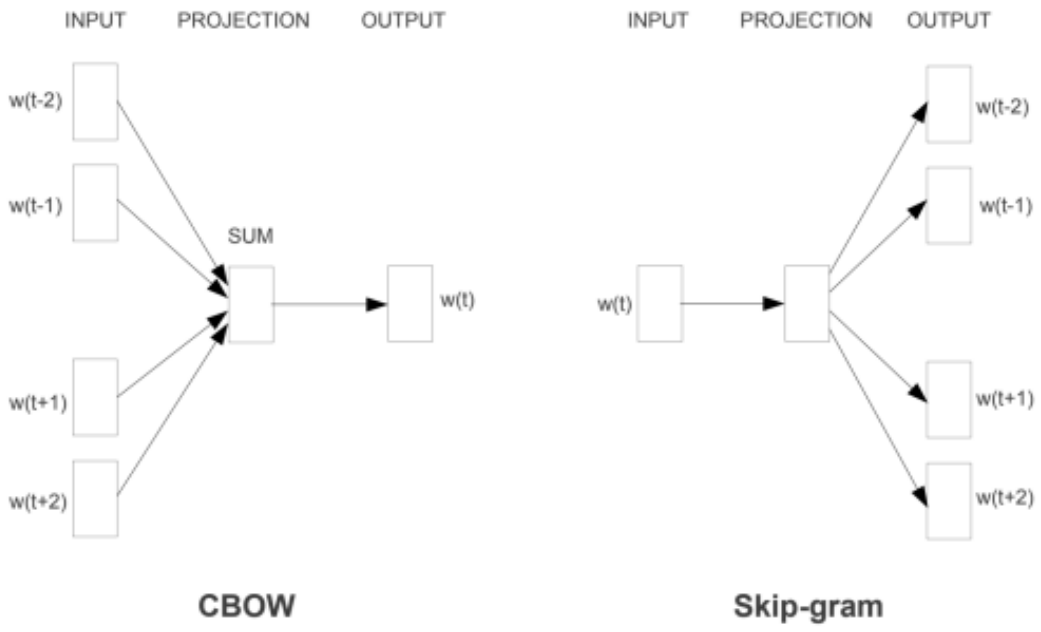


Figure 34: Word2Vec Training Models . [50]

Both models are focused on learning about words given their local usage context, where the context is defined by a window of neighboring words. This window is a configurable parameter of the model .

The key benefit of the approach is that high-quality word embeddings can be learned efficiently (low space and time complexity), allowing larger embeddings to be learned (more dimensions) from much larger corpora of text (billions of words) .

5.2.3 GloVe :

The Global Vectors for Word Representation, or GloVe, algorithm is an extension to the Word2Vec method for efficiently learning word vectors, developed by Pennington, et al. at Stanford. Classical vector space model representations of words were developed using matrix factorization techniques such as Latent Semantic Analysis (LSA) that do a good job of using global text statistics but are not as good as the learned methods like Word2Vec at capturing meaning and demonstrating it on tasks like calculating analogies

GloVe is an approach to marry both the global statistics of matrix factorization techniques like LSA with the local context-based learning in Word2Vec. Rather than using a window to define local context, GloVe constructs an explicit word-context or word co-occurrence matrix using statistics across the whole text corpus. The result is a learning model that may result in generally better word embeddings. [50]

5.3 Using Word Embeddings :

There are several options when it comes time to use word merging in a NLP project. This section explains those options.

5.3.1 Learn to embed :

There are two main options when training word embedding :

- **Independent learning :** where the model is trained to learn embedding, which is saved and used as part of another project task model. This is a good approach if you want to use the same embed on multiple forms .
- **Co-learning :** where embedding is learned as part of a model for a large task. This is a good approach if you only use embeds for one task .

5.3.2 Reuse an Embedding :

It is common for researchers to make pre-trained word embeddings available for free, often under a permissive license so that can use them on academic or commercial projects. For example, both Word2Vec and GloVe word embeddings are available for free download. These can be used instead of training embeddings from scratch. There are two main options when it comes to using pre-trained embeddings:

- **Static :** where the embedding is kept static and is used as a component of model.This is a suitable approach if the embedding is a good fit for problem and gives good results .
- **Updated :** where the pre-trained embedding is used to seed the model, but the embedding is updated jointly during the training of the model. This may be a good option if we are looking to get the most out of the model and embedding task .

5.3.3 Which option should we use? :

We explore different options and, if possible, test to see which gives the best results on the problem. We might start with quick methods, like using a pre-trained embed, and only use a new embed if it performs better in solving our problem . [50]

6. BLEU scores :

6.1 What it BLEU :

BLEU (BiLingual Evaluation Understudy) is a metric that is widely used to evaluate Natural Language Processing (NLP) systems which produce language, especially machine translation (MT) and Natural Language Generation (NLG) systems. Because BLEU itself just computes word-based overlap with a gold-standard reference text, its use as an evaluation metric depends on an assumption that it correlates with and predicts the real-world utility of these systems, measured either extrinsically (e.g., by task performance) or by user satisfaction.

Hence the usage of BLEU to evaluate NLP systems is only sensible in the presence of validation studies which show that BLEU scores correlate with direct evaluations of the utility of NLP systems. In rough terms, a validation study involves evaluating a number of NLP systems (or individual output texts) using both a metric such as BLEU and a gold standard

human evaluation, and then calculating how well the metric correlates with the gold-standard human evaluation .

BLEU's output is always a number between 0 and 1. This value indicates how similar the candidate text is to the reference texts, with values closer to 1 representing more similar texts. Few human translations will attain a score of 1, since this would indicate that the candidate is identical to one of the reference translations. For this reason, it is not necessary to attain a score of 1. Because there are more opportunities to match, adding additional reference translations will increase the BLEU score .[66]

6.2 How it works :

To conduct a BLEU measurement the following is necessary :

- One or more human reference translations. This should be data that has not been used in building the system (training data) and ideally should be unknown to the MT system developer.
- It is generally recommended that 1,000 or more sentences be used to get a meaningful measurement. Too small a sample set can sway the score significantly with just a few sentences that match or do not match well.
- Automated translation output of the exact same source data set.
- A measurement utility that performs the comparison and score calculation.

Scores are given to individual MT-translated segments – usually sentences – by comparing them with one or a set of good quality human reference translations. When a sentence is translated by two different MT systems, one might produce a translation that matches 75% of the words of the reference correct translation, while the translation of the second MT system might match 55% of the words. Both MT translations might be 100% correct, but the one with the 75% match will be assessed as having provided higher quality, which might seem somewhat arbitrary.

This potential problem is illustrated in the following example. Once we select one of these translations as the only reference, all the other correct translations will score lower.

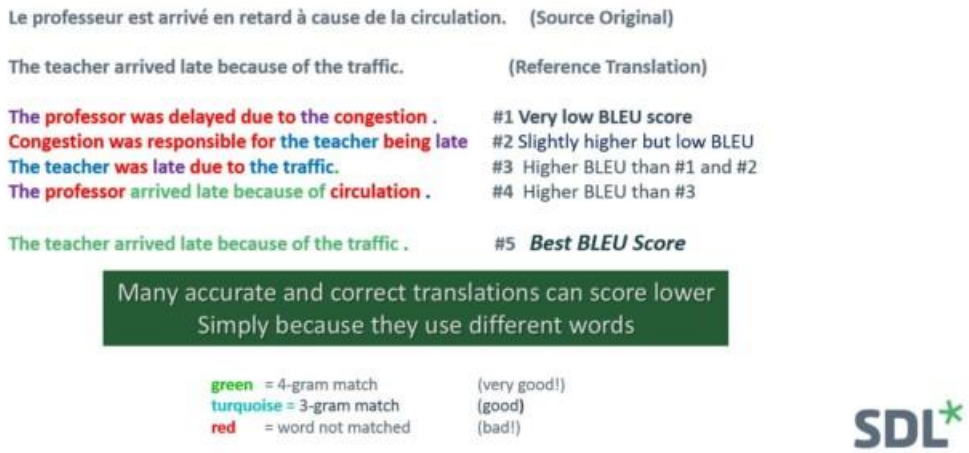


Figure 35: Example of Bleu score Work .[38]

The BLEU metric scores a translation on a scale of 0 to 1, in an attempt to measure the adequacy and fluency of the MT output. The closer to 1 the test sentences score, the more overlap there is with their human reference translations and thus, the better the system is deemed to be. BLEU scores are often stated on a scale of 1 to 100 to simplify communication, but this should not be confused with the percentage of accuracy .

The MT output would score 1 only if it is identical to the reference human translation. But even two competent human translations of the exact same material may only score in the 0.6 or 0.7 range as they are likely to use different vocabulary and phrasing. We should be wary of very high BLEU scores (in excess of 0.7) as it is probably measuring improperly or overfitting .

The BLEU metric also gives higher scores to sequential matching words. That is, if a string of four words in the MT translation match the human reference translation in the same exact order, it will have more of a positive impact on the BLEU score than a string of two matching words will. This means that an accurate translation will receive a lower score if it uses different, but correct words or matching words in a different word order .[38]

6.3 BLEU score algorithm :

BLEU compares the n-gram of the candidate translation with n-gram of the reference translation to count the number of matches. These matches are independent of the positions where they occur.

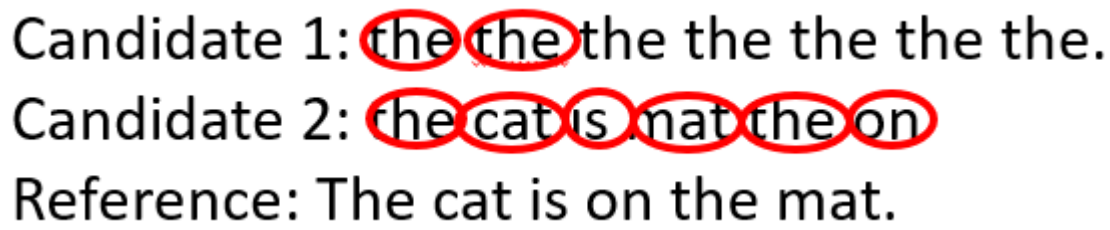
The more the number of matches between candidate and reference translation, the better is the machine translation.

- **Precision :**

In terms of Machine Translation, we define Precision as 'the count of the number of candidate translation words which occur in any reference translation' divided by the 'total number of words in the candidate translation.'

$$\text{Precision} = \frac{\text{No. of candidate translation words occuring in any reference translation}}{\text{Total no. of words in the candidate translation}}$$

- **Example :**



The precision for candidate 1 is 2/7 (28.5%)

The Precision for candidate 2 is 1(100%).

These are unreasonably high precision, and we know these are not good translations.

To solve the issue, we will use modified n-gram precision. It is computed in multiple steps for each n-gram.

- **Example :**

We have three human reference translation and a machine-translated candidate .

Candidate 1: It is a guide to action which ensures that the military always obeys the commands of the party.

Reference 1: It is a guide to action that ensures that the military will forever heed Party commands.

Reference 2: It is the guiding principle which guarantees the military forces always being under the command of the Party.

Reference 3: It is the practical guide for the army always to heed the directions of the party.

We first calculate Count clip for any n-gram using the following steps :

Step1: Count the maximum number of times a candidate n-gram occurs in any single reference translation; this is referred to as Count.

Step 2: For each reference sentence, count the number of times a candidate n-gram occurs. As we have three reference translations, we calculate, Ref 1 count, Ref2 count, and Ref 3 count.

Step 3: Take the maximum number of n-grams occurrences in any reference count. Also known as Max Ref Count.

Step 4: Take the minimum of the Count and Max Ref Count. Also known as Count clip as it clips the total count of each candidate word by its maximum reference count

$$Count_{Clip} = min(Count, Max_Ref_Count)$$

Step 5: Add all these clipped counts.

Below we have clip counts for unigram and bigrams

Candidate Count	Count	Ref1 Count	Ref2 Count	Ref3 Count	Max Ref Count	Clip Count
It	1	1	1	1	1	1
is	1	1	1	1	1	1
a	1	1	0	0	1	1
guide	1	1	0	1	1	1
to	1	1	0	1	1	1
action	1	1	0	0	1	1
which	1	0	1	0	1	1
ensures	1	1	0	0	1	1
that	2	2	0	0	2	2
the	3	1	4	4	4	3
military	1	1	1	0	1	1
always	1	0	1	1	1	1
obeys	0	0	0	0	0	0
commands	1	1	0	0	1	1
of	0	0	1	1	1	0
party	1	0	0	1	1	1
18						**17**

Table 3: Clip counts for unigram .

Candidate Count	Count	Ref1 Count	Ref2 Count	Ref3 Count	Max Ref Count	Clip Count
It is	1	1	1	1	1	1
is a	1	1	0	0	1	1
a guide	1	1	0	0	1	1
guide to	1	1	0	0	1	1
to action	1	1	0	0	1	1
action which	0	0	0	0	0	0
which ensures	0	0	0	0	0	0
ensures that	1	1	0	0	0	1
that the	1	1	0	0	1	1
the military	1	1	1	0	1	1
military always	0	0	0	0	0	0
always obeys	0	0	0	0	0	0
obeys the	0	0	0	0	0	0
the commands	0	0	0	0	0	0
commands of	0	0	0	0	0	0
of the	1	0	1	1	1	1
the party	1	0	0	1	1	1
17						**10**

Table 4: Clip counts for bigrams .

Step 6: Finally, divide the clipped counts by the total (unclipped) number of candidate n-grams to get the modified precision score.

$$p_n = \frac{\sum_{C \in \{Candidates\}} \sum_{n\text{-}gram \in C} Count_{clip}(n\text{-}gram)}{\sum_{C' \in \{Candidates\}} \sum_{n\text{-}gram' \in C'} Count(n\text{-}gram')}.$$

The modified precision score for the unigram is 17/18

The modified precision score for bi-gram is 10/17

- **Modified precision P :** Sum of the clipped n-gram counts for all the candidate sentences in the corpus divide by the number of candidate n-grams

♣ How does this modified precision score help?

Modified n-gram precision score captures two aspects of translation: adequacy and fluency.

- A translation using the same words as in the references tends to satisfy adequacy.
- The longer n-gram matches between candidate and reference translation account for fluency .

♣ What happens if the translations are too short or too long?

We add brevity penalty to handle too short translations.

Brevity Penalty(BP) will be 1.0 when the candidate translation length is the same as any reference translation length. The closest reference sentence length is the "best match length.".

With the brevity penalty, we see that a high-scoring candidate translation will match the reference translations in length, in word choice, and word order.

BP is an exponential decay and is calculated as shown below :

$$\text{Brevity Penalty} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases}$$

- r : count of words in a reference translation
- c : count of words in a candidate translation

Finally, we calculate BLEU .

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$

- BP- brevity penalty
- N: No. of n-grams, we usually use unigram, bigram, 3-gram, 4-gram
- W : Weight for each modified precision, by default N is 4, w is 1/4=0.25
- P : Modified precision

The BLEU metric ranges from 0 to 1. When the machine translation is identical to one of the reference translation, it will attain a score of 1. For this reason, even a human translator will not necessarily score 1.

Generally , BLEU measures the closeness of the machine translation to human reference translation taking translation length, word choice, and word order into consideration. In addition to translation, we can use the BLEU score for other language generation problems

with deep learning methods such as:

^ Language generation.

^ Image caption generation.

^ Text summarization.

^ Speech recognition.

^ And much more..[40]

7. Conclusion :

In this chapter, a general idea of Natural Language Processing is presented so that a simplified definition, its brief history, steps, major tasks, approaches and applications are presented .

After that, the Neural Network was discussed, and some information was provided about it, its definition, and a brief history , its construction, components, algorithms. And then more specialization was made in this section, so one of the most widely used types of neural network, which is called the Recurrent Neural Network, was presented.As well as its different architectures .

After that, the long-term memory was explained by providing a definition of it, as well as the way it works, in order to take a general idea of it because it is the basis of our work, and it and the method of Encoder-Decoder “Sequence to Sequence Model “, which also gave a brief idea.In this chapter we focused an the tools we will use in the next chapter, which is the used approach .



Chapter III : Proposed methodology.



1. Introduction :

After we have provided general ideas about some of the terms and methods, as well as the areas under which the idea of our work involves. We will present in this chapter in detail the used approach structure , proposed structure , and explain each step in detail.

In the approach we collect a corpus, which will pass through several stages of pre-processing, besides implementing LSTM, to see if there is a good improvement or not in the validation .

2. Global Architecture :

Here, we present the different phases followed to design our proposed approach, furthermore explaining the details of each phase, starting by giving a general diagram on our approach. The goal is Trying to reach an acceptable translation between the two languages English and Arabic.

We focus on the data set to be rich, the secret to correct translation is machine training.

In the other hand,LSTM is implemented to enhance our corpus to see if it will improve the results.

We organize our system according to a single categorization approach which is the machine learning approach, we can manually annotate a learning corpus and use a classifier to categorize the messages in the test corpus.

We used this approach to a Arabic corpus trained with LSTM models, then to a other languages corpus trained with same models, at the end we compare them.

We present bellow the general architecture .giving the possible steps :

• Clean and prepare data .

• develop encoder-decoder LSTMs Model .

• model training and evaluation model skill .

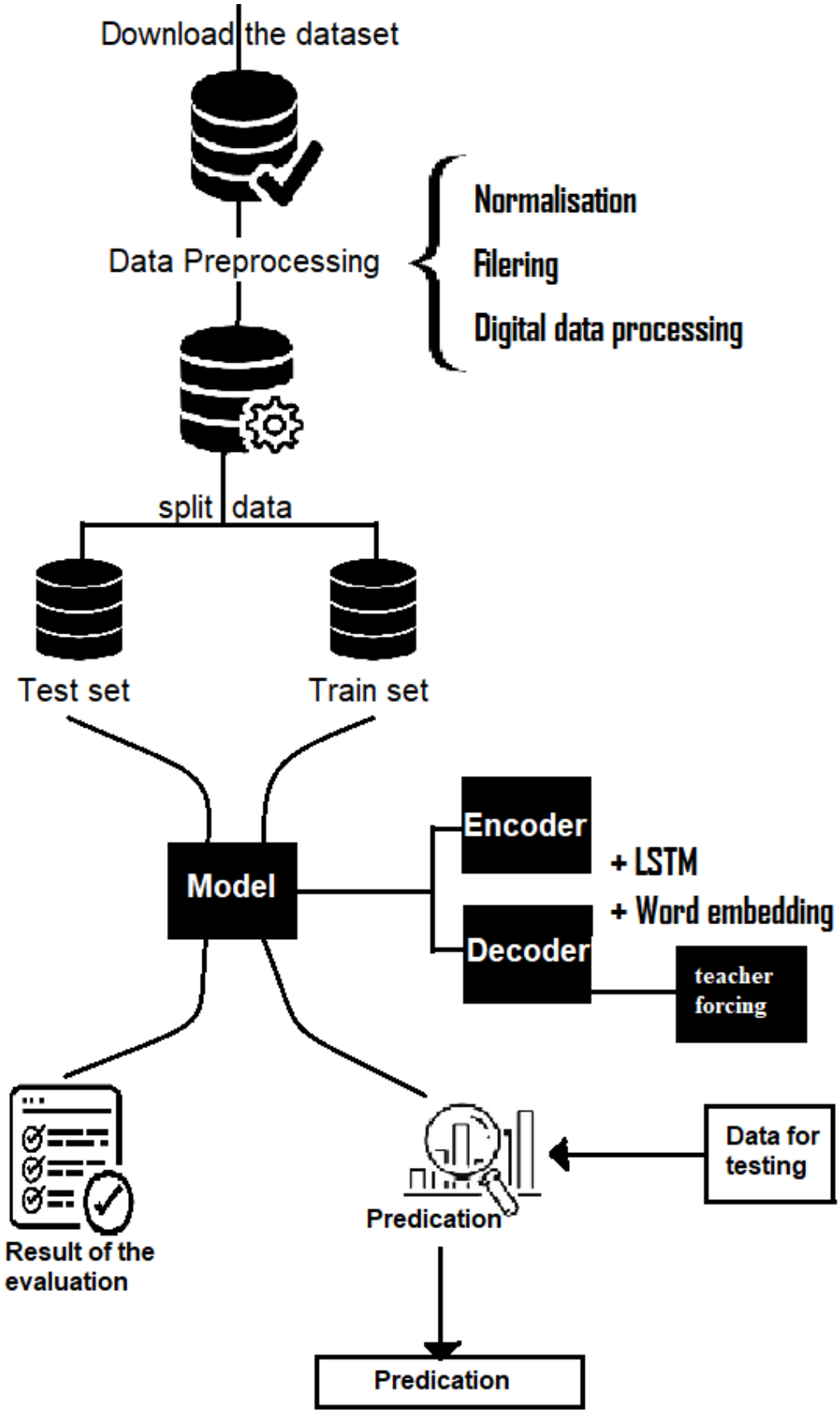


Figure 36: General architecture of our proposed approach .

3. Process steps :

In what follows, we will talk about each step of the process in our work

3.1 Data Preparation :

3.1.1 Dataset Definition :

A data set is a set of data that is collected for a specific purpose. There are many ways in which data can be collected: for example, as part of service delivery, one-off surveys, interviews, observations, and so on. In order to ensure that the meaning of data in the data set is clearly understood and data can be consistently collected and used.[64]

Our work is based on a set containing datasets for language translation tasks, prepared by the Tatoeba Project .

- **Tatoeba project :**

Tatoeba is a large database of sentences and translations. Its content is ever-growing and results from the voluntary contributions of thousands of members .

Tatoeba provides a tool for you to see examples of how words are used in the context of a sentence. You specify words that interest you, and it returns sentences containing these words with their translations in the desired languages. The name Tatoeba (for example in Japanese) captures this concept .

The project was founded by Trang Ho in 2006, hosted on Sourceforge under the codename of multilangdict .[39]

3.1.2 Load and split Data :

We load the text data so that we can work with it. The text is small and will load quickly and easily fit into memory. This will not always be the case and you may need to write code to memory map the file. Tools like NLTK will make working with large files much easier .[50]

We read the file into a list, one line at a time. You can do that with the readlines() method.

- **Cleaning Data :**

- **Normalizing Case :**

It is common to convert all words to one case. This means that the vocabulary will shrink in size, but some distinctions are lost (e.g. Apple the company vs apple the fruit is a commonly used example). We can convert all words to lowercase by calling the lower() function on each word . [50]

[I'm not OK] ➔ [i’m not ok]

- **Split by tabulation :**

One way would be to split the document into words by tabulation .

['Hi.\t.مرحبًا] ➔ ['Hi.’] [’.مرحبًا ’]

- **Remove Punctuation, Arabic diacritics, Numbers :**

It helps to get rid of unhelpful parts of the data, or noise .

Remove Punctuation : [كيف حالك؟] ➔ [كيف حالك]

[i'm not ok] ➔ [im not ok]

Arabic diacritics : [. مرحبًا] ➔ [. مرحبا]

Numbers : [1990 year] ➔ [year]

3.1.3 Digital Data :

in order to prepare sequence prediction data for use with LSTM models , We have 4 ways to do it :

scale numeric data and transform categorical data also pad and truncate input sequences with varied lengths and transform input sequences into a supervised learning problem .

- **Prepare Categorical Data :**

Categorical data are variables that contain label values rather than numeric values. The number of possible values is often limited to a fixed set. Categorical variables are often called nominal.

Each value represents a different category. Words in text may be considered categorical data, where each word is considered a different category. Additionally, each letter in text data may be considered a category. Sequence prediction problems with text input or output may be considered categorical data.

Some categories may have a natural relationship to each other, such as a natural ordering.

The place variable above does have a natural ordering of values. This type of categorical variable is called an ordinal variable. Categorical data must be converted to numbers when working with LSTMs .

- **Convert Categorical Data to Numerical Data :**

Either Integer Encoding. or One Hot Encoding.[49]

- **One Hot Encoding :**

The one hot encoding is converting each words to a binary vector with the same length and marking the specific word with a 1.

```
            ["Red" " Blue" "White" "Yellow" "Black" "Pink" ]
White    [  0    0    1    0    0    0 ]
Pink     [  0    0    0    0    0    1 ]
```

1.1 Building the Model :

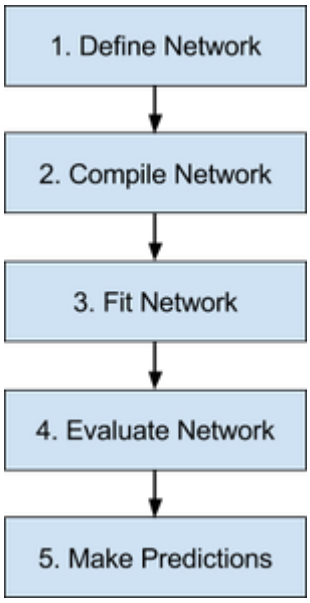


Figure 37: Steps in the neural network model life cycle .[33]

3.1.4 Define the Model :

3.1.4.1 Definition of Teacher forcing :

During training decoder receives the correct output from the training set as the previously decoded result to predict the next output. However, during inference decoder receives the previously decoded result to predict the next output. Teacher forcing improves training process.

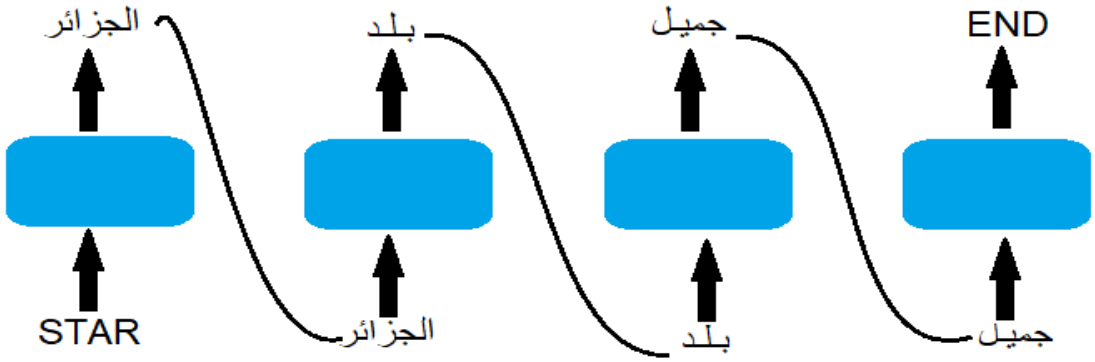


Figure 38: Example of Method of Teacher forcing .

3.1.4.2 Training encoder - decoder with teacher forcing :

- For the first output :
- Decoder produce the output sequece one by one .
- For each output, the decoder consumes a context vector and an input .
- The initial context vector is created by the encoder .
- The initial input is a special symbol for decoder to make it start, **e.g.** 'start' .
- Using initial context and initial input, decoder will generate first output .
- For the next output :
- Decoder will use its current state as context vector and generated (predicted) output as input .
- We (the teacher!) provide the correct output to the decoder as input .
- The difference is: decoder use the context vector and the correct input to next output rather than using its prediction in previous cycle .

- Decoder will work in such a loop using its state and output provided correct output as the next step context vector and input until the generated output is a special symbol 'stop' or the pre-defined maximum steps (length of output) is reached .[32]

3.1.4.3 Define the Encoder-Decoder LSTM :

We can think of the model as being comprised of two key parts: the encoder and the decoder.

The first step is to define your network for both parts. You create your layers and add them in the order that they should be connected .

The first layer in the network must define the number of inputs to expect , Then embedding layer and One or more LSTM layers can be used to implement the encoder model. The output of this model is a fixed-size vector that represents the internal representation of the input sequence. The number of memory cells in this layer defines the length of this fixed-sized vector.

The decoder must transform the learned internal representation of the input sequence into the correct output sequence.the first we define the number of inputs , then embedding layer and One or more LSTM layers can also be used to implement the decoder model. This model reads from the fixed sized output from the encoder model. And also a hidden layers (dense layers) ,and finally Dense layer is used as the output for the network .

3.1.4.4 Define the layers :

1- Input layer :

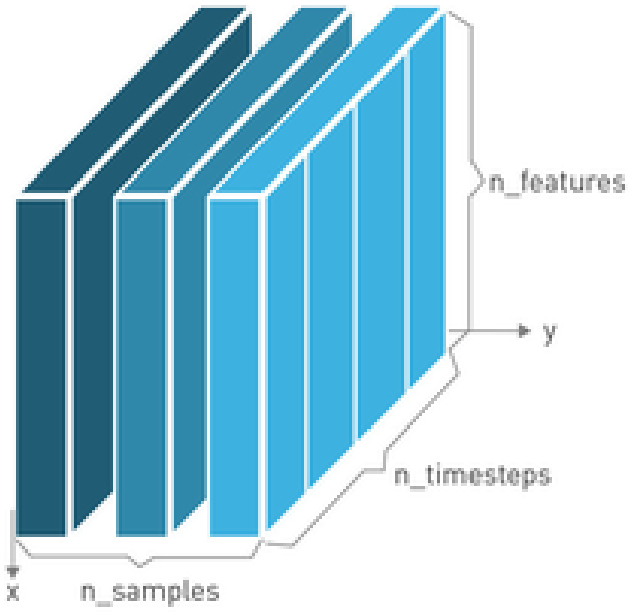


Figure 39: input layer .

[65]

The first layer in the network must define the number of inputs to expect(the shape of the input layer). Input must be three-dimensional, comprised of samples, time steps, and features in that order .

Samples. These are the rows in your data. One sample may be one sequence .

Time steps. These are the past observations for a feature, such as lag variables .

Features. These are columns in your data .

Assuming your data is loaded as a NumPy array, you can convert a 1D or 2D dataset to a 3D dataset using the reshape() function in NumPy .[49]

2- Embedding layer :

The Embedding layer(the first hidden layer) is initialized with random weights and will learn an embedding for all of the words in the training dataset. It is a flexible layer that can be used in a variety of ways, such as :

- It can be used alone to learn a word embedding that can be saved and used in another model later .
- It can be used as part of a deep learning model where the embedding is learned along with the model itself .
- It can be used to load a pre-trained word embedding model, a type of transfer learning .

The Embedding layer is defined as the first hidden layer of a network. It must specify 3 arguments :

- input dim: This is the size of the vocabulary in the text data .
- output dim: This is the size of the vector space in which words will be embedded.
- input length: This is the length of input sequences .[50]

3- LSTM layer :

LSTM has 3 important parameters

i. units: Positive integer, dimensionality of the output space
ii. return_sequences: Boolean, whether to return the last output. In the output sequence, or the full sequence. Default: False.
iii. return_state: Boolean, whether to return the last state in addition to the output. Default: False .[29]

The first parameter (units) indicates the dimension of the output vector/matrix.

The last 2 parameters (return_sequences and return_state) determines what the LSTM layer outputs. LSTM can return 4 different sets of results/states according to the given parameters:

1. return_sequences=False+ return_state=False(or Default): Last Hidden State (Hidden State of the last time step) .
2. return_sequences=True : All Hidden States (Hidden State of ALL the time steps) .
3. return_state=True : Last Hidden State+ Last Hidden State (again!) + Last Cell State (Cell State of the last time step) .
4. return_sequences=True + return_state=True: All Hidden States (Hidden State of ALL the time steps) + Last Hidden State + Last Cell State (Cell State of the last time step) .

➔When return_state and return_sequences parameters are False at the same time .

The size of output is 2D array of real numbers .

The first dimension is indicating the number of samples in the batch given to the LSTM layer .

The second dimension is the dimensionality of the output space defined by the units parameter in Keras LSTM implementation .

➔When return_sequences parameter is True .

The ouput is a 3D array of real numbers .

The first dimension is indicating the number of samples in the batch given to the LSTM layer .

The second dimension is the number of time steps in the input sequence. By indexing second dimension you can access all the hidden states of the units at a given time step .

The third dimension is the dimensionality of the output space defined by the units parameter in Keras LSTM implementation .

The content of the array is all the hidden states of each time steps of the LSTM layer .

➔When return_state parameter is True .

The ouput is a three 2D-arrays of real numbers.

The first dimension is indicating the number of samples (batch size) given to the LSTM layer .

The second dimension is the dimensionality of the output space defined by unit parameter in the Keras LSTM layer .

It returns 3 arrays in the result :

1-The LSTM hidden state of the last time step .
2-The LSTM hidden state of the last time step (again) .
3-The LSTM cell state of the last time step .

➔When return_state and return_sequences parameters are True at the same time .

In this situation, LSTM layer returns 3 results (as return_sequences=True)
1-the hidden states for each input time step (as return_state=True) .
2-the hidden state output for the last time step .
3-the cell state for the last time step .[31]

4- Dense layer :

The dense layer is a neural network layer that is connected deeply, which means each neuron in the dense layer receives input from all neurons of its previous layer. The dense layer is found to be the most commonly used layer in the models .

In the background, the dense layer performs a matrix-vector multiplication. The values used in the matrix are actually parameters that can be trained and updated with the help of backpropagation.

The output generated by the dense layer is an 'm' dimensional vector. Thus, dense layer is basically used for changing the dimensions of the vector. Dense layers also applies operations like rotation, scaling, translation on the vector .[30]

- **Keras Dense Layer Parameters used :**

  - units: Positive integer, dimensionality of the output space.
  - activation: Activation function to use. If you don't specify anything, no activation is applied (ie. "linear" activation: a(x) = x). [28]

5. **Output layer :**

It is the last dense layer, to which the above applies .

3.2 Training the Model :

3.2.1 Compile the Model :

After defining a model.Compilation requires a number of parameters to be specified, specifically tailored to training your network. Specifically, the optimization algorithm to use to train the network and the loss .

The type of predictive modeling problem imposes constraints on the type of loss function that can be used.below are some standard loss functions for different predictive model types :

- **Regression:** Mean Squared Error or mean squared error, mse for short .
- **Binary Classification (2 class):** Logarithmic Loss, also called cross entropy or binary Crossentropy .
- **Multiclass Classification (>2 class):** Multiclass Logarithmic Loss or categorical crossentropy .

The most common optimization algorithm is classical stochastic gradient descent, but Keras also supports a suite of other extensions of this classic optimization algorithm that work well . with little or no configuration. Perhaps the most commonly used optimization algorithms because of their generally better performance are :

- Stochastic Gradient Descent, or sgd .
- Adam, or adam .
- RMSprop, or rmsprop .

We can also specify metrics to collect while fitting your model in addition to the loss function. Generally, the most useful additional metric to collect is accuracy for classification problems (**e.g.** ‘accuracy’ or ‘acc’ for short). The metrics to collect are specified by name in an array of metric or loss function names .[49]

3.2.2 Fit the Model :

Once the network is compiled, it can be fit, which means adapting the weights on a training dataset. Fitting the network requires the training data to be specified, both a matrix of input patterns, X, and an array of matching output patterns, y .

The network is trained using the Backpropagation Through Time algorithm and optimized according to the optimization algorithm and loss function specified when compiling the model.

The backpropagation algorithm requires that the network be trained for a specified number of epochs or exposures to all sequences in the training dataset. Each epoch can be partitioned into groups of input-output pattern pairs called batches .

This defines the number of patterns that the network is exposed to before the weights are updated within an epoch. It is also an eciency optimization, ensuring that not too many input patterns are loaded into memory at a time .

- **Epoch :** One pass through all samples in the training dataset and updating the network weights. LSTMs may be trained for tens, hundreds, or thousands of epochs .
- **Batch :** A pass through a subset of samples in the training dataset after which the network weights are updated. One epoch is comprised of one or more batches .

Once fit, a history object is returned that provides a summary of the performance of the model during training. This includes both the loss and any additional metrics specified when compiling the model, a progress bar is displayed on the command line for each epoch .

Training can take a long time, from seconds to hours to days depending on the size of the network and the size of the training data .[49]

3.2.3 Evaluate the Model :

Once the network is trained, it can be evaluated. The network can be evaluated on the training data, but this will not provide a useful indication of the performance of the network as a predictive model, as it has seen all of this data before. We can evaluate the performance of the network on a separate dataset, unseen during training. This will provide an estimate of the performance of the network at making predictions for unseen data in the future .

The model evaluates the loss across all of the test patterns, as well as any other metrics specified when the model was compiled, like classification accuracy. A list of evaluation metrics is returned .[49]

One of the most common automatic evaluation metric is BLEU Score

- **Evaluate Generated Text With the BLEU Score :**

- **sentence bleu() :** the function for evaluating a candidate sentence against one or more reference sentences.
- **corpus bleu() :** the function for calculating the BLEU score for multiple sentences such as a paragraph or a document. The references must be specified as a list of documents where each document is a list of references and each alternative reference is a list of tokens

The BLEU score calculations allow us to specify the weighting of different n-grams in the calculation of the BLEU score.such as individual and cumulative n-gram scores.

- **Individual n-gram Scores :**

An individual n-gram score is the evaluation of just matching grams of a specific order, such as single words (1-gram) or word pairs (2-gram or bigram). The weights are specified as a tuple where each index refers to the gram order. To calculate the BLEU score only for 1-gram matches, you can specify a weight of 1 for 1-gram and 0 for 2, 3 and 4

- **Cumulative n-gram Scores :**

Cumulative scores refer to the calculation of individual n-gram scores at all orders from 1 to n and weighting them by calculating the weighted geometric mean. By default, the sentence bleu() and corpus bleu() scores calculate the cumulative 4-gram BLEU score, also called BLEU-4 .

3.3 Testing the Machine Translation :

- **Make Predictions on the Model :**

Once we are satisfied with our fit model, we use it to make predictions on new data .

The predictions will be returned in the format provided by the output layer of the network.

In the case of a regression problem, these predictions may be in the format of the problem directly, provided by a linear activation function .

For a binary classification problem, the predictions may be an array of probabilities for the first class that can be converted to a 1 or 0 by rounding. For a multiclass classification problem, the results may be in the form of an array of probabilities (assuming a one hot encoded output variable) that may need to be converted to a single class output prediction using the argmax() NumPy function .[49]

4. Conclusion :

In this chapter, our proposed system is described, which aims to design a system capable of improving machine translation .

This design will be put into practice in the next chapter .



Chapitre VI : Implementation , Results and discussions.



1. Introduction :

We present in this chapter the implementation of our proposed approach, we start with the presentation of the development environment, detail the different tools used, then we explain the steps of our implementation, in the end we interpret and comment the results obtained .

2. Development environment :

2.1 Python :

Python is a high-level, interpreted and general-purpose dynamic programming language that focuses on code readability. The syntax in Python helps the programmers to do coding in fewer steps as compared to Java or C++. The language founded in the year 1991 by the developer Guido Van Rossum has the programming easy and fun to do. The Python is widely used in bigger organizations because of its multiple programming paradigms. They usually involve imperative and object-oriented functional programming. It has a comprehensive and large standard library that has automatic memory management and dynamic features.[36]

2.2 Python Features :

2.2.1 Open source :

Python is publicly available open source software, anyone can use source code that doesn't cost anything.

2.2.2 Easy-to-learn :

Popular (scripting/extension) language, clear and easy syntax, no type declarations, automatic memory management, high-level data types and operations, design to read (more English like syntax) and write (shorter code compared to C, C++, and Java) fast.

2.2.3 High-level Language :

High-level language (closer to human) refers to the higher level of concept from machine language (for example assembly languages). Python is an example of a high-level language like C, C++, Perl, and Java with low-level optimization.

2.2.4 Portable :

High level languages are portable, which means they are able to run across all major hardware and software platforms with few or no change in source code. Python is portable and can be used on Linux, Windows, Macintosh, Solaris, FreeBSD, OS/2, Amiga, AROS, AS/400 and many more.

2.2.5 Object-Oriented :

Python is a full-featured object-oriented programming language, with features such as classes, inheritance, objects, and overloading.

2.2.6 Python is Interactive :

Python has an interactive console where you get a Python prompt (command line) and interact with the interpreter directly to write and test your programs. This is useful for mathematical programming.

2.2.7 Interpreted :

A Python program are interpreted, takes source code as input, and then compiles (to portable byte-code) each statement and executes it immediately. No need to compiling or linking.

2.2.8 Extendable :

Python is often referred to as a "glue" language, meaning that it is capable to work in mixed-language environment. The Python interpreter is easily extended and can add a new built-in function or modules written in C/C++/Java code.

2.2.9 Libraries :

Databases, web services, networking, numerical packages, graphical user interfaces, 3D graphics, others.

2.2.10 Supports :

Support from online Python community .[37]

2.3 Python module used :

♣ **RegEx Module :**

A RegEx, or Regular Expression is a sequence of characters that forms a search pattern . RegEx can be used to check if a string contains the specified search pattern .

➔Import the RegEx module:

Import re [41]

♣ **Numpy Module :**

NumPy is a Python library used for working with arrays . It also has functions for working in domain of linear algebra, fourier transform, and matrices.

➔Import the NumPy module:

import numpy **as** np [41]

♣ **Matplotlib Module :**

Matplotlib is a low level graph plotting library in python that serves as a visualization utility.

➔ Import the matplotlib module:

import matplotlib.pyplot **as** plt [41]

♣ **Keras Module :**

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible is key to doing good research .

➔ Import the Keras module :

from tensorflow **import** keras [45]

♣ **string Module :**

The string module contains a number of functions to process standard Python strings.

➔ Import the string module :

import string [51]

♣ **sklearn Module :**

Scikit-learn is a machine learning library for Python. It features several regression, classification and clustering algorithms including SVMs, gradient boosting, k-means, random forests and DBSCAN. It is designed to work with Python Numpy and SciPy.

➔ Import the sklearn module :

import sklearn [46]

♣ **NLTK :**

NLTK is the Natural Language Toolkit, a comprehensive Python library for natural language processing and text analytics. Originally designed for teaching, it has been adopted in the industry for research and development due to its usefulness and breadth of coverage.[52]

It contains text processing libraries for tokenization, parsing, classification, stemming, tagging and semantic reasoning. It also includes graphical demonstrations and sample data sets as well as accompanied by a cook book and a book which explains the principles behind the underlying language processing tasks that NLTK supports.[21]

➔ Import NLTK :

```
import nltk
```

2.4 Google Colab :

2.4.1 Google Colab definition :

Google Colaboratory widely known as Google Colab is an open source service provided by Google to any person having a Gmail account. Google Colab provides GPU for research to the people who do not have enough resources or cannot afford one. The Google Colab service provides 12.72 GB of RAM and 358.27 GB of hard disk space in one runtime. Every runtime lasts for 12 hours after which the runtime is reset and the user has to establish a connection again. This is to ensure that people do not use the GPU service for crypto currency mining and other illegal purposes. Different run cases are shown below.

Once the user opens a Google Colab file they need to select a runtime type. There are 3 available options for the same :

1. None (that will use the computer's CPU the user is using)
2. GPU

TPU (especially for tensor processing) .[67]

2.4.2 Benefits of Colab :

Besides being easy to use , the Colab is fairly flexible in its configuration and does much of the heavy lifting for you.

- Python 2.7 and Python 3.6 support .
- Free GPU acceleration .
- Pre-installed libraries: All major Python libraries like TensorFlow, Scikit-learn, Matplotlib among many others are pre-installed and ready to be imported .
- Built on top of Jupyter Notebook .
- Collaboration feature (works with a team just like Google Docs): Google Colab allows developers to use and share Jupyter notebook among each other without having to download, install, or run anything other than a browser .
- Supports bash commands .
- Google Colab notebooks are stored on the drive .[42]

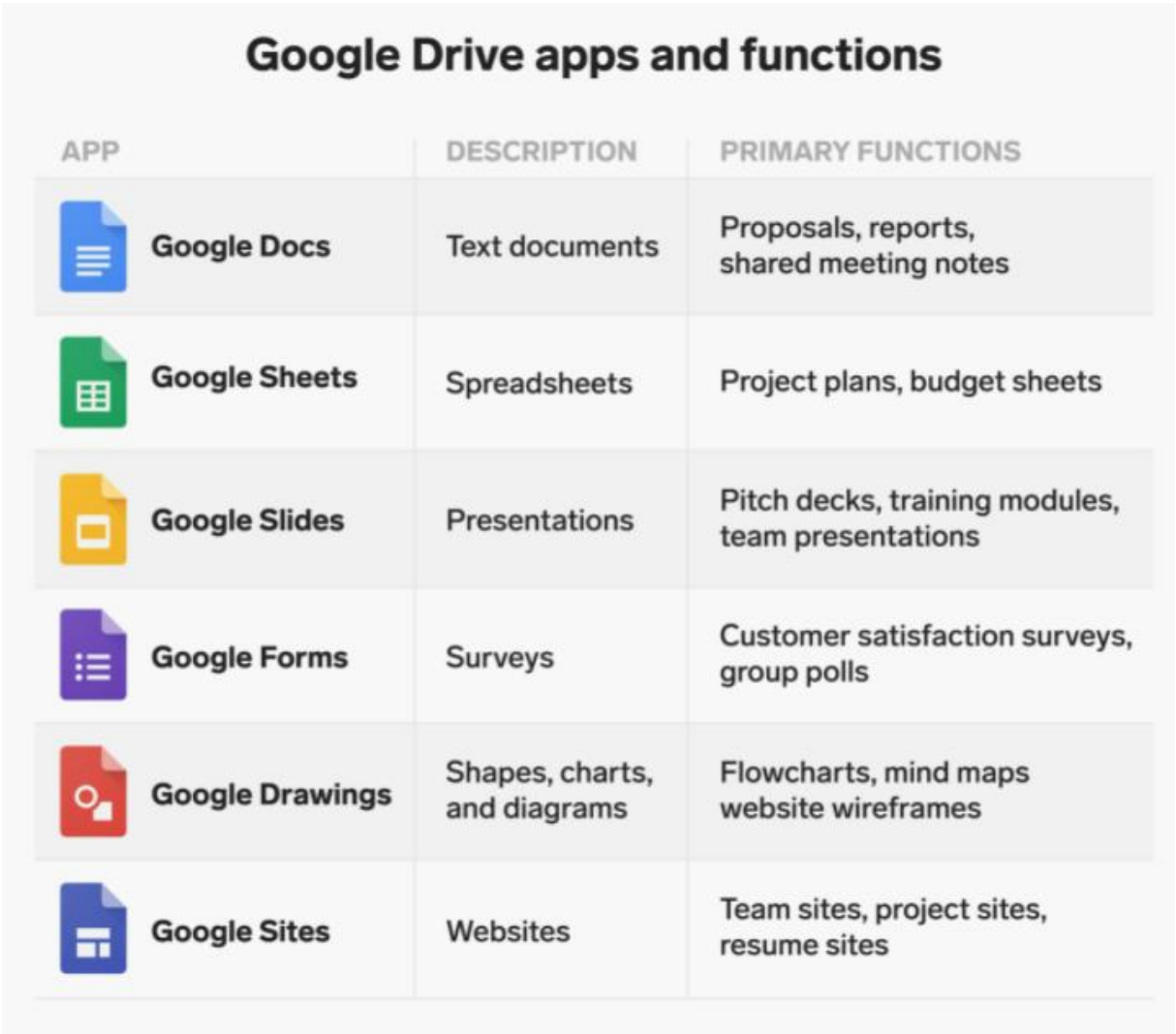
2.5 Google Drive :

Google Drive is a free cloud-based storage solution that enables users to :

- **Storing files :** By default, Google gives 15 GB of free storage space across Gmail, Google Drive and Google Photos. If you ever hit that limit, you can always pay for more space by upgrading to a Google One account.
- **Sharing files :** Files are private until you share them. But at any time, you can invite others to view, edit, or download files and documents.
- **Accessing files from anywhere :** Files can be accessed from any smartphone, tablet or computer. Offline access is also available.
- **Comprehensive search :** Drive can recognize images and even text in scanned documents, making searching for files even easier.
- **Secure storage :** Drive files are protected by an AES256 or AES128[3] encryption, the same security protocol used on other Google services.

[3] AES stands for Advanced Encryption Standard, which is the norm used worldwide to encrypt data.It is one of the most secure encryption methods.AES uses symmetric key encryption, which involves the use of only one secret key to cipher and decipher information.[44]
The main difference between 128 and 256-bit encryption algorithms is the length of the secret key that they use. The 128 and 256 in AES-128 and AES-256 means that the two algorithms use 128-bit and 256-bit keys respectively. [44]

Here is a breakdown of the most commonly used Google Drive applications :



Google Drive apps and functions

APP	DESCRIPTION	PRIMARY FUNCTIONS
Google Docs	Text documents	Proposals, reports, shared meeting notes
Google Sheets	Spreadsheets	Project plans, budget sheets
Google Slides	Presentations	Pitch decks, training modules, team presentations
Google Forms	Surveys	Customer satisfaction surveys, group polls
Google Drawings	Shapes, charts, and diagrams	Flowcharts, mind maps website wireframes
Google Sites	Websites	Team sites, project sites, resume sites

Figure 40: Google Drive applications and functions .[43]

Google Drive works with an integrated suite of apps powered by Google .[43]

3. Explanation of our approach :

3.1 Dataset :

We used the dataset ones mentioned in the chapter 3.

Arabic dataset [English + TAB + Arabic + TAB + Predicate] and Five datasets for other language pairs .

Each language has a number of pairs, uses 10,000 samples for each one

3.2 Data download :

```
Hi.     مرحبًا.  CC-BY 2.0 (France) Attribution: tatoeba.org #538123 (CM) & #629296 (Samer)
Run!    اركض!    CC-BY 2.0 (France) Attribution: tatoeba.org #906328 (papabear) & #1245450 (saeb)
Help!   النجدة!! CC-BY 2.0 (France) Attribution: tatoeba.org #435084 (lukaszpp) & #371293 (saeb)
Jump!   اقفز!   CC-BY 2.0 (France) Attribution: tatoeba.org #1102981 (jamessilver) & #6009426 (damascene)
Stop!   قف!     CC-BY 2.0 (France) Attribution: tatoeba.org #448320 (FeuDRenais) & #1245447 (saeb)
Go on.  داوم.    CC-BY 2.0 (France) Attribution: tatoeba.org #2230774 (CK) & #5118652 (damascene)
Go on.  استمر.   CC-BY 2.0 (France) Attribution: tatoeba.org #2230774 (CK) & #5118653 (damascene)
Hello!  مرحباً.  CC-BY 2.0 (France) Attribution: tatoeba.org #373330 (CK) & #1904219 (Asma)
Hurry!  تعجّل!   CC-BY 2.0 (France) Attribution: tatoeba.org #1182334 (Hideousss) & #6009436 (damascene)
Hurry!  استعجل!! CC-BY 2.0 (France) Attribution: tatoeba.org #1182334 (Hideousss) & #6009437 (damascene)
I see.  انا اري  CC-BY 2.0 (France) Attribution: tatoeba.org #2111796 (CK) & #6480319 (shoma79)
I won!  أنا فُزت!        CC-BY 2.0 (France) Attribution: tatoeba.org #2005192 (CK) & #2005329 (Asma)
Relax.  استرح.   CC-BY 2.0 (France) Attribution: tatoeba.org #1722133 (shanghainese) & #371606 (saeb)
Smile.  ابتسم.   CC-BY 2.0 (France) Attribution: tatoeba.org #2764108 (CK) & #6731986 (Safa_Alfulaij)
Cheers! في صحتك.         CC-BY 2.0 (France) Attribution: tatoeba.org #487006 (human600) & #4113375 (fadi)
Got it? هل فهمت؟        CC-BY 2.0 (France) Attribution: tatoeba.org #455353 (FeuDRenais) & #458469 (saeb)
He ran. ركض.     CC-BY 2.0 (France) Attribution: tatoeba.org #672229 (CK) & #1245424 (saeb)
I know. أعرف.    CC-BY 2.0 (France) Attribution: tatoeba.org #319990 (CK) & #372158 (saeb)
I know. أعلم ذلك.        CC-BY 2.0 (France) Attribution: tatoeba.org #319990 (CK) & #372159 (saeb)
I know. أنا أعلم         CC-BY 2.0 (France) Attribution: tatoeba.org #319990 (CK) & #6679425 (marhabaan)
I'm 19. أنا في 19        CC-BY 2.0 (France) Attribution: tatoeba.org #442400 (sacredceltic) & #6670033
(marhabaan)
I'm OK. أنا بخير.        CC-BY 2.0 (France) Attribution: tatoeba.org #433763 (CK) & #376401 (saeb)
Listen. استمع    CC-BY 2.0 (France) Attribution: tatoeba.org #1913088 (CK) & #6679426 (marhabaan)
No way! غير معقول!       CC-BY 2.0 (France) Attribution: tatoeba.org #2175 (CS) & #371702 (saeb)
Really? حقاً؟    CC-BY 2.0 (France) Attribution: tatoeba.org #373216 (kotobaboke) & #487784 (saeb)
Thanks. شكرا.    CC-BY 2.0 (France) Attribution: tatoeba.org #2057650 (nava) & #7238439 (OsoHombre)
Why me? لماذا أنا؟       CC-BY 2.0 (France) Attribution: tatoeba.org #24958 (CK) & #372548 (saeb)
```

Figure 41: Dataset example from the Dataset .

Assuming that the data is in the google drive , we use the following for to load it and divide it into lines .

```
from google.colab import drive

drive.mount('/content/gdrive')


with open("/content/gdrive/MyDrive/data/ara.txt", encoding="utf8") as f
:

  en2ar = f.readlines()
```

Figure 42: Load and divide dataset .

- ✓ readlines() : create a list of lines by splitting the text file at every occurance of '\n'

3.3 Dataset preprocessing :

Before we start building the models, we need clean the data by removing capital letters and punctuation and preparation.

3.3.1 Normalisation :

As we mentioned in chapter 3, we will by calling the lower() function on each word for convert all words to lowercase .

3.3.2 Dataset splitting :

- **Split the data into Train/Test :**

The clean data contains a little over 150,000 phrase pairs and some of the pairs toward the end of the file are very long.

This is a good number of examples for developing a small translation model. The complexity of the model increases with the number of examples, length of phrases, and size of the vocabulary.

We will simplify the problem by reducing the dataset to the first 10,000 examples in the file; these will be the shortest phrases in the dataset.

Further, we will then stake the first 9,000 of those as examples for training and the remaining 1,000 examples to test the fit model.

```
X_train, X_test, y_train, y_test = train_test_split(input_texts, target
_texts, test_size = 0.1,random_state=42)
```

Figure 43: Split dataset to Train / Test .

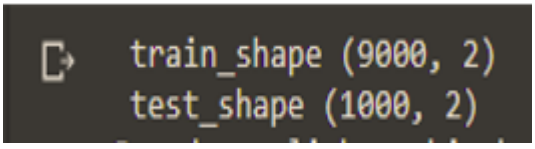


Figure 44: Shape Train / Test .

- **Split the data into Input/Target :**

Now , we split each line into input text and target text(we ignore the attribution field)

In our pre-processing we use the function preprocess sentence() for remove the arabic diacritics and numbers.

All of what we have mentioned is manifested in :

```
arabic_diacritics = re.compile("""   ّ|   ْ|   ً|   ٌ|   ٍ|   ِ|   ُ|
                        ٰ |  _|\u200d |\u200f """, re.VERBOSE)

def remove_diacritics(text):
    text = re.sub(arabic_diacritics, '', text)
    return text

table = str.maketrans('', '', string.punctuation)

def preprocess_sentence(w):
    w = remove_diacritics(w)
    w = w.translate(table)

    w = re.sub("[٩٨٧٦٥٤٣٢١٠0123456789]",r"", w)
    return w
```

Figure 45: Filtering dataset .

```
input_text = preprocess_sentence(l.split('\t')[0])
target_text = preprocess_sentence(l.split('\t')[1])
```

Figure 46: Split dataset to input and target sequence .

After preprocessing, add the START_ and _END to the Arabic sentence , this will later help our model determine when to start and end sequences(we need this we don't know a-priori how long the output sequences should be).

```
target_text = "STAR " + target_text + " END"
```

Figure 47: Determine the beginning and end of target sequence .

And create the vocabulary of all the unique English and Arabic words and compute the vocabulary sizes and the length of the maximum sentence of both the English and Arabic language.

We will represent this part in the following results:



Figure 48: Example of Arabic and English vocabulary .

3.3.3 Digital data :

These input texts still don't work as input to our model. We'll need to convert the words into numeric values. In our case, one-hot encodings are fine

We'll need input data for both the encoder and the decoder of the model, as well as the target data (used only in the decoder part).

first we compute the vocabulary for both English and Arabic. We also compute the vocabulary sizes and the length of maximum sequence for both the languages.

	English	Arabic
Number of samples	10000	10000
Vocabulary size	4361	8454
The length of maximum sequence	8	12

Table 5: Information in both English and Arabic sequence .

Finally we create 4 Python dictionaries (two for each language) to convert a given token into an integer index and vice-versa

```
# Create word to token dictionary for both source and target
input_token_index = [(char, i) for i, char in enumerate(all_arabic_words)]
target_token_index =[(char, i) for i, char in enumerate(all_arabic_words)]
# Create token to word dictionary for both source and target
reverse_input_char_index = dict((i, word) for word, i in input_token_index.
items())
reverse_target_char_index = dict((i, word) for word, i in target_token_inde
                                 x.items())
```

Figure 49: Creation of dictionaries .

3.3.4 generate the data : (one hot encoding)

The encoder_input_data will consist of samples of the maximum sequence length (4361) filled with the respective one-hot-encoded tokens (in this case a vector of length: 8).

The decoder_input_data and the decoder_target_data are both constructed in the same way as the input data for the encoder(with maximum sequence length:8454 and a vector of length:12)

We need to construct those two sequences because we're training our model through a process called teacher forcing, where the decoder learns to generate decoder_target_data[t+1...] given decoder_input_data[...t] while taking into account the input sequence via the encoder's internal state. Therefore we have to offset decoder_target_data by one timestep.

We need to construct those two sequences because we're training our model through a process called teacher forcing, where the decoder learns to generate decoder_target_data[t+1...] given decoder_input_data[...t] while taking into account the input sequence via the encoder's internal state. Therefore we have to offset decoder_target_data by one timestep.

then we iterate over all input and target texts and insert the respective one-hot encoding each word in the sequence.

Now, during inference (testing), we do not know the correct (expected) target data before hand

So we define the decoder_input_data as follows:

it begins with a special symbol start , it will continue with an input created by the decoder at the previous time step in other words, the decoder's output at time step t will be used decoder's input at time step t+1, this is the Teacher Forcing.

3.4 Model Development :

Next step we will define the model with encoder-decoder architecture, we have split the model into two parts, first, we have an encoder that inputs the English sentence and produces a hidden vector. The encoder is built with using Long Short-Term Memory (LSTM) layer.

Then the output of the encoder will be used as input for the decoder. For the decoder, we will be using LSTM layer again, as well as a dense layer that predicts the Arabic .

3.4.1 Encoder :

The first layer to be defined is the embedding layer. For this, we first must add an Input Layer, the only parameter to consider here is ‘shape’, which is the maximum length of the English sentences, in our case 8 .

```
encoder_inputs = Input(shape=(max_encoder_seq_length,))
```

Figure 50: Creation Input Encoder layer .

We will then connect it to the embedding layer, here the parameters to take into account as we mentioned before are ‘input_dim’ which is the length of the English vocabulary In this case 4361 and ‘output_dim’ which is the shape of the embedding vector 64. This layer will convert any of the English words into a vector of the shape of the output dimension.

```
enc_emb =  Embedding(num_encoder_tokens, latent_dim, input_length=max_e
ncoder_seq_length)(encoder_inputs)
```

Figure 51: Creation encoder embedding layer .

The concept behind this is to extract the meaning of the word in a form of a spatial representation where each dimension will be a characteristic defining the word. For example, the word ‘hello’ will be converted into a vector of shape 64. The higher the output dimension the more semantic meaning you can extract from each word, but also the higher the calculations required and the processing time. Finding a balance between speed and performance is required.

Next we will add the LSTM layer of size 64. Even though each time step of the LSTM outputs a hidden vector, we will focus our attention on the last one, therefore the parameter return_sequences is ‘False’, With the return_state = True .

```
encoder_lstm = LSTM(latent_dim, return_state=True,return_sequences=Fals
e)
```

Figure 52: Creation encoder LSTM layer .

```
encoder_outputs, state_h, state_c = encoder_lstm(enc_emb)
encoder_states = [state_h, state_c]
```

Figure 53: Creation encoder states .

The output of the encoder layer will be the hidden state of the last time step. We will then need to feed this vector into the decoder.
(state_h denotes a hidden state and state_c denotes cell state) .

3.4.2 Decoder :

Once we have the input ready, we will continue with the decoder. This is also built with a LSTM layer, the difference is the parameter return_sequences, which in this case is 'True', because In the encoder part we were expecting only one vector in the last time step and neglecting all the others, here we are expecting an output vector at every time step so the Dense layer can make a prediction.

```
decoder_inputs = Input(shape=(max_decoder_seq_length,))
dec_emb =  Embedding(num_decoder_tokens, latent_dim)(decoder_inputs)
decoder_lstm = LSTM(latent_dim, return_sequences=True, return_state=Tru
e)
decoder_outputs, _, _ = decoder_lstm(dec_emb, initial_state=encoder_sta
tes)

decoder_dense = Dense(num_decoder_tokens, activation='softmax')
decoder_outputs = decoder_dense(decoder_outputs)
```

Figure 54: Creation decoder layers .

We have one last step, to predict the translated word. For this we need to use a Dense Layer.

The parameter we need to define is the number of units, this number of units is the shape of the output vector and it needs to be the same as the length of the Arabic vocabulary (8454). The vector will be all values close to zero, except one of the units that will be close to 1. We then need to map the index of the unit that outputs a 1 with a dictionary where we map each

unit to a word. For example, if the input is the word ‘hello’ and the output is a vector where all are zeros and then the unit 472 is 1, we map this index against the dictionary containing the Arabic words and we get the value ‘مرحبا’.

A softmax activation function is used on the output layer to ensure the output has the properties of a probability distribution.

3.5 Create the model :

Now we glue both parts together using Keras' Model functional API. Specify the inputs needed to produce the outputs and the resulting model will automatically include all layers necessary to compute the outputs.

```
model = Model(inputs=[encoder_inputs, decoder_inputs], outputs=decoder_
outputs)
```

Figure 55: Creation of model .

Lastly, we compile the model defining the optimizer (the optimization algorithme is rmsprop) and loss(the function loss is categorical_crossentropy) .

Next, a summary of the defined model is printed, allowing us to confirm the model configuration .

```
model.compile(optimizer='rmsprop', loss='categorical_crossentropy' , me
trics=['accuracy'])
model.summary()
```

Figure 56: Compile and summary the model .

```
Model: "model"
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
=================================================================================================
input_1 (InputLayer)         [(None, 8)]               0
_________________________________________________________________________________________________
input_2 (InputLayer)         [(None, 12)]              0
_________________________________________________________________________________________________
embedding (Embedding)        (None, 8, 64)             279104      input_1[0][0]
_________________________________________________________________________________________________
embedding_1 (Embedding)      (None, 12, 64)            541056      input_2[0][0]
_________________________________________________________________________________________________
lstm (LSTM)                  [(None, 8, 64), (Non      33024       embedding[0][0]
_________________________________________________________________________________________________
lstm_1 (LSTM)                [(None, 12, 64), (No      33024       embedding_1[0][0]
                                                                   lstm[0][1]
                                                                   lstm[0][2]
_________________________________________________________________________________________________
dense (Dense)                (None, 12, 8454)          549510      lstm_1[0][0]
=================================================================================================
Total params: 1,435,718
Trainable params: 1,435,718
Non-trainable params: 0
_________________________________________________________________________________________________
```

Figure 57: Show Model Summary .

A plot of the model is also created providing another perspective on the model configuration**.**

```
plot_model(model, to_file='model.png', show_shapes=True)
```

Figure 58: Graph Drawing of model .

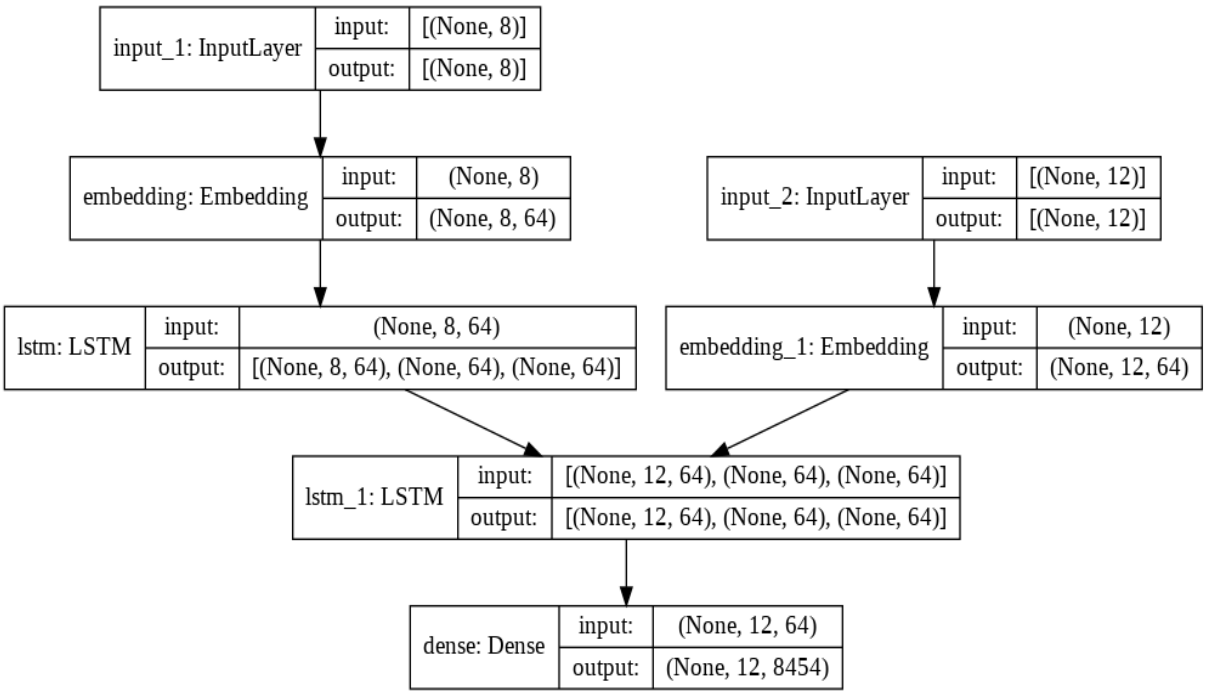


Figure 59: Plot of Model Graph for NMT .

Next, the model is trained.

3.6 training the model :

Once we define the model, we just need to train it, and check the accuracy .

Type of neural network used	**RNN**
Number of hidden layer in encoder/decoder	4
Word embedding size	64
Number of traing epochs	100
Batch size	64
Optimization methode	Rmsprop
Function loss	categorical_crossentropy
processing unit type	Graphical Processing Unit
time of execution	8m 50s
Percentage of validation split	10%
Activation function	Softmax

Table 6 : Training parameters .

```
history = model.fit([encoder_input_data, decoder_input_data], decoder_t
arget_data,batch_size=batch_size,epochs=epochs,validation_split=0.2)
```

Figure 60: Fit model .

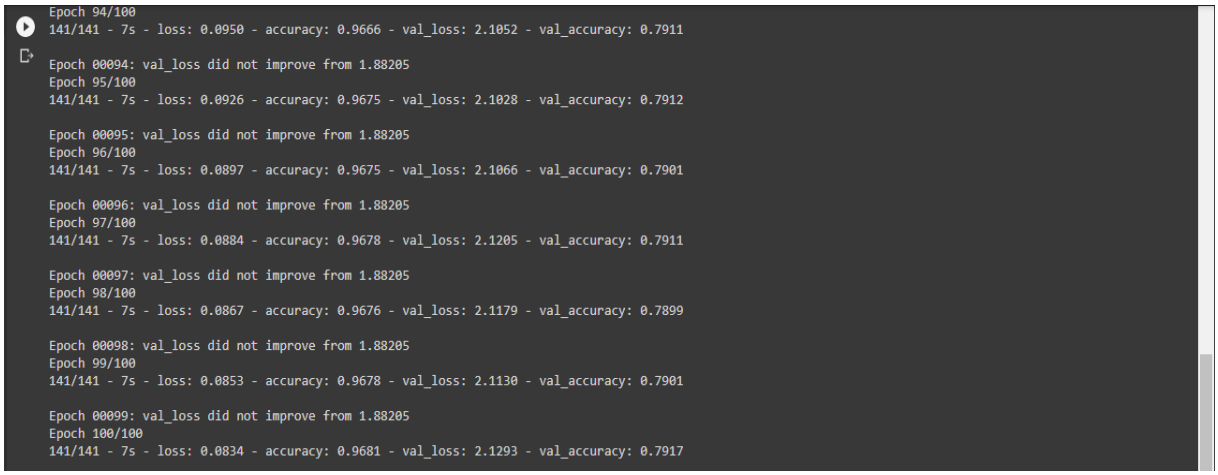


Figure 61: Result of fit model .

We will evaluate the model on the train and the test dataset.

The model should perform very well on the train dataset and ideally have been generalized to perform well on the test dataset.

- **Testing :**

Next, create a predict function named decode_sequence. After generating the empty sequence of length 1, the model should know when to start and stop reading the text. To read the model will check out for "STAR " in this case. Keep two conditions, either when the max length of sentence is hit or find stop character "END ". Keep on updating the target sequence by one and update the states .

- **Evaluation**

Evaluation of machine translation output is extremely important as it is the key factor that determines the quality of the translation, required level of post-editing, etc. Generally, MT outputs are evaluated both automatically and manually. [68]

We calculate the cumulative scores by corpus_bleu() function for BLEU-1, BLEU-2, BLEU-3 and BLEU-4

where BLEU-1 use the same weights (1, 0, 0, 0)

and the BLEU-2 : weights assign a 50% to each of 1-gram and 2-gram (0.5, 0.5, 0, 0)

The weights for the BLEU-3 are 33% for each of the 1, 2 and 3-gram scores.(0.33, 0.33, 0.33,0)

As for The weights for the BLEU-4 are 1/4 (25%) for each of the 1-gram, 2-gram, 3-gram and 4-gram scores.(0.25, 0.25, 0.25 ,0.25) .

3.7 Testing and Result :

We experimented our model on English and Arabic in both directions, and on five other language pairs

All the language pairs were trained with the same architecture and different corpus, and these systems were evaluated using the automatic evaluation method, BLEU (bilingual evaluation understudy) score . [27]

We compared the BLEU results for each language pair, the best one we saved and applied to the translation from English to Arabic

Also we have also translated new sentences in English into Arabic, because our Theme focuses on this direction.

Languge pair	English-Arabic		English-Frensh		English-German		English-Italian		English-Turkish		English-Spanish	
Language	Eng	Ara	Eng	Fre	Eng	Ger	Eng	Itl	Eng	Tur	Eng	Spa
Vocabulary size	4361	8454	2017	4301	2402	3856	1413	3649	2450	5558	2296	4537

Table 7 : Vocabulary in each language in five language pairs .

3.7.1 Test 1 : machine translation English to Arabic .

For example, 10 lines of the file .

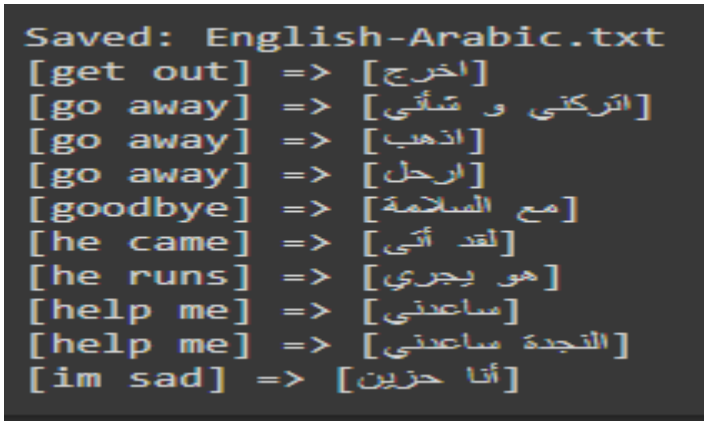


Figure 62: Example of dataset 'ara.txt' .

- ✓ **Presentation of results :**

  Next, we can repeat this for each source phrase in a dataset "ara.txt" and compare the predicted result to the expected target phrase in Arabic.

- o **Train :**

We print some of these comparisons to screen to get an idea of how the model performs in practice .

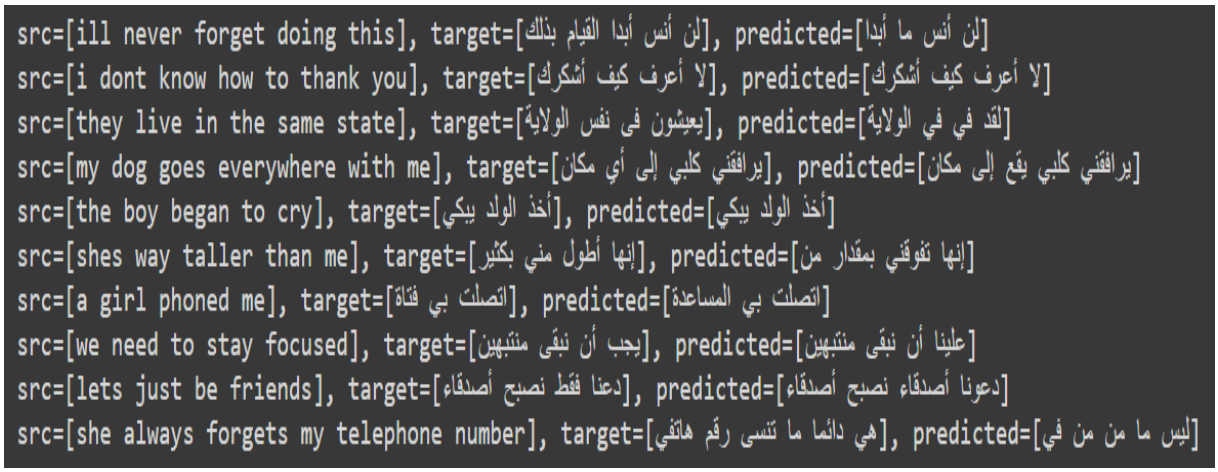


Figure 63: Show example of train Neural translation english-arabic .

Looking at the results for the test dataset first, we can see that the translations are readable and mostly correct .

- **For example :** "I don't know how to thank you" was correctly translated to "لا أعرف كيف أشكرك".

We can also see that the translations were not perfect, with "my dog goes 102nglish102re with me" translated to "يرافقني كلبي يقع إلى مكان" optimizer instead of the expected "يرافقني كلبي إلى أي مكان".

We will also calculate the BLEU scores to get a quantitative idea of how well the model has performed.

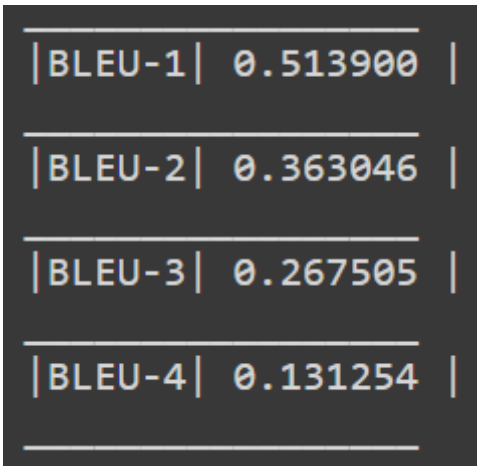


Figure 64: Show result of BLEU train .

A BLEU-4 score of about 0.1 31: The general idea is difficult to understand.

- **Test :**

Looking at the results on the test set, do see readable translations, which is not an easy task.

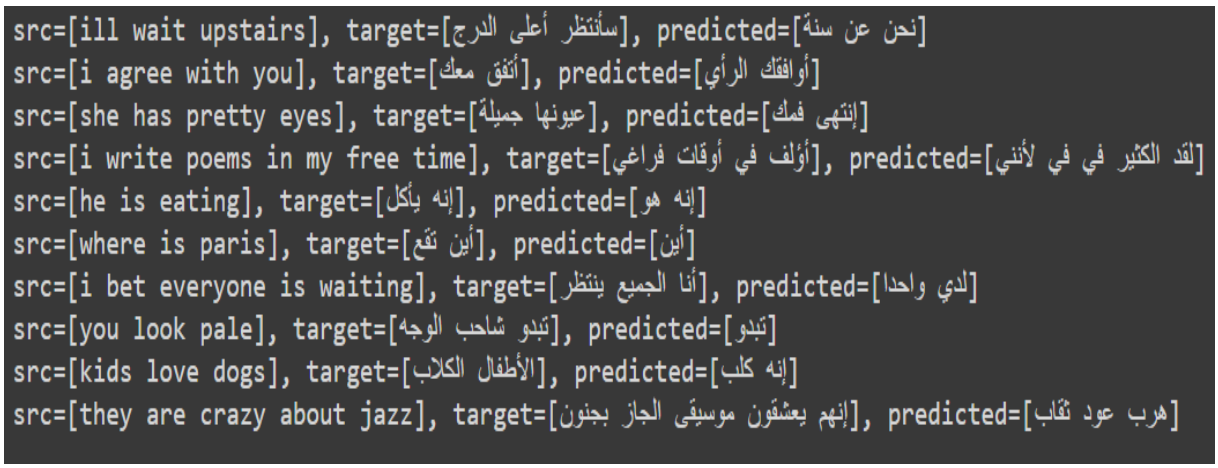


Figure 65: Show example of test Neural translation english-arabic .

- **For example :** we see "I agree with you" roughly correctly translated to "أتفق معك" or "أوافقك الرأي" .

We also see poor translations that the model, so it needs more tuning, such as "i write poems in my free time " translated as " لقد الكثير في في لأنني" instead of the expected " أؤلف في أوقات فراغي".

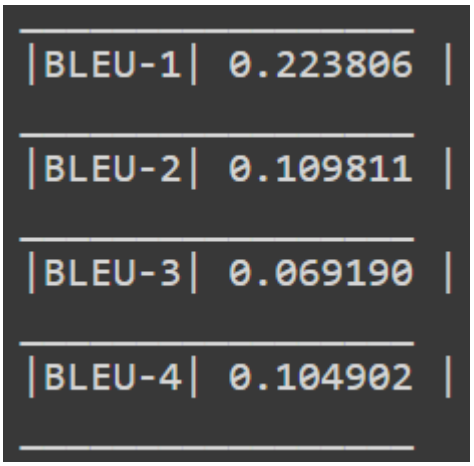


Figure 66: Show result of BLEU test .

A BLEU-4 score of about 0.1 04: The general idea is difficult to understand.

3.7.2 Test 2: machine translation Arabic to English .

For example, 10 lines of the file .

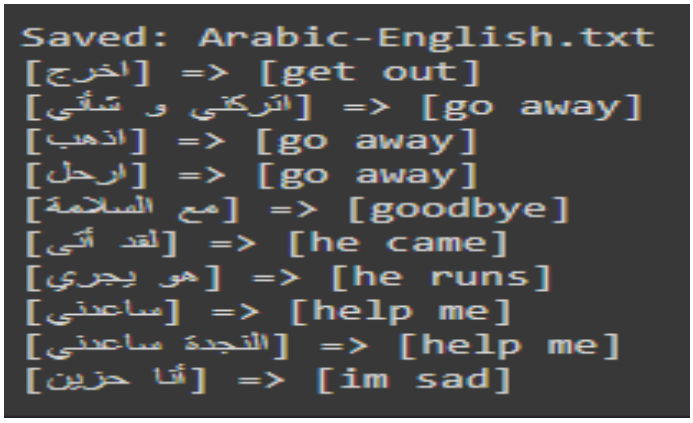


Figure 67: Example of dataset 'ara.txt' .

✓ **Presentation of results :**

Next, we can repeat this for each source phrase in a dataset "ara.txt" and compare the predicted result to the expected target phrase in english.

- **Train :**

We print some of these comparisons to screen to get an idea of how the model performs in practice.

```
src=[هل أنت], target=[are you japanese], predicted=[i is]
src=[أنا لم أفهم], target=[i do not understand], predicted=[i is to]
src=[يا لها من فتاة جميلة], target=[what a cute little girl], predicted=[i is to]
src=[لقد سمعت الكثير عنك], target=[ive heard quite a lot about you], predicted=[i you you to to]
src=[غنى أغنية], target=[he sang a song], predicted=[i is]
src=[مرحبا ادخل], target=[hi come on in], predicted=[i is]
src=[هذا خطير], target=[this is a dangerous thing], predicted=[i is]
src=[من أخذ], target=[who took my keys], predicted=[i is you]
src=[يشرب توم القهوة يوميا], target=[tom drinks coffee every day], predicted=[i is to]
src=[هو سعيد], target=[he is happy], predicted=[i is]
```

Figure 68: Show example of train Neural translation arabic-english .

We see very poor translations that the model, so it needs more tuning, such as " هذا خطير " translated as "i is " instead of the expected "this is a dangerous thing".

We will calculate the BLEU scores to get a quantitative idea of how well the model has performed.

```
|BLEU-1| 0.097170 |
|BLEU-2| 0.010076 |
|BLEU-3| 0.052450 |
|BLEU-4| 0.079224 |
```

Figure 69: Show result of BLEU train .

A BLEU-4 score of about 0.079 : Almost useless translations.

- **Test :**

Looking at the results on the test set, do see readable translations, which is not an easy task.

```
src=[أنا فقط سأترك هذه هنا], target=[ill just leave this here], predicted=[i you to to]
src=[دخل توم الشقة], target=[tom went inside the apartment], predicted=[i is to to]
src=[لا أفهم], target=[i dont understand], predicted=[i is]
src=[هي ولدت محظوظة], target=[she was born lucky], predicted=[i you]
src=[غادروا جميعا], target=[they all left], predicted=[i is]
src=[أتحس بوجع في], target=[do you feel any pain in your stomach], predicted=[i is you]
src=[الطيور على أشكالها تقع], target=[birds of a feather flock together], predicted=[i is to]
src=[أوقفه], target=[stop it], predicted=[i you]
src=[أنا تأخرت], target=[i am already late], predicted=[i is]
src=[تلقى توم كما لا بأس به من الردود الجيدة], target=[tom got quite a few positive responses], predicted=[i you you to to]
```

Figure 70: Show example of test Neural translation arabic-english .

We also see very poor translations that the model, so it needs more tuning, such as " هي ولـدت مـحظوظة" translated as "i you " instead of the expected "she was born lucky".

```
|BLEU-1| 0.094622 |
|BLEU-2| 0.008545 |
|BLEU-3| 0.046500 |
|BLEU-4| 0.071021 |
```

Figure 71: Show result of BLEU test .

A BLEU-4 score of about 0.071 : Almost useless translations.

3.7.3 Test 3 : Machine translation English to French .

For example, 10 lines of the file .

```
Saved: English-Frensh.txt
[wait] => [attends]
[wait] => [attendez]
[wait] => [attends]
[wait] => [attendez]
[begin] => [commencez]
[begin] => [commence]
[go on] => [poursuis]
[go on] => [continuez]
[go on] => [poursuivez]
[hello] => [bonjour]
```

Figure 72: Example of dataset 'fra.txt' .

✓ **Presentation of results**

we also repeat this for each source phrase in a dataset "fra.txt" and compare the predicted result to the expected target phrase in French .

- **Train :**

We print some of these comparisons to screen to get an idea of how the model performs in practice.

```
src=[i have to run], target=[je dois courir], predicted=[il dois faut courir]
src=[get out], target=[sors], predicted=[sortez]
src=[its unlikely], target=[cest improbable], predicted=[cest improbable]
src=[ill be nice], target=[je serai gentil], predicted=[je serai gentille]
src=[stay still], target=[navancez pas], predicted=[restez bouge pas]
src=[lets drink], target=[buvons], predicted=[buvons]
src=[theyre drunk], target=[elles sont saoules], predicted=[elles sont saoules]
src=[i trust you], target=[jai confiance en toi], predicted=[je confiance fie toi toi]
src=[toms deaf], target=[tom est sourd], predicted=[tom est sourd]
src=[listen to tom], target=[ecoute tom], predicted=[ecoute tom]
```

Figure 73: Show example of train Neural translation English-French .

Looking at the results for the test dataset first, we can see that the translations are readable and mostly correct.

- **For example:** "`theyre drunk`" was correctly translated to "`elles sont saoules`".

We can also see that the translations were not perfect, with "`I trust you`" translated to "`je confiancefie toi toi`" instead of the expected "`jai confiance en toi`".

We will also calculate the BLEU scores to get a quantitative idea of how well the model has performed.

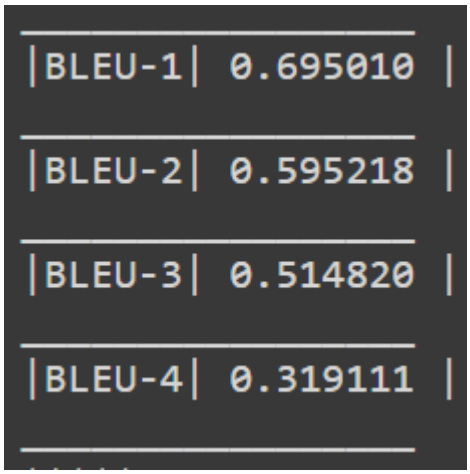


Figure 74: Show result of BLEU train .

A BLEU-4 score of about 0.31911 : Comprehensible results with correct translations.

- **Test :**

Looking at the results on the test set, do see readable translations, which is not an easy task.

```
src=[im amused], target=[ca mamuse], predicted=[je suis amusee]
src=[be fair], target=[sois sincere], predicted=[soyez juste]
src=[i feel weak], target=[je me sens faible], predicted=[je les sens mort]
src=[i must go], target=[il me faut men aller], predicted=[il dois faut aller]
src=[cheers], target=[merci], predicted=[tchintchin]
src=[i must insist], target=[je dois insister], predicted=[il me faut insister]
src=[make it quick], target=[faites vite], predicted=[revenez]
src=[are you busy], target=[etesvous occupee], predicted=[estu occupe]
src=[hes loaded], target=[il est plein aux as], predicted=[il est pete]
src=[look up there], target=[regarde lahaut], predicted=[regardez lahaut]
```

Figure 75: Show example of test Neural translation English-French .

- **For example :** we see "`look up there`" roughly correctly translated to "`regarde lahaute`".

We also see poor translations that the model, so it needs more tuning, such as "`i feel weak `" translated as "`je les sens mort`" instead of the expected "`je me sens faible`".

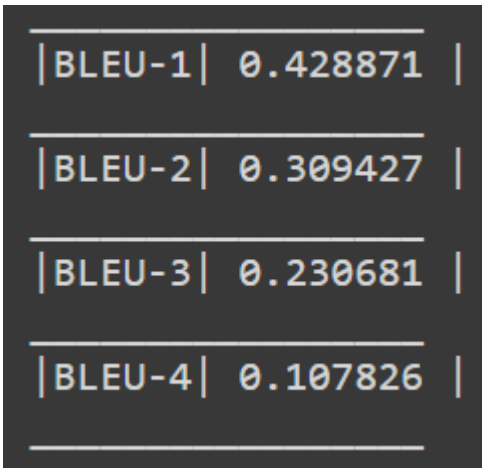


Figure 76: Show result of BLEU test .

A BLEU-4 score of about 0.107 : The general idea is difficult to understand.

3.7.4 Test 4: machine translation English to German .

For example, 10 lines of the file

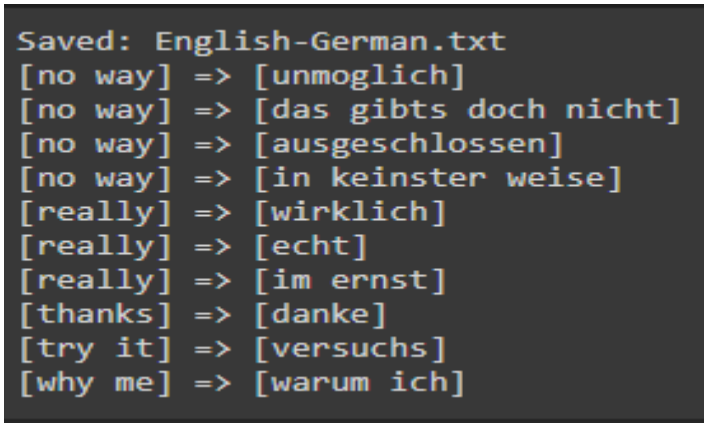


Figure 77: Example of dataset ‘deu.txt’ .

- ✓ **Presentation of results :**

we also repeat this for each source phrase in a dataset "deu.txt" and compare the predicted result to the expected target phrase in German.

- o **Train :**

We print some of these comparisons to screen to get an idea of how the model performs in practice.

```
src=[we want a car], target=[wir wollen ein auto], predicted=[wir wollen ein auto]
src=[did tom hurt you], target=[hat tom dir weh getan], predicted=[hat tom euch weh getan]
src=[happy new year], target=[gluckliches neues jahr], predicted=[frohes neues jahr]
src=[do you mean us], target=[meinst du uns], predicted=[hast du uns]
src=[stir the soup], target=[ruhr die suppe um], predicted=[ruhr die suppe um]
src=[he was naive], target=[er war zu treuherzig], predicted=[er war naiv]
src=[how is everyone], target=[wie geht es euch allen], predicted=[wie geht es dir]
src=[lets try], target=[lass es uns versuchen], predicted=[lass es uns]
src=[tom found one], target=[tom hat eine gefunden], predicted=[tom hat einen gefunden]
src=[i am in trouble], target=[ich sitze in der patsche], predicted=[ich bin in schwierigkeiten]
```

Figure 78: Show example of train Neural translation English-German .

Looking at the results for the test dataset first, we can see that the translations are readable and mostly correct.

- **For example:** "`we want a car`" was correctly translated to "`wir wollen ein auto`".

We can also see that the translations were not perfect, with "`how is everyone`" translated to "`wie geht es dir`" instead of the expected "`wie geht es euch allen `".

We will also calculate the BLEU scores to get a quantitative idea of how well the model has performed.

```
|BLEU-1| 0.755149 |
|BLEU-2| 0.668917 |
|BLEU-3| 0.610565 |
|BLEU-4| 0.431895 |
```

Figure 79: Show result of BLEU train .

A BLEU-4 score of about 0.431 : High quality translations.

- **Test :**

Looking at the results on the test set, do see readable translations, which is not an easy task.

```
src=[i got it], target=[ich habe es bekommen], predicted=[ich habs es]
src=[whats this for], target=[wozu ist das], predicted=[was fur ist das]
src=[its just a cat], target=[es ist nur eine katze], predicted=[es ist ein zu]
src=[who panicked], target=[wer ist in panik ausgebrochen], predicted=[wer brach in panik]
src=[return fire], target=[schiet zuruck], predicted=[schieen sie zuruck]
src=[no kidding], target=[echt], predicted=[wirklich]
src=[it wont hurt], target=[es wird nicht wehtun], predicted=[es wird nicht schaden]
src=[was that a yes], target=[war das ein ja], predicted=[war es ein geheimnis]
src=[its not so far], target=[es ist nicht so weit], predicted=[es ist kein nicht so]
src=[she might come], target=[vielleicht kommt sie], predicted=[sie wird vielleicht kommen]
```

Figure 80: Show example of test Neural translation English-German .

- **For example :** we see poor translations that the model, so it needs more tuning, such as " `i go it` " translated as "`ich habs es`" instead of the expected "`ich habe es bekommen`".

BLEU-1	0.507584
BLEU-2	0.379529
BLEU-3	0.300214
BLEU-4	0.155803

Figure 81: Show result of BLEU test .

A BLEU-4 score of about 0.155 : The general idea is difficult to understand.

3.7.5 Test 5: machine translation English to Italian .

For example,10 lines of the file .

```
Saved: English-Italian.txt
[wait] => [aspetti]
[wait] => [aspettate]
[do it] => [fallo]
[do it] => [falla]
[do it] => [lo faccia]
[do it] => [la faccia]
[do it] => [fatelo]
[do it] => [fatela]
[go on] => [vai avanti]
[go on] => [continua]
```

Figure 82: Example of dataset ‘ita.txt’ .

- ✓ **Presentation of results :**

we also repeat this for each source phrase in a dataset "ita.txt" and compare the predicted result to the expected target phrase in Italian .

- o **Train :**

We print some of these comparisons to screen to get an idea of how the model performs in practice.

```
src=[i like yoga], target=[a me piace lo yoga], predicted=[a me piace la]
src=[hold this], target=[tieni questo], predicted=[tenete questo]
src=[we ate], target=[noi mangiammo], predicted=[noi abbiamo]
src=[im sleepy], target=[io sono stanco], predicted=[io sono]
src=[we tried], target=[provavamo], predicted=[provammo]
src=[keep walking], target=[continuate a camminare], predicted=[continui a camminare]
src=[are you fat], target=[tu sei grasso], predicted=[e grassa]
src=[thanks], target=[grazie], predicted=[grazie]
src=[drop it], target=[la molli], predicted=[mollatela]
src=[it was fun], target=[era divertente], predicted=[e divertente]
```

Figure 83: Show example of train Neural translation English-Italian .

Looking at the results for the test dataset first, we can see that the translations are readable and mostly correct.

- **For example:** "thanks" was correctly translated to "grazie".

We can also see that the translations were not perfect, with "are you fat" translated to "e grassa" instead of the expected "tu sei grasso".

We will also calculate the BLEU scores to get a quantitative idea of how well the model has performed.

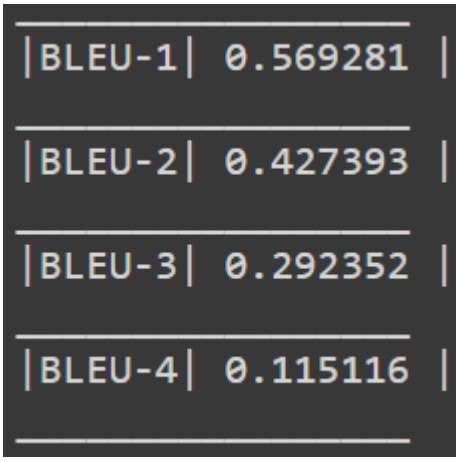


Figure 84: Show result of BLEU train .

A BLEU-4 score of about 0.115 : The general idea is difficult to understand.

- **Test :**

Looking at the results on the test set, do see readable translations, which is not an easy task.

```
src=[i fell], target=[caddi], predicted=[io sono]
src=[im calm now], target=[io sono calmo ora], predicted=[sono sono adesso]
src=[im a doctor], target=[io sono una dottoressa], predicted=[sono un un]
src=[i love fish], target=[io amo il pesce], predicted=[amo il il]
src=[im managing], target=[io ce la sto facendo], predicted=[io sto sto]
src=[im a vegan], target=[sono vegano], predicted=[sono sono]
src=[i need a job], target=[ho bisogno di un lavoro], predicted=[ho ho un un]
src=[be specific], target=[sii specifica], predicted=[sii precisa]
src=[i washed it], target=[la lavai], predicted=[io lavato]
src=[hes a prude], target=[lui e un puritano], predicted=[e un un]
```

Figure 85: Show example of test Neural translation English-Italian .

- **For example :** we see poor translations that the model, so it needs more tuning, such as " `i washed it` " translated as "`io lavato`" instead of the expected "`la lavai`".

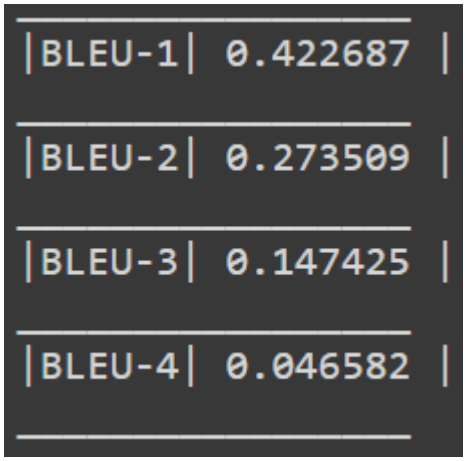


Figure 86: Show result of BLEU test .

A BLEU-4 score of about 0.046 : Almost useless translations.

3.7.6 Test 6: machine translation English to Spanish .

For example,10 lines of the file .

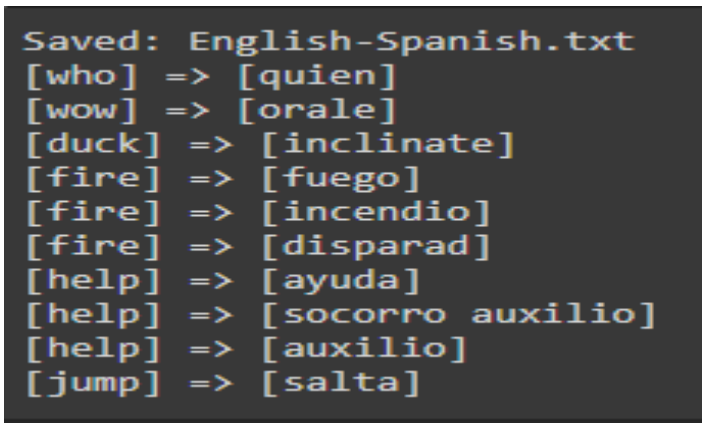


Figure 87: Example of dataset 'spa.txt' .

- ✓ **Presentation of results :**

we also repeat this for each source phrase in a dataset "spa.txt" and compare the predicted result to the expected target phrase in Spanish.

- **Train :**

We print some of these comparisons to screen to get an idea of how the model performs in practice.

```
src=[im attentive], target=[yo soy atento], predicted=[yo soy atento]
src=[start singing], target=[empiecen a cantar], predicted=[empieza a cantar]
src=[i love rock], target=[me gusta el rock], predicted=[me gusta el rock]
src=[is that legal], target=[eso es legal], predicted=[eso es legal]
src=[youre first], target=[tu eres el primero], predicted=[tu toca primero]
src=[i want eggs], target=[quiero huevos], predicted=[quiero huevos]
src=[i get you], target=[ya te pillo], predicted=[te te]
src=[i saw one], target=[vi uno], predicted=[yo encontre uno]
src=[im toms mom], target=[soy la madre de tom], predicted=[soy el el de tom]
src=[i never worry], target=[yo nunca me preocupo], predicted=[yo no lo preocupo]
```

Figure 88: Show example of train Neural translation English-Spanish .

Looking at the results for the test dataset first, we can see that the translations are readable and mostly correct.

- **For example :** "`i want eggs`" was correctly translated to "`quiero huevos`".

We can also see that the translations were not perfect, with "`I never worry`" translated to "`yo no lo preocupo`" instead of the expected "`yo nunca me preocupo`".

We will also calculate the BLEU scores to get a quantitative idea of how well the model has performed.

```
|BLEU-1| 0.703341 |
|BLEU-2| 0.598139 |
|BLEU-3| 0.476082 |
|BLEU-4| 0.244593 |
```

Figure 89: Show result of BLEU train .

A BLEU-4 score of about 0.244593 : The general idea is difficult to understand.

- **Test :**

Looking at the results on the test set, do see readable translations, which is not an easy task.

```
src=[wholl buy it], target=[quien lo comprara], predicted=[quien lo lo]
src=[are we lost], target=[estamos perdidos], predicted=[hemos perdimos]
src=[tom is a thief], target=[tom es un ladron], predicted=[es es timido]
src=[please go away], target=[vayase por favor], predicted=[por favor por]
src=[tom isnt sure], target=[tom no esta seguro], predicted=[tom no esta enfermo]
src=[i grinned], target=[sonrei], predicted=[me escondi]
src=[go for it], target=[ve a cogerlo], predicted=[ve a por ello]
src=[tom will hurry], target=[tom se dara prisa], predicted=[tom escucha]
src=[i blacked out], target=[quede inconsciente], predicted=[perdi la consciencia]
src=[how do we begin], target=[como empezamos], predicted=[como empiezo]
```

Figure 90: Show example of test Neural translation English-Spanish .

- **For example :** we see poor translations that the model, so it needs more tuning, such as "`tom isn't sure `" translated as "`tom no esta enfermo`" instead of the expected "`tom no esta seguro`".

BLEU-1	0.398481
BLEU-2	0.261136
BLEU-3	0.173459
BLEU-4	0.063454

Figure 91: Show result of BLEU train .

A BLEU-4 score of about 0.063 : Almost useless translations.

3.7.7 Test 7: machine translation English to Turkish .

For example, 10 lines of the file .

```
Saved: English-Turkish.txt
[eat it] => [bunu ye]
[eat up] => [silip supur]
[freeze] => [kmldama]
[freeze] => [oldugun yerde kal]
[go now] => [simdi git]
[go now] => [haydi git]
[go now] => [git artk]
[go now] => [gidin artk]
[got it] => [anladm]
[got it] => [anladn m]
```

Figure 92: Example of dataset ‘tur.txt’ .

- ✓ **Presentation of results :**

we also repeat this for each source phrase in a dataset "tur.txt" and compare the predicted result to the expected target phrase in Turkish .

- o **Train :**

We print some of these comparisons to screen to get an idea of how the model performs in practice.

```
src=[i want my mom], target=[annemi istiyorum], predicted=[ben istiyorum]
src=[will tom die], target=[tom olecek mi], predicted=[tom olecek mi]
src=[we admire you], target=[biz size hayranz], predicted=[biz size]
src=[i must do it], target=[onu yapmalym], predicted=[onu duyabiliyorum]
src=[toms smiling], target=[tom gulumsuyor], predicted=[tom gulumsuyor]
src=[did tom smile], target=[tom gulumsedi mi], predicted=[tom aglad m]
src=[tom is busy], target=[tom mesgul], predicted=[tom iyi]
src=[its a joke], target=[bu bir saka], predicted=[bu bir saka]
src=[i stammered], target=[ben kekeledim], predicted=[ben degistim]
src=[well check], target=[biz kontrol edecegiz], predicted=[biz kontrol]
```

Figure 93: Show example of train Neural translation English-Turkish .

Looking at the results for the test dataset first, we can see that the translations are readable and mostly correct.

- **For example :** "its a joke" was correctly translated to "bu bir saka".

We can also see that the translations were not perfect, with "we admire you" translated to "biz size" instead of the expected "biz size hayranz".

We will also calculate the BLEU scores to get a quantitative idea of how well the model has performed.

BLEU-1	0.605625
BLEU-2	0.447985
BLEU-3	0.278882
BLEU-4	0.095341

Figure 94: Show result of BLEU train .

A BLEU-4 score of about 0.063 : Almost useless translations.

- **Test :**

Looking at the results on the test set, do see readable translations, which is not an easy task.

```
src=[you go first], target=[siz once gidin], predicted=[buraya geri]
src=[is tom lost], target=[tom kayboldu mu], predicted=[tom hakl m]
src=[wait a while], target=[bir sure bekle], predicted=[bir sure]
src=[is tom better], target=[tom daha iyi mi], predicted=[tom yemek boylu mu]
src=[i had to stop], target=[durdurmak zorunda kaldm], predicted=[baglantm zorunda]
src=[count on it], target=[buna guven], predicted=[onu onu]
src=[im skeptical], target=[ben supheciyim], predicted=[ben yaralym]
src=[may i leave], target=[ayrlabilir miyim], predicted=[gidebilir miyim]
src=[look it up], target=[ona bak], predicted=[onu et]
src=[dont do that], target=[bunu yapma], predicted=[onu yapma]
```

Figure 95: Show example of test Neural translation English-Turkish .

- **For example :** we see poor translations that the model, so it needs more tuning, such as "im skeptical" translated as "ben yaralym" instead of the expected "ben supheciyim".

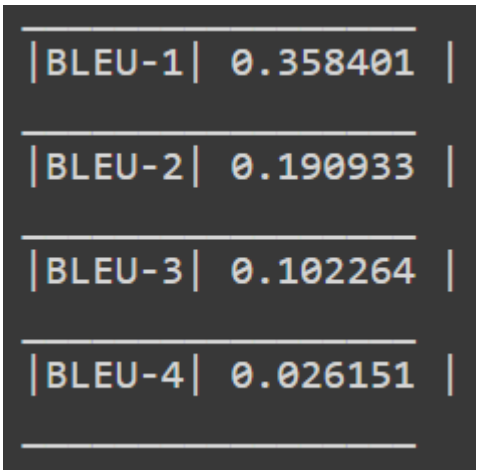


Figure 96: Show result of BLEU test .

A BLEU-4 score of about 0.063 : Almost useless translations.

❖ Comparison :

Languge pair		English-Arabic	English-Frensh	English-German	English-Italian	English-Turkish	English-Spanish
BLEU Score	BLEU-1	0.513900	0.680858	0.773200	0.600832	0.686288	0.696547
	BLEU-2	0.363046	0.582632	0.693826	0.461498	0.548312	0.585576
	BLEU-3	0.267505	0.501576	0.638806	0.324525	0.361859	0.461888
	BLEU-4	0.131254	0.309615	0.463195	0.138696	0.135117	0.228875

Table 8 : Results BLEU Score of train for all languages .

Languge pair		English-Arabic	English-Frensh	English-German	English-Italian	English-Turkish	English-Spanish
BLEU Score	BLEU-1	0.223806	0.433093	0.492238	0.452197	0.363416	0.392603
	BLEU-2	0.109811	0.309460	0.361562	0.301020	0.196673	0.266062
	BLEU-3	0.069190	0.222887	0.284944	0.175791	0.099636	0.172912
	BLEU-4	0.104902	0.103334	0.146697	0.049044	0.143767	0.060336

Table 9 : Results BLEU Score of test for all languages .

- **Discussion :**

By comparing the results of the BLEU Score , we find that the best result is the translation from English to Garman .

Therefore, we will try this model on both Arabic and English .

The results are as follows :

3.7.8 Test 8: machine translation English to Arabic with other model trained in English and German :

- **Train :**

We print some of these comparisons to screen to get an idea of how the model performs in practice.



Figure 97: Show example of train Neural translation English-Arabic with other model(English ang German)

We see very poor translations that the model, so it needs more tuning, such as "`what time is it`" translated as "الذي أن أن" instead of the expected "ما الساعة".

We will calculate the BLEU scores to get a quantitative idea of how well the model has performed.

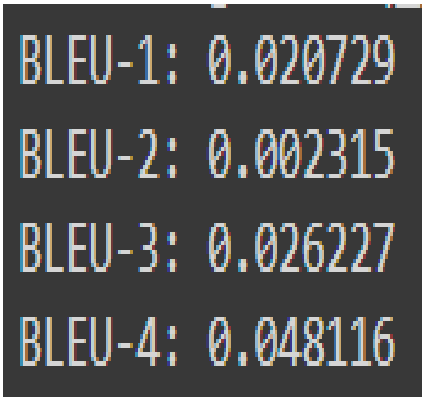


Figure 98: Show result of BLEU train .

A BLEU-4 score of about 0.048 : Almost useless translations.

- **Test :**

Looking at the results on the test set, do see readable translations, which is not an easy task.

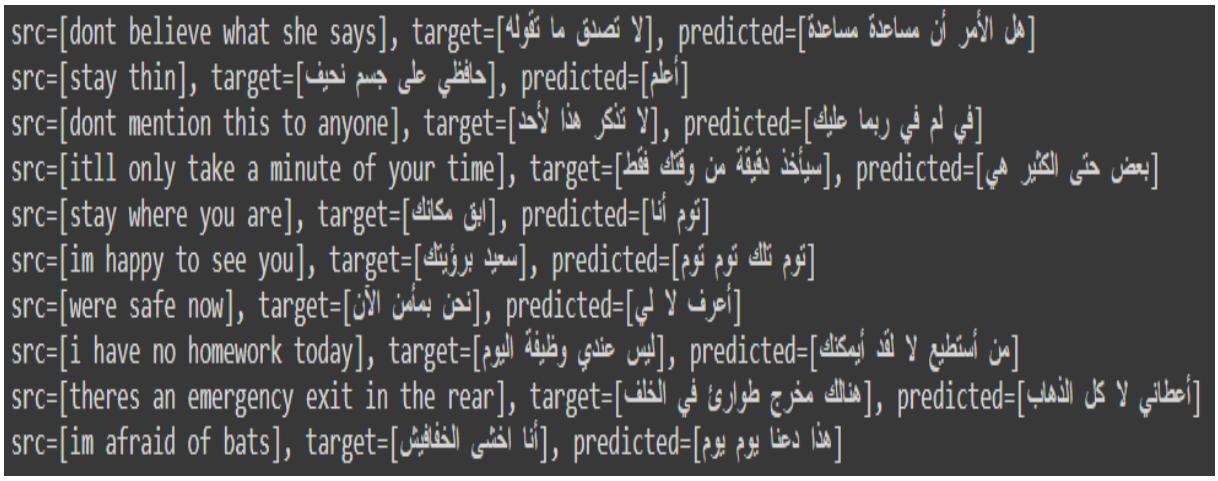


Figure 99: Show example of test Neural translation English-Arabic with other model(English ang German) .

We also see very poor translations that the model, so it needs more tuning, such as "dont believe what she says" translated as " هل الأمر أن مساعدة مساعدة" instead of the expected "لا تصدق ما تقوله".

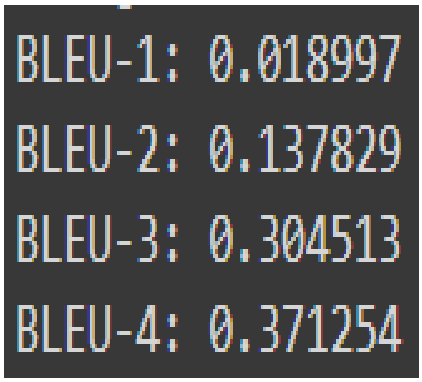


Figure 100: Show result of BLEU test .

A BLEU- 1score of about 0. 018: Almost useless translations.

- **Discussion :**

By comparing the first and last test of translating English into Arabic, we find that the results of the pure translation in the first test (model English-Arabic) were better than the results of the hybrid translation in the last test (model English-German).

3.7.9 Test 9 : machine translation English to Arabic (Translate new sentences) .

We tried new sentences, the results were the following :

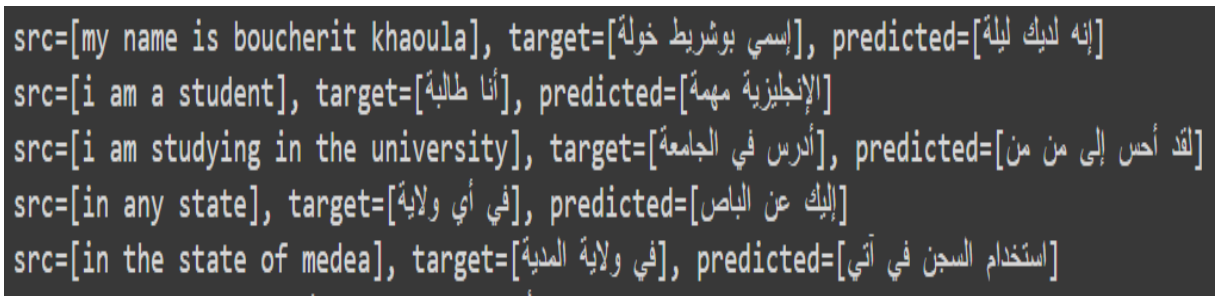


Figure 101: Show result of test new sentence english to arabic .

Results were obtained far from what we expected.

4. Conclusion :

In this chapter, we presented the implementation, the tests and the obtained results of our proposed approach, which mainly aims to improve the process of neural machine translation from English to Arabic .

General conclusion :

General conclusion :

Our work concern neural machine translation between English and Arabic, via encoder-decoder models combined with LSTM layers .

We used dataset of English-Arabic sentence pairs.we also used five sentence pairs of similar sentences for different languages to compare the results using BLEU Score, and apply the best result to the English- Arabic languages.

Several steps of pre-processing were implemented: Normalisation, Filtering(remove punctuation marks, arabic diacritics and Numbers), digital data processing and convert it to what is called one hot encoding to use as input and target for the encoder-decoder model.

An encoder decoder structure allows for a different input and output sequence length. First, we use an Embedding layer to create a spatial representation of the word and feed it into a LSTM layer that outputs a hidden vector, because we just focus on the output of the last time step we use return_sequences=False.

The decoder will be built with Embedding layer then LSTM layer and the parameter return_sequences=True, so each output of the time steps is used by the Dense layer.

During training we used the teacher forcing method where we offset the decoders input and target by one timestep. For inference, we used a slightly different setup, but still used the same modules we trained earlier. The input sequence was processed by the encoder and its final hidden states, along with the start-of-sequence "STAR ", were used as input for the decoder. Each predicted word was then fed back into the decoder while the hidden states were updated. We repeated this until the decoder predicted the end-of-sequence "END " telling us the predicted sequence is complete.

The results obtained were not satisfactory enough compared to Google Translate, for example, because the latter is used a more sophisticated encoder-decoder model .

Even though this model is already a nice improvement we can train it on a larger dataset and we also could increase the number of LSTM layers in the model , instead of just one layer in the encoder and one layer in the decoder or use bidirectional LSTM. We could also use a pre-trained embedding layer like word2vec or Glove. Finally, we could use the attention mechanism which is one of the major improvements in the natural language processing fiel

Web references :

Web references :

[1] : Ed Burns,Kate Brush.(2021,March).DEFINITION deep learning.Retrieved from https://searchenterpriseai.techtarget.com/definition/deep-learning-deep-neural-network(Article)

[2] : Kechit Goyal.(2019,DEC 13).Difference Between Deep Learning and Neural Networks.Retrieved from https://www.upgrad.com/blog/deep-learning-vs-neural-networks-difference-between-deep-learning-and-neural-networks/# (Blog)

[3] : Admin M2 IESC.(2018,avril 16).Artificial Intelligence, Machine Learning, and Deep Learning: Same context, Different concepts.Retrieved from https://master-iesc-angers.com/artificial-intelligence-machine-learning-and-deep-learning-same-context-different-concepts/ (Poster)

[4] : narayana reddy.(2020).Difference between Machine Learning and Deep Learning.Retrieved from narayana reddy.(2020).Difference between Machine Learning and Deep Learning.Retrieved from https://morioh.com/p/ed56b4fdbf1c(Poster)

[5] : Priya Pedamkar.(2020).7 Awesome Types of Deep Learning Networks - eduCBA.Retrieved from https://www.educba.com/deep-learning-networks/ (Tutorial)

[6] : Editorial.(2020,March 15).Different types of Deep Learning models explained.Retrieved from https://roboticsbiz.com/different-types-of-deep-learning-models-explained/ (Article)

[7] : Kenneth.(2019,Nov 26).Top 15 Deep Learning Applications In 2020.Retrieved from https://robots.net/tech-reviews/deep-learning-applications/# (Poster)

[8] : Muhammad Irfan.(2017,October).Bahria University.Machine Translation.Retrieved from https://www.researchgate.net/publication/320730405_Machine_Translation (Article)

[9] : What is Machine Translation? MACHINE TRANSLATION.Retrieved from https://kantanmt.com/documents/Machine_Translation.pdf (course)

[10] : Neural Machine Translation Definition.DeepAI.Neural Machine Translation.Retrieved from https://deepai.org/machine-learning-glossary-and-terms/neural-machine-translation (Poster)

[11] : What is Neural Machine Translation?.Lilt.What is Neural Machine Translation?.Retrieved from https://lilt.com/what-is-neural-machine-translation (Poster)

[12] : Expert.ai Team.(2020,June 9).Natural Language Processing Applications.Retrieved from https://www.expert.ai/blog/natural-language-processing-applications/ (Blog)

[13] : What is a Neural Network?.Databricks.Neural Network.Retrieved from https://databricks.com/glossary/neural-network (Poster)

[14] : upGrad.(2020,MAY 6).Neural Network: Architecture, Components & Top Algorithms.Retrieved from https://www.upgrad.com/blog/neural-network-architecture-components-algorithms/ (Blog)

Web references :

[15] : Diana Ramos.(2018,Oct 17):(Last modified on Jul 19, 2021).Real-Life and Business Applications of Neural Networks.Retrieved from https://www.smartsheet.com/neural-network-applications (Poster)

[16] : IBM Cloud Education.(2020,Sept14).Recurrent Neural Networks.Retrieved from https://www.ibm.com/cloud/learn/recurrent-neural-networks (Poster)

[17] : Priya Pedamkar.Recurrent Neural Networks (RNN) | Working | Steps - eduCBA.Recurrent Neural Networks (RNN).Retrieved from https://www.educba.com/recurrent-neural-networks-rnn/ (Poster)

[18] : Avijeet Biswal.(2021,June 2):(Last modified on 2021,Sep 7).Recurrent Neural Network (RNN) Tutorial for Beginners.Recurrent Neural Network (RNN) Tutorial: Types, Examples, LSTM & More.Retrieved from https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn (Tutorial)

[19] : Afshine Amidi,Shervine Amidi.CS 230 - Recurrent Neural Networks Cheatsheet.Recurrent Neural Networks cheatsheet.Retrieved from https://github.com/afshinea/stanford-cs-230-deep-learning/blob/master/en/cheatsheet-recurrent-neural-networks.pdf (GitHub)

[20] : Yash Prasad.Types of RNN (Recurrent Neural Network) - OpenGenus IQ.Types of RNN (Recurrent Neural Network).Retrieved from https://iq.opengenus.org/types-of-rnn/ (Poster)

[21] : What is the Natural Language Toolkit (NLTK)? - Techopedia.Natural Language Toolkit (NLTK). Retrieved from https://www.techopedia.com/definition/30343/natural-language-toolkit-nltk (Poster)

[22] : Devdarshan Mishra.Applications of Recurrent Neural Networks (RNNs).Applications of Recurrent Neural Networks (RNNs).Retrieved from https://iq.opengenus.org/applications-of-rnn/ (Poster)

[23] : Aditi Mittal.(2019,Oct 12).Understanding RNN and LSTM.Retrieved from https://aditi-mittal.medium.com/understanding-rnn-and-lstm-f7cdf6dfc14e (Poster)

[24] : Michael Phi.(2018,Sep 24).Illustrated Guide to LSTM's and GRU's: A step by step explanation.Retrieved from https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21 (Poster)

[25] : Jay Alammar.(2018,May 9).Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models With Attention).Retrieved from https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/ (Poster)

[26] : Harshall Lamba.(2019,Feb 8).Word Level English to Marathi Neural Machine Translation using Encoder-Decoder Model.Retrieved from https://towardsdatascience.com/word-level-english-to-marathi-neural-machine-translation-using-seq2seq-encoder-decoder-lstm-model-1a913f2dc4a7 (Poster)

Web references :

[27] : Aravind Pai.(2019,JUNE 10).Comprehensive Guide to Text Summarization using Deep Learning in Python.Retrieved from https://laptrinhx.com/comprehensive-guide-to-text-summarization-using-deep-learning-in-python-3082791976/ (Poster)

[28] : Dense layer - Keras.Retrieved from https://keras.io/api/layers/core_layers/dense/ (Poster)

[29] : (2021-09-04).tf.keras.layers.LSTM | TensorFlow Core v2.6.0.tf.keras.layers.LSTM..Retrieved from https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM (Poster)

[30] : Palash Sharma.(2020,Oct 20).Keras Dense Layer Explained for Beginners.Retrieved from https://machinelearningknowledge.ai/keras-dense-layer-explained-for-beginners/#:~:text=The%20dense%20layer%20is%20a,used%20layer%20in%20the%20models.&text=Thus%2C%20dense%20layer%20is%20basically,the%20dimensions%20of%20the%20vector (Poster)

[31] : Murat Karakaya Akademi.(2020,oct 28).LSTM: Understanding Output Types. INTRODUCTION - Medium.Retrieved from https://colab.research.google.com/drive/1nUazJkXcRZi-XiXCr6sEgCy2Z5Mhze1n?usp=sharing#scrollTo=XEdXw7pTYixX (Tutorial)

[32] : Murat Karakaya Akademi.(2020,nov 15).PART D: Encoder-Decoder with Teacher Forcing - Medium.Retrieved from https://colab.research.google.com/drive/1TNUB8vsZimVZl_wktWS1ac4wyw8kaNLW?usp=sharing#scrollTo=n0ztyXpsvVU1 (Tutorial)

[33] : IIEC Business school.(2019,DEC 11).5 step life cycle for neural network models in keras.Retrieved from https://tutorials.one/5-step-life-cycle-for-neural-network-models-in-keras/?fbclid=IwAR3aTLbg1KnRvXR8ctbSNDKyNssAQXYD1uDQqCX6HQ-_uDMpjNay2saHD4c (Tutorial)

[34] : ReLu Function in Python - JournalDev.ReLu Function in Python.Retrieved from https://www.journaldev.com/45330/relu-function-in-python (Tutorial)

[35] : Ankit Lathiya.(2020,Sep 9).Python Tanh: How To Use Math.Tanh() In Python.Retrieved from https://appdividend.com/2020/02/15/python-tanh-function-math-tanh-in-python-example/#:~:text=Python%20tanh()%20is%20an,returns%20the%20hyperbolic%20tangent%20value (Blog)

[36] : Mindfire Solutions.(2017,Apr 24).Advantages and Disadvantages of Python Programming Language.Retrieved from https://medium.com/@mindfiresolutions.usa/advantages-and-disadvantages-of-python-programming-language-fd0b394f2121 (Poster)

[37] : w3resource. (2018, April 14). Python Tutorial. Retrieved from https://www.w3resource.com/python/python-tutorial.php (Tutorial)

[38] : Kirti Vashee.(2019,12 Apr).Understanding machine translation Quality: BLEU scores.Retrieved from https://www.rws.com/blog/understanding-mt-quality-bleu-scores/ (Blog)

Web references :

[39] : Trang Ho.(2006).Qu'est-ce que Tatoeba ?. Retrieved from https://tatoeba.org/en/about (Tutorial)

[40] : Renu Khandelwal.(2020,Jan 25).BLEU — Bilingual Evaluation Understudy.Retrieved from https://towardsdatascience.com/bleu-bilingual-evaluation-understudy-2b4eab9bcfd1 (Poster)

[41] : Python Tutorial - W3Schools.Python Tutorial.Retrieved from https://www.w3schools.com/python/default.asp (Tutorial)

[42] : Vishakha Lall.(2018,Apr 1).Google Colab — The Beginner's Guide.Retrieved from https://medium.com/lean-in-women-in-tech-india/google-colab-the-beginners-guide-5ad3b417dfa (Poster)

[43] : Michael Nolledo.(2020,Jul 28).What is Google Drive? A guide to navigating Google's file storage service and collaboration tools.Retrieved from https://www.businessinsider.fr/us/what-is-google-drive-guide (Poster)

[44] : Ubiq Security.(2021,Feb 15).128 or 256 bit Encryption: Which Should I Use?.Retrieved from https://www.ubiqsecurity.com/blog/128bit-or-256bit-encryption-which-to-use/ (Blog)

[45] : Keras.About Keras https://keras.io/about/ (Tutorial)

[46] : Python SciKit Learn Tutorial - JournalDev.Python SciKit Learn Tutorial.Retrieved from https://www.journaldev.com/18341/python-scikit-learn-tutorial (Tutorial)

Books :

Books :

[47] : GOODFELLOW, Ian, BENGIO, Yoshua, et COURVILLE, Aaron. *Deep learning*. MIT press, 2016.

[48] : THE EUGUNE A.NIDA INSTITUTE FOR BIBLICAL SCHOLARSHIP. Similarity and Difference in Translation. Proceedings of the International conference on similarity and translation.New York City.May 31 - June 1,2001.STEFANO ARDUINI and ROBERT HODGSON JR.

[49] : BROWNLEE, Jason. Deep learning for natural language processing: develop deep learning models for your natural language problems. Machine Learning Mastery, 2017.

[50] : BROWNLEE, Jason. Long short-term memory networks with python: develop sequence prediction models with deep learning. Machine Learning Mastery, 2017.

[51] : LUNDH, Fredrik. Python standard library. " O'Reilly Media, Inc.", 2001.

[52] : PERKINS, Jacob. *Python text processing with NLTK 2.0 cookbook*. PACKT publishing, 2010.

Articles references :

Articles references :

[53] : Cf. Le Grand Robert, 1966.

[54]: Wehle, Hans-Dieter. "Machine learning, deep learning, and ai: What's the difference?." *Internationan Conference on Data scientist innovation day, Bruxelles, Belgium*. 2017.

[55] : AFLI, Haithem. La Traduction automatique statistique dans un contexte multimodal. 2014. Thèse de doctorat. Université du Maine. (Article)

[56] : TAGHBALOUT, Imane, ALLAH, Fadoua ATAA, et ELMARRAKI, Mohamed. Etude comparative des approches de traduction automatique.(Article)

[57] : KOEHN, Philipp. Statistical Machine Translation. Draft of Chapter 13: Neural Machine Translation. *Statistical Machine Translation. arXiv,* 2017.(Article)

[58] : JOSHI, Aravind K. Natural language processing. *Science*, 1991, vol. 253, no 5025, p. 1242-1249.(Article)

[59] : CHOWDHURY, Gobinda G. Natural language processing. *Annual review of information science and technology*, 2003, vol. 37, no 1, p. 51-89.(Article)

[60] : Johri, P., Kathait, M., Sabharwal, M., Al-Taani, A. T., & Suvanov, S. (2020). atural an ua Pro ssin istor, volution, A filiation and utur ork.

[61] : CHOPRA, Abhimanyu, PRASHAR, Abhinav, et SAIN, Chandresh. Natural language processing. *International journal of technology enhancements and emerging engineering research*, 2013, vol. 1, no 4, p. 131-134.

[62] : CHOPRA, Abhimanyu, PRASHAR, Abhinav, et SAIN, Chandresh. Natural language processing. *International journal of technology enhancements and emerging engineering research*, 2013, vol. 1, no 4, p. 131-134. (Article)

[63] : Liddy, Elizabeth D. "Natural language processing." (2001). (Article)

[64] : Australian Institute of Health and Welfare (AIHW) 2007, A guide to data development, AIHW Cat. no. HWI 94, Canberra: AIHW.(Artical)

[65] : Martınez, D., Fernández, A., Hernández, P., Cristóbal, S., Schwaiger, F., Nunez, J. M., & Ruiz, J. M. (2019). Forecasting Unstable Approaches With Boosting Frameworks and LSTM Networks. *9th SESAR Innovation Days, Athens, Greece*.

[66] : REITER, Ehud. A structured review of the validity of BLEU. *Computational Linguistics*, 2018, vol. 44, no 3, p. 393-401.(Article)

[67] : KANANI, Pratik et PADOLE, Mamta. Deep learning to detect skin cancer using google colab. *International Journal of Engineering and Advanced Technology Regular Issue*, 2019, vol. 8, no 6, p. 2176-2183.

Articles references :

[68] : K. Papineni, S. Roukos, T. Ward and W.-J. Zhu, BLEU: a method for automatic evaluation of machine translation, in: Proceedings of the 40th annual meeting on association for computational linguistics, pp. 311–318, Association for Computational Linguistics, 2002. (Article)