



Ahmed DRAYA University
Faculty of Science and Technology
Department of Mathematics and Computer
Science



A Thesis Presented to Fulfill the Master's Degree in Computer Science

Option: Intelligent Systems

Theme

Deep Transcoding

Realized by :

M. Abdelmounaim KEDDI

M. Khaled ABID

Supervised by :

Dr. Mohammed MEDIANI

2021/2022

Table of Contents

| | |
|---|------------|
| Table of Contents | II |
| List of Figures | III |
| List of Tables | IV |
| Dedication | V |
| Dedication | VI |
| Thanks | VII |
| Chapter 1: General Introduction | 1 |
| Chapter 2: Machine Translation | 3 |
| 2.1 what is Artificial intelligence ? (AI) | 3 |
| 2.2 History of Artificial Intelligence | 3 |
| 2.3 Natural Language | 4 |
| 2.4 Challenges of Natural Language | 4 |
| 2.5 Natural Language Processing (NLP) | 5 |
| 2.6 Goals of Natural Language Processing | 5 |
| 2.7 Natural Language Processing Application | 5 |
| 2.8 Supervised machine learning | 6 |
| 2.9 Unsupervised machine learning | 6 |
| 2.10 Machine Translation (MT) | 6 |
| 2.11 Translator | 7 |
| 2.12 Translator function | 7 |
| 2.13 History of Machine Translation | 7 |
| 2.14 Deep Learning (DL) | 9 |
| 2.15 Statistical Machine Translation (SMT) | 10 |
| 2.16 Neural Machine Translation (NMT) | 10 |

| | | |
|-------------------|--|-----------|
| 2.16.1 | Artificial Neural Networks | 10 |
| 2.16.2 | Today’s Artificial Neural Network Applications | 11 |
| 2.16.3 | some comme structures of Neural Networks | 12 |
| 2.16.4 | History of Neural Machine Translation | 12 |
| 2.16.5 | Neural Machine Translation (NMT) | 13 |
| 2.17 | Neural Translation Models | 14 |
| 2.17.1 | Encoder-Decoder Approach | 14 |
| 2.17.2 | Training | 14 |
| 2.18 | The foundations of successful neural machine translation | 15 |
| 2.18.1 | Deep learning | 15 |
| 2.18.2 | anticipation: | 16 |
| 2.18.3 | Attention | 16 |
| Chapter 3: | Collecting the Data-set | 17 |
| 3.1 | The idea and journey of searching for training data: | 17 |
| 3.2 | Problems and difficulties in training dataset formatting | 23 |
| Chapter 4: | Implementation | 24 |
| 4.1 | Experiments | 24 |
| 4.1.1 | Training details | 24 |
| 4.1.2 | Training data | 25 |
| 4.1.3 | Preprocessing | 25 |
| 4.1.4 | Evaluation | 26 |
| 4.2 | Execution of transformations on the pre-prepared data in Colab | 27 |
| Chapter 5: | Conclusion | 32 |
| | Appendices | 34 |
| Chapter A: | Statistics | 35 |
| | Bibliography | 38 |

List of Figures

| | | |
|-----|---|----|
| 2.1 | Exemple of Multiple Perceptron Neural Networks | 11 |
| 2.2 | Using neural networks to identify objects | 12 |
| 2.3 | Using neural networks to learn about things | 12 |
| 2.4 | seq2seq neural network includes two RNNs | 13 |
| 2.5 | Sequence-to-sequence encoder-decoder model | 14 |
| 2.6 | Fully unrolled computation graph for training example with 7 input tokens | 15 |
| 3.1 | Site of RosettaCode | 20 |
| 3.2 | Site of bitbucket | 21 |
| 3.3 | Data source from GitHub | 22 |
| 4.1 | Installation of fairseq | 27 |
| 4.2 | Preparing of Data For Training | 28 |
| 4.3 | Training Data on begining | 29 |
| 4.4 | Training Data on Interruption after begining | 30 |
| 4.5 | Testing and generation of results of the translation test | 30 |
| 4.6 | Results of the translation test | 31 |
| 4.7 | Add new Data without parallel translation | 31 |

List of Tables

| | | |
|-----|--|----|
| 2.1 | A non-exhaustive list of current and potential AI applications in medicine | 4 |
| 4.1 | corpus initial without devision | 25 |
| 4.2 | Dataset splitted on training, testing and validation corpus | 26 |
| 4.3 | BLEU score evaluation | 26 |

Dedication

I dedicate our work to:

The two most cherished beings in the world, candles who have always guided me on the right path:

My mother, the light of our life, who did everything to our success and happiness.

My father, to him we must offer all respect and love for her support and tenderness.

To our brothers and sisters for their help and encouragement

To my friend Khaled,

To all family members and friends.

KEDDI Abdelmounaim

Dedication

I thank Almighty God, I thank both my parents, my wife, my children, and all those who have the virtue of god after God over me from near or far without forgetting any of them.

Khaled ABID

Thanks

We thank God Almighty, who has enabled us to complete this scientific research, and who has given us health, wellness and determination. Praise be to God.

We extend our sincere thanks and appreciation to the supervising professor, Mohammed MEDIANI, for all the guidance and valuable information he provided us with, which contributed to enriching the subject of our study in all its various aspects. Everyone who helped us from near or far. And in the end, we can only pray to God, the Mighty and Sublime, to grant us repayment, guidance, chastity, and wealth, and to make us guided by the right path.

Chapter 1

General Introduction

Programming is the process of writing instructions and commands for a computer or informing about how to deal with data or how to carry out a series of works that you want in our time, as it has become dependent on and indispensable software.

Programming languages were discovered in general in the nineteenth century. The first programming language was invented by the English mathematician Ada Lovelace, who wrote the world's first machine algorithm in collaboration with Charles Babbage's, a famous mathematician, He is also a philosopher and mechanical engineer who is credited with inventing the concept of a programmable computer.

Programmers do research and development to discover more programming languages to make it easier to communicate with a computer and make it easier to perform many functions. Without programming languages, we would not have been able to use the Internet well, whether on computers or smartphones.

During the past decades, there has been a great development in programming languages, and this development has resulted in changes in the interface of software design and programming languages, as they have become easier, more interactive and more enjoyable to learn. Although there are a huge number of programming languages that differ in their goals and applications, most of these languages have gone through enormous stages of development over the centuries until they reached us in their current form, which is the most easy and simple, closer to the human language and easier to learn. Below are the stages of development of programming languages :

- The first generation Machine Language: It is the first language used in dealing with the computer, and it is a set of binary numbers (0,1) that the computer reads and was dealt with directly through the computer keyboard. Machine is one of the most difficult languages, and it becomes more difficult to modify when an error occurs.
- The second generation Assembly Language: It is a low-level programming language that shortens some of the phrases and symbols used, where the numeric symbols in the machine language are replaced by a group of abbreviated symbolic words using the English language, and this language was considered a giant leap in the world of programming languages.

- Third generation High-level procedural languages: They are a higher level than assembly languages and machine language, very similar to everyday texts and mathematical formulas in appearance. The computer is what it should do, and the most popular multi-use programming languages are C++, C, and Java.
- Fourth generation non-procedural high-level languages: These are high-level languages, but unlike third-generation languages, they allow users and programmers to define what the computer should do without having to specify how to do it.
- Fifth generation: These are languages designed so that the computer can act as a programmer and without the need to know how the code is written in detail. This language is mainly used in artificial intelligence applications.

Migrating a database from an old programming language like COBOL to a modern alternative like JAVA or C++ is a challenging, resource-intensive task that requires expertise in both source and target languages. For example, COBOL is still widely used today in mainframe systems around the world, so companies, governments, and others often have to choose whether to manually translate their code bases or commit to preserving code written in a language that dates back to the past. To the fifties of the last century 1950, it may be costly a lot of time and cost to do manual translation. For example, the Australian Commonwealth Bank spent about 750 million dollars to convert its database from COBOL to JAVA in a time frame of about 5 years. To solve this problem We must use intelligent systems capable of translating code from the source language to the target[1].

The recent developments in NMT have been widely accepted. Translating source code by these systems requires inference of variable types and training on many data to give a more accurate and efficient result. The applications of NMT to programming languages have been limited so far, due to To the lack of parallel resources available in this area[1].

In this note we propose applying modern approaches to unsupervised machine translation, by utilizing a large amount of monolingual source code from GitHub to train a model, TransCoder, for translation between three popular languages: C++, Java and C. To evaluate our model, we create a test set of 258 parallel jobs, along with associated unit tests. Although the model was not provided with parallel data, it was able to translate functions with high accuracy, and align functions correctly from the standard library across the three languages.

Chapter 2

Machine Translation

2.1 what is Artificial intelligence ? (AI)

Since the industrial revolution, there has been tremendous technological advancement. Many difficult manual tasks have been replaced by technology, which has greatly aided humanity. Artificial Intelligence (AI) is a technical advancement that has enabled humans to replace manual labor in a variety of industries. Artificial intelligence (AI) is an area of science and technology that develops intelligent computers and computer programs to do tasks that would normally need human intelligence. It is a system that can perform a variety of human-like functions. In order to attain outstanding performance for the given tasks, AI needs external data such as big data.[1]

Previously, AI was only a notion found in science fiction and arguments concerning the impact of technology in the current world. However, technology has now become an integral part of our daily lives. It has become a critical function in many technical and other industries. Artificial intelligence has a major impact on businesses such as manufacturing, healthcare, and supply chains, among others. Because AI can do things that humans can't, it has a wide range of applications that boost performance and efficiency.

2.2 History of Artificial Intelligence

Artificial intelligence has always been a science fiction novel that has become a reality. When Homer wrote that the gods were served at their dinner table by mechanical tripods who judged what the gods preferred, he was a pioneer of AI. The AI community did not create mechanisms that existed just as theories in the literature until the late twentieth century. René Descartes was intrigued by the concept "mechanical man." Although Gottfried Wilhelm Leibniz envisioned mechanical machines solving complicated problems using mathematical reasoning, he never claimed that machines could think for themselves. Abbé de Condillac Etienne Bonnot proposed that knowledge may be poured into the skull of a statue to make it think. Human-like behavior

Many AI researchers were inspired by the works of Isaac Asimov, L. Frank Baum, and Jules Verne, who explored the possibility of intelligent machines. [4] George Boole and Alan Turing,

respectively, defined the formal language and Turing machine for logical interpretation in 1847 and 1936. J. Neumann and O. Morgenstern proved the theory of decisions a year later. In 1965, Herbert Simon stated, "Machines can achieve practically whatever that man is capable of." The scientific community, on the other hand, concluded that algorithms could not be built for everything that man is capable of. Alain Colmerauer defined PROLOG, an Artificial Intelligence system construction language, in 1972.[1]

| Basic biomedical research | Translational research | Clinical practice |
|--|---------------------------------|--|
| Automated experiments | Biomarker discovery | Disease diagnosis |
| Automated data collection | Drug–target prioritization | Interpretation of patient genomes |
| Gene function annotation | Drug discovery | Treatment selection |
| Prediction of transcription factor binding sites | Drug repurposing | Automated surgery |
| Simulation of molecular dynamics | Prediction of chemical toxicity | Patient monitoring |
| Literature mining | Genetic variant annotation | Patient risk stratification for primary prevention |

Table 2.1: A non-exhaustive list of current and potential AI applications in medicine

2.3 Natural Language

Natural language describes how humans communicate with one another. Speech and text, to be specific.[2]

2.4 Challenges of Natural Language

The problem of working with natural language data has not been solved. It has been studied for almost a half-century and is extremely difficult. It's difficult for the child who must spend years learning a language... it's difficult for the adult language learner, it's difficult for the scientist who tries to model the relevant phenomena, and it's difficult for the engineer who tries to build systems that deal with natural language input or output. These activities are so difficult that Turing could justifiably make fluent natural language conversation the core of his intelligence test.[2]

Natural language is difficult primarily due to its disorder. There are only a few guidelines. Despite this, we can generally understand each other.[2]

Human communication is notoriously imprecise... It is also constantly evolving and changing. People are excellent at producing and understanding language, and they can communicate, perceive, and interpret exceedingly complex and subtle meanings. At the same time, while hu-

mans are excellent communicators, we are terrible at formally comprehending and explaining the laws that govern language.[2]

2.5 Natural Language Processing (NLP)

Natural Language Processing, or NLP for short, is broadly defined as the automatic manipulation of natural language, like speech and text, by software. The study of natural language processing has been around for more than 50 years and grew out of the field of linguistics with the rise of computers.[2]

2.6 Goals of Natural Language Processing

As stated above, the goal of NLP is to "achieve human-like language processing." The word 'processing' was chosen with care and should not be replaced with 'understanding.' Because, while NLP was once known as Natural Language Understanding (NLU) in the early days of AI, it is now widely acknowledged that, while the goal of NLP is real NLU, it has yet to be achieved. A complete NLU system might do the following[3]:

- Take an input text and translate it into a different language.
- Respond to questions concerning the text's content.
- Make deductions from the text

2.7 Natural Language Processing Application

For a variety of applications, natural language processing provides both theory and implementations. In fact, NLP can be used in any application that uses text. The following are some of the most common NLP applications[3]:

- Given the importance of text in this application, it's remarkable that relatively few solutions use natural language processing (NLP). Statistical approaches to NLP have been more popular recently, however few systems have built substantial systems based on NLP other than those developed by Liddy and Strzalkowski.
- Information Extraction (IE) — a more modern application area, IE focuses on recognizing, tagging, and extracting significant aspects of information from vast volumes of text into a structured representation, such as people, corporations, locations, and organizations. These extracts can then be used for a variety of tasks, such as question answering, visualization, and data mining.

- Question-Answering – unlike Information Retrieval, which returns a list of possibly relevant documents in response to a user’s query, question-answering returns either just the answer text or answer-providing passages.
- Summarization — The higher levels of NLP, notably the discourse level, can enable an application that condenses a longer text into a more compact, but still highly constructed, narrative representation of the original document.
- Machine Translation — MT systems have used various levels of NLP, ranging from a ‘word-based’ approach to applications that involve deeper levels of analysis, and are possibly the oldest of all NLP applications.
- Dialogue Systems may be the omnipresent application of the future, as envisioned by large end-user application providers. The phonetic and lexical levels of language are currently used by dialogue systems, which are usually focused on a narrowly specified application (e.g., your refrigerator or home sound system). It is thought that combining all of the above layers of language processing could result in fully livable discussion systems.

2.8 Supervised machine learning

Machine learning under supervision A machine-learning job that seeks to predict the intended output (such as the presence or absence of DR) from input data (such as fundus pictures). In the supervised machine-learning phase, the input–output correlation is identified, and the identified correlation is used to predict the proper output of subsequent cases.[4]

2.9 Unsupervised machine learning

Machine learning without supervision A machine-learning task that attempts to extract underlying patterns from unlabeled data. It can, for example, find sub-clusters in the original data, identify outliers, and generate low-dimensional representations of the data.[4]

2.10 Machine Translation (MT)

Machine translation (MT) from a translational perspective, conceptual and production, is a standard term that refers to the technique of using computer software (computer systems) to transfer the content of a text in a natural language called “SL” “source language” to a second natural language called “TL” “Target language” is also used for naming the original text that is supposed to be processed by translation with the input text where it is processed by computer and then producing a translated text that is called the output text, and the translation process is

carried out with or without human assistance.[5].

exemple:

x: La promesse de la liberté est une dette envers lui

y: The promise of freedom is a debt upon him

2.11 Translator

The translator is an artificial intelligence system that is trained with data in order to convert one language into another[5].

2.12 Translator function

After the programmer writes the sentences and commands according to the rules of the language he uses, the compiler does the following¹:

- The translator analyzes the sentences and programming commands written by the programmer and tells him whether they are correct or not according to the rules of the language (the source language).
- If the commands are correct, it converts the commands into the target language.
- The compiler applies sentences, commands, and code to the program in order to show the output of its execution.

2.13 History of Machine Translation

Machine Translation (MT) is the task of translating a sentence x from one language (the source language) to a sentence y in another language (the target language)[6]

The bet on machine translation in the beginning and the expectations of its success was so great that the specialists unanimously agreed that the results would be as follows:

¹<https://www.youtube.com/watch?v=J1EL3JzWWqo>

- That the computer does the work of the translator.
- The accuracy of the translation should be 95%.
- High speed of the machine in the completion of translation.
- That the machine translate any text, whether scientific or literary.

However, scientists were surprised by the complexity of human language and the vast amount of information used in translation. Human language is ambiguous in nature and most sentences have different meanings, and we do not realize this because we hear the sentences in a specific context and use our knowledge of the world to automatically choose the intended meaning.

We can divide machine translation according to the successive developments it has witnessed into:

1. Statistical Machine Translation.
2. Neural Machine Translation

The history of machine translation began in the 1950s, after World War II. The Georgetown experiment in 1954 included sixty sentences, all of which were machine-translated from Russian into English. The experiment also met with tremendous success, and financial funding for machine translation research. The creators claimed that the machine translation problem would be solved within three to five years[7].

In the year 1990, after the failure of translation based on rules and examples to embody the idea of machine translation as it was planned, IBM developed a system for translation from French to English in which it relied on previously translated texts to elicit the possibilities of the existence of an English word that would be a correct translation of the French word, which is The same method in which statistical machine translation works until today, but in a way that has a lot of complexity, and despite the acceptable performance of this approach, it has been criticized because of the linguistic obstacles that may arise if the linguistic repertoire is limited in resources, which prompted the scholars to think about Using what simulates the way humans work by designing an artificial neural network that can be learned and trained[7].

The development of neural or neural networks has spanned for many years, although they did not appear to the public until a short time ago, thanks to Maria Asuncion Castano and her team when they proposed in 1997 preliminary methods for using models based on artificial neural networks to translate sentences automatically. However, this did not receive much attention because computers were not so technologically advanced that would allow them to apply these methods and invest them on the ground to continue the research despite the adverse results due to successive studies such as the Scwenk 2007 experiments to integrate the artificial neural network model in statistical machine translation systems and research Kalchbrenner and Blunsom in 2013 to propose a two-party approach to encoder-decoder approach [7].

These researches have returned with good results on the research approach in neural machine translation by re-emerging its star in 2014 due to the availability of two basic conditions that were absent so far “the presence of sufficiently high computing power and the availability of large amounts of data to train neural networks with it.” The launch of the first neural model based on the translation of short clips (séquence to séquence modele) constituted a strong research impetus, especially after the strengthening of the artificial neural network with the attention mechanism in 2015 and the approval of the launch of the first experimental neural machine translation program at the International Machine Translation Conference in the same year. This encouraged many major institutions such as Google, Microsoft and Systran to convert their programs from the statistical approach to the neurological approach at the beginning of 2016, adding more language pairs and continuously feeding their databases to maintain their competitiveness in the global translation market.

2.14 Deep Learning (DL)

Deep learning is a type of machine learning that allows computers to learn from their mistakes and comprehend the world as a hierarchy of concepts. There is no need for a human computer operator to expressly specify all of the knowledge that the computer requires because the computer learns through experience. A graph of these hierarchies would be many layers deep, allowing the machine

to understand sophisticated concepts by building them out of simpler ones.[8]

2.15 Statistical Machine Translation (SMT)

Statistical machine translation (SMT) is the technology that powers popular MT tools like Google Translate, Microsoft®Translator, and Asia Online. It is based on an intuitively simple strategy: rather than attempting to encode all of the linguistic and world knowledge required to translate a text from one language into another a priori in the form of dictionaries, grammars, and knowledge bases (as rule-based and knowledge-based MT do), simply learn how to translate from existing human translations. In practice, this learning entails deducing statistical translation models from parallel corpora, i.e. source texts and human translations.[9]

2.16 Neural Machine Translation (NMT)

2.16.1 Artificial Neural Networks

Neural Networks is a group of artificial neurons located in layers on top of each other, and has a first layer, and a final layer. The first layer receives the raw information, processes it to pass it on later to the next layer and so on until we get the output from the final layer[10].

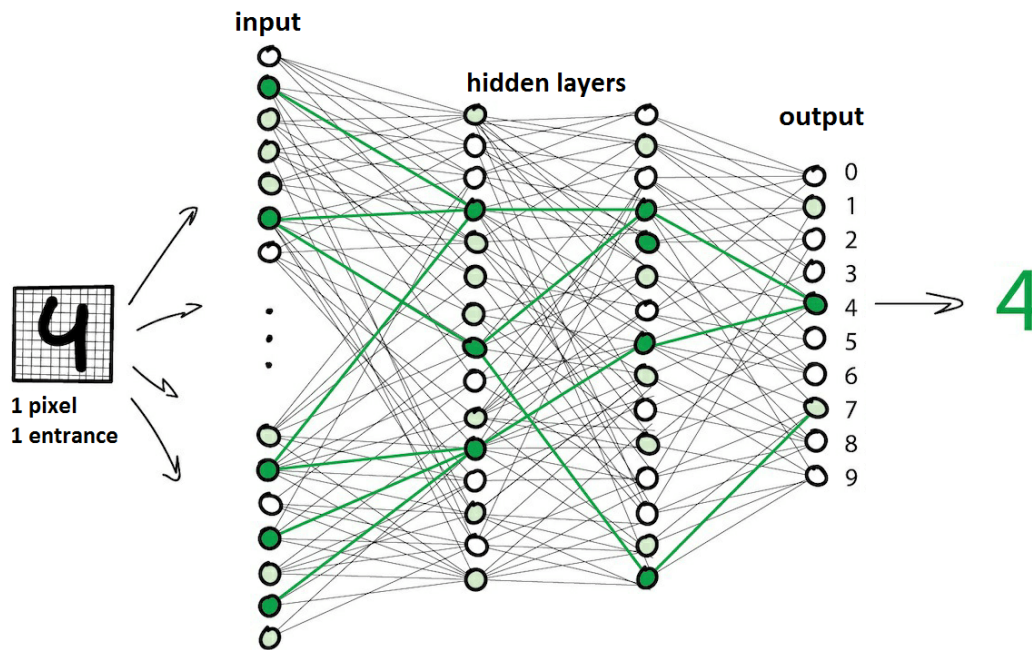


Figure 2.1: Exemple of Multiple Perceptron Neural Networks

2.16.2 Today's Artificial Neural Network Applications

There are various applications of artificial neural network. Every sector in this world want a system which is itself intelligent to solve any problem according to the inputs[11]:

1. Object identification in photos and videos.
2. Recognize speech and language structures.
3. Image processing and format conversion.
4. Machine translation[4].
5. Airline Security Control.
6. Investment Management and Risk Control.
7. Prediction of Thrift Failures.
8. Customer Research.
9. Prediction of Stock Price Index.

10. Prediction of Thrift Failures.

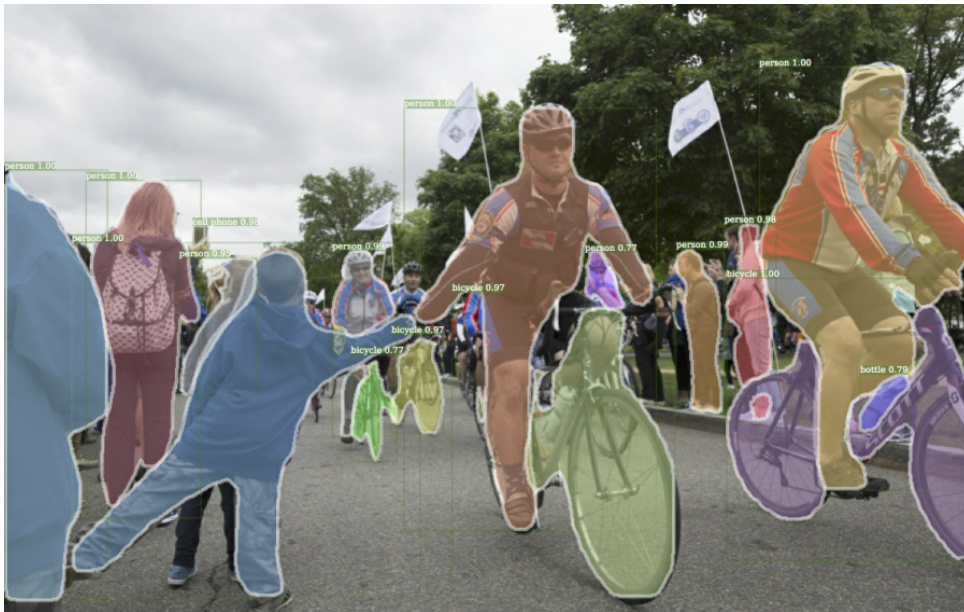


Figure 2.2: Using neural networks to identify objects

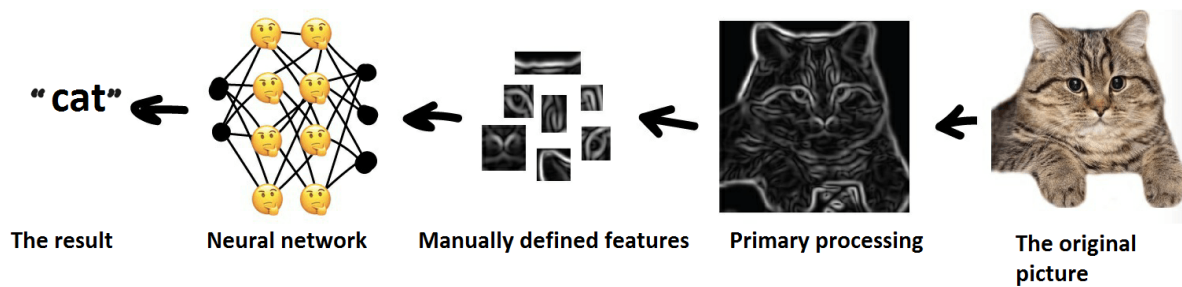


Figure 2.3: Using neural networks to learn about things

2.16.3 some common structures of Neural Networks

- Perceptron neural networks (Perceptron)[12].
- Convolutional Neural Networks (CNN)[12].
- Recurrent Neural Networks (RNN)[12].
- Autoencoders neural networks (Autoencoders)[12].

2.16.4 History of Neural Machine Translation

Machine translation was in full view of researchers researching these methods during the In the 1980s and 1990s, there was a last wave of neural network re-

search. Indeed, Forcada, et al (1997), and Castao et al. (1997) developed models that are substantially comparable to current popular neural machine translation techniques. However, none of these models have been trained on big enough data sets to produce satisfactory results for anything other than game scenarios. Because the computing complexity demanded much exceeded the computer resources available at the time, the project was shelved for over two decades.[13]

During this time, data-driven approaches such as phrase-based statistical machine translation climbed from obscurity to supremacy, making machine translation a viable tool for a wide range of applications, from information collection to enhancing professional translator productivity. With neural language models incorporated into classical statistical machine translation, systems, neural methods to machine translation have recently resurfaced. Schenck's ground-breaking research from 2007 showed considerable gains in public evaluation campaigns. However, due to computational considerations, these concepts were only gradually adopted. Many research groups have found it difficult to use GPUs for training since they do not have the necessary hardware or knowledge.[13]

2.16.5 Neural Machine Translation (NMT)

Neural Machine Translation is a new approach to machine translation Unlike the usual phrase-based translation system which is made up of a large number of words,neural machine translation seeks to construct and tune small sub-components that are tweaked separately train a single, massive neural network to comprehend a sentence and translate it correctly[14]

- Neural Machine Translation (NMT) uses a single neural network to do machine translation.
- The sequence-to-sequence (aka seq2seq) neural network architecture consists of two RNNs.

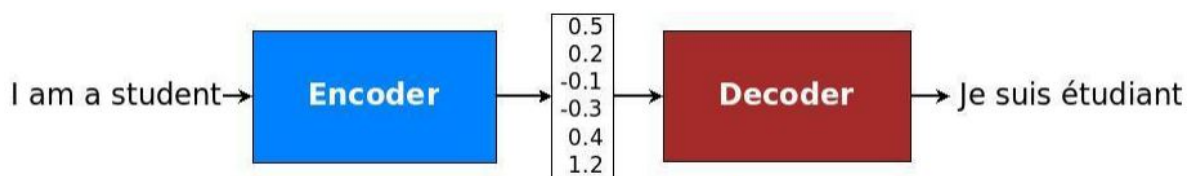


Figure 2.4: seq2seq neural network includes two RNNs

2.17 Neural Translation Models

2.17.1 Encoder-Decoder Approach

Our initial attempt at a neural translation model is simply a language model extension. Remember how a recurrent neural network can be used to model language as a sequential process. A model like this guesses the next word based on all preceding words. We now proceed to anticipate the translation of the sentence, one word at a time, after we reach the end of the sentence.[13]

for an example. We just concatenate the input and output sentences and train the model the same way we would a language model. We feed in the input sentence for decoding, then go through the model's predictions until it predicts an end of sentence token.[13]

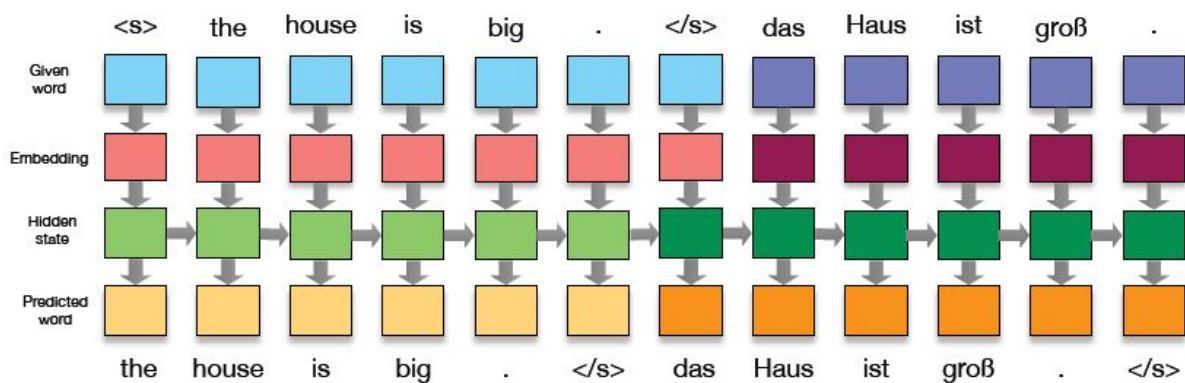


Figure 2.5: Sequence-to-sequence encoder-decoder model

2.17.2 Training

Dataset in the training corpus is used to train the parameters of a neural language model. The context words are fed into the network, and the output is compared to the onehot vector of the correct word to be predicted.[13]

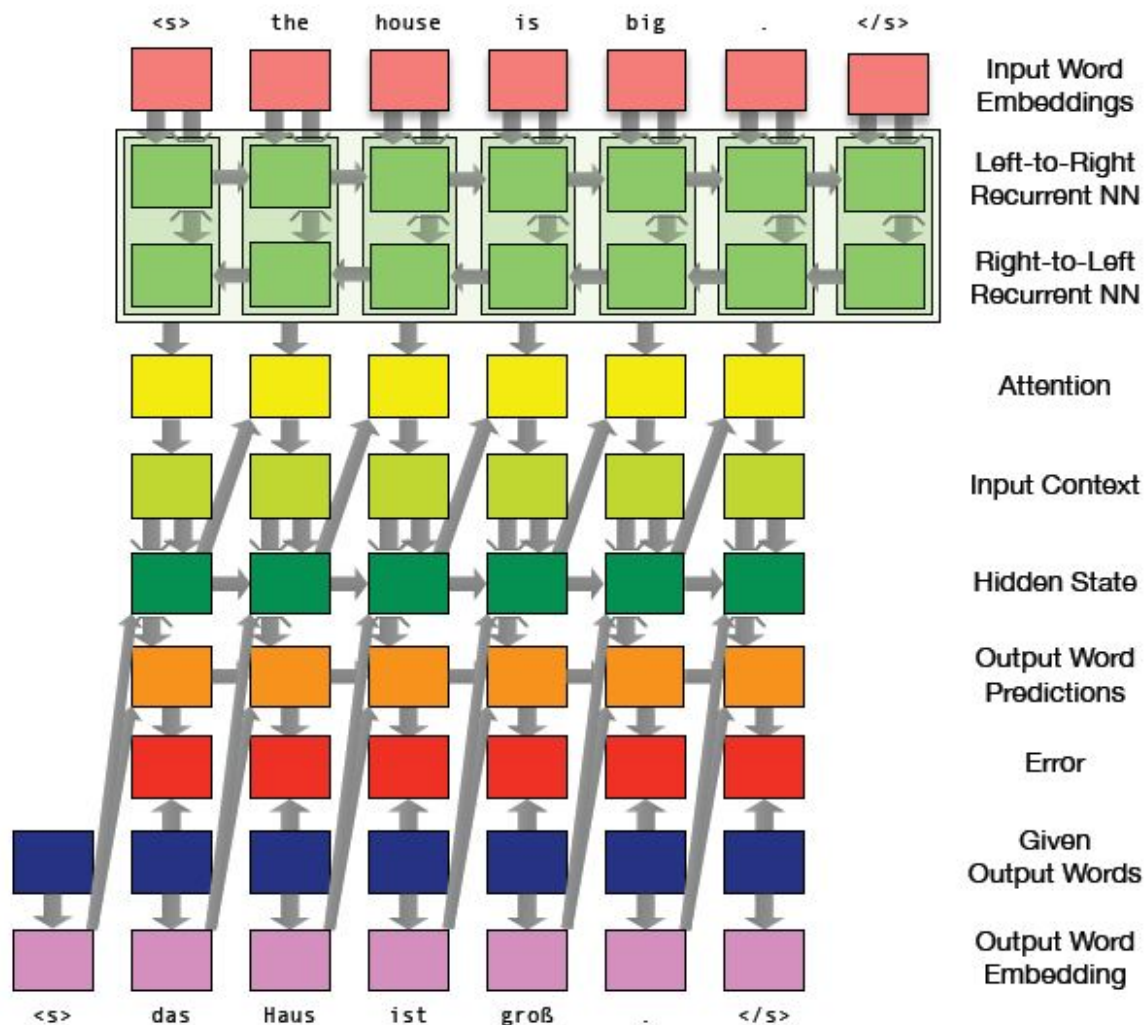


Figure 2.6: Fully unrolled computation graph for training example with 7 input tokens

2.18 The foundations of successful neural machine translation

2.18.1 Deep learning

The use of deep learning mechanism in machine translation programs began when digital data became available that could be invested in training artificial neural networks. Deep learning models are designed and trained by representing learning data instead of algorithms, as well as adding many hidden layers between the input layer and the output layer, where the deeper learning increases as the number of hidden layers increases.

2.18.2 anticipation:

Neural machine translation systems are designed in a way that enables them to develop a prediction mechanism and exploit it in translation. The program is trained in the same way as text-completion devices (Textcompletiondevices) are trained, that is, in an automatic and fast way, as shown in the keyboard of smart phones 3. When the user types For a word on the phone, the device suggests the next word based on the meaning of the previous word 32 . Neural machine translation systems work in the same way, when you enter text for translation, the program searches for all possible words To translate the first word, then move to the second word based on the meaning of the previous word, until he finishes translating the entire text.

2.18.3 Attention

In its infancy, machine translation systems were unable to translate long and complex sentences due to the failure of vectors to set the correct sequence of sentence elements in the target language, which prompted researchers to develop the attention mechanism to make it easier for the program to translate long sentences and enable it to synthesize its elements according to the rules and peculiarities of the target language.

Chapter 3

Collecting the Data-set

3.1 The idea and journey of searching for training data:

Our project comes with the idea of converting programming code from a source programming language into a target programming language. For example, the C source code to python source code, this idea is preceded, in fact we find many projects and programs based on this, But they differ only in terms of the linguistic specialization between them, or the method of conversion, either through the rules of the two languages, or through artificial neural networks, and the latter, we find those who do so facebook artificial intelligence, which publishes an open source project based on pre-trained models, based on three programming languages: C++, Java and Python.

This paves the way by saying that migrating a software base from an old programming language such as COBOL to a modern alternative such as Java or C++ is a difficult task that requires experience in both source and target languages. For example COBOL, it is still widely used today in central computer systems around the world, so companies, governments and others often have to choose whether to manually translate their code rules or commit to keeping code written in a language dating back to the 1950s.

A transcompiler, also known as source-to-source translator, is a system that converts source code from a high-level programming language (such as C++ or Python) to another. Transcompilers are used primarily for interoperability, and for translating code bases written in an old or deprecated language (such as COBOL and Python 2) into a modern language. They are usually based on

hand rewriting rules applied to the abstract syntax tree of the source code. Unfortunately, the resulting translations often lack readability, fail to respect the conventions of the target language, and require manual adjustments in order to function properly.

The overall translation process is time consuming and requires expertise in both the source and target languages, which makes code translation projects very expensive. Although neural models greatly outperform their rule-based counterparts in the context of natural language translation, their applications to transformation have been limited by the dearth of parallel data in this field.

It has been suggested to take advantage of recent methods in unsupervised machine translation to train a fully unsupervised neural translator. They train their model on the source code from the open source Github projects, and demonstrate that it could translate functions between C++, Java, and Python with high accuracy. Their method is exclusively based on monolingual source code, does not require any experience in the source or target language, and can easily be generalized to other programming languages. They also build and release a test suite of 852 parallel jobs, along with unit tests to validate translations. They show that their model outperforms rule-based trading baselines by a significant margin.

So they develop it and open its sources, a nervous system that can make migration a much easier and more efficient code. Their method is the first artificial intelligence system capable of translating code from one programming language to another without the need for parallel data for training. They claim that TransCoder can successfully translate functions between C++, Java, and Python 3. TransCoder outperforms open source and commercial rule-based translation software. In their evaluations, the model correctly translates more than 90 percent of Java functions to C++, 74.8 percent of C++ functions to Java, and 68.7 percent of functions from Java to Python. In comparison, the commercially available tool correctly translates only 61.0 percent of functions from C++ to Java, and the open source compiler is accurate for only 38.3 percent of Java functions compiled to C++.

In natural language, recent developments in neural machine translation have

been widely accepted, even among professional translators, who rely more and more on machine translation systems. However, its applications for code translation have been limited due to the scarcity of parallel data in this field.

We are looking for their pre-training data, but we don't find it, and when someone requests their training data, they are told that they are not authorized to participate.

In order to achieve our goal, it was necessary to provide training data, as we started searching for source code data on the Internet in several sites:

1. <https://searchcode.com/>
2. <https://www.codeproject.com/>
3. <https://codeanywhere.com/>

RosettaCode is a christomathy programming site. The idea is to provide solutions to the same task in as many different languages as possible, to show how languages are similar and different, and to help someone build upon one approach to a problem and learn another. RosettaCode currently contains 1184 tasks, 348 draft tasks, and is familiar with 865 languages, although we cannot (and cannot) have solutions for every task in every language.

RosettaCode is growing in code by programming language or in several languages, according to statistics that we have¹ increasing the number of examples for Cpp and Java from 744 and 842 in order in 2014 to 929 and 1051 in 2020, On 12/06/2022, it reached 1119 and 1134 in the same order as well, and is growing steadily, especially for more programming languages than others, which indicates the multiplicity of ideas with multiple problems and multiple solutions in the diversity of programming languages and the field of specialization.

¹See Appendix A Statistics

Pages in category "Programming Tasks"

The following 1,184 pages are in this category, out of 1,184 total.

- 1
 - 100 doors
 - 100 prisoners
 - 15 puzzle game
 - 15 puzzle solver
- 2
 - 2048
 - 21 game
 - 24 game
 - 24 game/Solve
- 4
 - 4-rings or 4-squares puzzle
- 9
 - 9 billion names of God the integer
 - 99 bottles of beer
- A
 - A+B
 - Abbreviations, automatic
- F
 - FTP
 - Function composition
 - Function definition
 - Function frequency
 - Function prototype
 - Functional coverage tree
 - Fusc sequence
- G
 - Galton box animation
 - Gamma function
 - Gapful numbers
 - Gauss-Jordan matrix inversion
 - Gaussian elimination
 - General FizzBuzz
 - Generate Chess960 starting position
 - Generate lower case ASCII alphabet
 - Generate random chess position
 - Generator/Exponential
 - Generic swap
 - Get system command output
 - Globally replace text in several files
 - Go Fish
 - Goldbach's comet
 - Graph colouring
 - Gray code
- Q
 - QR decomposition
 - Quaternion type
 - Queue/Definition
 - Queue/Usage
 - Quickselect algorithm
 - Quine
 - Quoting constructs
- P
 - Primes - allocate descendants to their ancestors
 - Primorial numbers
 - Priority queue
 - Probabilistic choice
 - Problem of Apollonius
 - Program name
 - Program termination
 - Proper divisors
 - Pseudo-random numbers/Combined recursive generator MRG32k3a
 - Pseudo-random numbers/Middle-square method
 - Pseudo-random numbers/PCG32
 - Pseudo-random numbers/Splitmix64
 - Pseudo-random numbers/Xorshift star
 - Pythagoras tree
 - Pythagorean quadruples
 - Pythagorean triples

Figure 3.1: Site of RosettaCode

Since planning the site is based on the problem and solving it in the manner of each programming language according to its ability, we choose two famous languages: cpp and java.

We have to extract the data available for both languages, so we search for sources extracted from the same site, and we find next two of several sites for source code data:

1. <https://bitbucket.org/nanzs/rosettacodedata/>

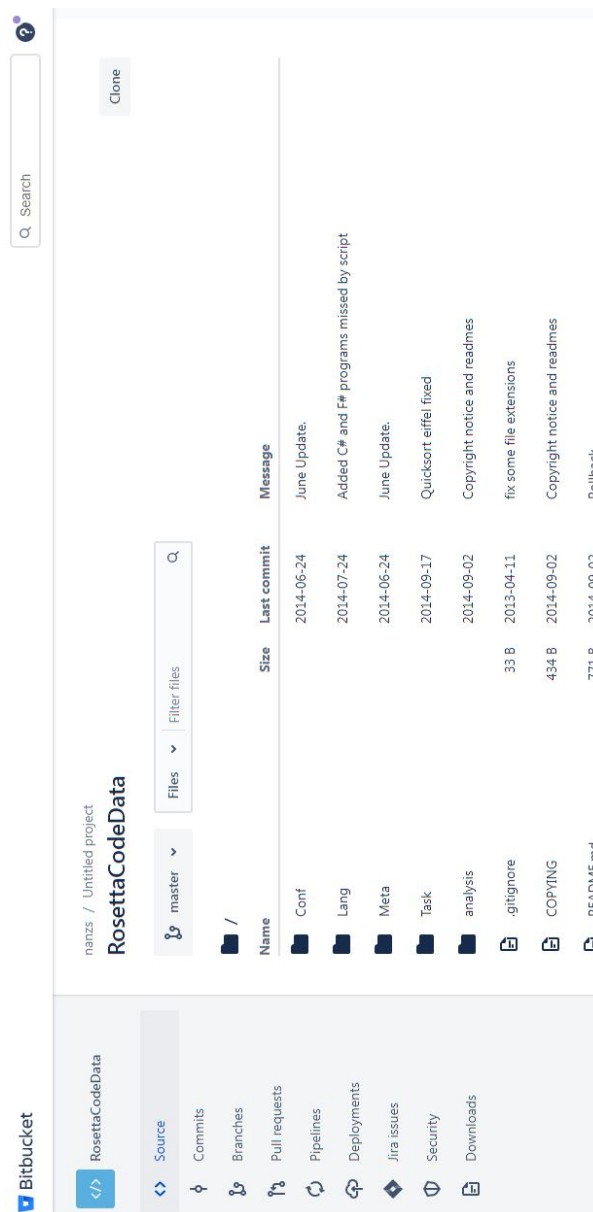


Figure 3.2: Site of bitbucket

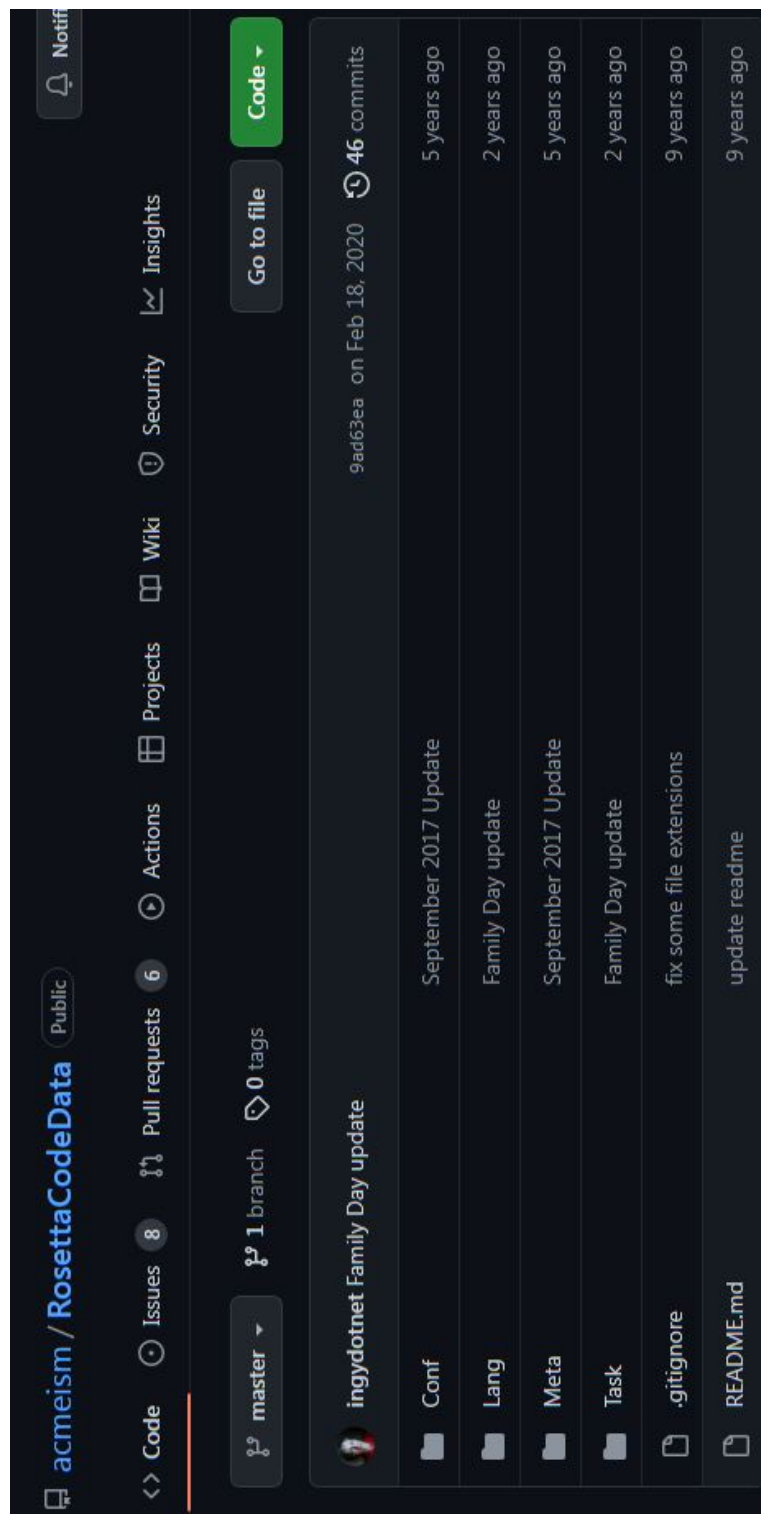
2. <https://github.com/acmeism/RosettaCodeData>

Figure 3.3: Data source from GitHub

We download copies of these sites, extract and collect data on the two programming languages mentioned above only so that each CPP source code corresponds to another java code, so that we break the code into parts at the function

level or less to reduce the length of the code.

3.2 Problems and difficulties in training dataset formatting

1. Each source code is written by its author according to his knowledge of the specific programming language without looking at another language except for a few who translate from one programming language to another.
2. The same problem is solved in the same programming language two or more times without it in other languages.
3. Long source codes in programming languages rather than in others.
4. Differences in values and comments according to each programming language for the same problem.
5. There is a difference in the formality structure and the way each programming language is written.
6. Differences in libraries that are included.
7. Difficulty checking, given the need to learn the target languages.
8. The difficulty of obtaining and extracting source data, especially since they are large, is more difficult to extract if it is by copying and pasting.
9. It takes a long time to process and prepare training data.
10. The existence of algorithms is too long for the size of the problem for the parallel solution.
11. Vocabulary is more coherent than natural speech, because of the imposed formalism.
12. Monitoring and auditing effort is costly because any software bug is a costly catastrophe.
13. Pass all sources one by one.

Chapter 4

Implementation

4.1 Experiments

4.1.1 Training details

We use to implement our project to create a source code adapter, Fairseq, which is a serial modeling toolkit written in PyTorch, allowing researchers and developers to train customized models for translation, summary, language modeling and other text types.¹

We use the following configurations:

- Optimizer adam betas '(0.9, 0.98)'
- Clip-norm 5.0
- Learning Rate 1e-3
- Encoder-layers 2 and Decoder-layers 2
- Encoder-attention-heads 4 and Decoder-attention-heads 4
- Batch-size 100
- Max-tokens 1024

¹<https://fairseq.readthedocs.io/en/latest/>

4.1.2 Training data

We download dataset available on [bitbucket.org](https://bitbucket.org/nanzs/rosettacodedata/)² and RosettaCodeData repository on GitHub³ we use It more than 10900 open source. We filter these sources from unusable comments, and select the C++, Java, files within those projects.

In this work, we decide to translate at function level. Unlike files or classes, functions are short enough to fit into a single batch, and working at function level allows for a simpler evaluation of the model.

We pre-train our model on all the source code available to us.

4.1.3 Preprocessing

After extracting data on Java and C++, we placed each source code for each programming language with the corresponding in the other language in parallel, all in one file to make it easier to verify the meaning and comparison, using comment breaks that mark the beginning of each source by each programming language in a unified order, the step after which we re-separated each source by programming language into a file alone, giving it a symbol indicating the programming language to which it belongs. For example, Java we code with "J" plus ranking number 1, 2, 3. The result is J1, J2, etc. Up to the number of source codes in place as in the next table :

| Corpus | Lines | Words |
|---------------------|-------|--------|
| corpus.cp | 5487 | 529194 |
| corpus.jv | 5487 | 534392 |
| corpus-nocomment.cp | 5487 | 525614 |
| corpus-nocomment.jv | 5487 | 529046 |

Table 4.1: corpus initial without deviation

The next step is to group the lines of source codes into one line for each code with a vocabulary by vocabulary according to each programming language for the source, and therefore all the separated files were regrouped by the specific language, each in one line followed by another in the same way to vocabulary

²<https://bitbucket.org/nanzs/rosettacodedata/>

³<https://github.com/acmeism/RosettaCodeData>

compatible with the format of the specific programming language as in the following table:

| Corpus | Lines | Words | Tokens | Unknowns |
|----------|-------|--------|--------|----------|
| train.cp | 4008 | 390967 | 394975 | 0.0% |
| train.jv | 4008 | 391611 | 395619 | 0.0% |
| test.cp | 700 | 67312 | 68012 | 2.88% |
| test.jv | 700 | 67562 | 68262 | 2.87% |
| valid.cp | 700 | 65781 | 66481 | 2.65% |
| valid.jv | 700 | 69192 | 69892 | 2.53% |

Table 4.2: Dataset splitted on training, testing and validation corpus

4.1.4 Evaluation

For evaluation, we rely on bleu assessment (bilingual evaluation under study) that is an algorithm to evaluate the quality of text that has been automatically translated from one natural language to another. Quality is the match between machine production and human production: "The closer machine translation to professional human translation, the better" is the central idea behind BLEU, which was one of the first metrics to claim a high relationship with human quality judgments, and remains one of the most popular automated and inexpensive metrics.⁴

For our experiences of our training data, the training takes a long time, especially between the epoch and the next, knowing that our begining starts from the process of preparing data in advance on a local computer, and due to the requirements of training equipment, we have to move to work on Google's cloud computer, Google Colab due to the appropriate advantages of our mission, yet it takes us a long time and still, and in the following table we present its results:

| | Cpp→Java | Java→Cpp |
|------|----------|----------|
| BLEU | 45.90% | 43.19 % |

Table 4.3: BLEU score evaluation

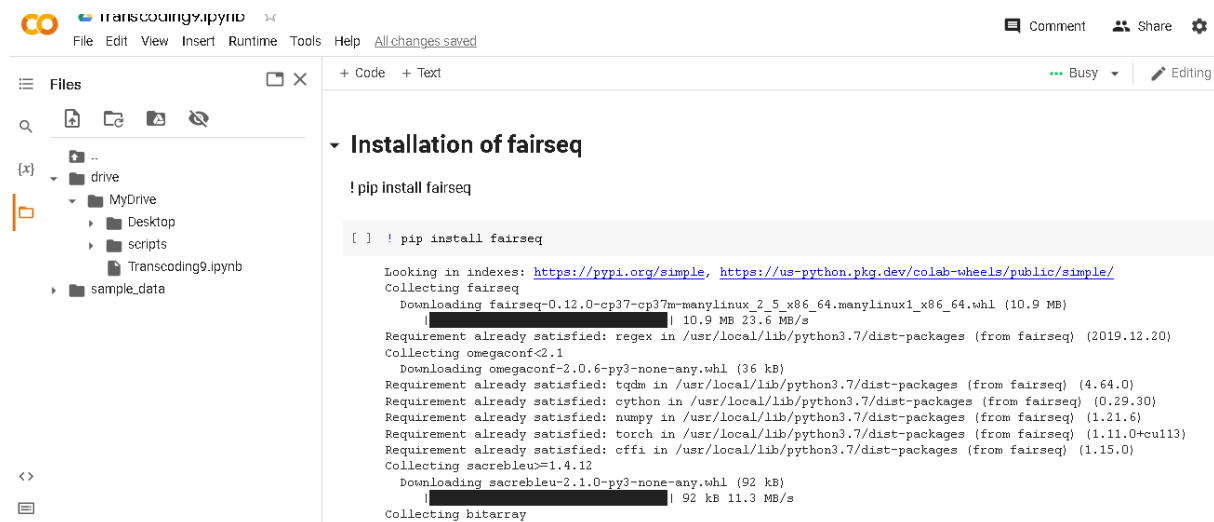
⁴<https://en.wikipedia.org/wiki/BLEU>

4.2 Execution of transformations on the pre-prepared data in Colab

1. Installation of fairseq:

Google Colab's environment is like the entire Linux environment with Python programming language installation, so all Linux commands and Python instructions are already in place, so we apply everything to Jupyter.

In this step, we start by installing fairseq within google colab, and this is to download all the libraries and orders needed to run the scripts with the data, to perform all the processes, exercises, tests and validations, otherwise nothing will work, since working on colab the process is very fast up to less than half a minute or a few seconds, and so far no data has been worked out, so that this step is one of the stages of pre-preparation on which the next steps depend depending on the next steps depending on Order. By pressing the cell button or through the keyboard by pressing Ctrl+Enter which is shown as in the following picture:



```
transcoding9.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share

Files
drive
MyDrive
Desktop
scripts
Transcoding9.ipynb
sample_data

Installation of fairseq
! pip install fairseq

[ ] ! pip install fairseq

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting fairseq
  Downloading fairseq-0.12.0-cp37-cp37m-manylinux_2_5_x86_64.whl (10.9 MB)
    |████████████████████████████████████████████████████████████████████████████████| 10.9 MB 23.6 MB/s
Requirement already satisfied: regex in /usr/local/lib/python3.7/dist-packages (from fairseq) (2019.12.20)
Collecting omegaconf<2.1
  Downloading omegaconf-2.0.6-py3-none-any.whl (36 kB)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from fairseq) (4.64.0)
Requirement already satisfied: cython in /usr/local/lib/python3.7/dist-packages (from fairseq) (0.29.30)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from fairseq) (1.21.6)
Requirement already satisfied: torch in /usr/local/lib/python3.7/dist-packages (from fairseq) (1.11.0+cu113)
Requirement already satisfied: cffi in /usr/local/lib/python3.7/dist-packages (from fairseq) (1.15.0)
Collecting sacrebleu>=1.4.12
  Downloading sacrebleu-2.1.0-py3-none-any.whl (92 kB)
    |████████████████████████████████████████████████████████████████████████████████| 92 kB 11.3 MB/s
Collecting bitarray
```

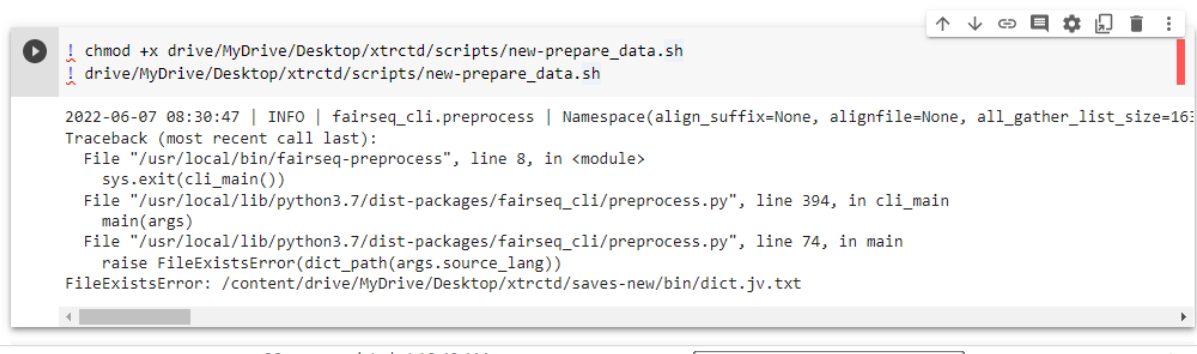
Figure 4.1: Installation of fairseq

2. Preparing of Data For Training:

Before you start, there's a process ahead of you, since training data is stored at Google Drive level, you have to access the storage site, so that when you press a folder or file button, the Drive button, which when you press it shows an order to access it, then a window pops up to choose your Google account, and then the colab access page shows a message at the full end of the process. Then we prepare the data on which we will train the form by pressing the cell button or through the keyboard by pressing Ctrl+Enter as in the following picture:

▾ Preparing of Data For Training

1. ! chmod +x drive/MyDrive/Desktop/xtrctd/scripts/prepare_data.sh
2. ! drive/MyDrive/Desktop/xtrctd/scripts/prepare_data.sh



```
! chmod +x drive/MyDrive/Desktop/xtrctd/scripts/new-prepare_data.sh
! drive/MyDrive/Desktop/xtrctd/scripts/new-prepare_data.sh

2022-06-07 08:30:47 | INFO | fairseq_cli.preprocess | Namespace(align_suffix=None, alignfile=None, all_gather_list_size=16:
Traceback (most recent call last):
  File "/usr/local/bin/fairseq-preprocess", line 8, in <module>
    sys.exit(cli_main())
  File "/usr/local/lib/python3.7/dist-packages/fairseq_cli/preprocess.py", line 394, in cli_main
    main(args)
  File "/usr/local/lib/python3.7/dist-packages/fairseq_cli/preprocess.py", line 74, in main
    raise FileExistsError(dict_path(args.source_lang))
FileExistsError: /content/drive/MyDrive/Desktop/xtrctd/saves-new/bin/dict.jv.txt
```

Figure 4.2: Preparing of Data For Training

3. Training Data on beginning:

As we read from the title, the step begins like this because colab's interactive environment reserves you several hardware and storage resources, they each time you monitor your presence with it to make sure you are not a machine, so we also have to monitor so that there is no interruption in the workflow, or interruption with storage resources, we start from this step we need a place to store data and work on it, because even this environment is a temporary reserved place, as you notice fairseq we have to install it Every time you break, here too we press the cell button to do the order, as in the following picture:

▾ Training Data:

1. On Beginning


- ! chmod +x drive/MyDrive/Desktop/xtrctd/scripts/train.sh
- ! drive/MyDrive/Desktop/xtrctd/scripts/train.sh

```
[ ] ! chmod +x drive/MyDrive/Desktop/xtrctd/scripts/new-train.sh
! drive/MyDrive/Desktop/xtrctd/scripts/new-train.sh
      (out_proj): Linear(in_features=512, out_features=512, bias=True)
    )
  (encoder_attn_layer_norm): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
  (fc1): Linear(in_features=512, out_features=4096, bias=True)
  (fc2): Linear(in_features=4096, out_features=512, bias=True)
  (final_layer_norm): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
)
)
(output_projection): Linear(in_features=512, out_features=9232, bias=False)
)
2022-06-07 08:31:07 | INFO | fairseq_cli.train | task: translation (TranslationTask)
2022-06-07 08:31:07 | INFO | fairseq_cli.train | model: transformer_wmt_en_de_big (TransformerModel)
```

Figure 4.3: Training Data on beginning

4. Training Data on Interruption after beginning:

As we say because colab's interactive environment reserves you several hardware and storage resources, they each time you monitor your presence with it to make sure you are not a machine, so we also have to monitor so that there is no interruption in the workflow, or interruption with storage resources, we start from this step we need a place to store data and work on it, because even this environment is a temporary reserved place, as you notice fairseq we have to install it every time when disconnected, here We also press the cell button to continue training, as in the following picture:



The top part of the image shows a terminal window with the following output:

```

2022-06-07 08:31:07 | INFO | fairseq.trainer | no existing checkpoint found /content/drive/MyD
2022-06-07 08:31:07 | INFO | fairseq.trainer | loading train data for epoch 1
2022-06-07 08:31:09 | INFO | fairseq.data.data_utils | loaded 6868 examples from: /content/drive/MyDrive/Desktop/xtrctd/
2022-06-07 08:31:10 | INFO | fairseq.data.data_utils | loaded 6868 examples from: /content/drive/MyDrive/Desktop/xtrctd/
2022-06-07 08:31:10 | INFO | fairseq.tasks.translation | /content/drive/MyDrive/Desktop/xtrctd/save_new/bin/train_iv

```

The bottom part of the image shows a code editor with the following code:

```

[ ] ! chmod +x drive/MyDrive/Desktop/xtrctd/scripts/new-train_continue.sh
! drive/MyDrive/Desktop/xtrctd/scripts/new-train_continue.sh

(encoder_attn_layer_norm): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
(fc1): Linear(in_features=512, out_features=4096, bias=True)
(fc2): Linear(in_features=4096, out_features=512, bias=True)
(final_layer_norm): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
)
(1): TransformerDecoderLayer(
  (dropout_module): FairseqDropout()
  (self_attn): MultiheadAttention(

```

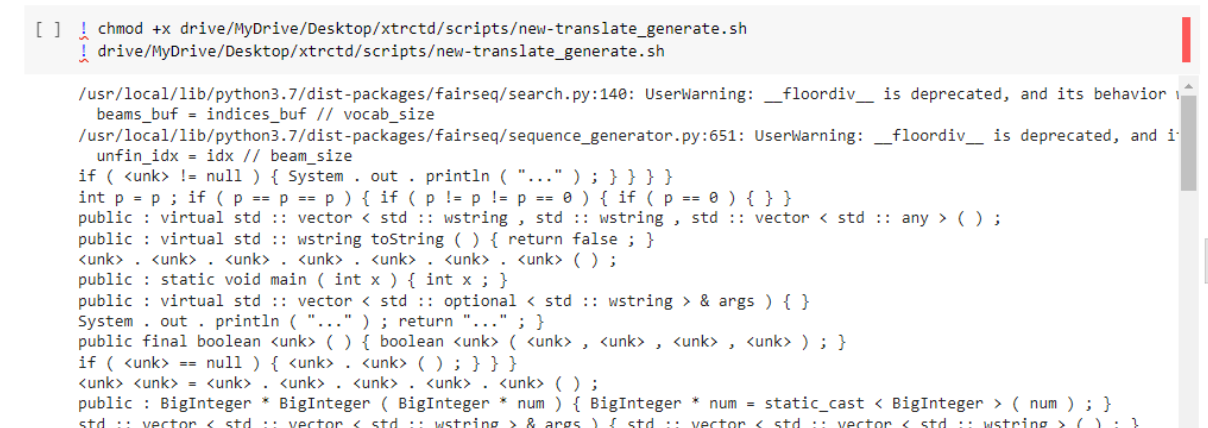
Figure 4.4: Training Data on Interruption after beginning

5. Testing:

In this step, we examine beyond training for data prepared for this, so that as the process progresses, we see target language lines line by line, up to the last line of data applied to them. This step is as in the following picture:

Testing

1. ! chmod +x drive/MyDrive/Desktop/xtrctd/scripts/translate_generate.sh
2. ! drive/MyDrive/Desktop/xtrctd/scripts/translate_generate.sh



The top part of the image shows a terminal window with the following output:

```

[ ] ! chmod +x drive/MyDrive/Desktop/xtrctd/scripts/new-translate_generate.sh
! drive/MyDrive/Desktop/xtrctd/scripts/new-translate_generate.sh

/usr/local/lib/python3.7/dist-packages/fairseq/search.py:140: UserWarning: __floordiv__ is deprecated, and its behavior
beams_buf = indices_buf // vocab_size
/usr/local/lib/python3.7/dist-packages/fairseq/sequence_generator.py:651: UserWarning: __floordiv__ is deprecated, and i
unfin_idx = idx // beam_size
if ( <unk> != null ) { System . out . println ( "... " ) ; } } }
int p = p ; if ( p == p == p ) { if ( p != p != p == 0 ) { if ( p == 0 ) { } } }
public : virtual std :: vector < std :: wstring , std :: wstring , std :: vector < std :: any > ( ) ;
public : virtual std :: wstring toString ( ) { return false ; }
<unk> . <unk> . <unk> . <unk> . <unk> . <unk> . <unk> ( ) ;
public : static void main ( int x ) { int x ; }
public : virtual std :: vector < std :: optional < std :: wstring > & args ) { }
System . out . println ( "... " ) ; return "... " ; }
public final boolean <unk> ( ) { boolean <unk> ( <unk> , <unk> , <unk> , <unk> ) ; }
if ( <unk> == null ) { <unk> . <unk> ( ) ; } } }
<unk> <unk> = <unk> . <unk> . <unk> . <unk> . <unk> ( ) ;
public : BigInteger * BigInteger ( BigInteger * num = static_cast < BigInteger > ( num ) ; }
std :: vector < std :: vector < std :: wstring > & args ) { std :: vector < std :: vector < std :: wstring > ( ) : }

```

The bottom part of the image shows a code editor with the following code:

```

[ ] ! chmod +x drive/MyDrive/Desktop/xtrctd/scripts/new-translate_generate.sh
! drive/MyDrive/Desktop/xtrctd/scripts/new-translate_generate.sh

/usr/local/lib/python3.7/dist-packages/fairseq/search.py:140: UserWarning: __floordiv__ is deprecated, and its behavior
beams_buf = indices_buf // vocab_size
/usr/local/lib/python3.7/dist-packages/fairseq/sequence_generator.py:651: UserWarning: __floordiv__ is deprecated, and i
unfin_idx = idx // beam_size
if ( <unk> != null ) { System . out . println ( "... " ) ; } } }
int p = p ; if ( p == p == p ) { if ( p != p != p == 0 ) { if ( p == 0 ) { } } }
public : virtual std :: vector < std :: wstring , std :: wstring , std :: vector < std :: any > ( ) ;
public : virtual std :: wstring toString ( ) { return false ; }
<unk> . <unk> . <unk> . <unk> . <unk> . <unk> . <unk> ( ) ;
public : static void main ( int x ) { int x ; }
public : virtual std :: vector < std :: optional < std :: wstring > & args ) { }
System . out . println ( "... " ) ; return "... " ; }
public final boolean <unk> ( ) { boolean <unk> ( <unk> , <unk> , <unk> , <unk> ) ; }
if ( <unk> == null ) { <unk> . <unk> ( ) ; } } }
<unk> <unk> = <unk> . <unk> . <unk> . <unk> . <unk> ( ) ;
public : BigInteger * BigInteger ( BigInteger * num = static_cast < BigInteger > ( num ) ; }
std :: vector < std :: vector < std :: wstring > & args ) { std :: vector < std :: vector < std :: wstring > ( ) : }

```

Figure 4.5: Testing and generation of results of the translation test

The result of the quality obtained on the BLEU scale, located in sub-folders of the data storage site, as in the following picture:



Figure 4.6: Results of the translation test

6. Add new Data without parallel translation:

In this step, we enter single-source data to output the target, which can also be used to increase machine data if it does not have sufficient training data for one of the two languages, in this way the background translation mentioned is done to create artificial data for the NLP task. So we can get more training for model training especially for NLP tasks and low-resource languages. Of course, let's not forget that in this step we're transforming from a source language into a target language, and that's when you enter new data to convert it, which we want. The following picture shows:

- Add new Data without parallel translation

```
1. ! chmod +x drive/MyDrive/Desktop/xtrctd/scripts/translate_interactive.sh.sh
2. ! drive/MyDrive/Desktop/xtrctd/scripts/translate_interactive.sh.sh > Result
```

```
[ ] ! chmod +x drive/MyDrive/Desktop/xtrctd/scripts/new-translate_interactive.sh
! drive/MyDrive/Desktop/xtrctd/scripts/new-translate_interactive.sh > /content/drive/MyDrive/Desktop/Result

/usr/local/lib/python3.7/dist-packages/fairseq/search.py:140: UserWarning: __floordiv__ is deprecated, and its behavior will
beams_buf = indices_buf // vocab_size
/usr/local/lib/python3.7/dist-packages/fairseq/sequence_generator.py:651: UserWarning: __floordiv__ is deprecated, and its
unfin_idx = idx // beam_size
```

Figure 4.7: Add new Data without parallel translation

Chapter 5

Conclusion

In this note, we are trying to inform one of the most important fields and applied areas interested in the processing of natural languages by computer, namely, the field of machine translation and specifically the conversion of source codes, and through our study we have found that researchers in this field sought to dispense with the human translator and replace it completely with computers, which is called transcoding, the translated machine is an important means within the reach of the professional programmer to counter the bets of the current information age, translation of codes has educational, economic and security benefits, including:

- Rosettacode, as an example, shows how to solve a problem from one programming language to another to show similarities or differences, and this site is growing more in terms of sources.
- Dissemination of information in all fields and multiple languages in response to the requirements of the era characterized by the flow of information that humanity has not been able to contain, especially in the translation of code.
- Update a language from an old version to a new version such as python2 to python3 especially when revising and reviewing.
- The huge amount of data requires the use of translation to profit time and reduce financial costs.

- code translation is the best way to correct errors and avoid harmful vulnerabilities.

Through this project, we want to move from natural machine translation to code machine translation, reach greater successes and solve all kinds of ambiguities and software problems, which requires a huge amount of data that requires a high-speed, powerful hardware structure and high quality quality due to the time required for training.

We hope to benefit from our findings and continue to implement and develop this project to achieve greater successes in the field of machine translation of codes in all fields of scientific, research, economic and security.

Appendices

Appendix A

Statistics

Rosetta Code: Popular Programming Languages

Generated: 2022-06-12T11:00:03

| # | Count | Name | Task not implemented in Racket | # | Phix | Wren | Julia | Go | Raku | Perl | Nim | Python | C | J | REXX | Java |
|----|-------|-------------|--|----|------|------|-------|----|------|------|-----|--------|---|---|------|------|
| 1 | 1540 | Phix | | | | | | | | | | | | | | |
| 2 | 1532 | Wren | 15 puzzle game in 3D | 3 | : | : | | : | | | | | | | | |
| 3 | 1508 | Julia | 16 puzzle game | 8 | : | : | : | : | : | : | : | | | | : | |
| 4 | 1494 | Go | 3d turtle graphics | 3 | : | : | | | | | | | | : | | |
| 5 | 1488 | Raku | ADFGVX cipher | 8 | : | : | : | : | : | : | : | : | | | | |
| 6 | 1449 | Perl | AVL tree | 13 | : | : | : | : | : | : | : | : | : | : | : | : |
| 7 | 1402 | Nim | Abelian sandpile model | 14 | : | : | : | : | : | : | : | : | : | : | : | : |
| 8 | 1382 | Python | Abelian sandpile model/Identity | 13 | : | : | : | : | : | : | : | : | : | : | : | : |
| 9 | 1204 | C | Achilles numbers | 9 | : | : | : | : | : | : | | | | : | | |
| 10 | 1152 | J | Active Directory/Search for a user | 12 | : | : | : | : | : | : | | : | : | | : | : |
| = | 1152 | REXX | Add a variable to a class instance at runtime | 12 | : | : | : | : | : | : | : | : | | : | | |
| 12 | 1134 | Java | Air mass | 11 | : | : | : | : | : | : | : | : | : | : | : | : |
| 13 | 1133 | Kotlin | Almkvist-Guillera formula for pi | 13 | : | : | : | : | : | : | : | : | : | : | : | : |
| 14 | 1130 | Mathematica | Alternated words | 14 | : | : | : | : | : | : | : | : | : | : | : | : |
| 15 | 1128 | Haskell | Append numbers at same position in strings | 10 | : | : | : | : | : | : | | : | : | | | |
| 16 | 1119 | C++ | Arithmetic coding/As a generalized change of radix | 12 | : | : | : | : | : | : | : | : | | : | | : |
| 17 | 1097 | Ruby | | | | | | | | | | | | | | |
| 18 | 1094 | Racket | Ascending primes | 8 | : | : | : | : | : | : | | | | : | | |

Popular Programming Languages on RosettaCode ¹

¹<http://timb.net/popular-languages.html>

```

1 UltraSearch Report, 12/06/2022 19:42
2
3
4
5 Results 929 objects found (995,69 KB)
6
7 Name Folder Path Last Modified
8 -----
9 9-billion-names-of-god-the-integer-1... D:\...ttaCodeData-master\Task\9-billion-names-of-God-the-integer\C++\ 18/02/2020 08:21
10 9-billion-names-of-god-the-integer-2... D:\...ttaCodeData-master\Task\9-billion-names-of-God-the-integer\C++\ 18/02/2020 08:21
11 9-billion-names-of-god-the-integer-3... D:\...ttaCodeData-master\Task\9-billion-names-of-God-the-integer\C++\ 18/02/2020 08:21
12 9-billion-names-of-god-the-integer-4... D:\...ttaCodeData-master\Task\9-billion-names-of-God-the-integer\C++\ 18/02/2020 08:21
13 24-game.cpp D:\RosettaCodeData-master\Task\24-game\C++\ 18/02/2020 08:21
14 24-game-solve.cpp D:\RosettaCodeData-master\Task\24-game-Solve\C++\ 18/02/2020 08:21
15 99-bottles-of-beer-1.cpp D:\RosettaCodeData-master\Task\99-Bottles-of-Beer\C++\ 18/02/2020 08:21
16 99-bottles-of-beer-2.cpp D:\RosettaCodeData-master\Task\99-Bottles-of-Beer\C++\ 18/02/2020 08:21
17 99-bottles-of-beer-3.cpp D:\RosettaCodeData-master\Task\99-Bottles-of-Beer\C++\ 18/02/2020 08:21
18 99-bottles-of-beer-4.cpp D:\RosettaCodeData-master\Task\99-Bottles-of-Beer\C++\ 18/02/2020 08:21
19 99-bottles-of-beer-5.cpp D:\RosettaCodeData-master\Task\99-Bottles-of-Beer\C++\ 18/02/2020 08:21
20 100-doors-1.cpp D:\RosettaCodeData-master\Task\100-doors\C++\ 18/02/2020 08:21
21 100-doors-2.cpp D:\RosettaCodeData-master\Task\100-doors\C++\ 18/02/2020 08:21
22 100-doors-3.cpp D:\RosettaCodeData-master\Task\100-doors\C++\ 18/02/2020 08:21
23 100-doors-4.cpp D:\RosettaCodeData-master\Task\100-doors\C++\ 18/02/2020 08:21
24 a+b-1.cpp D:\RosettaCodeData-master\Task\A+B\C++\ 18/02/2020 08:21
25 a+b-2.cpp D:\RosettaCodeData-master\Task\A+B\C++\ 18/02/2020 08:21
26 abc-problem.cpp D:\RosettaCodeData-master\Task\ABC-Problem\C++\ 18/02/2020 08:21
27 abstract-type.cpp D:\RosettaCodeData-master\Task\Abstract-type\C++\ 18/02/2020 08:21
28 abundant,-deficient-and-perfect-numb... D:\...ask\Abundant,-deficient-and-perfect-number-classifications\C++\ 18/02/2020 08:21
29 accumulator-factory-1.cpp D:\RosettaCodeData-master\Task\Accumulator-factory\C++\ 18/02/2020 08:21
30 accumulator-factory-2.cpp D:\RosettaCodeData-master\Task\Accumulator-factory\C++\ 18/02/2020 08:21

```

CPP Data count 2020 archive downloaded and counted on pc by ultrasearch program ²

```

1 UltraSearch Report, 12/06/2022 19:43
2
3
4
5 Results 1 051 objects found (1,16 MB)
6
7 Name Folder Path Last Modified
8 -----
9 9-billion-names-of-god-the-integer.j... D:\...ttaCodeData-master\Task\9-billion-names-of-God-the-integer\Java\ 18/02/2020 08:21
10 24-game.java D:\RosettaCodeData-master\Task\24-game\Java\ 18/02/2020 08:21
11 24-game-solve.java D:\RosettaCodeData-master\Task\24-game-Solve\Java\ 18/02/2020 08:21
12 99-bottles-of-beer-1.java D:\RosettaCodeData-master\Task\99-Bottles-of-Beer\Java\ 18/02/2020 08:21
13 99-bottles-of-beer-2.java D:\RosettaCodeData-master\Task\99-Bottles-of-Beer\Java\ 18/02/2020 08:21
14 99-bottles-of-beer-3.java D:\RosettaCodeData-master\Task\99-Bottles-of-Beer\Java\ 18/02/2020 08:21
15 99-bottles-of-beer-4.java D:\RosettaCodeData-master\Task\99-Bottles-of-Beer\Java\ 18/02/2020 08:21
16 100-doors-1.java D:\RosettaCodeData-master\Task\100-doors\Java\ 18/02/2020 08:21
17 100-doors-2.java D:\RosettaCodeData-master\Task\100-doors\Java\ 18/02/2020 08:21
18 100-doors-3.java D:\RosettaCodeData-master\Task\100-doors\Java\ 18/02/2020 08:21
19 100-doors-4.java D:\RosettaCodeData-master\Task\100-doors\Java\ 18/02/2020 08:21
20 a+b-1.java D:\RosettaCodeData-master\Task\A+B\Java\ 18/02/2020 08:21
21 a+b-2.java D:\RosettaCodeData-master\Task\A+B\Java\ 18/02/2020 08:21
22 a+b-3.java D:\RosettaCodeData-master\Task\A+B\Java\ 18/02/2020 08:21
23 a+b-4.java D:\RosettaCodeData-master\Task\A+B\Java\ 18/02/2020 08:21
24 abc-problem.java D:\RosettaCodeData-master\Task\ABC-Problem\Java\ 18/02/2020 08:21
25 abstract-type-1.java D:\RosettaCodeData-master\Task\Abstract-type\Java\ 18/02/2020 08:21
26 abstract-type-2.java D:\RosettaCodeData-master\Task\Abstract-type\Java\ 18/02/2020 08:21
27 abundant,-deficient-and-perfect-numb... D:\...sk\Abundant,-deficient-and-perfect-number-classifications\Java\ 18/02/2020 08:21
28 accumulator-factory-1.java D:\RosettaCodeData-master\Task\Accumulator-factory\Java\ 18/02/2020 08:21
29 accumulator-factory-2.java D:\RosettaCodeData-master\Task\Accumulator-factory\Java\ 18/02/2020 08:21
30 accumulator-factory-3.java D:\RosettaCodeData-master\Task\Accumulator-factory\Java\ 18/02/2020 08:21

```

JAVA Data count 2020 archive downloaded and counted on pc by ultrasearch program ³

²<https://github.com/acmeism/RosettaCodeData/tree/master/Task>

³<https://github.com/acmeism/RosettaCodeData/tree/master/Task>

```

1 UltraSearch Report, 12/06/2022 19:33
2
3
4
5 Results 744 objects found (707,98 KB)
6
7 Name Folder Path Last Modified
8 -----
9 9-billion-names-of-god-the-integer-1... D:\...edata-05dfbfc3415f\Task\9-billion-names-of-God-the-integer\C++\ 17/09/2014 12:07
10 9-billion-names-of-god-the-integer-2... D:\...edata-05dfbfc3415f\Task\9-billion-names-of-God-the-integer\C++\ 17/09/2014 12:07
11 9-billion-names-of-god-the-integer-3... D:\...edata-05dfbfc3415f\Task\9-billion-names-of-God-the-integer\C++\ 17/09/2014 12:07
12 9-billion-names-of-god-the-integer-4... D:\...edata-05dfbfc3415f\Task\9-billion-names-of-God-the-integer\C++\ 17/09/2014 12:07
13 24-game.cpp D:\nanzs-rosettacodedata-05dfbfc3415f\Task\24-game\C++\ 17/09/2014 12:07
14 99-bottles-of-beer-1.cpp D:\nanzs-rosettacodedata-05dfbfc3415f\Task\99-Bottles-of-Beer\C++\ 17/09/2014 12:07
15 99-bottles-of-beer-2.cpp D:\nanzs-rosettacodedata-05dfbfc3415f\Task\99-Bottles-of-Beer\C++\ 17/09/2014 12:07
16 99-bottles-of-beer-3.cpp D:\nanzs-rosettacodedata-05dfbfc3415f\Task\99-Bottles-of-Beer\C++\ 17/09/2014 12:07
17 99-bottles-of-beer-4.cpp D:\nanzs-rosettacodedata-05dfbfc3415f\Task\99-Bottles-of-Beer\C++\ 17/09/2014 12:07
18 99-bottles-of-beer-5.cpp D:\nanzs-rosettacodedata-05dfbfc3415f\Task\99-Bottles-of-Beer\C++\ 17/09/2014 12:07
19 100-doors-1.cpp D:\nanzs-rosettacodedata-05dfbfc3415f\Task\100-doors\C++\ 17/09/2014 12:07
20 100-doors-2.cpp D:\nanzs-rosettacodedata-05dfbfc3415f\Task\100-doors\C++\ 17/09/2014 12:07
21 100-doors-3.cpp D:\nanzs-rosettacodedata-05dfbfc3415f\Task\100-doors\C++\ 17/09/2014 12:07
22 a+b-1.cpp D:\nanzs-rosettacodedata-05dfbfc3415f\Task\A+B\C++\ 17/09/2014 12:07
23 a+b-2.cpp D:\nanzs-rosettacodedata-05dfbfc3415f\Task\A+B\C++\ 17/09/2014 12:07
24 abc-problem.cpp D:\nanzs-rosettacodedata-05dfbfc3415f\Task\ABC-Problem\C++\ 17/09/2014 12:07
25 abstract-type.cpp D:\nanzs-rosettacodedata-05dfbfc3415f\Task\Abstract-type\C++\ 17/09/2014 12:07
26 accumulator-factory-1.cpp D:\nanzs-rosettacodedata-05dfbfc3415f\Task\Accumulator-factory\C++\ 17/09/2014 12:07
27 accumulator-factory-2.cpp D:\nanzs-rosettacodedata-05dfbfc3415f\Task\Accumulator-factory\C++\ 17/09/2014 12:07
28 ackermann-function.cpp D:\nanzs-rosettacodedata-05dfbfc3415f\Task\Ackermann-function\C++\ 17/09/2014 12:07
29 anagrams.cpp D:\nanzs-rosettacodedata-05dfbfc3415f\Task\Anagrams\C++\ 17/09/2014 12:07
30

```

CPP Data count 2014 archive downloaded and counted on pc by ultrasearch program ⁴

```

1 UltraSearch Report, 12/06/2022 19:36
2
3
4
5 Results 842 objects found (805,50 KB)
6
7 Name Folder Path Last Modified
8 -----
9 string-interpolation--included-.java D:\...ttacodedata-05dfbfc3415f\Task\String-interpolation--included\Java\ 17/09/2014 12:07
10 knapsack-problem-continuous-3.java D:\...ttacodedata-05dfbfc3415f\Task\Knapsack-problem-Continuous\Java\ 17/09/2014 12:07
11 knapsack-problem-continuous-2.java D:\...ttacodedata-05dfbfc3415f\Task\Knapsack-problem-Continuous\Java\ 17/09/2014 12:07
12 knapsack-problem-continuous-1.java D:\...ttacodedata-05dfbfc3415f\Task\Knapsack-problem-Continuous\Java\ 17/09/2014 12:07
13 arithmetic-geometric-mean.java D:\...ettacodedata-05dfbfc3415f\Task\Arithmetic-geometric-mean\Java\ 17/09/2014 12:07
14 pernicious-numbers.java D:\nanzs-rosettacodedata-05dfbfc3415f\Task\Pernicious-numbers\Java\ 17/09/2014 12:07
15 box-the-compass.java D:\nanzs-rosettacodedata-05dfbfc3415f\Task\Box-the-compass\Java\ 17/09/2014 12:07
16 knights-tour.java D:\nanzs-rosettacodedata-05dfbfc3415f\Task\Knights-tour\Java\ 17/09/2014 12:07
17 terminal-control-display-an-extended... D:\...3415f\Task\Terminal-control-Display-an-extended-character\Java\ 17/09/2014 12:07
18 hough-transform.java D:\nanzs-rosettacodedata-05dfbfc3415f\Task\Hough-transform\Java\ 17/09/2014 12:07
19 character-matching.java D:\nanzs-rosettacodedata-05dfbfc3415f\Task\Character-matching\Java\ 17/09/2014 12:07
20 deal-cards-for-freecell.java D:\...rosettacodedata-05dfbfc3415f\Task\Deal-cards-for-FreeCell\Java\ 17/09/2014 12:07
21 pi.java D:\nanzs-rosettacodedata-05dfbfc3415f\Task\Pi\Java\ 17/09/2014 12:07
22 animation.java D:\nanzs-rosettacodedata-05dfbfc3415f\Task\Animation\Java\ 17/09/2014 12:07
23 fibonacci-n-step-number-sequences.java D:\...edata-05dfbfc3415f\Task\Fibonacci-n-step-number-sequences\Java\ 17/09/2014 12:07
24 levenshtein-distance-2.java D:\...zs-rosettacodedata-05dfbfc3415f\Task\Levenshtein-distance\Java\ 17/09/2014 12:07
25 levenshtein-distance-1.java D:\...zs-rosettacodedata-05dfbfc3415f\Task\Levenshtein-distance\Java\ 17/09/2014 12:07
26 greatest-subsequential-sum-2.java D:\...ettacodedata-05dfbfc3415f\Task\Greatest-subsequential-sum\Java\ 17/09/2014 12:07
27 greatest-subsequential-sum-1.java D:\...ettacodedata-05dfbfc3415f\Task\Greatest-subsequential-sum\Java\ 17/09/2014 12:07
28 test-a-function-2.java D:\nanzs-rosettacodedata-05dfbfc3415f\Task\Test-a-function\Java\ 17/09/2014 12:07
29 test-a-function-1.java D:\nanzs-rosettacodedata-05dfbfc3415f\Task\Test-a-function\Java\ 17/09/2014 12:07
30 galton-box-animation.java D:\...zs-rosettacodedata-05dfbfc3415f\Task\Galton-box-animation\Java\ 17/09/2014 12:07

```

JAVA Data count 2014 archive downloaded and counted on pc by ultrasearch program ⁵

⁴<https://bitbucket.org/nanzs/rosettacodedata/src/master/Task/>

⁵<https://bitbucket.org/nanzs/rosettacodedata/src/master/Task/>

Bibliography

- [1] F. A. PK, “What is artificial intelligence?,” “*Success is no accident. It is hard work, perseverance, learning, studying, sacrifice and most of all, love of what you are doing or learning to do*”., p. 65, 1984.
- [2] J. Brownlee, “Deep learning for natural language processing,” *Machine Learning Mystery, Vermont, Australia*, vol. 322, 2017.
- [3] E. D. Liddy, “Natural language processing,” 2001.
- [4] K.-H. Yu, A. L. Beam, and I. S. Kohane, “Artificial intelligence in health-care,” *Nature biomedical engineering*, vol. 2, no. 10, pp. 719–731, 2018.
- [5] M.-A. Lachaux, B. Roziere, L. Chanussot, and G. Lample, “Unsupervised translation of programming languages,” *arXiv preprint arXiv:2006.03511*, 2020.
- [6] J. Slocum, “A survey of machine translation: Its history, current status and future prospects,” *Computational linguistics*, vol. 11, no. 1, pp. 1–17, 1985.
- [7] J. Hutchins, “The history of machine translation in a nutshell,” *Retrieved December*, vol. 20, no. 2009, pp. 1–1, 2005.
- [8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.
- [9] D. Kenny and S. Doherty, “Statistical machine translation in the translation curriculum: overcoming obstacles and empowering translators,” *The Interpreter and translator trainer*, vol. 8, no. 2, pp. 276–294, 2014.
- [10] B. Yegnanarayana, *Artificial neural networks*. 2009.

-
- [11] S. R. Sharma, Vidushi and A. Dev., “A comprehensive study of artificial neural networks.,” *International Journal of Advanced research in computer science and software engineering* 2.10, 2012.
- [12] e. a. Chen, Mingzhe, “Machine learning for wireless networks with artificial intelligence: A tutorial on neural networks.,” *arXiv preprint arXiv:1710.02913* 9, 2012.
- [13] P. Koehn, “Neural machine translation,” *arXiv preprint arXiv:1709.07809*, 2017.
- [14] K. C. Bahdanau, Dzmitry and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *arXiv preprint arXiv:1409.0473*, 2014.