VIETNAM NATIONAL UNIVERSITY HOCHIMINH CITY
UNIVERSITY OF INFORMATION TECHNOLOGY
ADVANCED PROGRAM IN INFORMATION SYSTEMS

NGUYỄN VIỆT NAM

CHATBOT APPLICATION FOR ANSWERING QUESTIONS RELATED TO RULES AND REGULATIONS OF UNIVERSITY OF INFORMATION TECHNOLOGY

BACHELOR OF ENGINEERING IN INFORMATION SYSTEMS

HO CHI MINH CITY, 2018

NATIONAL UNIVERSITY HOCHIMINH CITY
UNIVERSITY OF INFORMATION TECHNOLOGY
ADVANCED PROGRAM IN INFORMATION SYSTEMS

NGUYỄN VIỆT NAM – 14520560

CHATBOT APPLICATION FOR ANSWERING QUESTIONS RELATED TO RULES AND REGULATIONS OF UNIVERSITY OF INFORMATION TECHNOLOGY

BACHELOR OF ENGINEERING IN INFORMATION SYSTEMS

THESIS ADVISOR

DR. NGÔ ĐỨC THÀNH
M.SC. NGUYỄN VINH TIỆP

HO CHI MINH CITY, 2018

ASSESSMENT COMMITTEE

The Assessment Committee is established under the Decision . ..., date ....by Rector of the University of Information Technology.

1 .... - Chairman.

2 .... - Secretary.

3 .... - Member

4 .... - Member.

UNIVERSITY OF INFORMATION TECHNOLOGY

ADVANCED PROGRAM

IN INFORMATION SYSTEMS

THESIS PROPOSAL

THESIS TITLE:

CHATBOT APPLICATION FOR ANSWERING QUESTIONS RELATED TO RULES AND REGULATIONS OF UNIVERSITY OF INFORMATION TECHNOLOGY

Advisors: Dr. Ngô Đức Thành
M.Sc. Nguyễn Vinh Tiệp

Duration: September 03, 2018 to January 21, 2019

Student: Nguyễn Việt Nam – 14520560

Contents:

1. **Descriptions**

At University of Information Technology and other universities, there are many problems arise while answering questions related to rules and regulations, which lead to late and unreliable answers. One of the most frequent problems come from the former process of making contact: phone call and email which has a long response time. On the other hand, the documents of rules and regulations of UIT are updated frequently but those new documents are separated from the original one, lead to difficulty when finding answers. Therefore, we decide to build a chatbot to help to solve the above problem, providing a new way to get quick and trustable answers for questions related to rules and regulations of UIT.

2. **Scope**

- Chatbot development.
- Natural language processing.
- Question answering
- Machine reading comprehension.

3. Objectives

- Understand core concepts and algorithms using in natural language processing and best practices to build chatbot.
- Build a Chatbot that capable to answer questions related to rules and regulations of UIT in Vietnamese, which is unprecedented in previous studies in Vietnam.
- Publish the Chatbot to support UIT staffs for answering questions related to rules and regulations.

4. Methodologies

- Use Sequence to Sequence model with word embeddings as the main approach of the thesis.
- Use Tensorflow with Python as the main library to write the chatbot model. Other libraries may be used for word embeddings and data preprocessing.
- Build server for chatbot using Django as webhook for later coupling with social network applications.

5. Expected results

- Understand fundamental algorithms and technologies using in chatbot.
- Working productively on chatbot development.
- Successfully develop the chatbot that could answer questions related to rules and regulations of UIT.
- Ensure that the chatbot is capable to use in supporting UIT staffs for answering questions related to rules and regulations.

Timeline:

Phase 1 (03/09/2018 - 27/09/2018):

- Study fundamentals of Deep learning and Natural language processing
- Research methodologies, algorithms and technologies for chatbot development.

- Decide the structure for the chatbot.

Phase 2 (28/09/2018 - 13/11/2018):

- Prepare training and testing dataset.
- Implement the chatbot.

Phase 3 (14/11/2018 - 14/12/2018):

- Test and validate the chatbot.
- Optimize and tune the chatbot.
- Publish the chatbot.

Phase 4 (15/12/2018 - 22/01/2019):

- Writing thesis document.
- Preparing for thesis review and defense.

Approved by advisors		**Ho Chi Minh city, 24/09/2018**
Signature of advisor	**Signature of advisor**	**Signature of Student**
Dr. Ngô Đức Thành	**M.Sc. Nguyễn Vinh Tiệp**	**Nguyễn Việt Nam**

Abstract

In the University of Information System, there exists a fact that students couldn't gain fast and accurate answers for questions related to rules and regulations using existing methods. In this thesis, I aim to develop a Vietnamese Question Answering System with reading comprehension ability that can skim and interprets documents to return a fast and accurate answer to the user.

Researches to build a Vietnamese Question Answering system is more challenging than English Question Answering System, especially system using deep learning approaches, since the lack of available Vietnamese resources and tools, and most Vietnamese Question Answering System is based on statistical approaches, or Information Retrieval approach.

From that, I proposed a combination of existing Information Retrieval approaches for quickly skim through documents, and deep learning approaches along with transfer learning for complex text understanding and inference, to build a Vietnamese Question Answering System for rules and regulations of University of Information System. A score of 60% is achieved that posed promising potentials for improvements.

Acknowledgements

First of all, I would like to express my sincere gratitude to my advisors Dr. Ngo Duc Thanh and M.Sc. Nguyen Vinh Tiep for the continuous support of my thesis and related researches, for their patience, motivation, and immense knowledge.

Besides my advisors, I appreciate the help and supports from researchers and fellow students of MMLab, especially Mr. Dang Van Thin, for their insightful feedback and inspiration.

I would like to express my thanks to staffs in Faculty of Information Systems as well as staffs of the University of Information Technology for their guidance and assistance as well as providing the best conditions for me to study in and finish this graduation thesis.

I also appreciate the help and support of VietAI – a non-profit organization for the wonderful talk about question answering and with the help of their GPU, model training and testing become much faster and easier.

Last but not least, I would like to thank my classmates, my friends, and my family for supporting me on my career path, so that I can stand here today with pride. Special thanks go to Tran Tri Nguyen for his wonderful ideas and assistance during the first part of this thesis.

December, 2018

Nguyen Viet Nam – student of AEP 2014

Contents

List of Tables

List of Figures

Abbreviations

IR	Information Retrieval
NLP	Natural Language Processing
QA	Question Answering
UIT	University of Information Technology

Chapter 1

Introduction

1.1 Overview

The idea of *question answering based intelligent assistants* (or *chatbots*) first appeared in the 1960s, but not until recently that *chatbots* have become *popular*, according to the data of Google Trends and Facebook IQ presented in Figure 1.1 and Figure 1.2. Such popularity is the result of *progressive achievements* in *Natural Language Processing* (NLP), *Artificial Intelligence*, and the global presence of text messaging applications. Unlike *traditional systems* where users interact with them through *simple and structured language*, *modern chatbots* allow users to interacts using *natural languages* such as text, speech or even images.

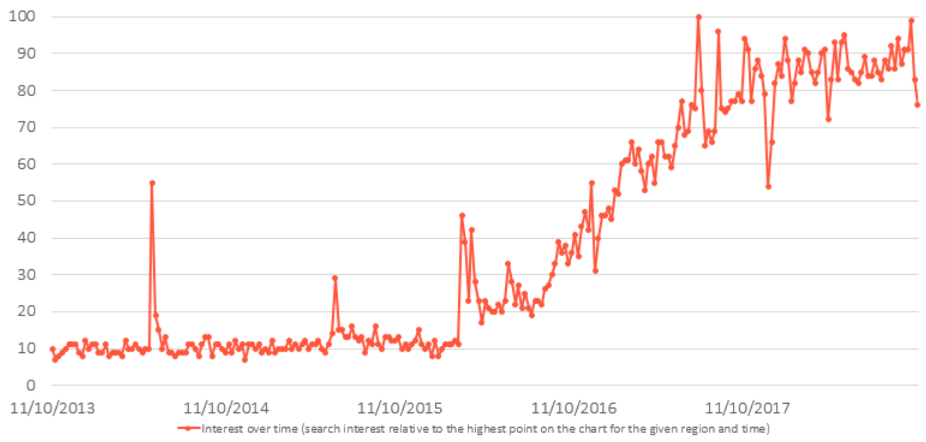


Figure 1.1: Chatbot search count by Google Trend in 2013-2018

Chatbots have many *aspiring and promising applications* as well as *potential benefits* as shown in Figure 1.3, and *customer service* is one field that *chatbots* can play a key role

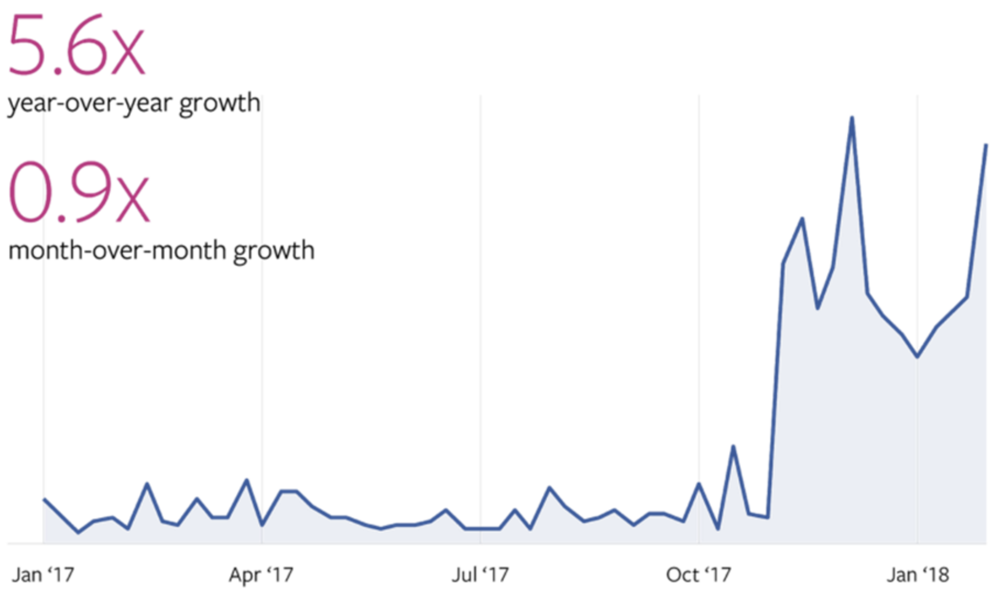


Figure 1.2: Activity of chatbots discussion by Facebook IQ

in. On one hand, with the fast-growing E-commerce market, there is a strong demand for customer service staffs on the field. On the other hand, with more and more people paying attention to experience and service quality nowadays, the *issues* of traditional customer service are becoming obvious, such as limited availability (only office hours) and inefficiency (customers often have to wait).

1.2 Problem Statement

As discussed in the previous section, the *issues* of traditional customer service are becoming conspicuous, and the *same issues* can be seen in *University of Information Technology* (UIT): when a student has *a question* in mind, besides meeting the staff in person, he/she have to *send a proper e-mail* to the university, which *takes days or even weeks* for an answer, not to mention the mail has to be sent to the department that is related to the question. The student can also *look up answer* on the university website, but these documents on the website are “dirty", such that various versions of the same document are presented, and they are barely organized & maintained), which make it *time-consuming* for the student to search for relevant information.

This raises an important question: is it possible to design a *QA system* that can relieve staff

[1]https://www.drift.com/blog/chatbots-report/

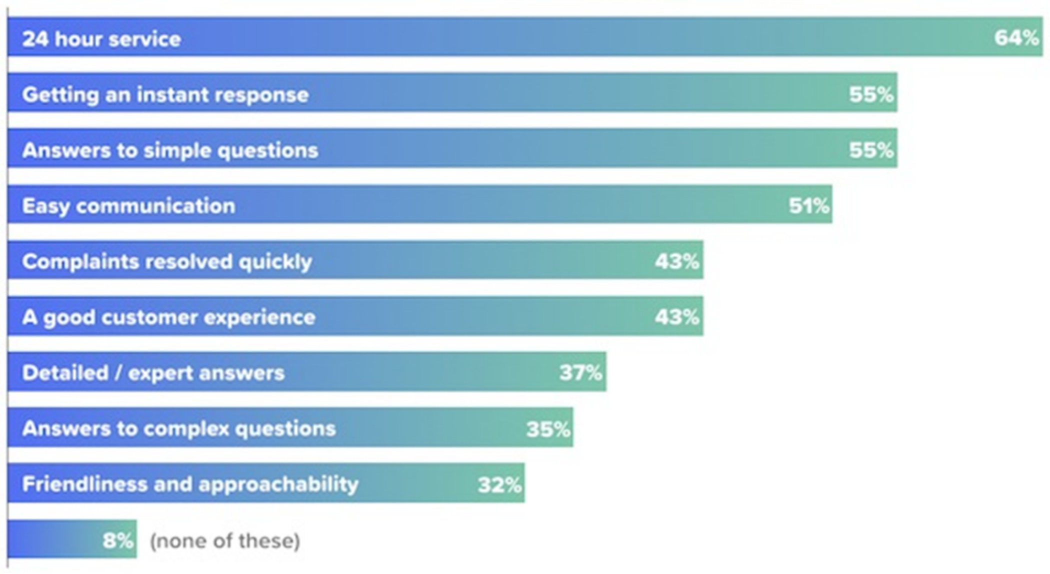


Figure 1.3: Potential benefits of chatbots by 2018 State of Chatbots Report [1]

from *answering simple, common and repetitive questions*, and let them focus on the cases that really need human participation? If so, we can largely *facilitate productivity*, *improve user experience* and *reduce cost*.

1.3 Literature Survey

Throughout the process of investigating *the current state of UIT* and *machine reading comprehension* researches, I have come to several facts:

- There are around 65% of survey individuals who ask friends and their teaching consultant for answers when they have a question related to UIT regulations, 15% of survey individuals search for answers on published documents on the UIT website, and 45% used the mailing system to consult the UIT staffs.
- Around 30% of people who have used the mailing system dissatisfied with the waiting time and the quality of answer given by UIT staffs
- *QA systems* based on machine reading comprehension have *surpassed human performance* in *English* (based on SQuAD leaderboard [2]), but still *underdeveloped* in *Vietnamese*.
- To the best of my knowledge, there are no *QA systems* developed to *tackle the problem of answering questions based on documents* using *deep learning approaches* in

[2]https://rajpurkar.github.io/SQuAD-explorer/

Vietnamese.

Given these facts, to solve the aforementioned problem, this thesis *investigates* and *build a QA system* that, given the *input as a question* and *a corpus represent all rules and regulations of UIT*, the system will return a *short answer retrieved from the corpus*.

The main contribution of this work is the *combination of traditional information retrieval (IR) approaches with deep learning approaches* for *reading comprehension* in *Vietnamese*. My hypothesis is by doing so, we can achieve a *better result* than *previous Vietnamese QA systems*, which only applied *traditional IR approaches*. In essence, the following are the main contributions of this work:

- Design and develop a *real-world intelligent assistant* that support UIT staffs in *answering questions* related to *regulations of UIT*.
- Investigate an approach that *combines deep learning approaches* for *reading comprehension* with *traditional IR approaches* in Vietnamese QA systems.

1.4 Scope & Objectives

This study will focus on developing a *Question Answering system* that combines *Information Retrieval* and *Machine Reading Comprehension* approaches, that can assists UIT staffs in answering questions related to rules and regulations of UIT (can be found in UIT regulation documents) in Vietnamese. I believe that the QA model developed in this thesis could be applied to various fields with little training data, but this thesis only studies the use of the aforementioned QA system in understanding regulation documents of a university. The *QA system* to be developed based on *Natural Language Processing* and *deep learning* technique.

Under limited time, the objectives of this thesis is to: understand core concepts and *algorithms* used in *NLP* for *machine reading comprehension* and *best practices* to *build a QA system*; *build a QA system* that capable to *answer questions related to rules and regulations of UIT in Vietnamese* using *deep learning*, which is unprecedented in previous studies in Vietnam; and publish the *QA system* to support UIT staffs for *answering questions related to rules and regulations*

1.5 Thesis Organization

This thesis consists of 5 chapters, which will cover the intuitions as well as the process and the result of the *QA system* that is capable of reading comprehension in Vietnamese. Each chapter is organized in the following manner:

- *Chapter 1* introduces the problem, the motivation for doing this project and gives an overview of the study. This chapter also discusses the scope, the significance of the study and its objectives.
- *Chapter 2* provides a *basic background* in *question answering* and *transfer learning*, whose purpose is to inform readers about some *basic knowledge*, *past and current states* of those fields, some of which will be required for the next chapters. This chapter also presents *related works* that have been done.
- *Chapter 3* introduces the system architecture, and will dive deeper into each of these components as well as techniques used in great details.
- *Chapter 4* describes the process to *build* and *evaluate* the system as well as the experimental results of this work.
- *Chapter 5* concludes the *work* and *results* of this thesis, as well as providing some guidelines on *future works* to further improve our QA system.

Chapter 2

Background

2.1 Question Answering

There has been an interest in *representing knowledge and automatically processing it* from the time of the first generation of computers. This interest has increased to become an *urgent necessity* nowadays, because of the *exponential growth* in the amount of *available digital information*, the growth in *the use of computers for communications*, and the increasing *number of users* that have access to all this information [1]. These circumstances have advanced research into *information systems* that can facilitate the *localization, retrieval, and manipulation* of these enormous quantities of *data*. *Question Answering (QA)* is one of these research fields.

The task of *QA systems* is to *retrieve answers* to question in *natural language* using either a *pre-structured database or a collection of natural language documents*, rather than "*document retrieval*", which relevant documents are ranked in response to users' query, as most traditional *Information Retrieval (IR) systems* or *search engines* currently do. *QA systems* are also considered an *advanced form of IR* [1].

Most of previous studies define the general architecture for retrieving answer according to the question in *3 macro modules* [1] as presented in Figure 2.1:

- *Question Processing*: This module *receives* the question from the user in natural language, then *analyze*, *classify* and *reformulate* the question.
  - The *analysis* step is necessary to determine the *type of question* in order to *avoid ambiguities* in the answer.

  - The *classification* is one of the main steps of a QA system. There are 2 approaches for this, namely *manual classification*, which hand-made rules are applied to identify expected answer type that may be accurate but time-consuming, tedious and non-extensible, and *automatic classification*, which can be extended to new question type with acceptable accuracy.
  - There are approaches that *classify* the question type as *What, Why, Who, How and Where questions*, and there are approaches that classify questions corresponding to the *type of answer expected*, namely *factoid, list, definition and complex* questions. These type of definitions help on a better answer detection.
  - The *reformulation* step is required in order to *enhance the question phrasing* and to *transform the question into queries* for IR systems (search engines).
- *Document Processing*: This module select a set of *relevant documents or paragraphs* depending or on the focus of the question or text understanding through NLP. Retrieved data can be *ranked* according to its *relevance* for the question.
- *Answer Processing*: This module use extraction techniques on the result of the *document processing* module to *find an answer*. The answer must be *simple*, but it might require *merging information* from different sources, *summarization* and dealing with *uncertainty* or *contradiction*.

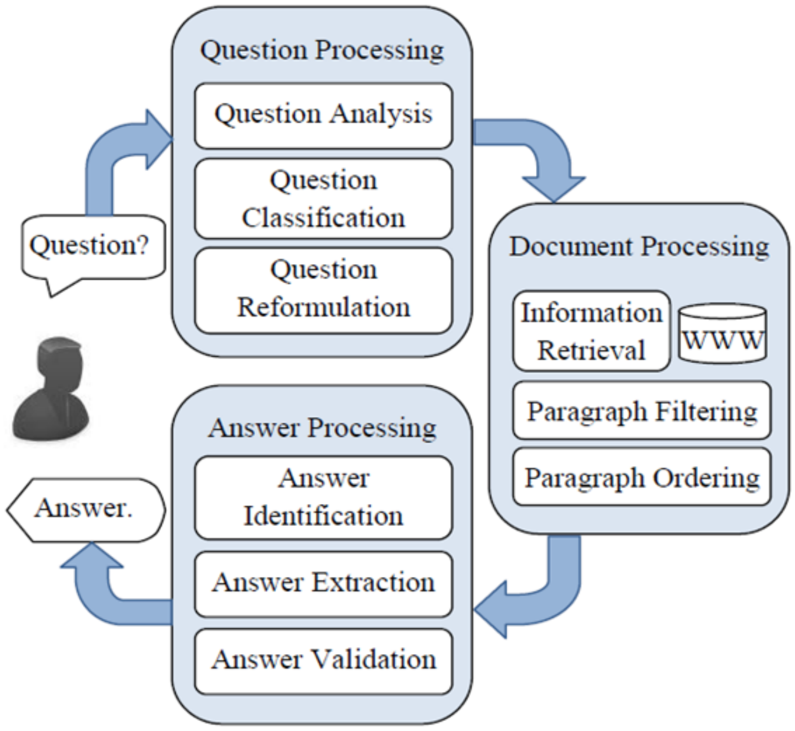


Figure 2.1: Basic Architecture of a Question Answering System

There are many approaches to *classify QA systems*, and one observation is to define *QA* *systems* by the *paradigm* each one implements [2]:

- *Information Retrieval QAs* use search engines to *retrieve answers*, apply filters and *ranking* to find the appropriate answer.
- *Natural Language Processing QAs* apply *linguistic intuitions* and *machine learning approaches* to extract answers
- *Knowledge Base QAs* find answers from a *structured data source* (or a *knowledge base*) rather than unstructured text.
- *Hybrid QAs* are high-performance QA systems that make use of as many types of resources as possible, which is a *combination* of IR QA, NLP QA, and Knowledge Base QA. The main example of this paradigm is IBM Watson [3].

Some of the *earliest QA systems* date back to BASEBALL [4] and LUNAR [5] in the 1960s, which tended to be *limited* in a *specific domain*, but they are nevertheless telling of our fascination with autonomous agents that can *understand* and *communicate* in *natural language* to answer questions.

Since the turning point in 1999, with the introduction of the QA track in the Text REtrieval Conference (TREC), researches in QA have been blooming, and the number of such *QA systems* are *enormous*. For the next few paragraphs, *trends* and *statistics* of QA systems since 2000 are introduced, based on a survey. [1].

From 130 popular studies taken from the initial 1842 set, with 34.59% of papers implemented a *knowledge base paradigm*, 33.08% implemented an *NLP paradigm*, there is a slightly higher usage of knowledge base and NLP implementations, compared to 28.57% implemented an *IR paradigm*, and 3.76% implemented a *hybrid paradigm*, as shown in Figure 2.2.

Most *QA systems* use *English* as the adopted language and the average oscillation over the year show a *constantly seek to better option* on QA research. One important finding is that each year, we could observe a *higher average accuracy* for NLP paradigm in 53% of the case, and the accuracy is reached by approaches that implemented *language independence*. An analysis in the relationship between *high average accuracy* reached and *independence language implementations* show that *NLP paradigms* are more fit in *Open-Domain QA* systems and *Knowledge Based paradigms* are more fit in *Restricted-Domain QA*.

QAs has been implemented in *various field of knowledge*, and the domains in which QA systems are implemented can be divided into *Open-Domain* that answers domain independence questions and *Restricted-Domain* that answer domain-specific questions [6]. According to

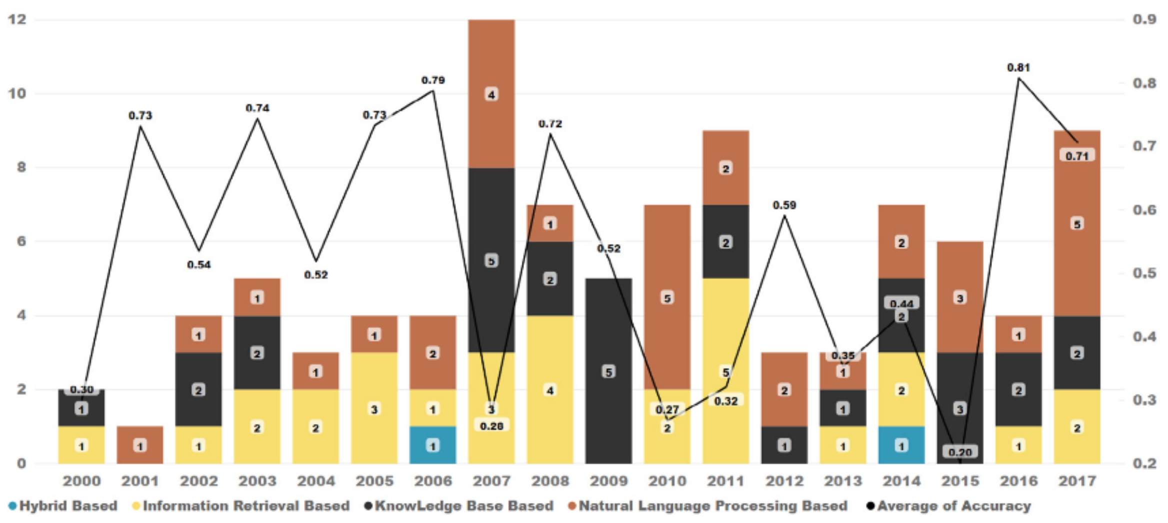


Figure 2.2: QA Publication timeline since 2000

Figure 2.3, *Open-Domain* based on *World Wide Web* implementations are the biggest part of the researches, and follow up with an important rate is *medicine* subject. We can also observe that QA intersects with many areas and domains, showing how these systems can be important for *any kind of user and need*.

NLP techniques are essential for *QA systems* to create an understanding between user and the machine and most researchers are using NLP on their approaches. For most researches, *regardless* of the used paradigms which are named over the main task, they use *NLP techniques* whether to classify the question or to retrieve the answer. In other words, *NLP techniques* are not only used in *NLP QA paradigm*, but also used in *IR QA* and *Knowledge Base QA*. Those techniques include *POS Tagging, Tokenization, Named Entity Recognition, Semantic Parser, and Similarity Distance*. Figure 2.4 represents this idea.

To implement the modules of a QA system (Question Processing, Document Processing, and Answer Processing), various techniques, algorithms, frameworks and systems related to information extraction, NLP and machine learning. The 15 most addressed techniques, algorithms, frameworks, and tools are observed and represented in Figure 2.5

Information Retrieval

Information Retrieval (IR) systems can be found in *intelligence systems* as long ago as in the 1960s, as the earliest computer-based search systems were built in the late 1940s, inspired by pioneer innovation. With the growth of computer technologies, *search systems capabili-*

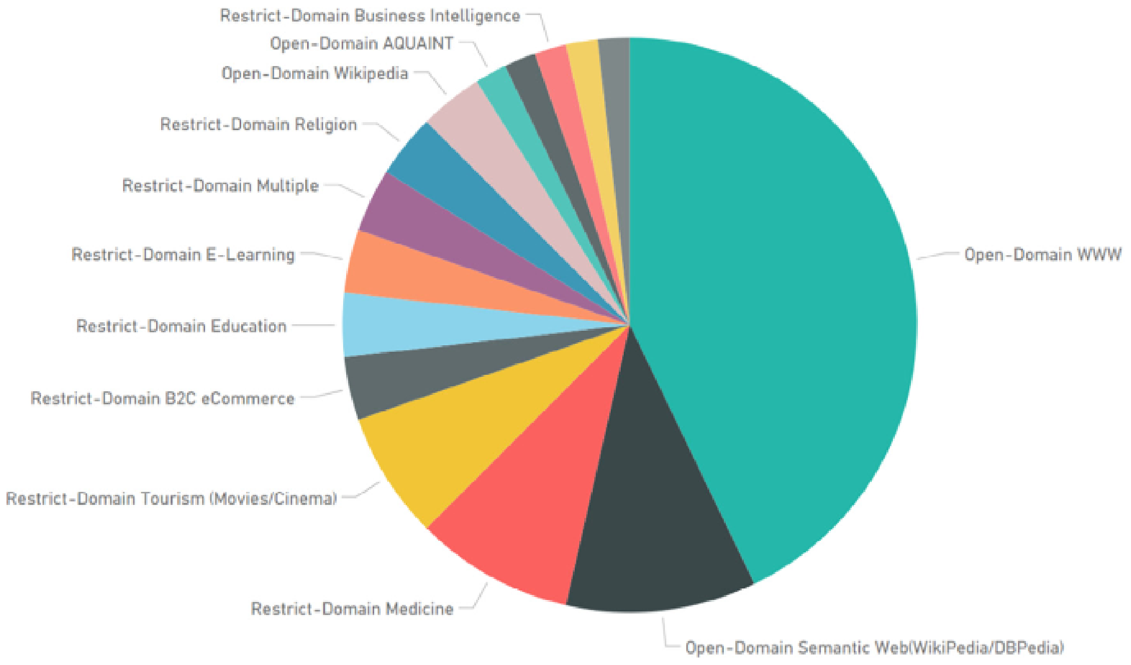


Figure 2.3: Fields where researchers in QA are implementing their approaches

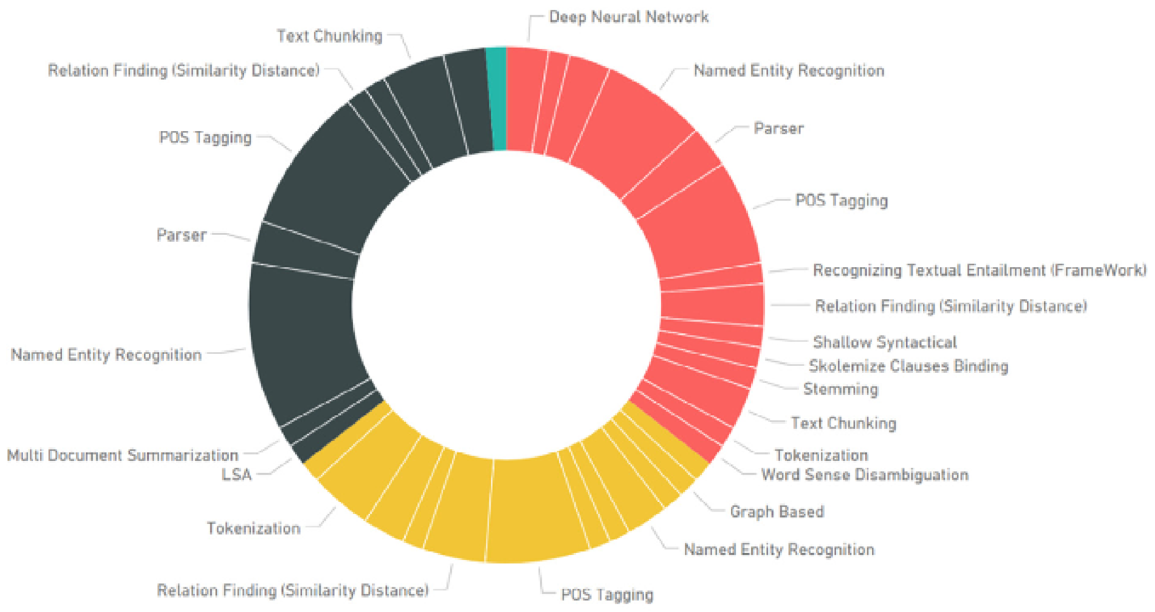
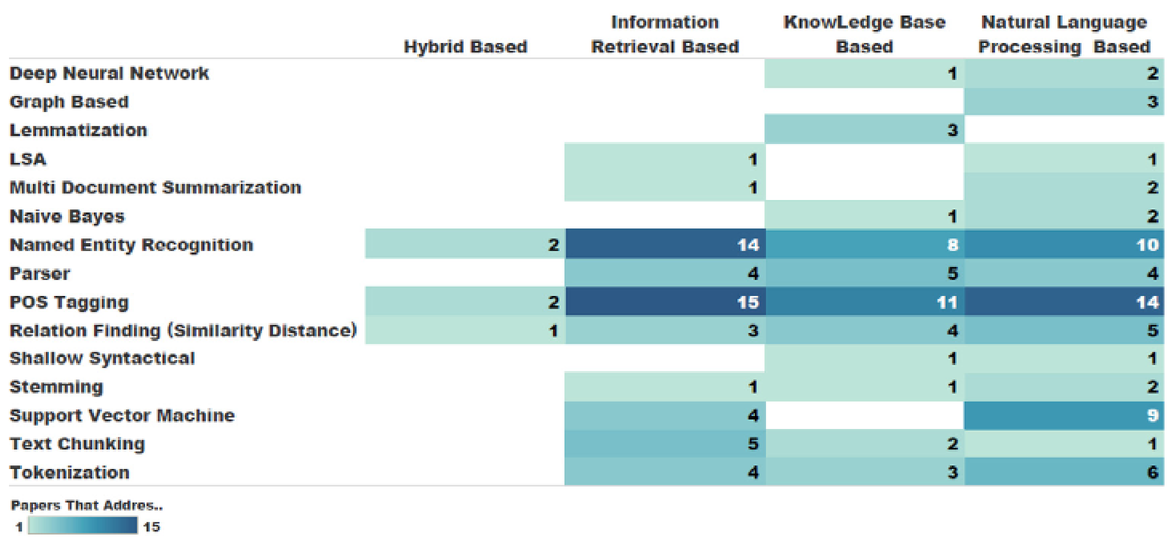


Figure 2.4: NLP techniques used in other paradigms beside NLP paradigm



	Hybrid Based	Information Retrieval Based	KnowLedge Base Based	Natural Language Processing Based
Deep Neural Network			1	2
Graph Based				3
Lemmatization			3	
LSA		1		1
Multi Document Summarization		1		2
Naive Bayes			1	2
Named Entity Recognition	2	14	8	10
Parser		4	5	4
POS Tagging	2	15	11	14
Relation Finding (Similarity Distance)	1	3	4	5
Shallow Syntactical			1	1
Stemming		1	1	2
Support Vector Machine		4		9
Text Chunking		5	2	1
Tokenization		4	3	6

Figure 2.5: Top 15 techniques, algorithms, frameworks and tools used

ties grow with increases in processor speed and storage capacity. The development of such system also reflects a *rapid progression* from the *manual library-based approaches* of acquiring, indexing and searching information to a *more automated method*. [7]

The *task of IR systems* is to *locate information relevant to a user' query*, which is in collections of unstructured data or semi-structured data (e.g web pages, images, videos, documents, ...). These *documents* are typically transformed into a *suitable representation* so that relevant ones can be *retrieved effectively*. Each *IR model* governs: (1) *how documents and queries are represented*, and (2) *how the relevance of a document to a user query* is defined. *IR models* can be categorized according to 2 dimensions: the *mathematical basis* and the *properties of the model* [8], which is presented in Figure 2.6 and will be described further below. In this section, past works focus on *IR models* based on the first dimension will be reviewed.

Mathematical based IR models can be further classified into 4 categories: *set-theoretic models*, *algebra models*, *probabilistic models* and *feature-based retrieval models*. Although each models have different strategies to represent documents and queries, they all treat each document or query as a *bag of terms* (a synonym of word in IR literature), means that a document is described by a set of distinctive terms, term sequence and position in a sentence of a document are ignored [7, 8].

Set-theoretic models represent documents as sets of words or phrases. From there, *Similarities* are usually derived from *set-theoretic operations* on those sets. Popular models fall on this category are: *Standard Boolean model*, *Extended Boolean model* and *Fuzzy Retrieval model* [8].

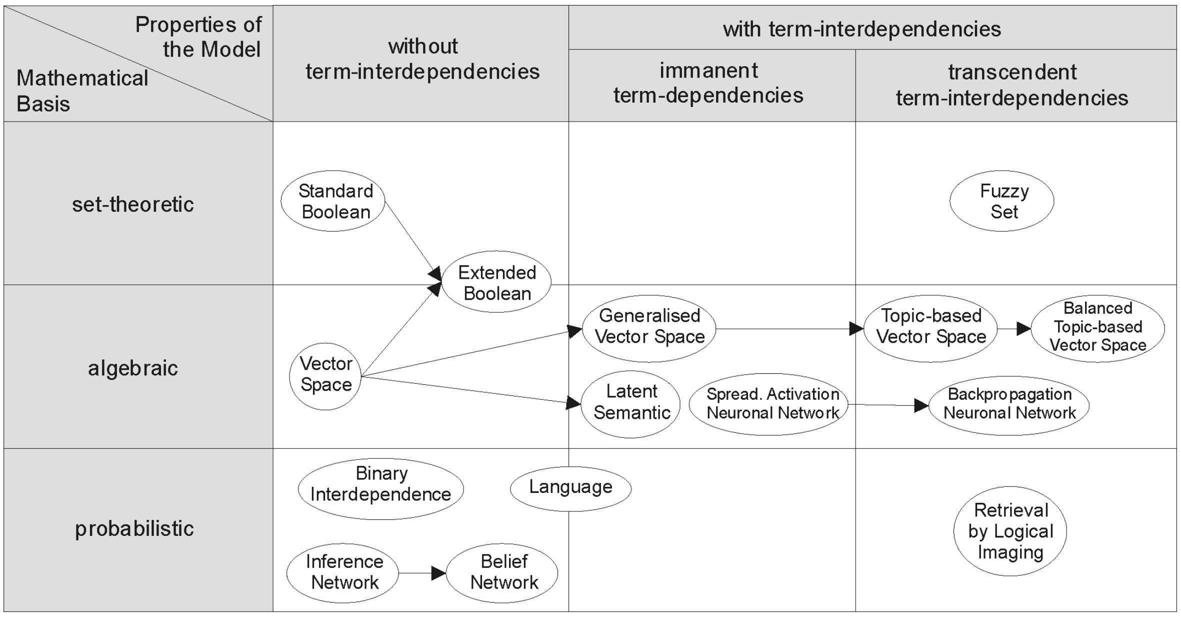


Figure 2.6: IR models categorization

Standard Boolean model [9] uses the notion of *exact matching* to equate documents to the user query, which is based on *boolean logic* and *classical set theory*. The *query* is defined as *a logical combination of terms*, which resulted in a set of those documents (also represented as sets of terms) that exactly matched the query. Some *setbacks* of this model are: similarities are based on whether or not the documents contain the query terms, and *does not consider term weights*; Exact matching may *retrieve to few or too many documents*; *difficult to translate* a query into a Boolean expression; and the approach is more like data retrieval than information retrieval.

Extended Boolean model [10] is developed to overcome the shortcomings of *Standard Boolean model*, that *combines* the characteristics of *Boolean Algebra* and *Vector Space Model* to employs *partial matching* and *term weights*. By doing so, a document can be *relevant if it matches some of the query terms*, but it is not the case in the *Standard Boolean model*.

Fuzzy Retrieval model is based on the *Extended Boolean model* and the *Fuzzy set theory*, allows the *manipulation and accumulation of weights* given to terms. This logic permits the definition of *intermediary truth values* between conventional evaluations of true and false. There are 2 classical *Fuzzy Retrieval model*, namely *Mixed Min and Max model* [11] and *Paice model* [12].

Algebraic models represent documents and queries as *vectors, matrices or tuples*. Some models that belong to this category are: *Vector Space model*, *Generalize Vector Space model*, *Topic-based Vector Space model*, *Extended Boolean model* and *Latent Semantic Indexing*

[8].

Vector Space model [13] represent textual information as *vectors in N-multidimensional space*, where N is the number of unique terms in the collection of being searched and each dimension corresponding to a separate term). *Similarities* can be easily calculated on these vectors, using *vector operations*. *Vector Space Model* gives good results when used along with approaches like ranking of documents, *Latent Semantic Indexing*. This model is perhaps the best known and most widely used, with some attractive properties: *simple* model based on linear algebra; *employs term weights* instead of binary; allows *computing a continuous degree of similarities* between queries and documents; *allows ranking documents according to their possible relevance*; allows *partial matching*. But *Vector Space model* also comes with some limitations: *long documents are poorly represented* since they have poor similarity values (a small scalar product and a large dimensionality); *search keywords must precisely match document terms*; documents with similar context but different term vocabulary would not be associated, resulting in a "*false negative*" match; theoretically *assumes that terms are statistically independent*; weighting is intuitive but not very formal. Many of these difficulties can, however, be overcome by the integration of various tools, including mathematical techniques such as *singular value decomposition* and lexical databases such as WordNet.

The *Generalized Vector Space model* [14] is a generalization of the *Vector Space Model* that *does not assume pairwise orthogonality* among the vectors representing index terms, where the model does not take into consideration that two index terms can be semantically related. In this model, *index terms are composed of smaller elements*.

However, all mentioned approaches do not take advantage of relationships of concepts presented in the document. The *Topic-based Vector Space model* [15] extends the *Vector Space model* by *removing the constraint that the term-vectors be orthogonal*, which causes problems with synonyms and strongly related terms. This model facilities *the use of stopword lists, stemming and thesaurus*, in contrast to the *Generalized Vector Space model* that depends on *concurrence-based similarities between terms*. The *Enhanced Topic-based Vector Space model* [16] has the capability of *representing linguistic phenomena* using a *semantic ontology*.

Lexical matching methods where information is retrieved by *exactly matching* terms in documents with user query are *imprecise*, since there are generally many ways to express a concept by the use of synonyms. Additionally, many words and phrases have multiple meanings. In those case, *Latent Semantic Indexing* [17] comes in handy, that uses a mathematical technique called *singular value decomposition*. The model *analyzes relationships between a set of documents and the terms* they contain by producing a set of concepts related to the documents and terms, which is based on the principle that *words that are used in the same context tend to have similar meanings*.

Probabilistic models treat the process of document retrieval as a *probabilistic inference*, where *similarities* are computed as *probabilities* that a document is relevant given a query. *Probabilistic theorem* is often used in these models. Popular models are: *Binary Independence model*, *Probabilistic Relevance model*, *Uncertain Inference model*, *Language model*, *Divergence-from-randomness model* and *Latent Dirichlet Allocation*. [8]

Binary Independence model [18] treats documents as *binary vectors*, such that only the present or absence of terms in documents is documented. This model is based on the *Binary Independence assumption* that terms are independently distributed in the set of relevant documents as well as independently distributed in the set of irrelevant documents. This postulate is very restrictive in as much as it gives better results in many situations. This model is treated as an instance of the *Vector Space model*.

Probabilistic Relevance model [19] makes an estimation of the probability of finding if a document is relevant to a query by *assumes that the relevance probability depends on the query and document representations* (the relevance of a document increases with the frequency of the query term). However, various setbacks are inherent in the probabilistic model: index terms are not weighted; there is no accurate estimate for the first run probabilities, since it is based on relevance feedback; and terms are assumed mutually independent. Several approaches were proposed to address these problems, one is the *Binary Independence model* from the same author, and one popular derivative is based on the *Okapi (BM25)* weighting scheme and its *BM25F* brother.

Uncertain Inference model [20] deals with ways of formally defining query and document relationship in IR, as a mechanism of deriving consequences from human knowledge through *uncertain set theory*. In this approach, *the query supplied by the user is treated as a set of assertions about the desired document*. The model's main activity is to *deduce, given a particular document, if the query statements are true*. In situations where the statements are true, the document is retrieved. The main challenge of this approach is that the contents of documents are not sufficient to assert the queries, thus a knowledge based on facts and rules is required.

A *Language model* is formalized as a probability distribution over a sequence of strings or words. Traditional methods usually involve making an *n-th order Markov assumption* and *estimating n-gram probabilities* through *counting and subsequent smoothing*, called the *n-gram models*, that assigns a probability distribution over a given word observed after a fixed number of previous words. One merit of such count-based model is that they are *simple to train*, but *probabilities of rare n-grams are poorly estimated* due to data sparsity, despite smoothing techniques. *Neural network models* are also popular that can achieve substantial improvements in perplexity over *count-based language model* that could simultaneously learn the conditional probability of the latest word in a sequence as well as a vector representation for each word in a predefined vocabulary.

Divergence-From-Randomness Model [21] is one type of probabilistic model where *term weights* are calculated by *determining the divergence between a term distribution produced by a random process and the actual term distribution*. This IR model employs term-document matching function that are computed by taking the product of two divergence functions

Latent Dirichlet Allocation [22] is a generative statistical model that *permits sets of observations to be explained by unobserved groups* so as to provide an explanation as to why some parts of the data are similar. In this approach, *each document is treated as a mixture of various topics*. It also assumes that *there are k underlying latent topics according to which documents are generated*, and that *each topic is represented as a multinomial distribution over the words in the vocabulary*. Therefore, a document is generated by *sampling a mixture of these topics* and then *amalgamating words from that mixture*.

Feature-based Retrieval models represent documents as vectors of values of feature functions (or just features) and seek the best way to combine these features into a single relevance score, typically by *learning to rank* methods. Feature functions are arbitrary functions of document and query, and as such can easily *incorporate almost any other retrieval model as just another feature*.

Aside from these models, there are popular techniques that are widely used in IR, such as:

- The use of *tf-idf* as a term-weighting scheme, which is the combination of 2 weights: (1) *Term frequency* (*tf*) based on the occurrence of words across the documents of a collection, and (2) *Inverse document frequency* (*idf*) that introduced the idea that the frequency of occurrence of a word in a document collection was inversely proportional to its significance in retrieval (less common words tended to refer to a more specific concepts, which were more important in retrieval)
- A mathematical technique named *singular value decomposition* that is applied to *Algebraic models*, to *reduce the dimensionality of the vector space of a document collection*, therefore *caused words with common semantic meaning to be merged*, resulting

in queries matching a wider range of relevant document

- The process of *Relevance Feedback*, which is a process to support iterative search, where *previously retrieved documents could be marked as relevant* in an IR system. *A user query was automatically adjusted* using information extracted from relevant documents. *Relevance Feedback* is one of the example uses of machine learning in IR.
- Other IR enhancements explored are: *clustering of documents with similar content*; the *statistical association of terms with similar semantic meaning*; *increasing the number of documents matched with a query by expanding the query with lexical variations* (so-called stems), or with semantically associated words

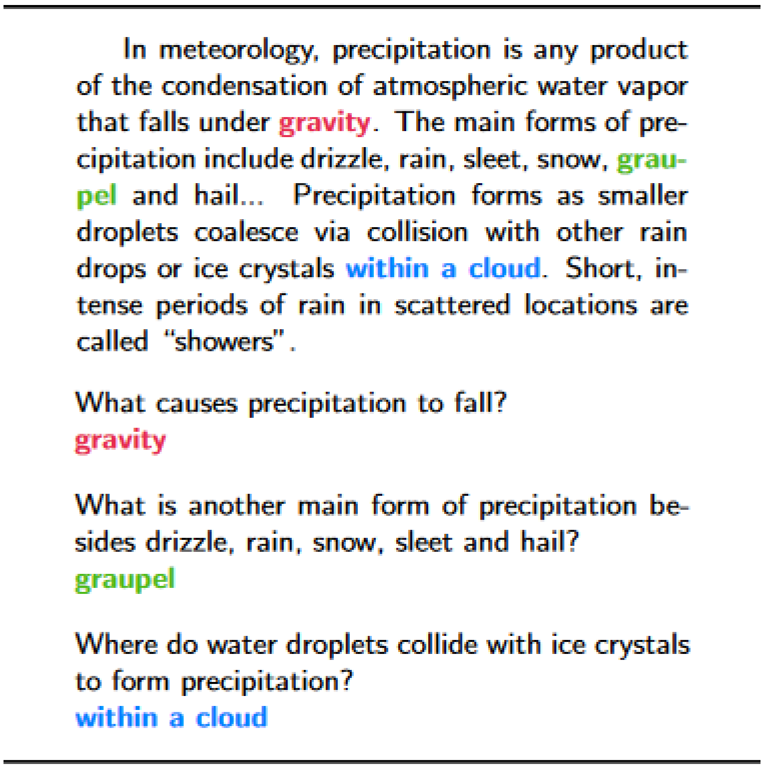
Reading Comprehension

Reading Comprehension, or the ability to read texts and then answer questions about it, is a challenging task for machines, require not only *natural language knowledge* but also *knowledge about the world*. Consider the question: "what causes precipitation to fall?" posed in the passage in Figure 2.7, first of all, one may locate the relevant part of the passage: "precipitation ... falls under gravity", then reason that "under" refers to a cause (not location), and thus determine the correct answer: "gravity". [23]

Over the last decay, it has been shown that *given enough data*, *remarkable progress* can be made on traditionally very hard tasks, and one of them is *Question Answering (QA)*. But not until 2015 that good datasets are available, therefore we really haven't had any statistical NLP systems that can understand even such simple passages. Before 2015, traditional solutions to *Reading Comprehension* rely on NLP pipelines that involve multiple steps of *linguistic analyses* and *feature engineering*, including *syntactic parsing*, *named entity recognition*, *question classification*, *semantic parsing*, etc. [24]

After 2015, with many high-quality datasets available (see more on 2.4), there have been approaches that apply *neural network* models for various NLP tasks, including several works on *machine reading comprehension*. A common approach to machine comprehension is to use *recurrent neural network*, which is a class of artificial neural network that *specializes with sequential information*, to process the given text and the question in order to predict or generate the answer.

Attention mechanism is also widely used on top of *recurrent neural networks* [25]. *Attention mechanism* can be described as *a mapping function from the query and a set of key-value pairs to an output*, where the query, keys, values, and outputs are all vectors. *The output*



In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called "showers".

What causes precipitation to fall?
gravity

What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
graupel

Where do water droplets collide with ice crystals to form precipitation?
within a cloud

Figure 2.7: Sample question-answer pairs taken from SQuAD dataset

is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key [26]. In terms of *encoder-decoder*, which is an architecture based on recurrent neural network that is widely used in various NLP tasks, including question answering, the query is usually the hidden state of the decoder. Whereas key, is the hidden state of the encoder, and the corresponding value is a normalized weight, representing how much attention a key gets.

The first approach that applies attention is proposed by Hermann et al. (2015) [24], in which *word-level attention* to model the interaction between question and context. This paper introduced the *Attention Reader*, which uses bidirectional document and query encoders to compute attention, and *Impatient Reader*, which compute the attention over document after reading each and every word in the query. Despite the complexity of *Impatient Reader* over *Attention Reader*, both models *perform almost identically* on the CNN/Daily Mail dataset.

Memory networks belong to a class of model that combine successful learning strategies developed in machine learning with a memory that can be read and written to, therefore accurately remember facts from the past [27]. There are also approaches that apply memory network into machine comprehension. Hill et al. (2015) [28] propose a *window-based mem-*

ory network, which uses windows of fixed length centered around the candidate words as memory cell, for the Children Book Test dataset. Sukhbaatar et al. (2015) [29] propose an *end-to-end memory network* on top of recurrent attentive model. Kumar et al. (2016) [30] propose *dynamic memory network*, in which episodic memories are formed and used to generate relevant answers. But its scalability when applied to large datasets is still an issue [31], therefore it is not widely used.

Successful machine comprehension models generally employ 2 key ingredients [32]:

- A *recurrent neural network* model to process sequential input.
- *Attention* component to cope with long-term interactions.

Numerous efforts have since continued to push the boundaries of such recurrent language models. Chen et al. [33] propose an *improved model over Attention Reader*. Kadlec et al. (2016) propose *Attention Sum Reader* [34] that apply pointer networks with one attention steps to predict a single entity from the document. *Pointer Network* [35] whose outputs "point" back to the input using attention mechanism, allows us to generate answers that consist of multiple tokens from the original text rather than from a larger fixed vocabulary. Sordoni et al. (2016) [36] propose *interactive alternating attention mechanism* to better model the link between question and context. Trischler et al. (2016) [37] solve cloze-style question answering task by *combining an attentive model with a reranking model*. Dhingra et al. (2016) [38] propose *Gated-attention Reader* that iteratively select important parts of the context by a multiplying gating function with the question representation. Cui et al. (2016) [39] propose *Attention-over-Attention Reader*, which is a 2-way attention mechanism to encode passage & question mutually. Wang et al. (2016) [31] propose to combine *match-LSTM*, which is a class of recurrent neural network whose input at each time step contains an additional attention premise and *Pointer Network* for machine comprehension. Xiong et al. (2016) [40] propose the concept of *dynamic coattention networks* and Seo et al. (2016) [41] introduced *Bidirectional attention flow* (BiDAF) that employ variant coattention mechanism to match the question and the context mutually. Yu et al. (2016) [42] and Lee et al. (2016) [43] deploy machine comprehension models that rank continuous text spans within passages. Yang et al. (2016) [44] propose a *fine-grained gating mechanism* to dynamically combine word-level and character-level representation and model the interaction between the question and context. Wang et al. (2016) [45] propose *matching the context of passage with the question from multiple perspectives*. Microsoft Language Team introduced *Reasoning Network* (or ResoNet) [46] that iteratively infer the answer with a dynamic number of reasoning steps, and *R-net* [47] with the concept of *self-matching attention* that refines the context representation by apply attention against itself. By far this is the state-of-the-art model on machine comprehension that based on *recurrent neural network* with *encoder-decoder* architecture.

Because of the *sequential nature* of the text coincides with the design philosophy that makes *recurrent neural network* popular, but also because of that sequential nature that *prevents Recurrent Neural Network from performing parallel computation*, as tokens must be fed into the Recurrent Neural Network in order, therefore prevents the machine comprehension systems from being deployed in real-time applications.

Some solutions, such as *factorization tricks* [48] and *conditional computation* [49], are proposed, and there are approaches that discard *recurrent neural network* and use exclusively *convolutional neural network* and *attention* as encoder building blocks. Extended Neural GPU [50], ByteNet [51] and ConvS2S [52] are popular architectures that follow this approach, and Yin et al. (2016) [53] is one example of this approach applied to machine comprehension. Google Language team proposed *the Transformer* [26], a network architecture that relies only on attention, applied to machine comprehension with the *QANet* model [32], and have achieved state-of-the-art speed and accuracy.

2.2 Transfer Learning

Machine learning has been successfully used in many applications where *patterns from past information* (training data) can be *extracted* in order to *predict future outcomes*. Traditional machine learning is characterized by training data and testing data having the *same input feature space and data distribution*. Otherwise, the result of the predictive learner can be degraded. In fact, obtaining training data that matches the feature space and predicted data distribution characteristic of testing data can be *tedious* and *expensive*. Therefore, the motivation of transfer learning is to create a good learner for a *target domain*, *trained from a related source domain* [54].

Transfer learning can be used to improve a learner from one domain by *transferring information from a related domain*. Take an example of 2 people learning to ride a motorcycle, one person have no prior experience in riding a bicycle, and another person have good knowledge by riding a bicycle for years, then the person have experience riding a bicycle will be able to learn how to ride a motorcycle in an efficient manner by transferring previously learning bicycle riding techniques to the task of learning to ride a bicycle. One person is able to take information from previously learned task and used it in a beneficial ways to learn related tasks.

The need for *transfer learning* occurs when there is a *limited supply of target training data*. With big data repositories becoming more prevalent, *using datasets that related to, but not exactly the same as the target domain* make transfer learning solutions an attractive approach. There are many successful machine learning applications that have applied transfer learning including *text sentiment classification*, *image classification*, *human activity classification*, *software defect classification*, and *multi-language text classification* [54].

According to [55], *transfer learning* can be categorized based on different settings, based on different situations between the source and target domains and tasks, and further described in Figure 2.8:

- *Inductive transfer learning* refers to a case when target domain data are available, and the target task can be different but related to the source task.
- *Transductive transfer learning* refers to a case of having labeled source and no labeled target domain data, and the source and target tasks must be the same.
- *Unsupervised transfer learning* refers to a case of having no labeled source and no labeled target domain data, and the target task could be different but related to the source task.

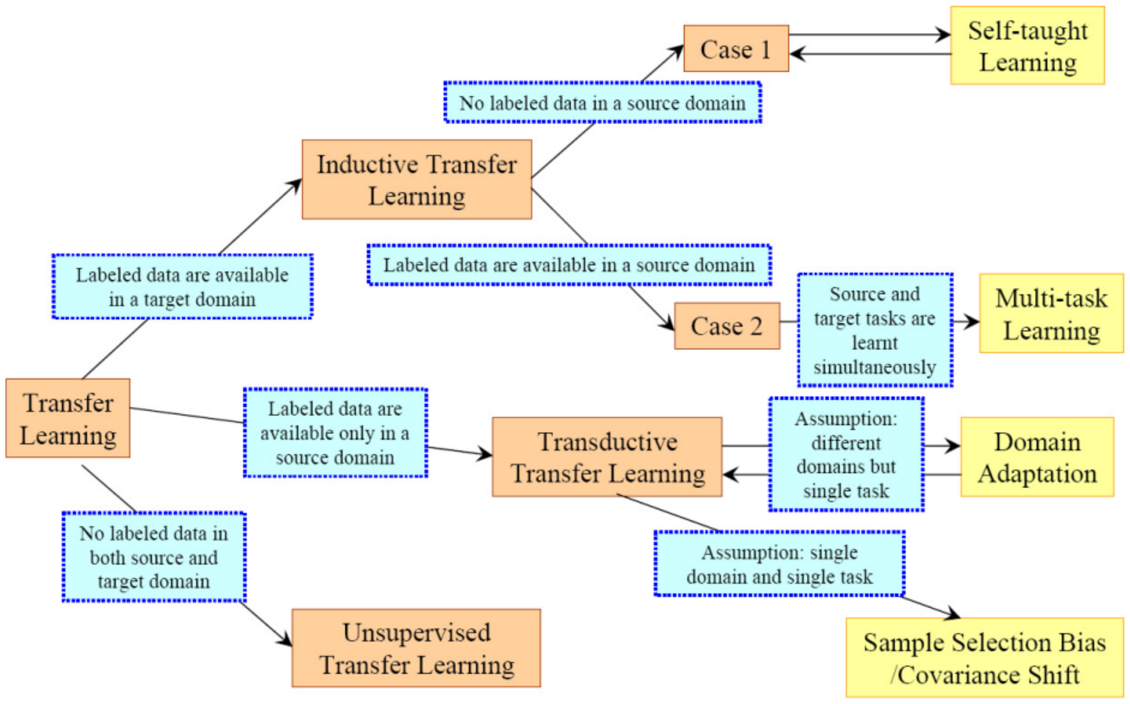

Case 1
Self-taught Learning
No labeled data in a source domain
Inductive Transfer Learning
Labeled data are available in a target domain
Labeled data are available in a source domain
Case 2
Source and target tasks are learnt simultaneously
Multi-task Learning
Transfer Learning
Labeled data are available only in a source domain
Transductive Transfer Learning
Assumption: different domains but single task
Domain Adaptation
No labeled data in both source and target domain
Assumption: single domain and single task
Unsupervised Transfer Learning
Sample Selection Bias /Covariance Shift

Figure 2.8: Different settings in Transfer Learning

There are different strategies for solving a transfer learning problem, and some of them are presented, as follow [55]:

- *Instance transfer* (or *transfer learning through instances*) approaches that re-weight some instances (or labeled data) in the source domain for use in the target domain for training. These algorithms work best when the conditional distribution is the same in both domains.
- *Feature representation transfer* (or *transfer learning through features* or *feature-based transfer learning*) approaches that find a "good" feature representation that reduces difference between the source and the target domains. These approaches can be categorized in 2 ways:
  - *Asymmetric feature transformation* approach that transforms features of the source through re-weight to more closely match the target domain.
  - *Symmetric feature transformation* approach that discovers underlying meaningful structures between source and target domains to find a common latent feature space.
- *Parameter transfer* approaches that discover and transfer knowledge through shared parameters of source and target domain learner models or by creating multiple source learner models and optimally combining the re-weighted learners (ensemble learners) to form an improved target learner.
- *Relational knowledge transfer* approaches that transfer knowledge based on some defined relationships between the source and target domains.

The cases where different methods are used for each transfer learning setting are described in Table 2.1

	Inductive transfer learning	Transductive transfer learning	Unsupervised transfer learning
Instance transfer	✓	✓	
Feature representation transfer	✓	✓	✓
Parameter transfer	✓		
Relational knowledge transfer	✓		

Table 2.1: Different approaches used in different transfer learning settings

2.3 Related Works

Although the *question answering research has made rapid advancement*, the *Vietnamese question answering research has still been under development*, mainly because of lack of resources and tools to process natural language, and most of Vietnamese question answering system mainly based on traditional information retrieval approaches. In this section, we will review some achievement in developing Vietnamese question answering systems.

Vu Mai Tran, Vinh Duc Nguyen, Oanh Thi Tran, Uyen Thu Thi Pham, and Thuy-Quang Ha [56] created a *Vietnamese QA system applied in travel domain* that can answer factoid questions about entity relations. First, the system extract patterns and tuples from a large text collection using the Snowball system [57] and stored in a database. When the system received a user's query, it extracts entities and looks for related candidate relations. Then, it calculated similarities between the query vector and all patterns of candidate relations, and the best pattern is chosen. Finally, a database query is built to get tuples contained the required entity and relation. This system achieved 89.7% precision and 91.4% ability to return an answer when testing on traveling domain.

Mai-Vu Tran, Duc-Trong Le, Xuan-Tu Tran, Tien-Tung Nguyen [58] built a *closed domain Vietnamese QA system for answering factoid questions related to Vietnamese person named entities* (Who, Whose or Whom). This system combined Conditional Random Fields (a machine learning methodology) in question analysis phase, and 2 automatic answering strategies: index sentences database-based and Google search engine based in answer extraction phase, in which if answers were not found in the collected database, the question would be pushed into Google search engine. The system filtered and ranked database-based candidate by using similarities with the question. This system obtains 74.63% precision and 87.9% ability to give an answer.

Dai Quoc Nguyen, Dat Quoc Nguyen, Son Bao Pham [59] proposed an *ontology-based Vietnamese QA system* that allowed users to make a query in natural language, which contains 2 modules: the question processing module and the answer retrieval module. The ontology included only two people relations. Despite the high cost for building the ontology, results were relatively high, achieve an accuracy of 95% in the question analysis module and 70% in the answer retrieval module.

Dang-Tuan Nguyen and Ha Quy-Tinh Luong [60] created a *Vietnamese search system that allows users to search courses in Vietnamese OpenCourseWare program using natural language*. First, the system extracted keywords from the Vietnamese OpenCourseWare page and stored in an ontology-based knowledge. When a user's query is input, the system analyzes the question based on a set of defined syntax rules to build a query in SPARQL, and the answer is extracted from the ontology.

Dang Truong Son, Dao Tien Dung [61] built a *community-based Vietnamese QA system which answers questions by finding similar question-answer pairs from the Internet*. With returned answers, authors hope that one of them will satisfy users' demand. They extracted 90,000 question-answer pairs from Yahoo Answers, and they claimed that higher results are obtained compared with answering results from Yahoo.

Huu-Thanh Duong, Bao-Quoc Ho [62] developed a *Vietnamese question answering system that answers factoid questions related to law on enterprises in Vietnamese*. By far, this is the most similar research compared to this work. The system calculates similarities between question and documents using the tf-idf weight and latent semantic indexing, therefore return the most relevant part of the document as the answer. If answers can't be found in the predefined documents, the question will be pushed into Google to retrieve more related documents. The system architecture is described in Figure 2.9. This system achieved 70% precision when applied in legal documents.

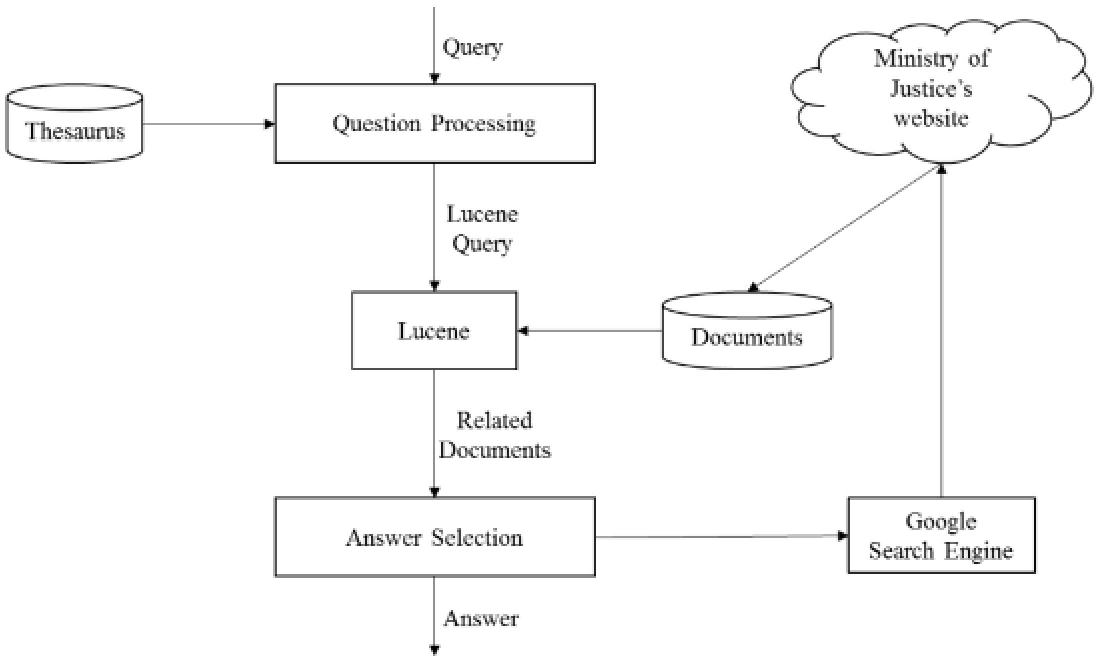


Figure 2.9: The vLawyer System

In this work, the same approach to Huu-Thanh Duong, Bao-Quoc Ho [62] (a keyword-based search based on similarity) is applied to our *search engine module* to *retrieve relevant documents*, but unprecedented to previous researches in Vietnamese QA, a *deep learning approach* is used for *reading comprehension*, in hope that the system will achieve greater text understanding, and thus return better results.

2.4 Datasets

As stated before, the success of a machine learning based question answering model depends heavily on a good dataset, but not until 2015 that good datasets become available. *Existing datasets before 2015* for Reading Comprehension are (1) *high in quality but too small for training modern data-intensive model* or (2) *large in size but are semi-synthetic and do not share the same characteristics as explicit reading comprehension questions*. [23].

Some of the machine comprehension dataset before 2015 are stated:

- *MCTest* [63]: A *open-domain reading comprehension dataset on texts* that contains 500 fictional stories and 2000 multiple choices questions that require casual, child-level reasoning and understanding of the world
- *ProcessBank* [64]: A *biology reading comprehension dataset* contains 585 questions spread over 200 paragraph that require an understanding of the relations between entities and events in the process

Since 2015, *many high-quality and large datasets have been developed*, and some of the popular ones are presented:

- *CNN/Daily Mail* [24]: *A news article machine comprehension dataset* contains nearly 300,000 articles with nearly 1.3 million cloze-style question-answer pairs, whose questions are constructed by deleting a single entity from abstractive summary points that accompany each article. Because the cloze process is automatic, the task only requires limited reasoning [33].
- *Children Book Test* [28]: A *machine comprehension dataset* where nearly 700,000 questions are taken from 108 *children' books* available through Project Gutenberg. Text passages are 20-sentence excerpts, and questions are generated by deleting a single word in the next 21st sentence. Machine comprehension based on context is necessary for this task.
- *WikiReading* [65]: A *machine comprehension dataset* with 18 million instances over 4.7 million *Wikipedia articles*, where the task is to predict textual values from the structured knowledge base *Wikidata* by reading the text of the corresponding Wikipedia articles.
- *LAMBADA* [66]: A *dataset to evaluate text understanding by means of a word prediction task* over 10,000 passages. For this task, information tracking in context understanding is important.
- *SQuAD* [23]: *A machine comprehension dataset* contains more than 100,000 *question-*

answer pairs from 536 *Wikipedia articles*, where the answer is a segment of text from the corresponding reading passage. Reasoning is required to solve this task.

- *Who did What* [67]: *A machine comprehension dataset* of over 200,000 *cloze-style multiple choice problem* constructed from the *LDC English Gigaword newswire corpus*, whose scalability and flexibility is suitable for neural methods.
- *NewsQA* [68]: A challenging *machine comprehension dataset* with over 100,000 crowd-sourcing *question-answer pairs* from over 10,000 *news articles from CNN*, with the answer being a span. NewsQA demands abilities beyond traditional word matching and recognizing textual entailment.
- *MS MARCO* [69]: A *machine comprehension dataset* with more than 1 million *anonymized questions* over nearly 9 million *passages from Bing's search query logs*. The large-scale and real-world nature of this dataset makes it attractive for machine comprehension and question answering models.
- *TriviaQA* [70]: A challenging *machine comprehension dataset* contains over 650,000 *question-answer-evidence* triples, provide high quality distant visions for question answering.

Next, we will focus on the SQuAD dataset since this dataset is thought to be the most suitable to the problem.

SQuAD: The Stanford Question Answering Dataset

The *SQuAD dataset*, or the Stanford Question Answering Dataset is a reading comprehension dataset contains *107,785 questions posed by crowdworkers* on a set of *536 randomly chosen Wikipedia articles*, where the *answer to each question is a segment of text (or span) taken from the corresponding reading passage*. In contrast to some datasets, SQuAD does not provide a list of answer choices, systems must select the answer from all possible spans in the passage. An example can be seen in Figure 2.7.

According to an analysis of the dataset [23], this dataset contains *diversity in answers*, as shown in Table 2.2: dates and numbers make up 19.8% of the data; 32.6% of answers are proper nouns; 31.8% are common noun phrases; and the last 15.8% are adjective phrases, verb phrases, clauses, and other types.

The SQuAD dataset also requires many types of reasoning for inference. According to an analysis [23], from 192 question-answer pairs from 48 articles, all examples have some sort of lexical or syntactic divergence between the question and answer in the passage, as shown

Answer type	Percentage	Example
Date	8.9%	19 October 1512
Numeric	10.9%	12
Person	12.9%	Thomas Coke
Location	4.4%	Germany
Other Entity	15.3%	ABC Sports
Common Noun Phrase	31.8%	property damage
Adjective Phrase	3.9%	second-largest
Verb Phrase	5.5%	returned to Earth
Clause	3.7%	to avoid trivialization
Other	2.7%	quickly

Table 2.2: Categories of answer types in SQuAD dataset

in Table 2.3.

Since the release of the SQuAD dataset and many other datasets, machine comprehension research has made rapid progress, with the best model now outperform human performance on the task. Table 2.4 presents the Exact Match and F1 scores of various models trained and evaluated on the SQuAD dataset.

Reading comprehension systems can often find the correct answer to a question in a context document, but they also tend to make unreliable guesses on questions for with the correct answer is not presented in the context. To address this problem, the SQuAD dataset combines existing SQuAD dataset with over 50,000 unanswerable questions, prove a challenge for existing systems that must not only answer questions when possible but also determine when no answer can be found in the context [71].

2.5 Evaluation Methods

Evaluation is a part of developing QA systems. Nowadays, as more and more QA systems are being developed, *reliable evaluation methods* to compare these models are *required*. According to [2], the most commonly used evaluation metrics in QA tasks are *F1* and *accuracy*. These metrics are usually used for *binary classification* when the prediction is either correct or incorrect. When used in IR, the prediction is whether the document is relevant or not relevant.

Reasoning	**Description**	**Example**	**Percentage**
Lexical variation (synonymy)	Major correspondences between the question and the answer sentence are synonyms.	Q: What is the Rankine cycle sometimes **called**? Sentence: The Rankine cycle is sometimes **referred** to as a practical Carnot cycle.	33.3%
Lexical variation (world knowledge)	Major correspondences between the question and the answer sentence require world knowledge to resolve.	Q: Which **governing bodies** have veto power? Sentence: **The European Parliament and the Council of the European Union** have powers of amendment and veto during the legislative process.	9.1%
Syntactic variation	After the question is paraphrased into declarative form, its syntactic dependency structure does not match that of the answer sentence even after local modifications.	Q: What Shakespeare scholar **is currently on the faculty**? Sentence: **Current faculty include** the anthropologist Marshall Sahlins, ..., Shakespeare scholar David Bevington.	64.1%
Multiple sentence reasoning	There is anaphora, or higher-level fusion of multiple sentences is required.	Q: What collection does the **V&A Theatre & Performance galleries** hold? Sentence: **The V&A Theatre & Performance galleries** opened in March 2009. ... **They** hold the UK's biggest national collection of material about live performance.	13.6%
Ambiguous	We don't agree with the crowdworkers' answer, or the question does not have a unique answer.	Q: What is the main goal of criminal punishment? Sentence: **Achieving crime control via incapacitation and deterrence** is a major goal of criminal punishment.	6.1%

Table 2.3: Types of reasoning used throughout SQuAD dataset

	Exact Match	F1
Random Guess*	1.3%	4.3%
Logistic Regression*	40.4%	51.0%
BiDAF	73.7%	81.5%
R-Net	76.5%	84.3%
BERT	**87.4%**	**93.2%**
Human	82.3%	91.2%

Table 2.4: Model performance on the SQuAD v1.1 dataset

The *accuracy* metric is calculated using Formula 2.1:

$$Accuracy = \frac{TruePositive + TrueNegative}{TruePositive + FalsePositive + FalseNegative + TrueNegative} \tag{2.1}$$

For any particular piece of data being evaluated, the system will classify it into 2 classes: a fragment that was correctly selected (true positive) or correctly not selected (true negative), and a fragment that was incorrectly selected (false positive) or incorrectly not selected (false negative), which is described in the following 2x2 contingency table as shown in Table 2.5.

		Predicted	
		Negative	Positive
Actual	Negative	True Negative	False Positive
	Positive	False Negative	True Positive

Table 2.5: The confusion matrix for binary classification tasks evaluation

For QA systems evaluation, an issue that can be observed on this metric is the *high rate of true negative* a system can find. For example, for a fact question in which there is one correct answer, everything else are deemed as incorrect and not selected. In this case, due to the high true negative rate, the system could have *high accuracy but meaningless*. To address this issue, the *f measure* is introduced, which is based on the same contingency table using 2 additional measures: *precision* and *recall*. *Precision* (Formula 2.2) is the *percentage of selected answers that are correct*, and *recall* (Formula 2.3) is the opposite measure, which calculates the *percentage of correct answers selected*. By using *precision* and *recall*, according to to [1], the issue of high rate of true negative answer is not relevant anymore.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{2.2}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{2.3}$$

Because of the contrary in natures between the precision and recall metrics, the key concept here is the *trade-off* researchers must do in each measure, looking for the best metrics to evaluate their systems. Most of factoid QA systems should use the recall metric since it does not matter how high the false positive rates are, if there are high true positive rates, the result will be good. On the other hand, for list or definition QA systems, maybe precision metric would be better. To balance this trade-off, the *f measure* is presented, as shown in Formula 2.4

$$F_\beta = \frac{(1+\beta^2) \times precision \times recall}{\beta^2 \times precision + recall} \tag{2.4}$$

With β decides *how much we weight recall with respect for precision*. For example, if $\beta = 0.5$, then recall is weighted half as important as precision. Usually we take $\beta = 1$ to value precision and recall equally, as shown in Formula 2.5

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{2.5}$$

There are other metrics that can be used to evaluate QA systems, such as the *Mean Average Precision* (MAP) metrics, and *Mean Reciprocal Rank* (MRR), used to calculate the answer relevance, which are standard measures for IR.

MAP and MRR are defined on the entire retrieved list of Q queries. MAP is an *averaged metric over ranks*, presented in Formula 2.6.

$$MAP = \frac{\sum_{q=1}^{Q} AveP(q)}{Q} \tag{2.6}$$

where the Average Precision, AveP is calculated over a single query q, presented in Formula 2.7:

$$AveP = \frac{\sum_{k=1}^{n} P(k) \times rel(k)}{|\{relevant\}|} \tag{2.7}$$

where k is the rank of each document and $rel(k)$ is a binary indicator $\in \{0, 1\}$ about whether document k is relevant or not. $P(k)$ is the precision up to document k. The existence of $rel(k)$ decides whether we will count $P(k)$ into AveP if document k is relevant, otherwise not.

On the other hand, MRR only cares about the rank of the 1st relevant document, presented on Formula 2.8.

$$MRR = \frac{1}{Q} \sum_{q=1}^{Q} \frac{1}{rank_q} \tag{2.8}$$

where $rank_q$ is the rank of the first relevant document for query q

Chapter 3

System Architecture

Consider using a *traditional keyword-based search engine* as shown in Figure 3.1. When you have a question in mind and hope that using the search engine, you can have an answer. First, you *formulate the question* in type it in the search box. Then the search engine *returns a set of relevant documents*. After that, *you pick top-ranked documents* and quickly *scan* them, as answer to the question often hides somewhere in the document. Finally, *you combine ideas and understanding* from what you read and *draw a conclusion* about the answer.

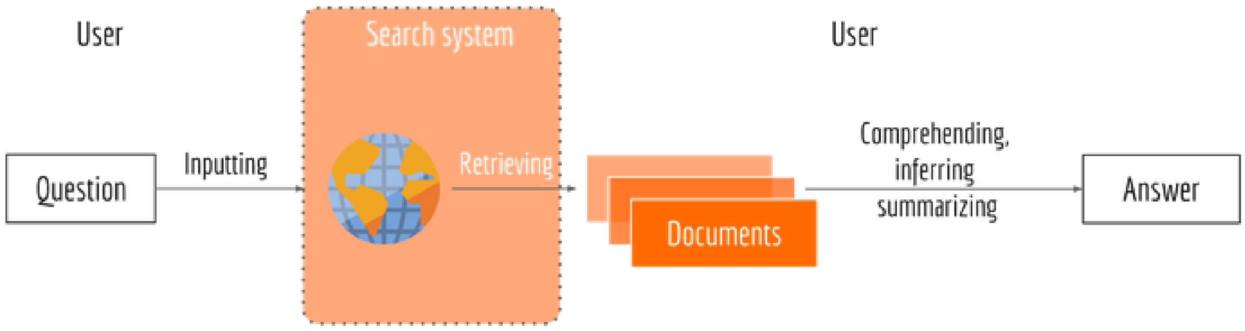


Figure 3.1: From question to answer: Using traditional search engine

Note that in these processes, the *search engine only helps you filtering out irrelevant documents*, *leaving the hard tasks* such as reading, comprehending and summarizing to yourself. In other words, the search engine doesn't actually answer your question, you answer it by yourself.

The work in this thesis is to build a *search system* with *reading comprehension ability* (and in Vietnamese, of course) that can make user experience smoother and more efficient, as all time-consuming tasks from retrieving to inferring and summarizing are left to machines, as shown in Figure 3.2. Such systems can be challenging, that not only *deploy an algorithm that returns query-relevant document*, but also deploy another algorithm with the abilities to follow organization of documents, to draw inferences from the document about it contents, and to summarize and *answer questions with relevant information in the document*.

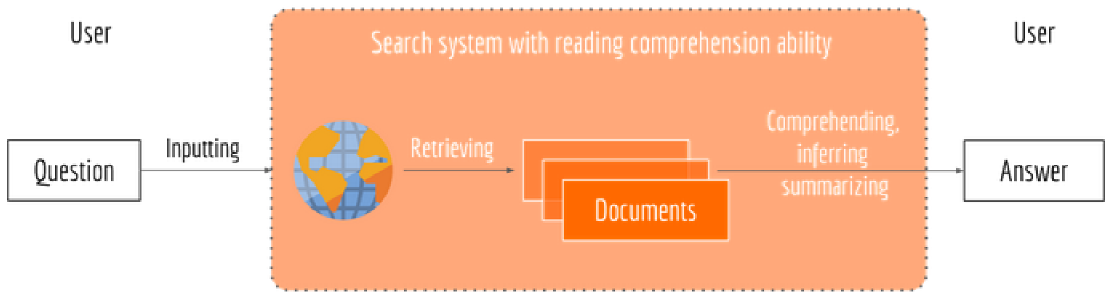

User
Search system with reading comprehension ability
User
Question
Inputting
Retrieving
Documents
Comprehending, inferring summarizing
Answer

Figure 3.2: From question to answer: Using search engine with reading comprehension ability

This system consists of 2 main components:

- A *search engine module* to *retrieve relevant documents*.
- A *deep learning module* that do *reading comprehension* on retrieve documents and *extract answers* to the question.

The system processes are described in Figure 3.3, as follow:

- First, *A corpus is created*, consists of organized documents related to rules and regulations of UIT. The corpus is *indexed* and *stored* as the *knowledge base* of the system.
- Given a question in natural language, the system receives the question as an input to the search engine.
- The search engine receives the query, then *returns top k relevant documents*.
- Top k relevant documents are treated as input to the deep learning module along with the question. The system returns two finite integers $s, e \in [0, L)$, with L being the length of the input document, represent the start position and end position of the answer, and a confidence value. From there, *the answer is extracted*.
- All answers are compared, and the *answer with the highest confidence value* is *returned as the answer to the user*.

In the following section, I will describe in details the structure of each module in the proposed system.

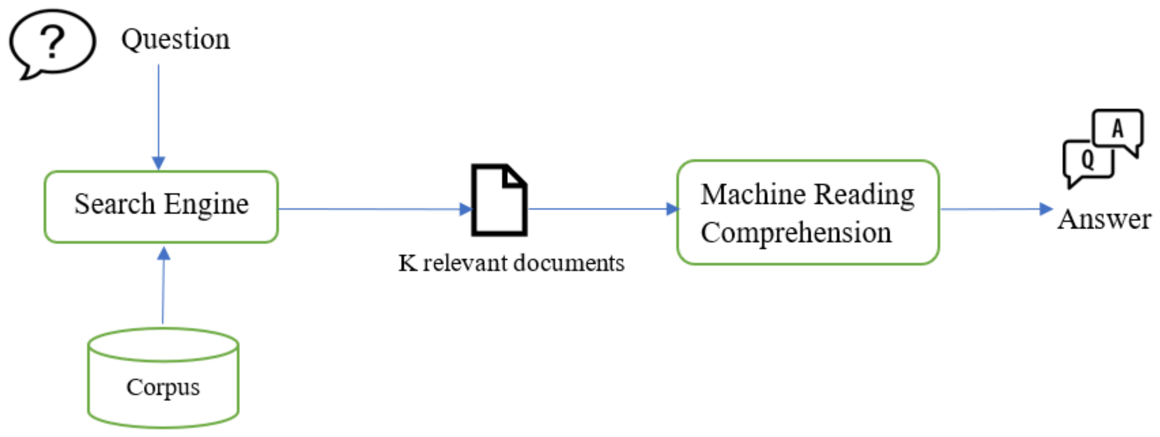


Figure 3.3: System architecture

3.1 Search Engine

The *task of the search module* is, same as any traditional search engines, to *retrieve relevant documents* based on a user' query in natural language. Search engines involve 3 stages, and further described in Figure 3.4:

- *Document indexing* where content-bearing terms are extracted from the document text, and weighing the index terms to enhance retrieval of relevant documents to the user.
- *Question processing* where the question in natural language is transformed into a logical query.
- *Retrieve and ranks documents* with respect to the query according to a *similarity measure*.

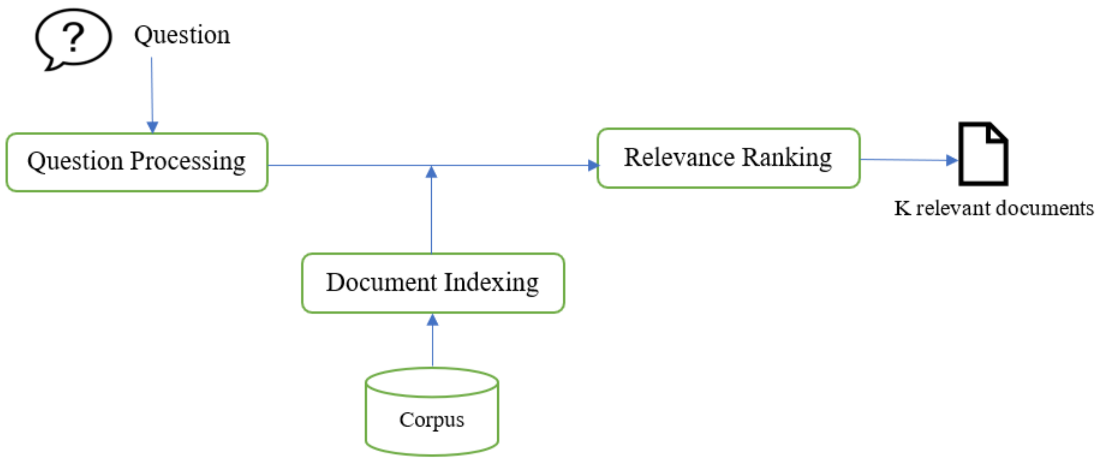


Figure 3.4: Search engine architecture

My search module adapts the *Extended Boolean model* (mentioned in Chapter 2.1) for IR, which will be further described.

Document Indexing

Document indexing involves steps to represent documents in a format for effective retrieval. Document indexing also involves *term weighting*, which is a process to compute and assign a numeric value to each term, in order to *weight its contribution* in distinguish a particular document from others. In this case, I represent documents as vectors as follow:

$$d_j = (w_{1,j}, w_{2,j}, ..., w_{t,j})$$

for document j in the corpus, where each dimension correspond to a separate term. Consider the example:

Document 1: Hello world, how are you?
Document 2: Cat runs behind another cat

Then the vector representations of the 2 documents will look like this:

Document 1: $(w_{Hello}, w_{world}, w_{how}, w_{are}, w_{you})$
Document 2: $(w_{cat}, w_{runs}, w_{behind}, w_{another})$

Where w represents term weight, which is used for relevant ranking. The most popular term-weighting scheme, and also what I will apply to the search module, is the *term-frequency-inverse document frequency* (or tf-idf), which is the combination of 2 scores: *term frequency* (tf) and *inverse document frequency* (idf), and is calculated as follow:

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D)$$

Term frequency $tf(t, d)$ represents the frequency that term t occurs in document d, furnishes a useful measurement for word significance. The simplest choice is to count the number of occurrence of the term t in document d, denoted by $f_{t,d}$, then the simplest tf scheme is

$$tf(t, d) = f_{t,d} \tag{3.1}$$

However, many variants of *term frequency* are introduced, such as:

- *Boolean frequency*: $tf(t, d) = 1$ if t is in d, and 0 otherwise.
- *Term frequency adjusted for document length*: $tf(t, d) = \dfrac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$
- *Logarithmically scaled frequency*: $tf(t, d) = log(1 + f_{t,d})$

- *Augmented frequency*: $tf(t,d) = K + K \cdot \frac{f_{t,d}}{max\{f_{t',d} : t \in d\}}$, which is used to prevent bias towards longer documents. Usually, K = 0.5 is used.

The *Inverse document frequency* is used to measure how much information a term provided, which is calculated as follow:

$$idf(t,D) = log\frac{N}{n_t} \tag{3.2}$$

Where:

N	: total number of documents in the corpus ($N = \|D\|$)
$n_t = \|\{d \in D : t \in d\}\|$	: total number of documents where term t occurs, or in other words: $tf(t,d) \neq 0$. However, if the term t is not in any documents, this will lead to a division-by-zero. In practice, it is common to change the denominator to $1 + \|\{d \in D : t \in d\}\|$

Some variants are introduced, such as:

- *Inverse document frequency smooth*: $idf(t,D) = log\left(1 + \frac{N}{n_t}\right)$
- *Inverse document frequency max*: $idf(t,D) = log\left(\frac{max_{\{t' \in d\}} n_{t'}}{1 + n_t}\right)$
- *Probabilistic inverse document frequency*: $idf(t,D) = log\left(\frac{N - n_t}{n_t}\right)$

The original *Extended Boolean model* used tf-idf weighting scheme for terms weighting, but in this model, I apply the *BM25F weighting scheme*, which is thought as an improvement over *tf-idf* [1], and is used widely by state-of-the-art search services. *BM25F* has its roots in *probabilistic IR*, but can be combined with another model, results in a more robust model. *BM25F* is based on the *Okapi BM25 ranking function*, in which the document is considered to be *composed from several fields* (such as headlines, main text, anchor text) with possibly *different degrees of importance, term relevance saturation and length normalization.*

Given a term t, then the *BM25 term weight* of a document D is:

$$w_{t,D} = idf(t,D) \cdot \frac{f_{t,D} \cdot (k_1 + 1)}{f_{t,D} + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{avgdl}\right)}$$

Where:

[1]BM25 The Next Generation of Lucene Relevance: https://opensourceconnections.com/blog/2015/10/16/bm25-the-next-generation-of-lucene-relevation/

$f_{t,D}$	: Term frequency (tf) of a term t in the document D (defined in Formula 3.1).
$idf(t,d)$	: Inverse document frequency (idf) of a term t (defined in Formula 3.2).
$\|D\|$	: Total length of the document D in words.
$avgdl$	: Average document length in the corpus from which documents are drawn.
k_1	: Chosen parameter, $k_1 \in [1.2, 2]$, usually 2.
b	: Chosen parameter, $b \in [0, 1]$, usually 0.75.

In the case of *BM25F*, terms are weighted according to their *field importance*. Suppose that a document is decomposed into 2 fields of streams: title and body. If we assign a weight of 3 to terms in the title and a weight of 2 to terms in the body, then it is equivalent to replacing the document by itself but this time repeating the title 3 times and the original body twice. Based on this intuition, the *BM25F* weighting scheme is defined as follow:

$$w_{t,D} = idf(t,D) \cdot \frac{\widetilde{f_{t,D}} \cdot (k_1 + 1)}{\widetilde{f_{t,D}} + k_1 \cdot \left(1 - b + b \cdot \frac{|\widetilde{D}|}{\widetilde{avgdl}}\right)} \tag{3.3}$$

Where:

$\widetilde{f_{t,D}}$	: Term frequency (tf) of a term t in the document D), defined by a linear combination of weighted fields of a term $\sum_{s=1}^{s} w_s \dot{f}_{s,t,D}$. For example: a document has 2 fields: title (weighted $w_{title} = 3$) and body (weighted $w_{body} = 2$), and the word “cat" appear once in the title and 10 times in the body. Then the new term frequency is calculated as follow: $\widetilde{f_{t,D}} = w_{title} \cdot f_{title,cat,D} + w_{body} \cdot f_{body,cat,D} = 3 * 1 + 2 * 10 = 23.$
$\|\widetilde{D}\|$	: Total length of the document D in words after considering term weights for each field, and can be thought as term frequency for all terms: $\sum_{t \in D} \widetilde{f_{t,D}}$.
$\widetilde{avgdl}$	: Average document length in the corpus from which documents are drawn after consider term weights for each field.

In short, the process for *document indexing* is presented as follow for each document in the corpus:

- The search module takes in a document and *retrieve keywords* from that document, normally with the help of a tokenizer.
- *Each keywords will be weighted* by a weighting scheme (some of them are defined above), and each keyword along with its weight will be stored in an (forward) index.

Question Processing

Question processing involves steps to transform a user query in natural language into a logical query that can be processed on a machine. Some techniques, such as *relevance feedback* and *query expansion* can be applied to better retrieval. The processes applied to my model are: Keyword extraction, query expansion and query reformulation, which is presented in Figure 3.5

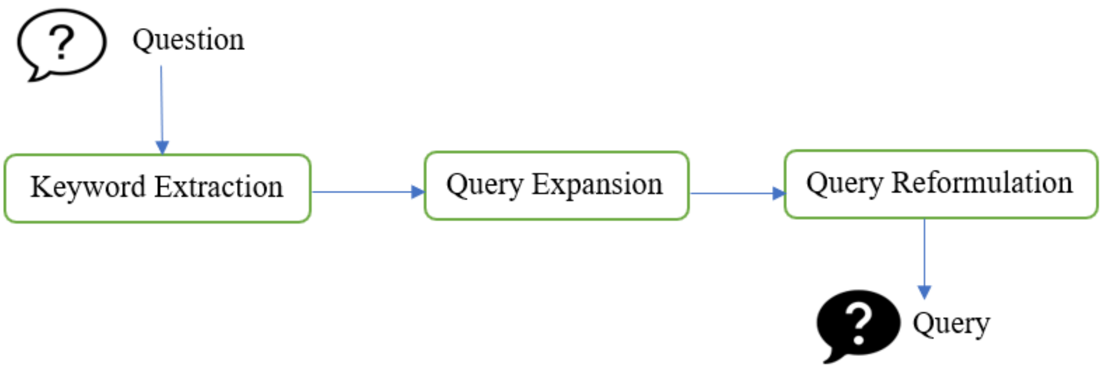


Figure 3.5: Question processing process

Derived from the *Extended Boolean model*, queries are represented as boolean expressions in normal form, described as follow:

$$Q = (W_1 \vee W_2 \vee ...) \wedge ... \wedge (W_i \vee W_{i+1} \vee ...)$$

where W_i is true for a document D_j when the term according to W_i is in the document, or in other words, $t_i \in D_j$. Equivalently, Q could be expressed in disjunctive normal form.

Keyword Extraction

The first step the process is *Keyword extraction*, where *important keywords are extracted*, and unimportant keywords are removed. Consider the question in natural language:

Q: How can a student sign up for researches in the university?

We can observe the following:

- The words ’a’, ’the’ are very common and occur in almost any sentences, but they carry insufficient information for searching purpose, thus they are deemed irrelevant and should be removed. These words are defined as *stopwords*.

- Words that are used to add meaning to another word are also deemed unimportant (pronouns, adjectives, adverbs, preposition, determiners, conjunctions and interjections), since the task of the search module is to retrieve relevant documents, not to find an appropriate answer, therefore complex question understanding is not required. In fact, those words may harm search results, since relevant documents may not contain such words. In the example, those words are: “how", “can", “for", “in".
- After removing non-essential words, that leaves us with 2 types of keywords: noun and verb, that define the subject, the object and the action emerged in the question.
- Assume that the corpus holds information about rules and regulations for a university (in my case is the UIT), the words “university" and “student" are also deemed unimportant, since the whole corpus is related to that university, thus we gained no new information by using such keywords.

Based on those intuitions, the keyword extraction process is as follow:

1. From the user question in natural language, the search module lowercases the question, removes punctuations, then *tokenizes the question into keywords*, and *assigns part-of-speech tagger to each keyword.*

2. From the list of keywords along with part-of-speech tagger for each keyword, the search module *remove stopwords* using a stopword list, pronouns, adjectives, adverbs, preposition, determiners, conjunctions and interjections using the part-of-speech tagger, *leaves with only noun and verb phrases*.

3. From the remaining noun and verb phrases, the search module *assigns a score to each keyword* that represent rareness of each keyword (based on the percentage of documents in which the keyword appear). Keywords that are deemed unimportant according to the context of the corpus (have a score that exceeds x%) are removed.

Query Expansion

For a question, there are various ideas to express the question through the use of synonyms. For example, if a student wants to join a research group, that student can formulate the question as follow:

Q1: How can a student sign up for researches in the university?

Q2: How can a student join a research group?

and if we have a document as follow:

D: .... to join in research groups, students must satisfy the following conditions: ...
The search module will return the document D if question Q2 is used as input, but it is not the case for question Q1, which is called a *false negative* result. To avoid such cases, the task of the second process, *Query expansion*, is to *find synonyms for each keywords*, thus *better formulate* the query and *reduce such false negative cases*.

The process of the *Query expansion* module is described as follow:

1. For each keyword that didn't appear in any document (the assigned score is 0 in the previous step), the search engine finds k synonyms, and assign appearance score for each synonym (percentage of documents that each synonym appears).

2. Similarly in the *Keyword extraction* module, for all keywords and each word' synonyms, the search module *remove unimportant keywords* that have appearance score exceeds x%.

3. Furthermore, the search modules *removes keywords that don't appear in any documents* (appearance score = 0), since those keywords may have negative effects on the retrieval process.

Query Reformulation

From the list of noun and verb phrases with the corresponding synonyms for each keyword, the final task is to join them together and describe them as a boolean expression. The process is as follow:

1. Join each keyword and its synonym together using the OR operator, result in k boolean expressions for k keywords.

2. Join all k boolean expression together into a single query using the AND operator.

For example, if we have 2 keywords “sign up" and “research" with the following synonyms for each keywords: (“recruit", “join", “enlist") and (“analysis", “inquiry", “investigation"), then the resulting query would look like this:

$$Q = (signup \vee recruit \vee join \vee enlist) \wedge (research \vee analysis \vee inquiry \vee investigation)$$

Relevance Ranking

Given a user query in boolean form, and indexed documents, the task of *Relevance Ranking* is to retrieve relevant documents based on their relevance. The *Extended Boolean model* applied the *p-norm* concept in IR to extend the *Standard Boolean model* and *Vector Space model* for relevance ranking, which is described below.

Let consider an simple example, where the query only contains 2 terms, associate with 2 term weights w_1 and w_2, then we can calculate similarity between the query and the document using the following formula (which is based on Euclidean distance):

For the query $q_{or} = K_x \vee K_y$:

$$sim(q_{or}, d) = \sqrt{\frac{w_1^2 + w_2^2}{2}}$$

For the query $q_{and} = K_x \wedge K_y$:

$$sim(q_{and}, d) = 1 - \sqrt{\frac{(1 - w_1)^2 + (1 - w_2)^2}{2}}$$

Note that in order to apply the formula, the local and global weights of terms presented in the document are first normalized by their maximum score. For example, if we use the tf-idf score for term weights, then the normalized weight for term i in document j is:

$$w_{i,j} = \frac{f_{i,j}}{max\, f_{i,j}} \frac{idf_j}{max\, idf_j}$$

Thus term weights adopt values from 0 to 1.

The previous example applied for 2 query terms using $L_2 - norm$. We can generalize the previous 2D extended boolean model example to higher dimensional space for n query terms using $L_p - norm$ which extends the notion of distance to include p-distance ($1 \leq p \leq \infty$) (in other words, $p =$ (number of terms in the query)), as follow:

Consider the query $q_{or} = K_1 \vee K_2 \vee ... \vee K_t$ with t terms:

$$sim(q_{or}, d) = \sqrt[p]{\frac{w_1^p + w_2^p + ... + w_t^p}{t}} \tag{3.4}$$

Consider the query $q_{and} = K_1 \wedge K_2 \wedge ... \wedge K_t$ with t terms:

$$sim(q_{and}, d) = 1 - \sqrt[p]{\frac{(1-w_1)^p + (1-w_2)^p + ... + (1-w_t)^p}{t}} \tag{3.5}$$

We can also evaluate query with both and/or boolean operators. In our case, where the query have the form of:

$$Q = (K_{11} \vee K_{12} \vee ...) \wedge ... \wedge (K_{n1} \vee K_{n2} \vee ...)$$

where $(K_{t1} \vee K_{t2} \vee ...)$ represent a keyword in the natural language question and its associate synonyms, we could treat the expression $(K_{t1} \vee Kt2 \vee ...)$ as part of a single term k, then the query will becomes as follow

$$Q = K_1 \wedge K_2 \wedge ... \wedge K_n$$

. Then, the similarity between the query and a document is calculated as follow, using Formula 3.5:

$$sim(Q, d) = 1 - \sqrt[p]{\frac{(1-w_{K_1})^p + (1-w_{K_2})^p + ... + (1-w_{k_n})^p}{n}}$$

Where W_{k_i} can be calculated using Formula 3.4. Thus:

$$sim(Q, d) = 1 - \sqrt[p]{\frac{\left(1 - \sqrt[p]{\frac{w^p_{k_{1_1}} + w^p_{k_{1_2}} + ... + w^p_{k_{1_t}}}{t}}\right)^p + ... + \left(1 - \sqrt[p]{\frac{w^p_{k_{n_1}} + w^p_{k_{n_2}} + ... + w^p_{k_{n_u}}}{u}}\right)^p}{n}} \tag{3.6}$$

3.2 BERT - Deep Learning Model for Reading Comprehension

BERT, or *Bidirectional Encoder Representations from Transformers* [72], is a general-purpose "language understanding" model that is trained on a large text corpus, which is claimed by the authors to be *unsupervised* and "*deeply bidirectional*", unprecedented to previous works on language understanding. Furthermore, *BERT can be fine-tuned* on various NLP downstream tasks, such as question answering, natural language inference, sentiment analysis, and have achieved *state-of-the-art results* on 11 tasks, including *Machine Reading Comprehension* (described on Table 2.4). In the subsequent subsections, I will describe in details the *BERT original task and architecture*, and the *fine-tuning process* on *Machine Reading Comprehension* task.

Original Task

The *transfer learning* approaches first appear in the field of computer vision, where researchers pretrained a neural network model on a known task (such as ImageNet), and then performing fine-tuning using the trained neural network as the basis of a new purpose-specific model. Researches throughout recent years have shown that transfer learning approaches *can be applied in a wide range of fields*, including *natural language tasks*.

Pretrained language representations (or *word embeddings*, which are the vector representations of words), are *useful* for various NLP tasks that deploy deep learning approaches. Currently, there are two existing strategies for applying pretrained language representations to downstream tasks [72]:

- *Feature-based* approaches, which apply such representations as *additional features*.
- *Fine-tuning approaches*, in which the model is trained on downstream tasks by simply *fine-tuning the pretrained parameters* (Refer as Parameter-based transfer learning).

Current *pretrained language representations* can be either *context-free* (such as WordNet or Glove) (where a word have a single word representation, so “bank" would have the same representation in “bank deposit" and “river bank") or *contextual*. Contextual language models can be *unidirectional* (either using a left-to-right or a right-to-left architecture where a word representation is based on words before or after it) or *shallowly bidirectional* (an example is ELMo) (where left-to-right and right-to-left word representations are concatenated together), as described below.

“The quick brown fox jumps right over the lazy dog"

$P(the \,|\, <s>) * P(quick \,|\, <s>\ the) * P(brown \,|\, <s>\ the\, quick) * \ldots$
$P(dog \,|\, </s>) * P(lazy \,|\, dog\ </s>) * \ldots$

The authors of *BERT* argue that *current techniques severely restrict the power of language representations*, especially for fine-tuning approaches. Such limitations are sub-optimal for sentence-level tasks, but can be *crucial for token-level tasks* such as *Reading Comprehension*, where it is *necessary to incorporate context from both directions*. *BERT* is the first language model that *achieved "deeply bidirectional" word representation*, or it is more accurate to say that it is "nondirectional".

Unlike previous works, where language models are trained in a left-to-right or right-to-left manner, *BERT* deploys two novel unsupervised prediction tasks, defined in 2 subsequent sections.

Task 1: Masked Language Model

Training a *"deeply bidirectional" language model* can be *challenging* because if we build such model like a normal language model, that would create cycles where *words can indirectly see itself* in a multi-layered context, and the *predictions will become trivial*. *BERT* solved this problem by deploying a task that is used in de-noising auto-encoders, where we *mask some of the words* from the input, and ask the language model to *construct those words* from the context, provided by the other, non-masked, words in the sequence. The task is called *Masked Language model* (or *Cloze task*).

In this task, the authors *mask 15% of all tokens* in each sequence at random. In particular, they *feed the input through a deep Transformer encoder* (described further below), and *use the final hidden states* corresponding to the masked positions to *predict the masked words*, exactly what a language model is trained.

```
Input:
The quick brown fox [MASK] right over the [MASK] dog
Label:
[MASK1] = runs; [MASK2] = dog
```

There are *two downsides* to the aforementioned approach:

- Since the [MASK] token is never seen during fine-tuning, there is a *mismatch between pre-training and fine-tuning*. To solve this problem, the author *does not always replace masked words with the actual [MASK] token*. Instead, in 15% of masked words, the training data generator will:
  - *Replace the word with the [MASK] token* in 80% of the time.
  - *Replace the word with a random word* 10% of the time.
  - *Keep the word unchanged* 10% of the time.

  The intuition behind this approach is the following:

  - If masked words are replaced by [MASK] 100% of the time the model wouldn't necessarily produce good token representations for non-masked words, since the model is trained for predicting masked words only.
  - If masked words are replaced by [MASK] 90% of the time and random words 10% of the time, this would teach the model that the observed masked word is never correct.
  - If masked words are replaced by [MASK] 90% of the time and kept the same word 10% of the time, then the model could just trivially copy the non-contextual embedding.
- Since only 15% of tokens are predicted in each batch, more pre-training steps are required for the language model to converge. Although such model are trained slower than traditional left-to-right language model, the improvements far outweigh the increased training cost

The *language model* does not know which words will be asked for prediction or which words have been replaced by random words, the model is *forced to learn distributional contextual representations of every word* so that it may be asked later. Furthermore, words only masked in 15% of the time, *the model's language understanding capability is not harmed.*

Task 2: Next Sentence Prediction

Training such a language model using cloze task is not sufficient for fine-tuning to downstream tasks, since the language model *doesn't understand relationships between sentences*, which is essential for many NLP tasks. To tackle the problem, the authors *apply a simple binary classification approach*, in which 2 sentences A and B are concatenated, the task is to *predict whether B actually comes after A in the original text.*

Specifically, when choosing the sentences A and B for each pretraining example, 50% of the time B is the actual next sentence that follows A, and 50% of the time it is a random sentence from the corpus. Despite its simplicity, the authors have proved that pretraining toward this task is beneficial for various NLP tasks.

```
Input:
[CLS] the man went to [MASK] store [SEP] he bought a gallon [MASK] milk [SEP]
Label: IsNext

Input:
[CLS] the man [MASK] to the store [SEP] penguin [MASK] are flight ##less birds [SEP]
Label: NotNext
```

([SEP] is used to separate 2 sentences)

Model Architecture

BERT's model architecture is a multi-layers, bidirectional *Transformer Encoder* [26], which is the first transduction model developed that relying entirely on self-attention, dispensing with recurrent and convolutional neural networks entirely, thus allowing high-speed training and inference. The *Transformer architecture* is an encoder-decoder model, but the *task of BERT is to learn word representation* only, as we only need to *encode data*, *BERT* only *uses the encoder block* of the *Transformer*

Figure 3.6 described high-level description of the Transformer encoder:

- The input is a sequence of tokens, which are first embedded into vectors and then processed in the network.
- The output is a sequence of vectors of size H, in which each vector corresponds to an input token with the same index

Aside from the architecture of BERT described in Figure 3.6, to be able to pretrain the model for the 2 tasks, several additions are applied as follow:

- For task 1, *a classification layer is added* on top of the encoder output, then the *output vectors* of the classification layer *are multiplied* by the *embedding matrix*, transform them into the vocabulary dimension. Finally, the *probability of each word is calculated using softmax*.
- For task 2, *a simple classification layer is added* that transform the output of the [CLS] token (see the Input Representation subsection for more details) into a 2×1 shaped vector. Finally, *the probability of IsNextSentence is calculated with softmax*.

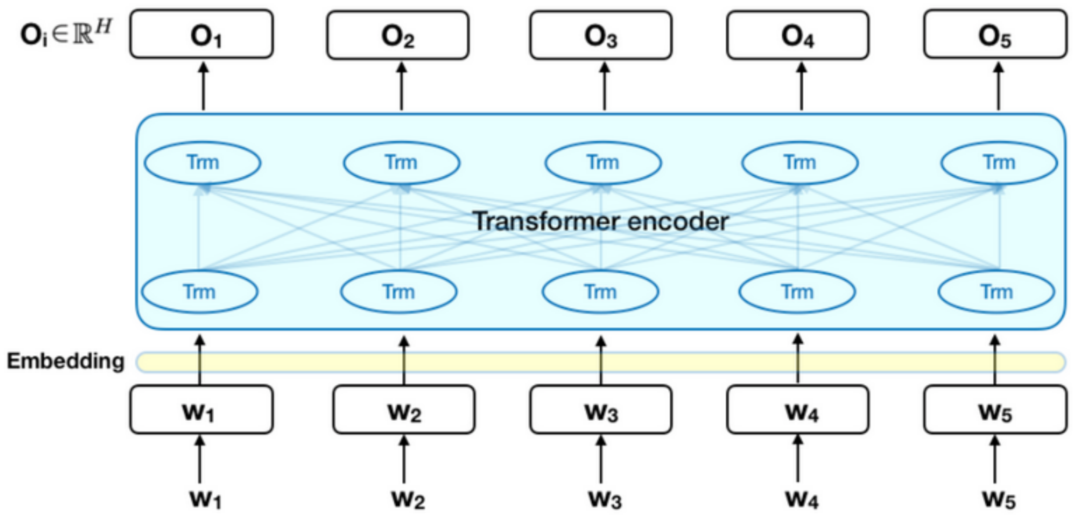


Figure 3.6: BERT architecture

Transformer: Attention Is All You Need

Transformer [73] is a model architecture that *relies entirely on attention mechanism* to draw global dependencies between input and output, derived from the encoder-decoder structure. *The encoder maps* an *input sequence* of symbol representations $(x_1, x_2, ..., x_n)$ *to a sequence of continuous representations* $z = (z_1, z_2, ..., z_n)$. From z, the decoder then *generates an output sequence* $(y_1, y_2, ..., y_m)$ of symbols, one element at a time. At each step the model is *auto-regressive*, consuming the previously generated symbols as additional input when generating the next. In this subsection, the encoder architecture of Transformer will be described, as it is used as a *BERT*'s building block.

The *Transformer* deploys *self-attention* and *point-wise, fully connected layers* for both encoder and decoder, described in the left and right part of Figure 3.7.

The encoder component of the Transfomer consists of $N = 6$ identical layers, where all layers are all identical in structure, yet they do not share weights. Each layer is composed of 2 sub-layers: (1) *A multi-head self-attention mechanism*, and (2) *a simple, position-wise fully connected feed-forward network*. Between each sub-layer is a *residual connection*, which simply adds the input of the current layer to its output via a short-cut connection, followed by *layer normalization*. To facilitate these residual connections, all sub-layers in the model, as well as the embedding layers, produce outputs of dimension d_{model}. In short, an encoder layer's process is described as follow:

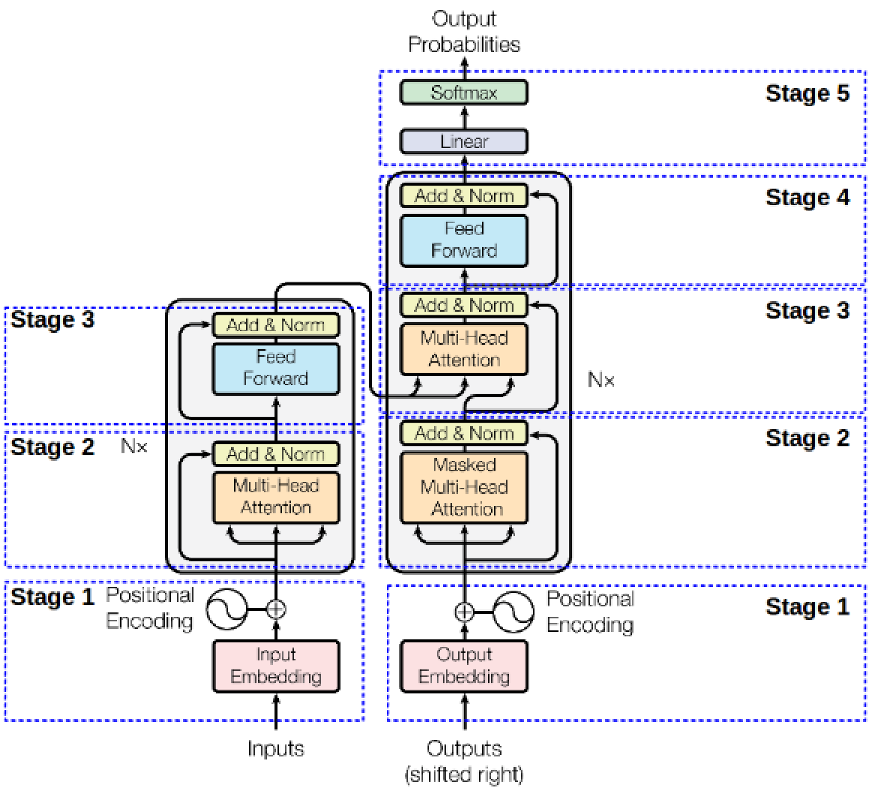


Figure 3.7: Transformer architecture

$$\boxed{\begin{aligned}&stage1\_out = Embedding() + TokenPositionEncoding()\\&stage2\_out = layer\_normalization(multihead\_attention(stage1\_out) + stage1\_out)\\&stage3\_out = layer\_normalization(FFN(stage2\_out) + stage2\_out)\\&\\&out\_enc = stage3\_out\end{aligned}}$$

Each of the layers in the Transformer encoder contains a fully connected feed-forward network, which is applied to each position separately and identically. This consists of two linear transformations with a ReLU activation in between.

$$FFN(x) = max(0, xW_1 + b_1)\dot{W}_2 + b_2$$

The key concept in the Transformer architecture is the *multi-head attention*. (See Chapter 2.1 for a definition of attention). The base attention function used in *multi-head attention* is called the *Scaled Dot-Product Attention*. The input consists of queries and keys of dimension d_k, and values of dimension d_v. We compute the dot products of the query with all keys, divide each by $\sqrt{d_k}$, and apply a softmax function to obtain the weights on the values, as described in Figure 3.8.

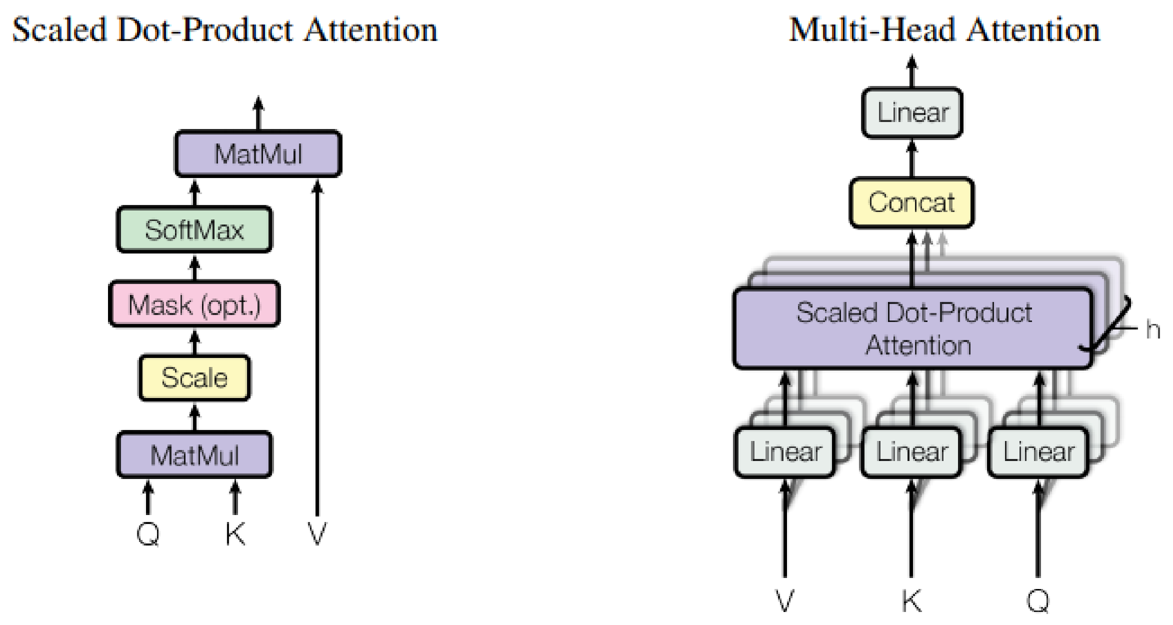


Figure 3.8: Scaled Dot-Product attention & Multi-head attention

In practice, we compute the attention function on a set of queries simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V. We compute the matrix of outputs as:

$$Attention(Q, K, V) = softmax\left(\frac{Q\dot{K}^T}{\sqrt{d_k}}\right)\dot{V}$$

Multi-head attention allows the model to *jointly attend to information from different representation subspaces at different positions*. Each of these heads is a linear transformation of the input representation. This is done so that different parts of the input representations could interact with different parts of the other representation to which it is compared to in the vector space. These attention are concatenated and once again projected, resulting in the final values, as show in Figure 3.8.

$$MultiHead(Q, K, V) = Concat(head_1, head_2, ..., head_h)\dot{W}^O$$

Where $head_i = Attention(Q\dot{W}_i^Q, K\dot{W}_i^K, V\dot{W}_i^V)$, the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ and $W^O \in \mathbb{R}^{h \times d_v \times d_{model}}$. This provides the model to capture various different aspects of the input and improve its expressive ability.

The *Transformer model* uses *multi-head attention* in 3 different ways: attention in “encoder-decoder attention" layers, and self-attention in the encoder and the decoder. I will focus on the use of such attention in the encoder layer. It is called “self-attention" since *all of the keys, values and queries come from the same place*, in this case, the output of the previous layer in the encoder.

One missing thing from the model described so far is that the order of the words in the input sequence does is not taken into account. To address this problem with no recurrence or convolution present, some information about the relative or absolute position of each token in the sequence to the embeddings must be injected.

The Transformer *adds a vector*, called “positional encodings", to each input embedding that follows a specific pattern, which *helps it determine the position of each word, or the distance between different words in the sequence*. The positional encodings have the same dimension d_{model} as the embeddings, so that the two can be summed. Here, *2 sinusoids (sine, cosine functions) of different frequencies* are used:

$$PE_{pos,2i} = sin\left(\frac{pos}{10000^{2i/d_{model}}}\right) \quad and \quad PE_{pos,2i+1} = cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{3.7}$$

Where pos is the position, and i is the dimension. That is, each dimension of the positional encoding corresponds to a sinusoid. The wavelengths form a geometric progression from 2π to $10000 \cdot 2\pi$. The intuition behind why this function is choosed is that it would allow the model to easily learn to attend by relative positions, since for any fixed offset k, PE_{pos+k} can be represented as a linear function of PE_{pos}.

Input Representation

The *input representation of BERT* must be able to *unambiguously represent a single text sentence* (for the masked language model task) and *a pair of sentences* (for the next sentence prediction task and various downstream NLP task, such as question answering, e.g., [Question, Answer] pair) in one token sequence. For a given token, its input representation is constructed by *summing the corresponding token, segment and positional embeddings* as displayed in Figure 3.9.

The details are described as follow:

- *The input is passed in as token using WordPiece embeddings*, which is described below. ## is used to denote split words.

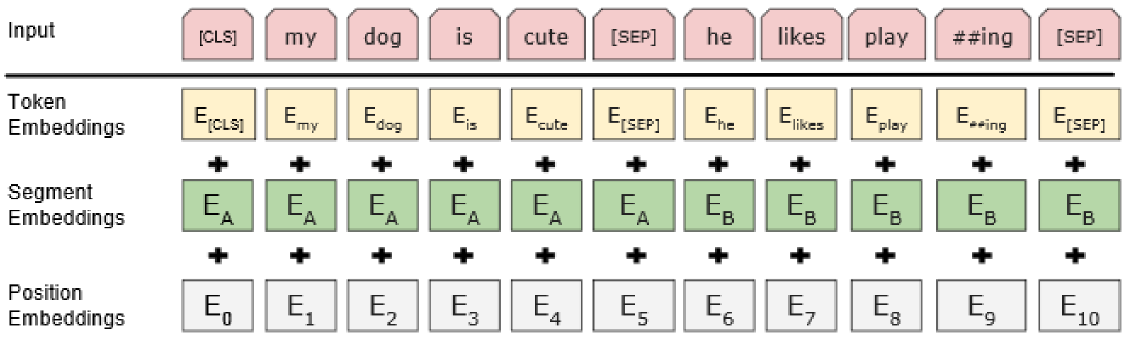


Figure 3.9: BERT input representation

- *Learned positional embeddings is applied* (described in Formula 3.7) for token embeddings with the supported sequence length up to x tokens.
- *The first token of every input sequence is always the special embeddings [CLS]*. The final hidden state (i.e., output of Transformer) corresponding to this token is used as the aggregate sequence representation for classification tasks. For non-classification tasks, this vector is ignored.
- *Sentence pairs are packed together into a single sequence*, whose sentences are *differentiated by a special token [SEP]*. Furthermore, a learned sentence embedding indicating sentence A or sentence B (*segment embeddings*) are applied to every token of the first sentence and the second sentence respectively.

WordPiece [74] is an *unsupervised text tokenizer and detokenizer* mainly for neural network-based systems, where the vocabulary size is predetermined prior to the neural model training. *WordPiece implements subword units* (e.g., byte-pair-encoding (BPE) [75]) and *unigram language model* [76]) with the extension of direct training from raw sentences. Furthermore, *WordPiece is language-independent*.

Machine Reading Comprehension Task on SQuAD

The SQuAD task of span prediction is quite different from the pre-training task, but the modification required to fine-tune BERT on SQuAD task is straightforward. The input is represented as a single packed sequence, [Question, Paragraph]. In the fine-tuning process, BERT learns 2 new vectors for answer prediction: a start vector $S \in \mathbb{R}^H$ and an end vector $E \in \mathbb{R}^H$. Figure 3.10 gives an understanding of the fine-tuning process.

Let the final hidden vector from BERT for the i^{th} input token be denoted as $T_i \in \mathbb{R}^H$. Then the probability of word i being the start/end of the answer span is computed as a dot product

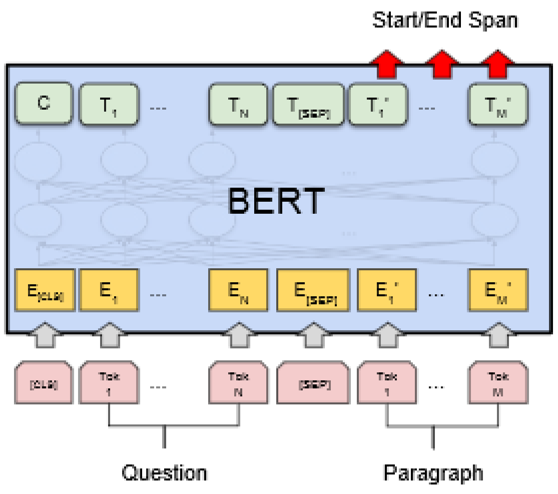


Figure 3.10: BERT fine-tuned on Machine Comprehention Task: SQuAD

between T_i and S or E followed by a softmax, as follow:

$$P^i_{start} = \frac{e^{S \cdot T_i}}{\sum_j e^{S \cdot T_i}} \quad and \quad P^i_{end} = \frac{e^{E \cdot T_i}}{\sum_j e^{E \cdot T_i}}$$

Chapter 4

Implementation & Testing

The purposes of this chapter are to state detailed implementations as well as testing processes, which are described in the following subsections.

4.1 Implementation

This section describes the processes taken to build the QA system, including required steps, settings as well as environments for:

- Building a *search engine* module.
- Prepare datasets for training and testing.
- Fine-tune and optimize BERT for *Vietnamese machine comprehension*.

Note that all of the source code for building this system is built using *Python* as the main programming language, compiled and run on Python 3.x

Search Engine Implementation

For developing the *search engine module*, I used *Whoosh* [1] as the Python library for writing the module, since it is *written in pure Python* that perfectly compatible with other Python modules. Furthermore, compared to other search engines like Lucene, Whoosh is *faster* as

[1] Whoosh documentation: https://whoosh.readthedocs.io/en/latest/

well as *easier to implement in small-scaled domains*, which make it a good search engine library for our module.

The search engine architecture is described in Figure 3.4, which includes 3 submodules, namely *Question Processing*, *Relevance Ranking* and *Document Indexing*. *Whoosh supports basic functions for Relevance Ranking and Document Indexing*, therefore the search engine will be implemented based on such functions. The required processes will be described in the subsequent sections.

Document Indexing

The first step to document indexing is to *define a schema* in which the document indexes will be stored. For the corpus representing regulations of UIT, the following fields are defined for indexing:

- *The document title*, which described the content of the document.
- *The document body*, contains detailed information.

Aside from above fields, I also define other fields, which is not used for indexing (therefore can't be searched).

- *Document path*, contains the file path of the document, which is used for retrieval purpose.
- *Last saved time*, contains the last time that a document has been indexed. Using this field, we only update new indexes, but not changes existing indexes, therefore improve performance.

Aside from the schema, *the algorithm in which keywords are retrieved from the document* (so that index terms can be extracted from each document) also need to be defined. My search module includes the following:

- *A Vietnamese Tokenizer* that tokenizes documents into keywords. My search module applies the ViTokenizer (or pyvi) [2] Tokenizer, which applied Conditional Random Field approach that has achieved an F1 score of 97.86%.
- *A lowercase filter* that transforms words into their lowercase form.
- *A stopwords filter* that removes stopwords (words that appears many times but hold little information), therefore these words won't be indexed and used in searches. A

[2]https://pypi.org/project/pyvi/

Vietnamese stopwords list [3] is required.

Other settings that are applied in the search module are:

- The *term frequency score*, which is the raw count of appearance of each word in the document, described in Formula 3.1, is stored as term weight in the document index. When needed, the *BM25F term weights* (described in Formula 3.3) will be derived from the term frequency.
- The *title field is having a field boost of 1.5*, mean that search terms appeared in the title will be weighted slightly more important than search terms appeared in the text, and is also applied for the BM25F term weighting. (Example of term weighting is described in the Document Indexing section in Chapter 3.1). This approach will be useful for relevance ranking that applies *BM25F ranking function*.
- In practice, the module store document indexes in a *reverse index* (which maps terms to documents instead of mapping documents to terms in a forward index, described in the Document Indexing section in Chapter 3.1), which is mainly for search performance purpose.

After these steps, we can call the indexing function provided by the Whoosh library that can create indexes in a short amount of time.

Query Expansion

The key process of the *Query Expansion* is to *find synonyms of rare keywords*. *The most suitable choice* for this task is to use the *Vietnamese Wordnet*, which is a lexical database of Vietnamese nouns, verbs, adjectives and adverbs grouped into sets of synonyms (or synsets).

However, by the time this thesis is written, both the Vietnamese Wordnet and Asian Wordnet (an alternative Wordnet that includes Vietnamese) are not available, therefore I implement an *alternative approach using Word2Vec*, which is a language model that are used to produce word embeddings. In Word2Vec, every word is described as a vector in an n-dimensional space. In order to find synonyms of a word, the following steps are required:

- Find the vector representation of the the word.
- From the vector representation, find k most similar vectors. There are various measures for calculating similarities, but the most popular measure that the search module

[3]https://github.com/stopwords/vietnamese-stopwords

derived is the cosine similarity, calculated as follow (given 2 vectors A and B that represent 2 words):

$$similarity = cos(\theta) = \frac{A \cdot B}{|A||B|} = \frac{\sum_{i=1}^{n} A_i \dot{B_i}}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Where A_i and B_i are component of vector A and B respectively.

- From those vector representations, retrieve k words as synonyms of the original word, since in Word2Vec, words with similar meaning tend to be grouped together.

The choice of settings for this module is as follow:

- For Vietnamese Word2Vec, I used a pretrained Word2Vec on the Vietnamese Wikipedia with a vector size of 100 and a vocabulary size of 10000.
- For each rare terms, we retrieve $k = 5$ synonyms for each term.
- Terms that have an appearance score larger than $x = 50\%$ (percentage of documents that a term appear) or 0% will be removed.

Query Reformulation

The process for creating a query with the result of the query expansion module is pretty straight forward: the query is built from the list of terms and their associated synonyms based on Whoosh's defined query structure. The final query will look like this:

“(university OR school) AND (graduate)"

Furthermore, a Whoosh-defined *fuzzy search* plugin is also used, which allow “fuzzy term" (terms that don't have to match exactly) to be searched. The fuzzy term will match any similar term within a certain number of “edits” (character insertions, deletions, and/or transpositions – this is called the “Damerau-Levenshtein edit distance”). The prefix distance and postfix distance are 1 and 2 respectively.

Searching

For a question in natural language posed to the search engine, the module *returns 5 relevant documents* with the highest relevance ranking, based on the *BM25F ranking function*.

Datasets Preparation

The dataset is the main challenge of this thesis, since there is *no such reading comprehension dataset in Vietnamese*. In this thesis, the straightforward solution is applied: *to translate the SQuAD dataset from English to Vietnamese*. However, *only using the translated SQuAD dataset yields disappointing results* with the F1 score of 52.08%, with the presence of poorly translated data that neither machine or human can find the right answer. Thus an approach for filtering bad translation is described [77], as follow:

1. *A small-scale, manual labour reading comprehension dataset* is created.
2. *The human-created dataset is used for training a weak QA model* (using BERT).
3. The weak model is used to *predict the translation quality of machine translated data* using the following score:

   $$score = P(y_1 = s) * P(y_2 = e)$$

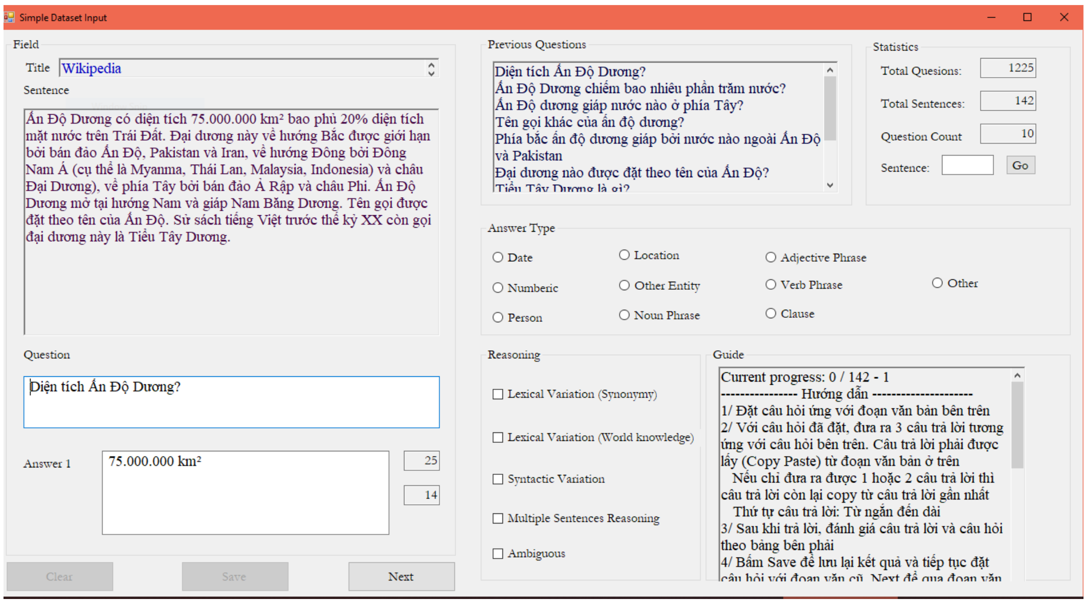
   Where s and e are the start and end position of the translated answer span, and $P(Y)$ is the softmax output of the final hidden layer, represents probability distribution of the start and the end indices of the answer in the passage.
4. After that, *the translated data is filtered by taking $x\%$ question-answer pairs with best scores*.
5. The filtered data is fused with our small-scale data for the final dataset used for training.

The process of translating SQuAD is described as follow:

1. *Each question in the dataset is translated normally* from English to Vietnamese.
2. If the context and answers (for each question in that context) are translated separately, the position of answer span changes or us lost in translation, therefor *each answer span in the context is marked* by some special characters to preserve the span boundary. There are rare cases that answer spans overlap each other, we repeat the sentence where the overlap occurs, and place them separately.
3. *The context is then translated* from English to Vietnamese.

4. We *retrieve each answer* span in the translated context and *mark them as the new answer* for each question. The special character is then removed.

The API used for translation is the *Google Neural Machine Translation*. Aside from the translated data, *a small scaled, manual labor dataset for training, translated quality prediction and testing is also created* based on Wikipedia, with the process similar to SQuAD creation process, with the input form described in Figure 4.1. Furthermore, the UIT regulations documents are also used for machine comprehension dataset.



Title: Wikipedia

Sentence: Ấn Độ Dương có diện tích 75.000.000 km² bao phủ 20% diện tích mặt nước trên Trái Đất. Đại dương này về hướng Bắc được giới hạn bởi bán đảo Ấn Độ, Pakistan và Iran, về hướng Đông bởi Đông Nam Á (cụ thể là Myanma, Thái Lan, Malaysia, Indonesia) và châu Đại Dương), về phía Tây bởi bán đảo Ả Rập và châu Phi. Ấn Độ Dương mở tại hướng Nam và giáp Nam Băng Dương. Tên gọi được đặt theo tên của Ấn Độ. Sử sách tiếng Việt trước thế kỷ XX còn gọi đại dương này là Tiểu Tây Dương.

Question: Diện tích Ấn Độ Dương?

Answer 1: 75.000.000 km² 25 14

Previous Questions:
Diện tích Ấn Độ Dương?
Ấn Độ Dương chiếm bao nhiêu phần trăm nước?
Ấn Độ dương giáp nước nào ở phía Tây?
Tên gọi khác của ấn độ dương?
Phía bắc ấn độ dương giáp bởi nước nào ngoài Ấn Độ và Pakistan
Đại dương nào được đặt theo tên của Ấn Độ?
Tiểu Tây Dương là gì?

Total Quesions: 1225
Total Sentences: 142
Question Count: 10
Sentence:

Current progress: 0 / 142 - 1
---------------- Hướng dẫn ---------------------
1/ Đặt câu hỏi ứng với đoạn văn bản bên trên
2/ Với câu hỏi đã đặt, đưa ra 3 câu trả lời tương ứng với câu hỏi bên trên. Câu trả lời phải được lấy (Copy Paste) từ đoạn văn bản ở trên
Nếu chỉ đưa ra được 1 hoặc 2 câu trả lời thì câu trả lời còn lại copy từ câu trả lời gần nhất
Thứ tự câu trả lời: Từ ngắn đến dài
3/ Sau khi trả lời, đánh giá câu trả lời và câu hỏi theo bảng bên phải
4/ Bấm Save để lưu lại kết quả và tiếp tục đặt câu hỏi với đoạn văn cũ. Next để qua đoạn văn

Figure 4.1: The crowd-facing interface used to collect the dataset

Under limited time, the statistics of the Vietnamese machine comprehension dataset is described as follow:

	QA pairs	Number of paragraphs
Trained data on Wikipedia	599	67
Trained data on UIT regulations	626	75
Translated SQuAD	78978	536
Testing data on Wikipedia	111	14
Testing data on UIT regulations	95	13
Total	80409	705

Table 4.1: Vietnamese machine comprehension dataset statistics

BERT Fine-tune Process

To fine-tune BERT for Vietnamese machine comprehension, I derived from the open source code that the authors of BERT have published to Github [4], which contains source code for pre-training and for fine-tuning on machine comprehension on the English dataset SQuAD (see Chapter 2.4). From the existing source, a fine-tuned model for Vietnamese machine comprehension task is trained, and an inference module used for demonstration purposed is implemented. For testing the validity of the code, I tested BERT on the English SQuAD dataset and have achieved the stated accuracy stated in their paper.

BERT offers 2 models with different size for used: The $BERT_{base}$ model and $BERT_{large}$ model. $BERT_{large}$ produces better results than $BERT_{base}$ (an 3% accuracy boost on SQuAD task), but also comes with high resources usage. It is currently not possible to reproduce most of $BERT_{large}$ results on paper using a GPU with 12GB - 16GB of RAM, since the maximum batch size can be put into memory is too small, therefore we will use $BERT_{base}$ for the work, with the following architecture settings:

- Number of layers (Transformer blocks) $L = 12$.
- Number of parameters: 110M.
- In each Transformer block:
  - Number of self-attention heads (in each Transformer block) $A = 12$.
  - The size of each feed-forward/filter layer $H = 768$

The pretraining corpus is the *Vietnamese Wikipedia*, in which only text passages are extracted, lists, tables and headers are ignored. Unfortunately, I won't be able to do re-produce the pre-train model on Vietnamese, since the process is computationally expensive (which takes 2 weeks on a single preemptible Cloud TPU v2 at a cost of about $500USD). Therefore, I derived from a multilingual pretrained model on 102 languages, including Vietnamese. The authors offer 2 kinds of multilingual pretrained model:

- *"Uncased model"*, in which accent markers are stripped and texts are lowercased.
- *"Cased model"* preserves true case and accent markers.

The *uncased model* may affect the Vietnamese text understanding, but the benefits of reducing the effective vocabulary make up for this.

Below are the hyperparameters used for fine-tuning for Vietnamese machine comprehension:

[4]https://github.com/google-research/bert

- *Adam optimizer* [78] is used, which is a combination of 2 extensions of stochastic gradient descent: (1) Adaptive Gradient Algorithm (AdaGrad) and (2) Root Mean Square Propagation (RMSProp) that computes adaptive learning rates for each parameter, with:
  - Initial learning rate is set at $\alpha = 3e^{-5}$.
  - The exponential decay rate for the first moment (the mean) estimates $\beta_1 = 0.9$.
  - The exponential decay rate for the second-moment (the uncentered variance) estimates $\beta_2 = 0.999$.
  - The learning rate is linearly increased for the first $warmup\_steps$ training steps, then decreases it thereafter proportionally to the inverse square root of the step number. *The $warmup\_steps$ is 10% of total training steps.*
- Along with Adam optimizer, *L2 regularization* is also used with *L2 weight decay of 0.01* and linear decay of the learning rate.
- *Dropout probability of 0.1* for all layers for training.
- *gelu is used as the hidden activation function* instead of the standard relu.
- The *batch size* of 8 will be used.
- The data will be trained for *2 epochs*.
- The *maximum number of input tokens after WordPiece tokenization is 384*. Sequences longer than this will be truncated, and sequences shorter than this will be padded.
- One approach to take into account long inputs is *the sliding window approach*, with the *stride of 128*.
- When testing and inference, *3 best answers* will be returned for each question.
- The *maximum answer length is 500*. (This is needed because the start and end predictions are not conditioned on one another).
- The *maximum token length for the question is 64*. Questions longer than this will be truncated to this length.

The fine-tuning is done using Google Colab, a free cloud service with a 12GB Tesla K80 GPU, which costs several hours.

4.2 Testing

In this section, the system performance and results will be expressed through test cased. Furthermore, I try to gain insights into the type of mistakes my approach produces in an attempt to improve our model in the future.

For each test, the test set will be split according to each question domain (Wikipedia and UIT regulations) to better rate the system on various perspective.

First of all, an experiment is done to (1) test the quality of hand-crafted data and (2) decided whether the $BERT_{uncased}$ or the $BERT_{cased}$ model will be used, as follow: Each BERT model is fine-tuned on the (hand-crafted) Wikipedia dataset, UIT regulations dataset, both dataset combined and the translated SQuAD. The result is displayed as follow, in Table 4.2 and Table 4.3:

Uncased	All Tests		Wikipedia Tests		UIT Tests	
	EM	F1	EM	F1	EM	F1
Starting Point	0	11.44	0	8.09	0	15.88
Training using Wikipedia file only	10.29	28.73	**13.51**	28.02	8.42	29.73
Training using UIT file only	10.70	30.99	9.01	22.84	**9.47**	36.28
Training using both files	**10.67**	**35.43**	11.71	**30.30**	7.37	**38.70**
Training using translated SQuAD	**25.10**	**47.56**	**30.63**	**47.76**	**18.95**	**44.94**

Table 4.2: The $BERT_{uncased}$ model on hand-crafted training dataset

Cased	All Tests		Wikipedia Tests		UIT Tests	
	EM	F1	EM	F1	EM	F1
Starting Point	0	10.38	0	7.93	0	11.93
Training using Wikipedia file only	23.05	44.08	32.43	48.34	12.63	38.91
Training using UIT file only	23.45	44.73	32.43	48.28	12.63	39.41
Training using both files	**27.16**	**49.78**	**36.03**	**53.07**	**15.78**	**43.33**
Training using translated SQuAD	26.33	**52.08**	33.33	51.81	**18.94**	**52.15**

Table 4.3: The $BERT_{cased}$ model on hand-crafted training dataset

Comparing the results of 2 models, we can clearly see that the results of the $BERT_{cased}$ model are far higher than the $BERT_{uncased}$ model, which suggest that *accents are important in Vietnamese* after all. I have decided to use the $BERT_{cased}$ model for fine-tuning. Furthermore, using both the question-answer pairs from different domains doesn't seem to have a negative effect on fine-tuning, therefore I will *use both files for training*, in hope that the model can transfer its knowledge from large-quantity Wikipedia question-answer pairs to the smaller size UIT regulations question-answer pair. However, results of the model when trained with hand-craft data just barely higher than the results of the model when trained with translated data along with bad translation with a variance of 2.0 F1 for the cased model suggests that *the amount of hand-crafted data isn't enough for filtering bad translations*, leaving a room for improvement on future works.

For the process of fusing the hand-craft data and the translated data for training is described in the Datasets Preparation section of Chapter 4.1, an experiment is done that yield results as follow when used for fine-tuning when both datasets are fused for x = 25%, 50%, and 75%.

Cased	All Tests		Wikipedia Tests		UIT Tests	
	EM	F1	EM	F1	EM	F1
Hand-crafted + 25% best-translated	**35.39**	57.87	43.24	59.00	**26.31**	**55.72**
Hand-crafted + 50% best-translated	34.57	60.61	43.24	65.97	23.15	53.90
Hand-crafted + 75% best-translated	34.98	**59.00**	**45.05**	**66.09**	22.10	50.22

Table 4.4: Fine-tuning result on $BERT_{cased}$

The increasing of around 10% across all scores (49.78 F1 using hand-crafted data to 59.00 for all tests, 53.07 F1 to 66.09 for the Wikipedia QA, and 43.33 F1 to 55.72 for the UIT regulations QA) suggests *the approach is reasonable* for Vietnamese question answering. However, when increasing the number of translated training data that is based on Wikipedia, the model seems to be biased by those types of question-answer pairs, therefore the results for the UIT regulations task decreased, despite a continuous increase in the original task (Wikipedia QA in Vietnamese).

Using the fine-tuned BERT model on the UIT regulations task, the 2 modules are combined. From there, we create various test cases to test the performance of the system on each type of question, described below. From the corpus I created from the UIT webpage (126 documents) [5] (note that I exclude tables and forms), 3 documents are taken out and used for the questioning process:

Document 1: Không hoàn tất học phần, Điểm I

> Vì những lý do chính đáng không thể dự thi, kiểm tra (ốm đau, tai nạn, việc gia đình đột xuất,. . .) sinh viên sẽ được xem xét giải quyết cho nhận điểm chưa hoàn tất học phần (hoãn thi), ký hiệu bằng chữ I. Trước khi kết thúc học kỳ, sinh viên phải nộp đơn trình bày rõ lý do không thể hoàn tất học phần cùng các giấy tờ xác nhận cần thiết cho cán bộ giảng dạy học phần đó và P.ĐTĐH. Trường hợp đột xuất và có lý do chính đáng, sinh viên phải nộp trong vòng 3 ngày kể từ ngày thi để được xem xét. Cán bộ giảng dạy phụ trách học phần, khoa/bộ môn quản lý ngành đào tạo và P. ĐTĐH sẽ xem xét và quyết định sinh viên có được nhận điểm I hay không. Nếu không được chấp thuận, sinh viên tự ý bỏ thi sẽ nhận điểm không (0) cho học phần đó. Nếu được nhận điểm I, trong thời gian tối đa là 2 học kỳ chính tiếp theo, sinh viên phải làm đơn đăng ký thi lại học phần đó. Sau khi thi, điểm I sẽ được đổi thành điểm mà sinh viên đạt được. Ngược lại, qua

[5]https://daa.uit.edu.vn/content/qui-che-qui-dinh-qui-trinh-dao-tao-dai-hoc

hai học kỳ, nếu sinh viên không đăng ký thi lại thì điểm I sẽ bị đổi thành điểm không (0). Sinh viên nhận điểm I trong học kỳ nào sẽ không được xét học bổng khuyến khích của học kỳ đó.

Document 2: Tín chỉ học phí TCHP

Tín chỉ học phí (TCHP) là đơn vị dùng để lượng hóa chi phí của các hoạt động giảng dạy tính cho từng học phần. Số TCHP của mỗi học phần được xác định căn cứ vào đề cương và cách thức tổ chức học phần; cụ thể như sau: Phần giảng dạy lý thuyết tại lớp: 15 tiết tương đương với 1 TCHP. Phần giảng dạy thực hành, thí nghiệm, thảo luận: 15 tiết tương đương với 1 TCHP. Một số học phần đặc biệt được xác định số TCHP riêng như: học phần Giáo dục quốc phòng-An ninh, Giáo dục thể chất, Thực tập doanh nghiệp, Khóa luận tốt nghiệp, Môn học đồ án bao gồm chuyên đề nghiên cứu khoa học, seminar,... sẽ được Hiệu trưởng ban hành. Có 3 loại TCHP: 1. TCHP học lại (TCHPHL): là tín chỉ học phí của học phần học lại. 2. TCHP học cải thiện (TCHPCT): là tín chỉ học phí của học phần học cải thiện. 3. TCHP học mới (TCHPHM): là tín chỉ học phí của học phần mới học lần đầu. Đầu năm học, Trường sẽ công bố mức học phí cho mỗi loại TCHP.

Document 3: Xét duyệt đề tài khóa luận tốt nghiệp KLTN

Khoa chịu trách nhiệm xét duyệt đề tài KLTN. Việc xét duyệt phải đảm bảo các yêu cầu sau: Đề tài có thể do giảng viên hay các doanh nghiệp đề xuất. Đề tài và đề cương nghiên cứu phải đảm bảo: Tính khoa học, tính thực tiễn của đề tài; Tính khả thi của đề tài (điều kiện thực hiện, thời gian, khối lượng...); Đề tài phù hợp với mục tiêu đào tạo và chuẩn đầu ra của ngành đào tạo tương ứng; Các đề tài đã qua xét duyệt phải được công bố để sinh viên đăng ký. Mỗi đề tài KLTN được giao cho từ 1 đến 2 SV thực hiện với sự phân công trách nhiệm cụ thể cho từng SV có ghi rõ trong đề cương nghiên cứu. Mỗi đề tài KLTN phải do tối thiểu 1 CBHD có học vị từ thạc sĩ trở lên hướng dẫn, nếu đề tài do doanh nghiệp đề xuất thì có thể có thêm 1 đại diện của doanh nghiệp đồng hướng dẫn. CBHD đề tài không được là người có quan hệ ruột thịt với SV thực hiện (cha, mẹ, vợ, chồng, anh, chị, em ruột). CBHD được hướng dẫn tối đa 5 đề tài hoặc 10 SV trong mỗi đợt giao đề tài KLTN. Đối với các đề tài có sử dụng dữ liệu hoặc tư liệu của doanh nghiệp, CBHD có trách nhiệm kiểm tra việc thực hiện các qui định về tác quyền của đơn vị cung cấp dữ liệu hoặc tư liệu liên quan.

For each test case, 3 questions will be asked. The system performance is noted and described as follow:

Test case 1: Factoid questions

- Question 1.1: Sinh viên không thể dự thi phải nộp đơn trước bao nhiêu ngày để xét?
- Question 1.2: TCHP là gì?
- Question 1.3: Ai có thể đề xuất đề tài tốt nghiệp?

For question 1.1, the model returns the correct document with the highest ranking and return the correct answer: “3 ngày" with confidence of 92%.

For the second question 1.2, the correct document is returned with the highest score, and the most confident answer is “đơn vị dùng để lượng hóa chi phí của các hoạt động giảng dạy tính cho từng học phần", which is correct.

For question 1.3, the expected document is returned with the highest ranking, and the correct answer “giảng viên hay các doanh nghiệp" is also returned with 100% accuracy. Furthermore, another document is returned with the answer of “Khoa" in the context of “Khoa có thể đề xuất bổ sung các tiêu chuẩn xét chọn làm KLTN của riêng Khoa", which further explain another possibility, which is good.

From this result, I conclude that *the model can easily answer factoid questions*.

Test case 2: Questions with lexical variation (synonym)/ syntactic variations

- Question 2.1: Nếu không được đồng ý mà em không thi thì sẽ ra sao?
- Question 2.2: 1 tín chỉ học phần bằng bao nhiêu tiết học?
- Question 2.3: Ai xác nhận đề tài tốt nghiệp?

For the first question 2.1, the model returns an unrelated document about university transfer. After I rephrase the question as: “Nếu không được chấp thuận mà em không thi thì sẽ ra sao?", the system returns the correct answer and document with highest confidence “sinh viên tự ý bỏ thi sẽ nhận điểm không (0) cho học phần đó".

For question 2.2, the correct document and answer are returned with the highest confidence: “15". Furthermore, another document with the same information is also returned.

For question 2.3, no document is returned, but when the question is repharsed as “Ai chấp nhận đề tài tốt nghiệp?", the correct document and answer are returned with the highest ranking. Change the question to “Ai xác nhận khóa luận tốt nghiệp?" and an irrelevant document is returned.

I suspect that the reason behind this shortcoming come from the lack of ability to return synonyms by the Word2Vec leads lack to a weak query expansion module that relevant documents can't be found. To improve the query expansion process, *a better synonym searcher is required*, such as an ontology for university purpose in Vietnamese, or the Vietnamese Wordnet.

Test case 3: Questions require world knowledge

- Question 3.1: Sau khi được miễn thi thì khi nào em bị đánh rớt?
- Question 3.2: Ai quyết định tiền phải đóng?
- Question 3.3: Ai là người có quan hệ ruột thịt với sinh viên?

In this test case, the system can't find the good enough answer for all 3 questions (In question 3.1, no document is returned; In question 3.2, a document with more matched keywords is returned with a good enough answer, but the document I expected is not returned, which require world knowledge; In question 3.3, the expected document is returned as the second-highest ranking document, but the correct answer can't be inferred by BERT, with the answer of "CBHD" when the correct answer is "cha, mẹ, vợ, chồng, anh, chị, em ruột").

Those results are expected, since the BERT model *can only understand knowledge written in the context, not something we referred as "common sense"*, and the search module can only match document by terms. There are approaches to fuse common sense into a deep learning model, but we are still far off from such an achievement.

Test case 4: Questions require multiple sentence reasoning

- Question 4.1: Khi bị tai nạn không thi và đăng kí thi lại thì sao?
- Question 4.2: Tín chỉ học phí gồm những gì ngoài tín chỉ học lại và tín chỉ học mới?
- Question 4.3: Thạc sĩ được nhận bao nhiêu đề tài tối đa?

For question 4.1, the expected document and a good enough answer: "điểm I sẽ bị đổi thành điểm không (0)" are returned with the highest confidence, which resulted in a passing case, suggest that *the model can solve easy questions involve inference through multiple sentences*.

For question 4.2, the expected document is returned, but the correct answer "TCHP học cải thiện (TCHPCT)" can't be inferred from the model. Other tests suggest that the fine-tuned BERT model is weak against inference from lists. More training data on this type of question can solve the problem.

For the final question 4.3, the correct document is returned, but the fine-tuned BERT returns the answer of "1", which is incorrect (the correct answer is " tối đa 5 đề tài"). Rephrase the question lead to the same answer, which suggests that *the system is still weak in question understanding and inference through multiple sentence. More training data is required to achieve better results.* Furthermore, the system fails to find the answer inferred from a very large sentences span, due to a lack of memory and computation required to store the data. Better hardwares are required for better inference.

Test case 5: Questions with long answers.

- Question 5.1: Sau khi nhận điểm I thì sao?
- Question 5.2: Định nghĩa 3 loại tín chỉ học phí?
- Question 5.3: Yêu cầu về đề tài khóa luận?

For question 5.1, no relevant document is returned, most likely due to a lack of matching terms. For question 5.2, the expected document is returned, but the system can only return the definition for the first credit. Question 5.3 poses correct document and answer with the highest confidence.

Since the model is mostly trained on Wikipedia documents-based question-answer pairs, in which short answers are many, *the system lacks skills to infer long answers from the sentences. One approach to address this problem* is to balance the number of Wikipedia question-answer pairs (most of which is on the translated dataset) and the number of UIT regulations question-answer pairs (in which long answers are many) using *data augmentation.*

Chapter 5

Conclusion

Under limited time, I have successfully build a question answering system with reading comprehension ability that can somehow ease the current workloads of UIT staffs as well as troubleshoots UIT students with their troubles, so they can focus more on their study and career path.

Though the results aren't high, and there are rooms for more improvements, I have learned many things during the work of this thesis. From almost zero experience in machine learning and NLP, I have learned various algorithms and approaches, optimization tricks and best practices when building a machine learning model, as well as problem-solving skills when facing obstacles during development phrases.

Although the system is not excellent enough, but having trained on a general-purpose dataset, I believe that the system can be used not only in UIT but also in any other organizations, and it can also be applied to many problems require reading comprehension skill. The code as well as the Vietnamese dataset for this system are open-source and available at my Gitlab page `gitlab.com/phateopera/UIT_Regulations_QAHelper` in hope that the system can be developed for better results, therefore advancing the question answering field in Vietnamese.

Some future directions on this QA system are described as follow:

- The result on hand-craft dataset suggests that the amount of data is not enough for fine-tuning, though it poses positive results when fused with filtered translated data. With more hand-craft data serves at a better base for translation quality prediction and fine-tuning, I believe that the model will yield better results.

- Another approach to find synonyms in Vietnamese is required for better performance and accuracy, since Word2Vec is slow and unreliable.
- With a little training data, the system can be applied for a wide range of domains that involves reading comprehension.
- In the original BERT implementation, the fine-tuning process involves learning 2 additional vectors by using softmax, but more complex layers can be made in between to better simulate the inference process, therefore improve predictions.
- Provide mobile support and speech recognition for the questioning process for better convenient. Furthermore, a speech recognition system can be combined with this QA system for speech understanding (such as question answering based on professor's lecture).

Bibliography

[1] M. A. C. Soares and F. S. Parreiras, "A literature review on question answering techniques, paradigms and systems," *Journal of King Saud University-Computer and Information Sciences*, 2018.

[2] X. Yao, "Feature-driven question answering with natural language alignment," Ph.D. dissertation, Johns Hopkins University, 2014.

[3] D. A. Ferrucci, "Introduction to "this is watson"," *IBM Journal of Research and Development*, vol. 56, no. 3.4, pp. 1:1–1:15, May 2012.

[4] B. F. Green Jr, A. K. Wolf, C. Chomsky, and K. Laughery, "Baseball: an automatic question-answerer," in *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*. ACM, 1961, pp. 219–224.

[5] W. A. Woods and R. Kaplan, "Lunar rocks in natural english: Explorations in natural language question answering," *Linguistic structures processing*, vol. 5, pp. 521–569, 1977.

[6] A. Mishra and S. K. Jain, "A survey on question answering systems with classification," *Journal of King Saud University-Computer and Information Sciences*, vol. 28, no. 3, pp. 345–361, 2016.

[7] M. Sanderson and W. B. Croft, "The history of information retrieval research," *Proceedings of the IEEE*, vol. 100, no. Special Centennial Issue, pp. 1444–1451, 2012.

[8] M. Fridah Nyamisa, "A survey of information retrieval techniques," *Advances in Networks*, vol. 5, p. 40, 01 2017.

[9] H. P. Luhn, "A statistical approach to mechanized encoding and searching of literary information," *IBM Journal of research and development*, vol. 1, no. 4, pp. 309–317, 1957.

[10] G. Salton, E. A. Fox, and H. Wu, "Extended boolean information retrieval," Cornell University, Tech. Rep., 1982.

[11] E. A. Fox and S. Sharan, "A comparison of two methods for soft boolean operator interpretation in information retrieval," 1986.

[12] C. D. Paice, "Soft evaluation of boolean search queries in information retrieval systems," *Information Technology Research Development Applications*, vol. 3, no. 1, pp. 33–41, 1984.

[13] G. Salton, A. Wong, and C.-S. Yang, "A vector space model for automatic indexing," *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, 1975.

[14] S. M. Wong, W. Ziarko, and P. C. Wong, "Generalized vector spaces model in information retrieval," in *Proceedings of the 8th annual international ACM SIGIR conference on Research and development in information retrieval.* ACM, 1985, pp. 18–25.

[15] J. Becker and D. Kuropka, "Topic-based vector space model," in *Proceedings of the 6th international conference on business information systems*, 2003, pp. 7–12.

[16] A. Polyvyanyy and D. Kuropka, "A quantitative evaluation of the enhanced topic-based vector space model," 2007.

[17] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American society for information science*, vol. 41, no. 6, pp. 391–407, 1990.

[18] S. E. Robertson and K. S. Jones, "Relevance weighting of search terms," *Journal of the American Society for Information science*, vol. 27, no. 3, pp. 129–146, 1976.

[19] S. Robertson, H. Zaragoza *et al.*, "The probabilistic relevance framework: Bm25 and beyond," *Foundations and Trends® in Information Retrieval*, vol. 3, no. 4, pp. 333–389, 2009.

[20] C. J. Van Rijsbergen, "A non-classical logic for information retrieval," *The computer journal*, vol. 29, no. 6, pp. 481–485, 1986.

[21] G. Amati and C. J. Van Rijsbergen, "Probabilistic models of information retrieval based on measuring the divergence from randomness," *ACM Transactions on Information Systems (TOIS)*, vol. 20, no. 4, pp. 357–389, 2002.

[22] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[23] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," *arXiv preprint arXiv:1606.05250*, 2016.

[24] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, "Teaching machines to read and comprehend," in *Advances in Neural Information Processing Systems*, 2015, pp. 1693–1701.

[25] A. P. Parikh, O. Täckström, D. Das, and J. Uszkoreit, "A decomposable attention model for natural language inference," *arXiv preprint arXiv:1606.01933*, 2016.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[27] J. Weston, S. Chopra, and A. Bordes, "Memory Networks," *ArXiv e-prints*, Oct. 2014.

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

[29] S. Sukhbaatar, J. Weston, R. Fergus *et al.*, "End-to-end memory networks," in *Advances in neural information processing systems*, 2015, pp. 2440–2448.

[30] A. Kumar, O. Irsoy, P. Ondruska, M. Iyyer, J. Bradbury, I. Gulrajani, V. Zhong, R. Paulus, and R. Socher, "Ask me anything: Dynamic memory networks for natural language processing," in *International Conference on Machine Learning*, 2016, pp. 1378–1387.

[31] S. Wang and J. Jiang, "Machine comprehension using match-lstm and answer pointer," *arXiv preprint arXiv:1608.07905*, 2016.

[32] A. W. Yu, D. Dohan, M.-T. Luong, R. Zhao, K. Chen, M. Norouzi, and Q. V. Le, "Qanet: Combining local convolution with global self-attention for reading comprehension," *arXiv preprint arXiv:1804.09541*, 2018.

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

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

[35] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 2692–2700.

[36] A. Sordoni, P. Bachman, A. Trischler, and Y. Bengio, "Iterative alternating neural attention for machine reading," *arXiv preprint arXiv:1606.02245*, 2016.

[37] A. Trischler, Z. Ye, X. Yuan, and K. Suleman, "Natural language comprehension with the epireader," *arXiv preprint arXiv:1606.02270*, 2016.

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

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

[40] C. Xiong, V. Zhong, and R. Socher, "Dynamic coattention networks for question answering," *arXiv preprint arXiv:1611.01604*, 2016.

[41] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi, "Bidirectional attention flow for machine comprehension," *arXiv preprint arXiv:1611.01603*, 2016.

[42] Y. Yu, W. Zhang, K. Hasan, M. Yu, B. Xiang, and B. Zhou, "End-to-end reading comprehension with dynamic answer chunk ranking," *CoRR, abs/1610.09996*, 2016.

[43] K. Lee, S. Salant, T. Kwiatkowski, A. Parikh, D. Das, and J. Berant, "Learning recurrent span representations for extractive question answering," *arXiv preprint arXiv:1611.01436*, 2016.

[44] Z. Yang, B. Dhingra, Y. Yuan, J. Hu, W. W. Cohen, and R. Salakhutdinov, "Words or characters? fine-grained gating for reading comprehension," *arXiv preprint arXiv:1611.01724*, 2016.

[45] Z. Wang, H. Mi, W. Hamza, and R. Florian, "Multi-perspective context matching for machine comprehension," *arXiv preprint arXiv:1612.04211*, 2016.

[46] Y. Shen, P.-S. Huang, J. Gao, and W. Chen, "Reasonet: Learning to stop reading in machine comprehension," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 1047–1055.

[47] W. Wang, N. Yang, F. Wei, B. Chang, and M. Zhou, "Gated self-matching networks for reading comprehension and question answering," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2017, pp. 189–198.

[48] O. Kuchaiev and B. Ginsburg, "Factorization tricks for lstm networks," *arXiv preprint arXiv:1703.10722*, 2017.

[49] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," *arXiv preprint arXiv:1701.06538*, 2017.

[50] Ł. Kaiser and S. Bengio, "Can active memory replace attention?" in *Advances in Neural Information Processing Systems*, 2016, pp. 3781–3789.

[51] N. Kalchbrenner, L. Espeholt, K. Simonyan, A. v. d. Oord, A. Graves, and K. Kavukcuoglu, "Neural machine translation in linear time," *arXiv preprint arXiv:1610.10099*, 2016.

[52] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," *arXiv preprint arXiv:1705.03122*, 2017.

[53] W. Yin, S. Ebert, and H. Schütze, "Attention-based convolutional neural network for machine comprehension," *arXiv preprint arXiv:1602.04341*, 2016.

[54] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big Data*, vol. 3, no. 1, p. 9, 2016.

[55] S. J. Pan, Q. Yang *et al.*, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[56] V. M. Tran, V. D. Nguyen, O. T. Tran, U. T. T. Pham, and T. Q. Ha, "An experimental study of vietnamese question answering system," in *2009 International Conference on Asian Language Processing*. IEEE, 2009, pp. 152–155.

[57] E. Agichtein and L. Gravano, "Snowball: Extracting relations from large plain-text collections," in *Proceedings of the fifth ACM conference on Digital libraries*. ACM, 2000, pp. 85–94.

[58] M.-V. Tran, D.-T. Le, X.-T. Tran, and T.-T. Nguyen, "A model of vietnamese person named entity question answering system," in *Proceedings of the 26th Pacific Asia Conference on Language, Information, and Computation*, 2012, pp. 325–332.

[59] D. Q. Nguyen, D. Q. Nguyen, and S. B. Pham, "A vietnamese question answering system," in *Knowledge and Systems Engineering, 2009. KSE'09. International Conference on*. IEEE, 2009, pp. 26–32.

[60] D. T. Nguyen and H. Q.-T. Luong, "Document searching system based on natural language query processing for vietnam open courseware library," *arXiv preprint arXiv:0912.1829*, 2009.

[61] D. T. Son and D. T. Dung, “Apply a mapping question approach in building the question answering system for vietnamese language,” in *Journal of Engineering Technology and Education and The 2012 International Conference on Green Technology and Sustainable Development (GTSD2012)*, 2012, pp. 380–386.

[62] H.-T. Duong and B.-Q. Ho, “A vietnamese question answering system in vietnam’s legal documents,” in *IFIP International Conference on Computer Information Systems and Industrial Management*. Springer, 2014, pp. 186–197.

[63] M. Richardson, C. J. Burges, and E. Renshaw, “Mctest: A challenge dataset for the open-domain machine comprehension of text,” in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013, pp. 193–203.

[64] J. Berant, V. Srikumar, P.-C. Chen, A. Vander Linden, B. Harding, B. Huang, P. Clark, and C. D. Manning, “Modeling biological processes for reading comprehension,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1499–1510.

[65] D. Hewlett, A. Lacoste, L. Jones, I. Polosukhin, A. Fandrianto, J. Han, M. Kelcey, and D. Berthelot, “Wikireading: A novel large-scale language understanding task over wikipedia,” *arXiv preprint arXiv:1608.03542*, 2016.

[66] D. Paperno, G. Kruszewski, A. Lazaridou, Q. N. Pham, R. Bernardi, S. Pezzelle, M. Baroni, G. Boleda, and R. Fernández, “The lambada dataset: Word prediction requiring a broad discourse context,” *arXiv preprint arXiv:1606.06031*, 2016.

[67] T. Onishi, H. Wang, M. Bansal, K. Gimpel, and D. McAllester, “Who did what: A large-scale person-centered cloze dataset,” *arXiv preprint arXiv:1608.05457*, 2016.

[68] A. Trischler, T. Wang, X. Yuan, J. Harris, A. Sordoni, P. Bachman, and K. Suleman, “Newsqa: A machine comprehension dataset,” *arXiv preprint arXiv:1611.09830*, 2016.

[69] T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng, “Ms marco: A human generated machine reading comprehension dataset,” *arXiv preprint arXiv:1611.09268*, 2016.

[70] M. Joshi, E. Choi, D. S. Weld, and L. Zettlemoyer, “Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension,” *arXiv preprint arXiv:1705.03551*, 2017.

[71] P. Rajpurkar, R. Jia, and P. Liang, “Know what you don’t know: Unanswerable questions for squad,” *arXiv preprint arXiv:1806.03822*, 2018.

[72] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[73] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[74] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[75] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," *arXiv preprint arXiv:1508.07909*, 2015.

[76] T. Kudo, "Subword regularization: Improving neural network translation models with multiple subword candidates," *arXiv preprint arXiv:1804.10959*, 2018.

[77] K. Lee, K. Yoon, S. Park, and S.-w. Hwang, "Semi-supervised training data generation for multilingual question answering," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, 2018.

[78] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.