MASTER OF SCIENCE
IN ENGINEERING

Haute Ecole Spécialisée
de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts
Western Switzerland

Master of Science HES-SO in Engineering
Av. de Provence 6
CH-1007 Lausanne

Master of Science HES-SO in Engineering

Orientation: Information and Communication Technologies (ICT)

Deepening Project: GenBot

Author:

Romain Claret
romain.claret@master.hes-so.ch

Under the direction of:
Prof. Dr. Jean Hennebert
HES-SO//Fribourg
Institute of Complex Systems (iCoSys)

External expert:
Dr. Christophe Gisler

Lausanne, HES-SO//Master, June 3, 2019

Accepted by the HES-SO//Master (Switzerland, Lausanne) on a proposal from:

Prof. Dr. Jean Hennebert, deepening project supervisor
Dr. Christophe Gisler, iCoSys, main expert

Place, date: ____

Prof. Dr. Jean Hennebert	M. Philippe Joye
Supervisor	ICT MRU Leader at HES-SO//Fribourg

Contents

Contents

Acronyms

AGI
Artificial General Intelligence.

AI
Artificial Intelligence.

AIML
Artificial Intelligence Markup Language.

ANI
Artificial Narrow Intelligence.

ANN
Artificial Neural Networks.

AWS
Amazon Web Services.

BD
Big Data.

CBOW
Continuous Bag of words.

DL
Deep Learning.

DM
Data Mining.

DNN
Deep Neural Networks.

DP
Deepening Project.

FAQ
Frequently Asked Questions.

Acronyms

ICT
: Information and Communications Technologies.

IR
: Information Retrieval.

ML
: Machine Learning.

MRU
: Master Research Units.

NL
: Natural Language.

NLG
: Natural Language Generation.

NLP
: Natural Language Processing.

NLU
: Natural Language Understanding.

NN
: Neural Network.

RNN
: Recurrent Neural Network.

Sci-Fi
: Science Fiction.

Seq2Seq
: Sequence to Sequence.

SNN
: Shallow Neural Network.

TF-IDF
: Term Frequency-Inverse Document Frequency.

Abstract

In the scope of this DP, and as the technology of NLP is in constant evolution, the author will be focusing on the exploration of the word embedding algorithm Word2Vec, which is, at the beginning of 2019, commonly used as a foundation for DNN Chatbots. As a result of this project, the student is demonstrating what the Word2Vec technology, its extensions, and its applications are.

Keywords: Word Embedding, Word2Vec, NLP, Natural Language Understanding (NLU), Machine Learning (ML), Data Engineering, Conversational Agent, Chatbot, Generic

Chapter 1

Introduction

Start of 2019, chatbots are everywhere but very limited to narrow tasks, and are, in most cases, sequences of if-else conditions resulting in a very weak Artificial Intelligence (AI). Indeed, hard-coded connections are requiring an infinite amount of human power to create generic Chatbots able to maintain a conversation at a human level. However, the progress in the field of ML is demonstrating that providing large corpora to an unsupervised algorithm is enough to maintain a passive conversation with users, which results into a shifting of the human power into data engineering. Multiple algorithms and technics are emerging monthly, demonstrating promising conversational performance improvements; however, they are all still narrow AI. Indeed, even if they are getting better at providing meaningful sentences, they are still not able to generalize all tasks linked to a conversation, such as: understanding the context, search and learn for missing information, initiate conversation in a meaningful manner, be intuitive, and more. The generalization of those features would allow a significant step forward into general Chatbots.

1.1 Aim of Study

In harmony with the author's interest, the goal of this semester's DP is to suggest and demonstrate strategic approaches as a premise to the AGI and to get a step closer to general Chatbots, which can initiate and maintain human-like conversations in a pro-active manner.

1.2 Scope and Study Borders

As a red line for this DP, the focus will be on the Word2Vec technology, from a research perspective. Indeed, this technology is seen as a foundation for the modern NLP and DNN Chatbots, which makes it an exciting vector of study about its current usage, its extensions, and its potential evolution.

1.3 Industrial Interest

iCoSys, the Institut of Complex Systems at University of Applied Sciences and Arts at Fribourg, Switzerland, is interested into the result of this project as a study for their AI-News project, whose goal is to provide a chatbot as a tool to reader, to help them narrow their interests and deliver the right information. AI-News is in collaboration with the Swiss Innovation Agency from the Swiss Confederation, and La Liberté, the daily newspaper from Fribourg.

Chapter 2

Questions

To help the student find a red line to focus its research on; he was required to work on the subject: *"What should be the initial questions to ask in order to make AGI Chatbots"* as a preliminary study, before the beginning of the DP itself; and to write down the outcome as a set of questions related to his interests and the field of AGI Chatbots.

2.1 Initial and Broad Questions

As a result of the preliminary study, the following questions were extracted. Please take into account that those questions were not meant to be answered as part of the project itself but as part of the process of appropriation of the field of study.

- Is the Artificial Neural Networks (ANN) approach appropriate to represent the world?
- Can agents be made exclusively from a language?
- Are agents able to experience an environment?
- Is a narrative environment enough to understand an environment?
- Is the language able to provide to an agent an understanding of the world?
- Is the knowledge of the language syntax enough to gain an understanding?
- Is the result of unsupervised learning enough to discover all nuances?
- Is the unsupervised learning sufficient to make sense to an environment?
- Is a descriptive explanation of the world in a language enough to express it?
- Is the description good enough to catch all the nuances?
- Is the language good enough to explain?
- Can we augment or make a semantic language?
- Can we create a common symbolic language?
- Is the language multi-dimensional?
- How many dimensions are needed for a complex language?
- Is it possible to give a word equivalence to machines for human-specific words?
- Are all emotions describable into words?
- Are emotions altering language descriptions?
- Is an approximation of the real world enough to understand the environment?
- Would a the simulated world be a good approximation of the real world?

2.2 Narrowed Questions

In a second time, the student was asked to narrow the initial questions above into potential fields of study.

- Common human-machine language
  - Is it possible to create a multi-dimensional human-machine language, which includes a common semantic, symbolic, and emotion definition?
  - Is it possible to create an abstract world for machines to understand human symbolic based on a real world, and define fundamentals for machine representation of the language?
- Machine intuition
  - Is it possible to provide to machines an human-like intuition (inside voice), which would help to keep a long term context and specialize in specific fields?
- Evaluate human-machine communication
  - Is it possible to provide a protocol to test the communication skills and machine understanding?

2.3 Potential Red lines

From the potential fields above, the following suggested red lines were proposed.

- How to quantify a chatbot understanding?
- What is the premise to make chatbots general with today's technology?
- How can chatbot be proactive?
- How to simulate human-like intuition in chatbots?

2.4 The DP Question and Red line

Based on reflective work and discussions, the concluding red line and question for this DP are:

- What is Word Embedding and can it be used to make chatbots proactive?

Chapter 3

Plan

3.1 Contraints

Timeframe: 15 weeks
Starting date: 18.02.2019
Ending date: 31.05.2019

3.2 Initial Plan

As the first milestone for the DP, the student was required to create an initial plan, with the purpose to help himself and the teacher to visualize the project's main red line.

3.2.1 Tasks

1. Initial research about general chatbots
2. Determine the project target
3. Play with the subject
4. Explore the Word2Vec methodology
5. Explore the Word2Vec extensions
6. Combine and test ANN algorithms with Word2Vec
7. Explore ANN algorithm topology for the chatbot
8. Analyze of the chatbot intuition with parallel algorithms
9. Analyze of a protocol to evaluate proactive chatbots
10. Analyze Profile-based initiatives
11. Analyze and experiment profile nurturing
12. Analyze and experiment with chatbot initiatives with no profiles
13. Make overall improvements
14. Autonomous data gathering
15. Make suggestions
16. Determine possible continuation and future outcomes for the project

3.2.2 Milestones

1. Initial DP plan and specification document
2. Basic multi-dimensional word embedding space
3. Basic conversational agent
4. Basic proactive chatbot
5. DP report

3.2.3 Sprints

18.02.19 to 08.03.19 (3 weeks)

- Do the initial research about general chatbots
- Determine the project target
- Play with the subject
- **DELIVERABLE:** Plan and Initial Specification document

11.02.19 to 29.03.19 (3 weeks)

- Explore the Word2Vec methodology and its extensions
- Combine and test ANN algorithms with Word2Vec
- **MVP:** Basic multi-dimensional word embedding space

01.04.19 to 19.04.19 (3 weeks)

- Explore ANN algorithm topology for the chatbot
- Analysis of the chatbot intuition with parallel algorithms
- Analysis of a protocol to evaluate proactive chatbots
- **MVP:** Basic conversational agent

22.04.19 to 10.05.19 (3 weeks)

- Profile-based initiatives
- Analysis and experiment of the profile nurturing
- Analyze and experiment with chatbot initiatives with no profiles.
- **MVP:** Basic proactive chatbot

13.05.19 to 31.05.19 (3 weeks)

- Overall improvements
- Autonomous data gathering
- Make suggestions
- Determine possible continuation and future outcomes for the project
- **DELIVERABLE:** Report + Sources

3.2.4 Gantt chart

Figure 3.1 represents the visual gantt chart for the initial plan.

3.3 Effective Plan

As expected the initial plan served as an initial model, and evolved iteratively based on the student and teacher feedback while exploring the subject.

3.3.1 Tasks

1. Initial research about general chatbots
2. Determine the project target
3. Set the initial plan
4. Make LaTeX report template
5. Explore the Word2Vec subject
6. Explore the Word2Vec algorithm
7. Build a Word2Vec model on the latest english wikipedia dump
8. Explore Word2Vec parameters
9. Explore Word2Vec analogies
10. Explore Word2Vec sentence generation
11. Explore visual representations of Word2Vec vectors
12. Explore Word2Vec applications with chatbots
13. Write the report

3.3.2 Milestones

1. Initial DP plan and specification document
2. Basic Word2Vec Word Embedding Model
3. Conclusions for Word2Vec based chatbots
4. Ideas to make chatbots proactive
5. Deliver the report

3.3.3 Gantt chart

Figure 3.2 represents the visual gantt chart for the effective plan.

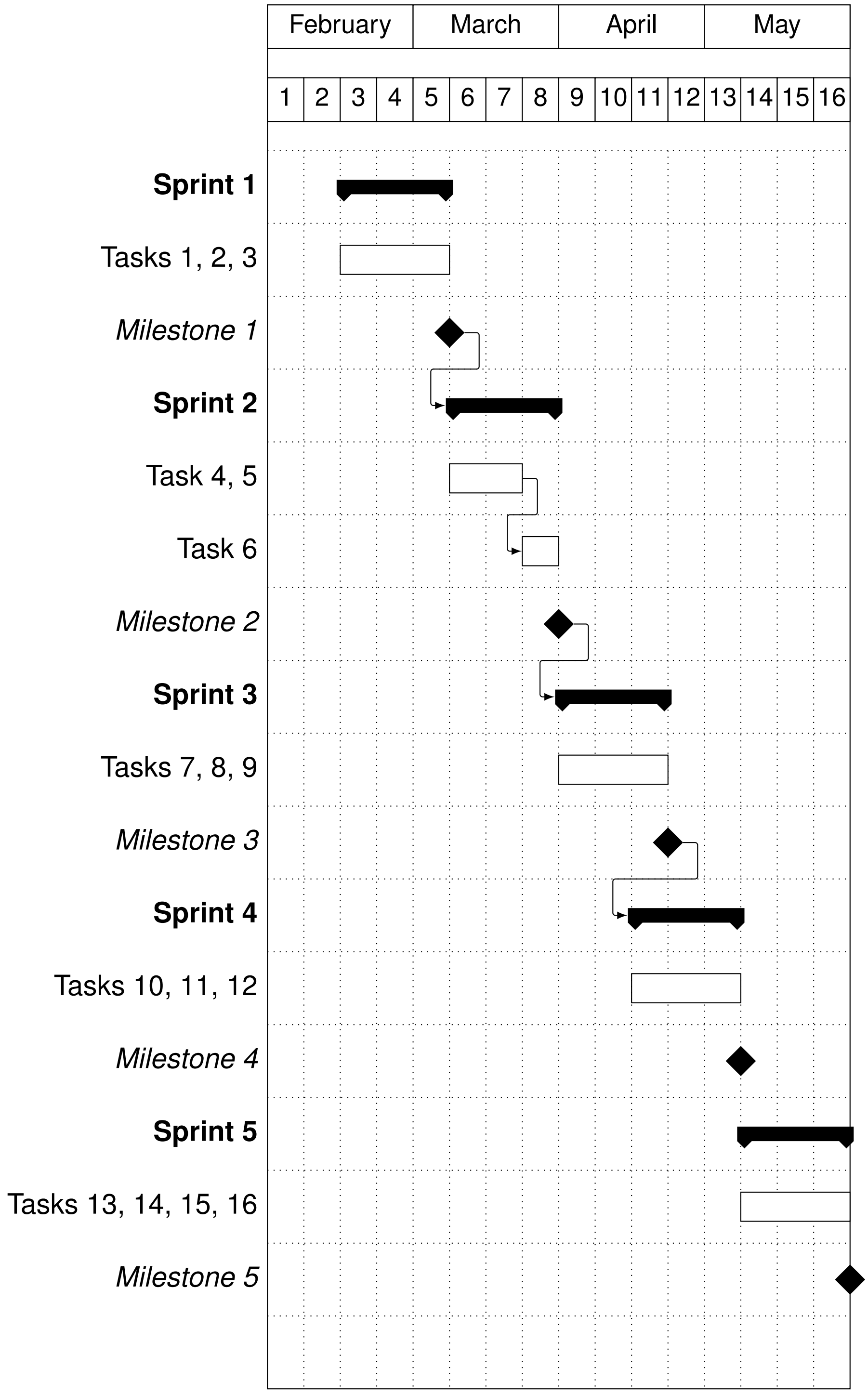


Figure 3.1: Initial Gantt Chart

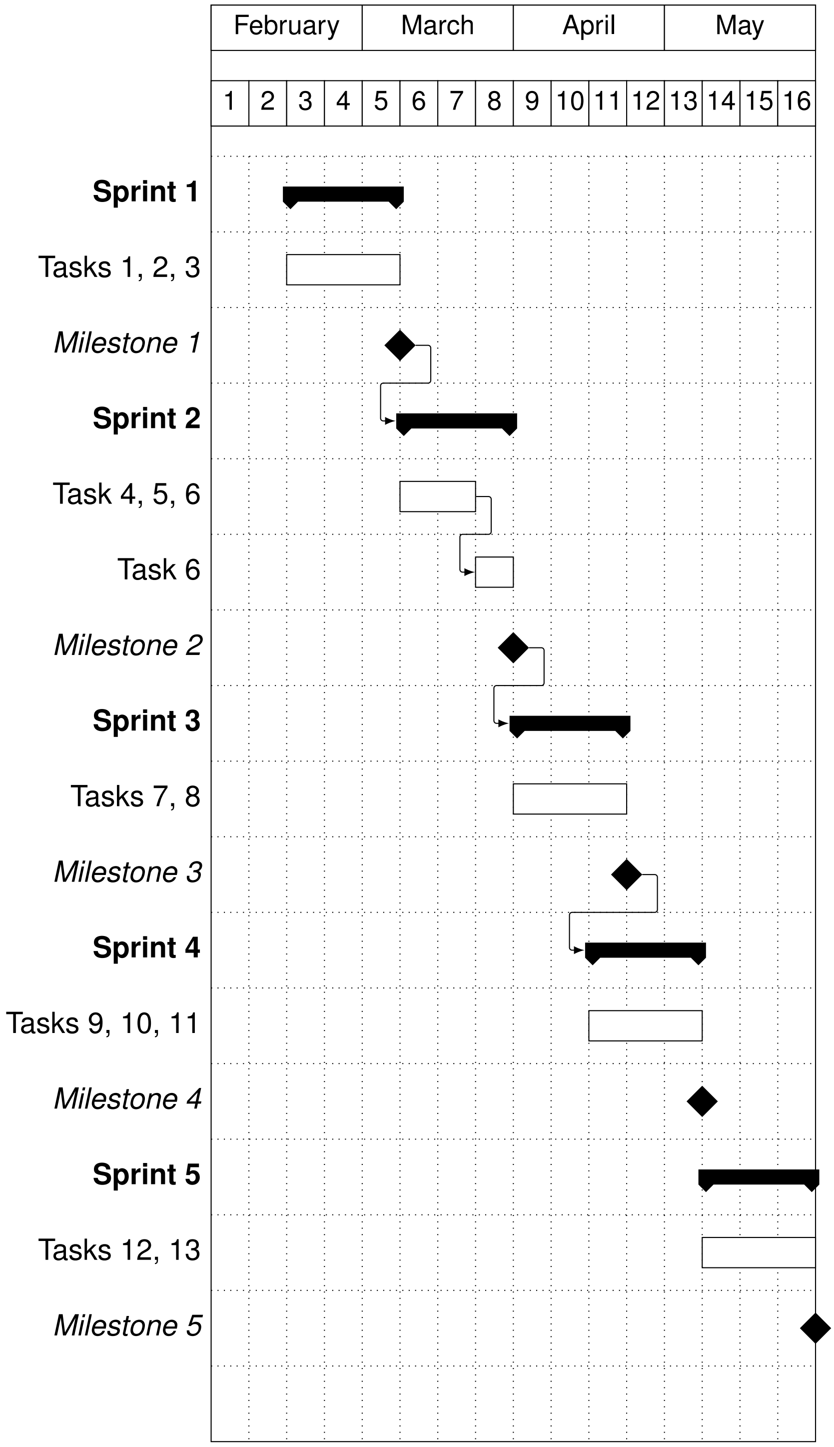


Figure 3.2: Effective Gantt Chart

Chapter 4

State of the art

4.1 Chatbots

From a user point of view, chatbots are trendy nowadays. Big companies such as *Google* or *Apple* are pushing to make the technology mainstream. Even if not every lambda people understand the word “chatbot”, they all have at least a mental representation of it. Indeed, whether they call it Digital Assistant, Siri, Ok Google, and so on, in the end, they all get the concept of an AI narrowed to more or less human-like conversations.

4.1.1 History of Chatbots

From when are they coming? Not mentioning *Alan Turing* or *Joseph Weizenbaum*, considered as the fathers of AI and chatbots, would not be fair. Indeed, they forecasted in 1950 that computers would be able to use human-like communication, and proposed a test to distinguish humans from machines, called the Turing Test[1]: where a human is asked to talk to a masked entity and determine if he is talking to a human or a computer. If the human cannot determine who is the computer, then the machine passed the Turing test, as seen on figure 4.1.

In 1966, Joseph Weizenbaum wrote Eliza[31], a computer program simulating a psychotherapist, seen as one of the first well-known attempts to make a Chatbot passing Turing test. Note that due to technical restrictions, Eliza is not performing well at all time. As it is for today, it is possible to play with it on a dedicated website.

Since Eliza, a lot of progress has been made, indeed, to only cite a few noticeable chatbots: *Parry*[25] (1972), *Jabberwack*[47] (1988), *Dr. Sbaitso*[13] (1991), *A.L.I.C.E*[45] (1995), *Smarterchild*[46] (2001), *Watson*[24] (2006), *Siri*[8] (2010), *OK Google*[21] (2012), *Alexa*[7] (2014), *Cortana*[32] (2014), Facebook Bots[16] (2016), and *Tay*[39] (2016), which where all part of the Chatbot history [17].

From IF-ELSE, Artificial Intelligence Markup Language (AIML), up to ML with ANN and Deep Neural Networks (DNN), the improvements in the field of chatbots increased drastically over the years. At every iteration, the algorithms are becoming more sophisticated and better at using the human language, which is now called the field of the NLP and NLU.

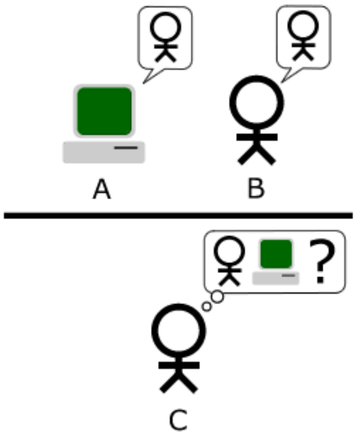


Figure 4.1: The "standard interpretation" of the Turing Test, in which player C, the interrogator, is tasked with trying to determine which player - A or B - is a computer and which is a human. The interrogator is limited to only using the responses to written questions in order to make the determination. [10]

4.1.2 Narrow Chatbots

Once again, chatbots are almost everywhere nowadays. Indeed, it became a common tool for companies of any size to communicate with their customers and a toy for users. However, most of the time, Chatbots are not understood by their users and are leading to a high level of frustration. Even if they are becoming increasingly mainstream and sophistical, people do not realize their limits. Today's chatbots are often mistaken for AGI in Science Fiction (Sci-Fi) and are expected to do much more than they can do. Indeed, making ANI chatbots implies a specialization into a specific field.

Not to forget that the primary purpose of chatbots is to provide a conversational service to the user from text to vocal or even visual format. However, its purpose can be derivated in an almost unlimited amount of solutions such as Health, Weather, Customer Service, Games, and much more.

4.1.3 IR Chatbots

Most of the time used by Frequently Asked Questions (FAQ) chatbots, which are probably the most common type of chatbots, its goal is to answer specific questions, based on a specific keyword. Indeed, the communication skills are limited to pre-made sentences and a question/answer database, which often results, in the best case scenario, in a perfect match, or the worst case scenario, in the return of something unexpected.

Technically speaking, IR is part of the Data Mining (DM) in the field of ML. It is well suited for search engines, as it works in a query mode. Indeed, the algorithm tries to find the best match to the submitted query in its database, usually with pattern extraction and a rank.

4.1.4 Sequential Chatbots

“If he says this, then say that, then do so.” This sentence is a good example of the concept of sequential chatbots. From a communication point of view, it does not have to talk to accomplish its purpose; it is indeed usually based on a keyword detection technic to determine what pre-made action to do. However, as the whole system works on pre-made actions, the development of such algorithms requires a lot of brain power from the developers. Indeed, as all actions result from anticipated specific keywords, and even specific order of keywords, the complexity can quickly increase, which most of the time, makes sequential chatbots seen as command line terminals instead of conversational chatbots.

4.1.5 Forwarding Chatbots

Often used by companies for customer service, it has become the most popular type of chatbots and seen as a hybridization of the IR and sequential chatbots. Its goal is to simulate an agent that is available 24/7 to help the customer. Indeed, it will try its best to answer the most popular questions based on its FAQ database and forward the user seamlessly to a human agent if its knowledge is getting limited. In the best case scenario, it is greatly appreciated by the user as the transition from Chatbot to human is not noticeable.

4.1.6 Learning Chatbots

As ML evolves at an incredible rate and is boosted by DNN, new NLP algorithms emerges, and most of the time leaves the previous generation far behind. Modern learning chatbots algorithms are what come closer to human-like conversations. Leaving the algorithm progress alone through iterations on a large dataset or commonly named Big Data (BD) of real conversations, it will learn patterns by itself. However, the output generated by the trained model is dependent on the data the training occurred on. The most well-known example is *Tay*[39] (2016), the Twitter chatbot from Microsoft, that was influenced by the 4chan community to make it speak like a Young Racist Girl.

However, it is essential to take note that learning chatbots have been existing for a long time now. *A.L.I.C.E*[45] had already basic learning skills, as AIML was taking care of saving variable on the run, such as the first name of the user. Even if this methodology could be seen archaic if compared to new DL algorithms such as LSTM, it is still used today likewise the AIML technology.

4.1.7 Proactive Chatbots

“Hey, I saw that you are on the website for some minutes now, do you need some more dedicated information?”. It is almost impossible that someone never received a message alike. Indeed, proactivity is not new in the field of chatbots. Mimicking an interest from the Chatbot to initiate conversations has become a standard in marketing and customer support chatbots. However, the limitations are hit fast, beyond asking general questions, not much progress has been made until now.

True proactive chatbots are implying that the algorithm is capable of initiating conversations from a human-like perspective, initiating the conversation or asking information in a meaningful manner based on the user, the context and the relationship with the user. The state-of-the-art search could not find any evidence of existing real proactive chatbots as described.

4.1.8 Chatbot Examples

As a help to get a feeling about narrow chatbots, a none-exhaustive list of applications is available below, and for more references about chatbots, Chatbot .org[12] is an excellent, up to date, place about referencing old and new chatbots.

- Receive relevant information about a trip, book flights, and hotels, and get updated on the boarding and weather conditions at the airport.
- Keep track and order coffee remotely at the office.
- Monitor customer's satisfaction.
- Convert potential customer into paying customers by interacting with them at the right moment.
- Personal assistant on-the-go, get the schedule and the next meetings.
- Relay for people on hold at a service.

4.1.9 Narrow Chatbots compared to General Chatbots

Before going further into the world of general chatbots, it is required to understand the following two axes of AI defined by Tasks and Knowledge. Indeed, narrow chatbots are limited by the range of tasks they can accomplish and the knowledge they can use. However, most of the time, they are very good at a particular task for a particular knowledge requirement. The table 4.1 tries to represent the position of Narrow and General Chatbots on those two axes.

Tasks: Talk, FAQ, Remote Control, Customer, or Placing orders are just a few tasks that a chatbot could accomplish.

Knowledge: Health, Weather, Customer Service, or Games are just a few knowledge examples chatbots could excel at.

	Knowledge	
Tasks	Expert in a specific Field Expert at all Tasks	**General Chatbots** Expert in all Fields Expert at all Tasks
	Narrow Chatbots Expert in a specific Field Expert at specific Task	Expert in all Fields Expert at specific Task

Table 4.1: Tasks versus Knowledge in the field of Chatbots

4.1.10 General Chatbots

Much effort is being made to get chatbots that can perform well simultaneously in various tasks and knowledge. Indeed, general chatbots are not limited to previously learned tasks and subjects; they should also be able to learn and relearn.

Those type of chatbots have not been found during the state of the art phase, and are probably by this mean either none-existant at the moment or hidden in laboratories, far from public knowledge.

However, big companies like Amazon are providing to the public a feel of general chatbots with *Alexa*[7]. Users can converse with it, command their smart houses, use it as a personal assistant, and even program it to perform custom actions. However, it is not yet able to learn by itself and generate out of the blue none-programmed skills.

Note that general chatbots could be scary for lambda people if it starts mimicking human being too well, as in the user's mind, talking to a machine should be differentiable from talking to humans. Admittedly, in the case of the *Turing Test*[1], the human does not know if it is talking to a human or a machine, which makes it probably more comfortable to accept than talking to a machine directly. Sci-Fi is conditioning people to believe that human-like performing machines are dangerous for the human species.

4.1.11 From ANI to AGI

On a side, even if it is not part of the DP, it is interesting to write a few lines about AI. New incredible algorithms outperforming the previous one, and experiments reports are emerging almost every month and redefining the standard of AI. Paradigms are shifting and technologically speaking; we are entering a new era of computer-assisted humankind.

4.1.12 ANI

More than a sequential algorithm, narrow artificial intelligence in modern terminology is the definition of "being good at something". ANI has been made possible with the huge progress in ML, the arrival of the DL, and the need for humans to store data about everything (BD). In medicine, for instance, it is sometimes performing so well that humans, who spent years studying, are left behind by an algorithm trained on large datasets for a few days.

4.1.13 AGI

The next step into the field of AI, when supervision has been banned as a teaching method for algorithms as the human interaction is inputting more errors than machine themselves if unsupervised. In addition to teaching themselves, algorithms are teaching each other, and improve over the iteration with auto corrections and optimizations. They are excelling at all tasks requiring repetition, precision, and safety. Besides, they are also all able to retrieve any available information and use it for their need. "In the future, machines will be able to understand and do everything, much more efficiently than humans."

4.2 NLP

Present in our daily lives, this technology is used massively to automate the extraction of information from human communication. In other words, it is seen as the given skill to machines to comprehend human language.

Examples The following is an non-exhaustive list of NLP use:

- Customer Support Chatbots
- Translation into foreign languages
- Voice recognition
- Spam filters
- Interpreting written queries
- Generation of the responses

4.2.1 NL

Naively, it is the language naturally used by humans. The goal of the NLP is to mimic the NL to create a human-like verbal interaction. However, it is not an easy task as it is nearly not possible to teach a machine to talk like a human. Indeed, even if machine are given the same language rules as humans, they do not understand by themselves, and are just applying the provided rules, resulting in a problem during conversational ambiguities. It would be necessary to sequentially teach the missing pieces of information, which would result in an almost an unlimited amount of conditions.

NL decomposition Beyond the grammar and orthography, human language is composed of an incredible amount of subtleties, which makes sense most of the time intuitively for humans, but not for machines. To help understand the complexity behind NL, the following list expresses the foundation of human language:

- Semantics: express the relations between words, sentences, paragraphs, etc.
- Morphology: maintain a structure and the content of word forms
- Phonology: sounds used to express words
- Syntax: rules applied to the bag of words to create valid texts
- Pragmatics: how the context influences the meaning of words

4.2.2 Current NLP technics

Most of the following technics have been developed in the IR field.

- Term Frequency-Inverse Document Frequency (TF-IDF): Used to set the word importance in corpora.
- CBOW [5.1.6]: Counts the words occurrences throughout in corpus.
- Skip-Grams [5.1.7]: Counts the occurrences of the character throughout in corpus.

- Topic modeling: Text clustering providing meaningful information to discover hidden structures via text chunking to identify the parts of the sentence in relation to each other.
- Segmentation: Split corpus into predefined parts, such as: *sentences, paragraphs, chapters, etc.*
- Tokenization: Split the sentences into words.
- Tagging: Based on a pre-made dictionary, it gives a new layer of meaning to the word, such as: *verb, adverb, noun, people name, locations, number, etc.*
- Dictionary: Use of tokenized words to build a dictionary, which could contain the word occurrences.
- Stop Words: Ignoring words only used as liaisons, and not containing information, such as: *and, or, etc.*
- Stemming: Uniformizing words to their root by removing the prefix and suffix, such as: *remake and loveable*.
- Lemmatization: Replace the words to their base form, such as *conjugated verb*.

4.2.3 NLP declensions

Further into the field of NLP, it is today commonly split into two groups:

NLU It is a subdivision of DM, and it involves the processing of the text by analyzing its content by extracting relevant information, usually called keywords.

Natural Language Generation (NLG) In combination with NLU, often applied to text classification, NLG is useful to generate custom sentences, using custom keyword extracted to make the response to a query even more relevant and usually keeps track of the context.

4.3 Word Embedding

It can be summarised as the vector representation of a word and often using between 100 and 400 dimensions. Its position in the multi-dimensional space keeps track of the word context and semantic to a dictionary of words and the corpora. Due to the vector nature of the words, geometrical operations can be applied to those words to find word similarities and relationship between them.

4.3.1 Word2Vec

Published by Google in 2013, Word2Vec[34], probably became, the most popular algorithm in the word embedding field, nowadays. It uses a Shallow Neural Network (SNN) 4.2, similar to a conventional supervised model. Indeed, it is a two-layer Neural Network (NN), its input is text corpora, and its output is word vectors based on a given dictionary. Even if it is easy to train and test, it is often difficult to tweak the algorithm, and as a result, makes it harder to make a good generalization. Even if it is not using DL itself as output, the input text form of the words are transformed into their value form, which makes it incredibly powerful and useful for DNN algorithms as input.

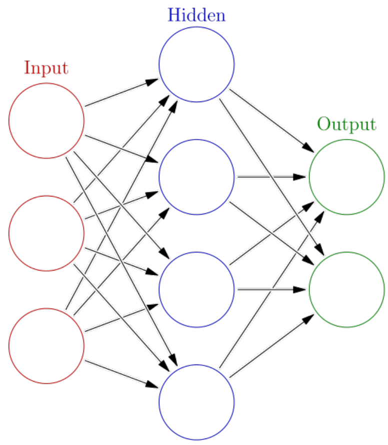


Figure 4.2: Artificial neural network with layer coloring [19]

4.3.2 Gensim Framework

Since its publication in 2013, Word2Vec[34] various actors implemented the algorithm and companies like RaRe Technologies[43], specialized in NLP, made it easier for Scientist and Hobbyists to jump right in. Which probably influenced its popularity by making it accessible for the community instead of big companies and institutions only. In the case of this DP, the author will be focusing on the framework made available by RaRe Technologies and more specifically by RaRe Consulting[42], Gensim[44]. Other frameworks and solutions are available, they are in the end implementing the same algorithm generating a Word2Vec model by capturing the context of the words, but they are different by their language and their different integration with custom features.

Gensim is a python implementation of Word2Vec, and it is not a general purpose ML or even DL framework. However, in order of magnitude, it does one thing, but does it very well, as it does its job faster that Tensorflow[2] for instance.

4.4 Word2Vec Alternatives

Word2Vec does not define Word Embedding; indeed the concept of the vector representation in a multi-dimensional space has multiple solutions to it. Without going into too many details, and to name a few, the following could be alternatives to Word2Vec, with their pros and cons.

4.4.1 Word2Vec[34]

With the purpose to make the following alternatives comparable, the following are the pros and cons for Word2Vec itself.

Pros

- First word embedding solution to be able to generate a model on a large corpus with a dictionary containing millions of words.
- Outputs word vectors based on corpora with raw text.

Cons

- Difficulties to extract the sense of words with multiple meanings depending on the context. e.g., the word **Doctor**, it could be the Academic Title, a Physician, or even the name of a TV-Show.

4.4.2 FastText[11]

Made by Facebook.

Pros

- Same as Word2Vec, except that technically it use the Skip-Grams [5.1.7] technics to train on characters composing the words instead of CBOW [5.1.6], which traines with words.

Cons

- Same as Word2Vec, words with multiple meanings are not managed well.

4.4.3 Glove[38]

Global Vectors for Word Representation is a contribution to the Word Embedding by the University of Standford in NLP.

Pros

- Less time consuming than Word2Vec.
- Depending on the scenario and benchmarks, it sometimes performs better than Word2Vec at tasks related to semantic.

Cons

- Larger memory usage than Word2Vec.
- Same as Word2Vec for multiple word meanings.

4.4.4 Adagram[9]

Adaptive Skip-Gram is a Russian contribution by National Research University of Moscow.

Pros

- It claims, contrary to Word2Vec, to be able to manage different word meanings as it should extract the context of the surrounding words.

Cons

- In the current state, it is not designed for corpora with a dictionary larger than tens of millions of words.
- Does not keep track of the word order.

4.5 Word Embedding Extensions

Based on the Word Embedding technics, proposals were raised about its extension to the sentences and even documents. A simple solution to generate a sentence/documents representation would be to sum word vectors composing the sentences/documents; however, the following non-exhaustive technics are performing better than naive addition.

4.5.1 Doc2vec[28]

Adaptation of Word2Vec for document embedding.

Pros

- Based on Word2Vec.
- Performs well in most cases.

Cons

- In few cases the embedding could be biased towards the specific content words.[27]

4.5.2 Skip-thought[26]

Made for corpus with semantically related sentences.

Pros

- Works well with corpus having a sentence continuity.

Cons

- Adjacent sentences must be semantically related.

4.5.3 RNN

With the current market and institutional need to make everything DL, the subject of DNN must be slightly overviewed. All the technics described previously in this chapter are not using DNN, and there is a good reason for their success. Indeed, they do not require labeled data for training, which is required by DL algorithms. However, the idea has not been abandoned; solutions are raising to overcome this drawback, such as using crowd-based solution, which uses users as signals to determine document similarities [37].

4.6 Beyond Word Embedding

As NLP usage increases over the years, Word Embedding technics and its extensions are becoming increasingly more sophisticated and are getting closer to human-like generalization. As controversially suggested by Geoffrey Hinton, famous DL researcher, it would be possible to get to human-like conversational capabilities via a method he calls Thought Vectors [14]. Without going into exciting details, it implies for the AGI, at this stage, even if the algorithm does not understand the meaning of the sentences, the reasoning behind a thought would be well enough emulated to make it human-like.

4.6.1 Though Embedding

A vectorized thought would be trained to generate a thought's context. As for Word Embedding and Doc Embedding, Thought Embedding are linked by a chain of reasoning.

4.6.2 Contextual embeddings

Thought Embedding paper and implementation is yet to be made, however, progress has been made in this direction with contextual embeddings Context2Vec[36]. It is a bidirectional LSTM[23] unsupervised model generating a Word Embedding based on its occurrence in the sentences, which unlike Adagram[4.4.4] is taking into account order.

However, Context2Vec does not define Contextual Embeddings. New emerging 2019 algorithms, based on the attention mechanism [50], such as Transformers[4] or even further its bidirectional extension BERT[15], which makes LSTM almost obsolete.

Summarized

- Context dependent word embeddings.
- Can generate sentence embeddings.
- The output can be used almost as it is for NLP.
- Tracking using selective "forget" gates.

4.7 Datasets

Nowadays, in data engineering, the new gold is data. Luckily, today is driven by BD, and almost any kind of data is available to who knows where to look at. For the case of the DP, the author is interested in conversational data. Even though the English Wikipedia Dump from 09th May 2019 will be using exclusively, the following is a non-exhaustive list of corpora gathered during the DP.

- Wikipedia Dumps Index: https://dumps.wikimedia.org/backup-index.html
- English Wikipedia Dumps (bzip2: 16Gb, raw: 90Gb): https://dumps.wikimedia.org/enwiki/latest/ [enwiki-latest-pages-articles.xml.bz2]
- Reddit Comments (bzip2: 6Gb, raw: 32Gb): https://www.reddit.com/r/datasets/comments/3bxlg7/i_have_every_publicly_available_reddit_comment/
- The Open American National Corpus: http://www.anc.org
- Santa Barbara Corpus of Spoken American English: https://www.linguistics.ucsb.edu/research/santa-barbara-corpus
- Leipzig Corpora Collection http://wortschatz.uni-leipzig.de/en/download/

- Legal Case Reports: http://archive.ics.uci.edu/ml/datasets/Legal\+Case\+Reports
- Cornell Newsroom: https://summari.es
- DeepMind Q&A: https://cs.nyu.edu/~kcho/DMQA/
- Large Movie Review: http://ai.stanford.edu/~amaas/data/sentiment/
- Project Gutenberg - Free eBooks: https://www.www.gutenberg.org

4.8 Word2Vec Models

Training own models are very resource consumptive, and often the resources are not available, would it be datasets or computer power. Luckily, big companies like Google did the work for us, and pre-trained models that could be used out of the box. The following is a non-exhaustive list of models for Word2Vec:

- Gensim directory: https://github.com/RaRe-Technologies/gensim-data/releases
- Fasttext: https://fasttext.cc/docs/en/english-vectors.html
- estnltk: http://ats.cs.ut.ee/keeletehnoloogia/estnltk/word2vec/

Chapter 5

Analysis

5.1 Word Embedding: Word2Vec

The main focus for this DP is for the author to get some expertise with the Word Embedding [4.3] and more specifically the Word2Vec [4.3.1, 4.4.1] algorithm, which is already very complete on features and parameters[48].

5.1.1 Word2Vec Operations

Analogies

Word2Vec is known for being able to handle analogies and more specifically for the famous:

Man is to Woman as King is to? ***[Queen]***

Which translated with Gensim [4.3.2] into the vectorial form and the geometric operation as:

```
model.wv.most_similar(positive=['king', 'woman'],
                      negative=['man'], topn=1)
```

Outputing the following: **[('queen', 0.6848626732826233)]**

Geometric Operations

As each word in Word2Vec is a vector representation in a multi-dimensional space, it implies that standard geometrical operations are applicable. Indeed, as seen with analogies [5.1.1], some operations are being made, implying *positive* and *negative*. Below are the top 3 operations used in the Word2Vec.

- Addition
- Soustraction
- Cosine

Common Tasks

As seen with analogies [5.1.1], the *most_similar* function is a massively used task for Word2Vec, however it's not the only one. Indeed, the following is the top 3 functions used with Word2Vec space:

- most_similar('king'): outputs the top most similar words to a given word.
- similarity('woman','man'): outputs the degree of similarity between two words.
- doesnt_match('dog cat computer bird'.split): outputs the non similar words.

5.1.2 Word Length

As Word2Vec is a multi-dimension space, in which words are positioned, it is important to understand how those vectors are positioned and what information they carry. The word length represents how often a word is used in a context. Indeed, if a word is used a lot in a context, its length will be greater than the same word used at the same frequency but split up in multiple contexts. Meaning that word length, in combination with the term frequency, is useful to measure the word significance in contexts. [49]

5.1.3 Word Angle

Word vectors are not only carrying the length [5.1.2], they also have a direction, popularly described by the Cosine function. The process applied to vectors in Word Embedding is known as the Cosine Similarity, which is the vectors normalized dot product and making it very efficient for evaluation, in particular with sparse vectors such as word vectors.

Positive Space

Often, the Word2Vec space is kept into a positive space, implying that the output is between 0 and 1, where 1 is for the angle at 0°, which implies that vectors are above each other and the vectors are most probably the same.

Negative Space

However, it is also possible to go into the negative similarities, which is described by vectors being in opposite directions, with an angle higher than 90°, which transcribes into the -1 value if the vectors are in the exact opposites, independently to their magnitude.

5.1.4 Normalization

During the similarity calculation, mathematically speaking normalizing vectors are making cosine [5.1.3] and dot-product equivalent. In word embedding, it is usually the relations between word vectors that are required, implying that to enhance the similarity function performances, the vector normalization is commonly used as in this case the length does not carry any useful information. However, if the relation to the context is required, the normalization should be avoided.

5.1.5 Lemmatization

Expect from its direct implication with the dictionary size and its implication related to the processing powertime required to compute the model. Lemmatization should be considered in specific cases. Indeed, using it makes the Word2Vec space sparser, which is useful for small datasets; however, for big datasets, the gain is often negligible.

However, for the case where the words must carry different information depending on the context, for instance with abbreviations, it is necessary to use the lemmatization feature, for example, IR could be mistaken with Infrared (IR).

Another case would be the need for the use of tokens for written words such as *New York*.

5.1.6 CBOW

Continues Bag of Words is the original method presented in the Word2Vec paper [34] and involves NN to predict a word vector based on its context. In the case of a unique word vector, it is similar to an encoder-decoder architecture.

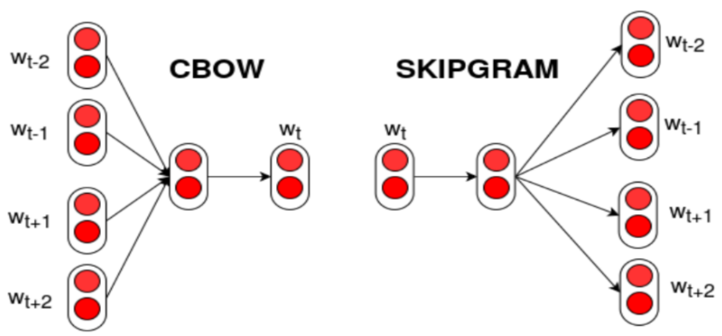
The concept behind CBOW is to take, for a given word vector, the *n* neighboring word vectors as the context, then predict the given word vector, as seen on the figure 5.1.

Concerning the prediction quality, its measure requires to provide the hot encoding of the given word as input, so it calculates the output error and learns the vector representation of the word.

5.1.7 Skip-Grams

Introduced by Facebook, Fasttext[11] Word Embedding is using a variant of CBOW that looks like its flipped version, as seen on figure 5.1. It also involves a NN, but instead of predicting the word based on the context, it predicts the context based on the word.

Given the target word vector as input, the model outputs the probability distribution for the given word.



$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} logp(w_t|w_{t+j}) \qquad \frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} logp(w_{t+j}|w_t)$$

Figure 5.1: CBOW and Skip-Gram neural network representations[3]

5.1.8 Dimensions

Typically between 100 and 400, Word2Vec dimensions are defining the accuracy in the similarity between word vectors. It is difficult to find the sweet spot. However, there is some experimental curve for the dimension amount, which takes into account the final accuracy, time, and power consumption.

Low accuracy It is recommended to use at least 50 dimensions to avoid loosing too many properties of high dimensions.

Normal accuracy Using around 200 dimensions provides acceptable accuracy for the time spent computing.

High accuracy Around 300 dimensions, the results are considered very good; however, the time is high.

Beyond high accuracy It seems that beyond 300 dimensions, the accuracy gain is not worth anymore the extra time used during training.

In other words, the amount of dimensions usually reflects the over and underfitting of the model. Each model is dependent on the dataset, and it is often required to test the different number until the accuracy is acceptable. Plus, as this subject is still debated at the moment in the ML community, there is no absolute value, except try and fail.

5.1.9 Window

As explained for CBOW5.1.6 and Skip-Grams5.1.7, those algorithms are either taking the context as input, or outputting the context for a given word. However, to capture the context, it is required to define its size, which is in our case called the window.

By default, the value of 5 of the window is used and generally works well; however, the accuracy of the model will depend on the dataset used.

As a measure, the analogy score could be used. However, this score is often not the solution as the impact of the quality of a poor dataset is higher than the size of the window if using the default value. Plus it is essential to be careful with the sentence sizes of the corpus used; indeed, if the window value is higher to the average sentences length, the algorithm will not capture meaningful relations.

In other words, large windows size usually capture more information related to the main topic, and small windows capture the information related to the word itself.

5.1.10 Epochs

As the number of epochs is directly proportional to the amount of computation and time spent training the model, the common question to ask is: *How much epochs will be giving the best accuracy for the time spent?* The answer is as always: it depends on the dataset and the parameters.

However, based on various sources, including the author of Gensim[41], it is suggested that increasing the number of epochs have, in most of cases, an accuracy performance. The default value is usually five epochs.

Finally, the quest to find the right parameters is still a matter of experimentations.

5.1.11 Gensim API

It would not make much sense to go in details of each Gensim parameter, as a list of the parameters, and their description is available on their official API website[18].

However, to initiate the curiosity of the reader, the following list represents the most used parameters during the DP:

- size: dimensions used [5.1.8]
- window: window size used [5.1.9]
- min_count: minimum frequency a word should have to be considered
- workers: cpu cores used
- sg: set at 1 to use Skip-Grams [5.1.7] and set at 0 to use CBOW [5.1.6]
- iter: epochs used [5.1.10]

5.1.12 Retrain Model

In ML, models are the resulting artifacts from training processes and provide the rules to generate an output for an input. A Word2Vec model is the representation of the relations between the dictionary of words and their contexts for the corpus it was trained on.

As seen in the state of the art section, pre-trained Word2Vec models are available on the internet [4.8]. However, to save disk space and probably business knowledge, it is infrequent that the full model is shared. Indeed, the model provided is only a model with frozen NN weights, which is perfectly fine for regular usage as it behaves the same as a full model, except that it is not containing the multi-dimensional matrix representing the Word2Vec space.

To get the full model, it requires to train it ourselves, with all the side-effects it implies, such as having a high amount of power and time consumption. However, a full model provides the ability to *retrain* the model. One could add words to the dictionary or customize the weights to match a new verbal style or context. For instance, starting from on a Wikipedia model, it would be possible to influence the words to match the style of an author such as *Edgar Allan Poe*.

5.1.13 Evaluation

Determining if the model works correctly is difficult. A naive solution would be to create a complex supervised protocol to evaluate the success of a model. However, it would not require an enormous amount of work from a human perceptive. A solution is to exploit the analogies capabilities from Word2Vec 5.1.1, which, in theory, should perform well at.

The concept is called the *Analogy Evaluation* and it uses a list of pre-made analogies in various domaines, such as the famous: **Man is to King, as Woman is to?**, and evaluates if the results from the model match the expect answer: *Queen*. In Gensim, the following function does the evaluation:

```
evaluate_word_analogies(analogies,
                        restrict_vocab=300000,
                        case_insensitive=True,
                        dummy4unknown=False)
```

Another evaluation method would be based on the word similarities itself. Indeed, based on a pre-made list of pairs of words, an input such as *cup* would output *mug*. This evaluation is done in Gensim via the following function:

```
evaluate_word_pairs(pairs, delimiter='\t',
                    restrict_vocab=300000,
                    case_insensitive=True,
                    dummy4unknown=False)
```

5.1.14 CPU VS GPU

As mentioned in the state of the art section, Word2Vec is performing better than DL solutions, in the most cases, because it is not using labeled data [4.5.3], even with the use of GPUs. However, in the context of Word2Vec, an experiment from Gensim has been made to compare the computation made CPU and GPU on the Gensim Framework [51]. The result is unexpected from a generalization point of view: indeed, it appears that CPUs are performing better than GPUs during the Word2Vec training.

Chapter 6

Experiments & Results

Intending to get a hold on the Word2Vec technology and as a complement to the analysis chapter 5, the following are the experiments made during the DP and their results.

6.1 Build a Word2Vec model

Getting started with Word2Vec As recommended to the author, this section has been made following the Gensim tutorial by Machine Learning Plus [30]. There is not much to say about this section except that the author, with its current Word2Vec knowledge, would recommend this tutorial to anyone willing to get started with Word2Vec.

As overviewed in the tutorial mentioned above, using pre-trained models is good to get started quickly, and building a model on a small dataset is fair enough to get for various tasks; however, in the case of this DP, the author wanted to understand how everything works behind the scene.

6.1.1 Build the Vocabulary

The first step is to build a Word2Vec model is to build its vocabulary. Luckily, it is straight forward, as it does not require much tweaking. Indeed, it is only required to decide if the lemmatization will be present if stop words are included, if there is word frequency threshold, and if the symbols such as punctuation are used.

For the DP, two dictionaries were built, both with default settings from the WikiCorpus method but differentiated by the presence or not of the lemmatization. This function is from the Gensim package corpora.wikicorpus and is specialized for Wikipedia dump. The following is the function definition [18]:

```
gensim.corpora.wikicorpus.WikiCorpus(
        fname, processes=None, lemmatize=True,
        dictionary=None, filter_namespaces=('0', ),
        tokenizer_func=<function tokenize>,
        article_min_tokens=50, token_min_len=2,
        token_max_len=15, lower=True,
        filter_articles=None)
```

Notebook in Appendices

- pa-build-dictionary

6.1.2 Build the Wikipedia Model

This section could resume about half of the DP, which in theory should not be too difficult; however, hours passed at a high rate to finally being able to produce Wikipedia models easily. The following is the function definition [18]:

```
gensim.models.word2vec.Word2Vec(sentences=None,
        corpus_file=None, size=100, alpha=0.025,
        window=5, min_count=5, max_vocab_size=None,
        sample=0.001, seed=1, workers=3,
        min_alpha=0.0001, sg=0, hs=0, negative=5,
        ns_exponent=0.75, cbow_mean=1,
        hashfxn=<built-in function hash>, iter=5,
        null_word=0, trim_rule=None, sorted_vocab=1,
        batch_words=10000, compute_loss=False,
        callbacks=(), max_final_vocab=None)
```

Notebook in Appendices

- pa-w2v-mono-training

6.1.3 Split and Retrain Technic

During the initial phases of the DP, the English Wikipedia Dump was too large for the machine available to the author. [6.2] As a result, it was required to find a workaround to make it work anyway.

First step Split the Wikipedia corpora into chunks.

Second step Retrain [5.1.12] the model on each chunks.

Notebooks in Appendices

- pa-wikidump-splitter
- pa-build-word2vec-on-splits

Issue

Sadly, it was found out later, while using the CPU Dedicated Server [6.2.6], that the vocabulary was not updated, resulting in a model containing only the initial chunk vocabulary. Even if the limited vocabulary represents well the whole contexts from the English Wikipedia, it does not make much sense as a whole.

Due to the original purpose to fix specific hardware limitation [6.2.2], the solution was discontinued as the limitations were solved with new hardware. If the code must be reused, the vocabulary should be updated at each chunk iteration.

6.2 Environments

The language used during the whole chapter is Python with the Gensim framework and Jupyter Notebook. The main dataset for the experiment is the English Wikipedia Dump from 09th May 2019 [4.7].

6.2.1 Local Machines

At the beginning of the project, it was suggested to the author that his local machines would be enough to use Word2Vec; however, even after various tweaking, it concluded that the hardware on author's local machines was limited. Indeed, the RAM, CPU, and Disk Space were not enough to handle the English Wikipedia dump dataset (bzip2: 16Gb, raw: 90Gb) [6.2.7].

Macbook Pro Specifications

- CPU: 2.3GHz Intel Core i7, 4 cores
- RAM: 16GB 1600MHz DDR3

Windows Specifications

- CPU: 2.50GHz Intel Xeon E5-2680 v3
- RAM: 12GB 2400MHz DDR4

6.2.2 iColab GPU Server

As a solution, the iCoSys Institut at HES-SO//Fribourg provided access to a remote machine dedicated to DL student projects. This infrastructure could hold the first experiments; however, due to the nature of the server, the Word2Vec training was highly impacted for the large Wikipedia dataset. Indeed, the server was build for GPU usage instead of CPU intensive computation, in addition to the RAM being shared across all users, the author was not able to train the full model in one shot, and had to use a custom made data splitter with a retraining [5.1.12] workaround to make it work [6.1.3].

However, even with the used workaround, the time to train the whole dataset took 3 days and 20 hours, and sadly for what was discovered afterward, an incomplete model [6.1.3].

iColab GPU Server Specification

- CPU: 2.10GHz Intel Xeon E5-2620 v4
- RAM: 128GB

6.2.3 AWS

With the hope to solve the author's local machines [6.2.1] limitations and to decrease drastically the time spent on the iColab GPU Server [6.2.2], the author tried deploying a virtual machine on the AWS EC2 service[35].

The outcome was not satisfying at all. Indeed, the author's account at AWS is restricted from powerful machines [6]. Indeed, useful machines for the project with a high amount of RAM and CPUs were not available to use.

However, for the sake of the experiment, the author tried the largest machine available to him, the c4.8xlarge [5]. Except for being expensive; the results were not as good as it should have been expected: indeed the performance was similar to the iColab GPU Server.

c4.8xlarge

- vCPU: 64
- ECU: 132
- RAM: 60 GB
- Price: 1.591 per Hour

6.2.4 Microsoft Azure Notebook

Desperate to train the wikipedia model, the author also tried the Azure Notebook service[33]. The results are similary limited as for AWS [6.2.3],.

6.2.5 Google Colab

Even the Google Colab server, which is meant for GPU and TPU processing was tested [20]. Even by taking care of keeping the session alive which implies bypass an inactivity timeout after 90 minutes[22], the maximum lifetime per instance of 12 hours [40] and RAM limitation made it impossible to accomplish the training.

6.2.6 CPU Dedicated Server

After insisting, a dedicated server with enough CPU and RAM from the iCoSys Institut was provided to the author.

It has become the machine of reference for all the results for the DP.

CPU Monster Specification

- CPU: 8x 1.2Ghz AMD Opteron 6176
- RAM: 192GB DIMM

6.2.7 Memory Issues

The main issue encountered with the machines was the memory allocation. Indeed, the whole dataset is loaded into the RAM and that the multi-core merging function is using temporary additional RAM [6.1].

Dataset English Wikipedia Dump is weighting 16Gb in its compressed bzip2 form, and about 90Gb in its raw form.

Local machine [6.2.1] The problem is clearly coming from the RAM available, 12GB and 16GB.

iColab [6.2.2] Even if the RAM is in theory enough for the dataset, it seems that the limitation comes from the server configuration; indeed, the RAM is shared across all users and is probably limited.

AWS [6.2.3]

Azure Notebook [6.2.4] Free azure notebook is limiting the at 4GB per session.

Google Colab [6.2.5] As an honorable mention, the ram is limit to 13GB per session.

```
2019-03-25 08:31:18,867 : INFO : PROGRESS: pass 0, dispatched chunk #34 = documents up to #70000/4614519, outstanding queue
 size 31
Exception in thread Thread-1:
Traceback (most recent call last):
  File "/usr/lib/python3.5/threading.py", line 914, in _bootstrap_inner
    self.run()
  File "/usr/lib/python3.5/threading.py", line 862, in run
    self._target(*self._args, **self._kwargs)
  File "/usr/lib/python3.5/multiprocessing/pool.py", line 366, in _handle_workers
    pool._maintain_pool()
  File "/usr/lib/python3.5/multiprocessing/pool.py", line 240, in _maintain_pool
    self._repopulate_pool()
  File "/usr/lib/python3.5/multiprocessing/pool.py", line 233, in _repopulate_pool
    w.start()
  File "/usr/lib/python3.5/multiprocessing/process.py", line 105, in start
    self._popen = self._Popen(self)
  File "/usr/lib/python3.5/multiprocessing/context.py", line 267, in _Popen
    return Popen(process_obj)
  File "/usr/lib/python3.5/multiprocessing/popen_fork.py", line 20, in __init__
    self._launch(process_obj)
  File "/usr/lib/python3.5/multiprocessing/popen_fork.py", line 67, in _launch
    self.pid = os.fork()
OSError: [Errno 12] Cannot allocate memory
```

Figure 6.1: Memory Allocation Error

6.3 Play with Word2Vec

The second most time consuming and informative part of the DP is when the author could finally play with the Word2Vec model.

6.3.1 Common Word2Vec Operations

As seen in the analysis chapter, the common operations such as Geometrical Vector Operations [5.1.1], Similarities [5.1.1], Analogies [5.1.1] and Normalization [5.1.4] were played with on the large wikipedia model.

Notebooks in Appendices

- pa-play-with-w2v-enwiki-lemmatized.ipynb
- a-play-with-w2v-enwiki-unlemmatized.ipynb

6.3.2 Models Diversity and Time Benchmarks

To profit the most from the CPU Dedicated Server, and as building Wikipedia models are time-consuming, the author started a process of keeping the machine busy by producing Word2Vec models with different parameters based on the Wikipedia dump. All the following parameters were altered:

- CBOW 5.1.6
- Skip-Grams 5.1.7
- Lemmatization 5.1.5
- Dimensions 5.1.8
- Window 5.1.9
- Epochs 5.1.10

Spreadsheet in Appendices

- pa-models-created-and-time-benchmarks

6.3.3 Visual Representation

To get a hold on what a Word2Vec model is, the author implemented a visual representation of the multi-dimensional space using the T-SNE [29] technic, which flatters high dimensional points into a two-dimensional space. For example, the top 100 similar vectors to the word *Woman* on the figure 6.2.

Notebook in Appendices

- pa-w2v-explore-models

6.3.4 Word Vector Influences

To pursue the understanding of how vectors are interacting with each other, the author implemented a solution that applies an arbitrary value to each element of a word vectors and evaluates the impact of this change on the word similarities. For example, for the word vector *Federer*, the top 5 similar words will be tracked while each element of the Federer vector will be alternated elements. As a way to keep track of the altered element on the visual representation, each vector has been labeled with its altered index as seen in figure 6.3.

Notebook in Appendices

- pa-w2v-explore-vector-influence

6.4 Proactivity

As part of the DP red line [2.4], it was asked to explore the Word2Vec possibilities to make chatbots proactive.

6.4.1 Sentence Generator

To explore the possibilities of proactivity, the author tried to used geometric operations [5.1.1] to generate sentences. Various vector manipulations were used to alternate and generate new sentences out of *Proverbs* and *Facts*, such as word additions, arbitrary value subtractions, and even the use of random vectors.

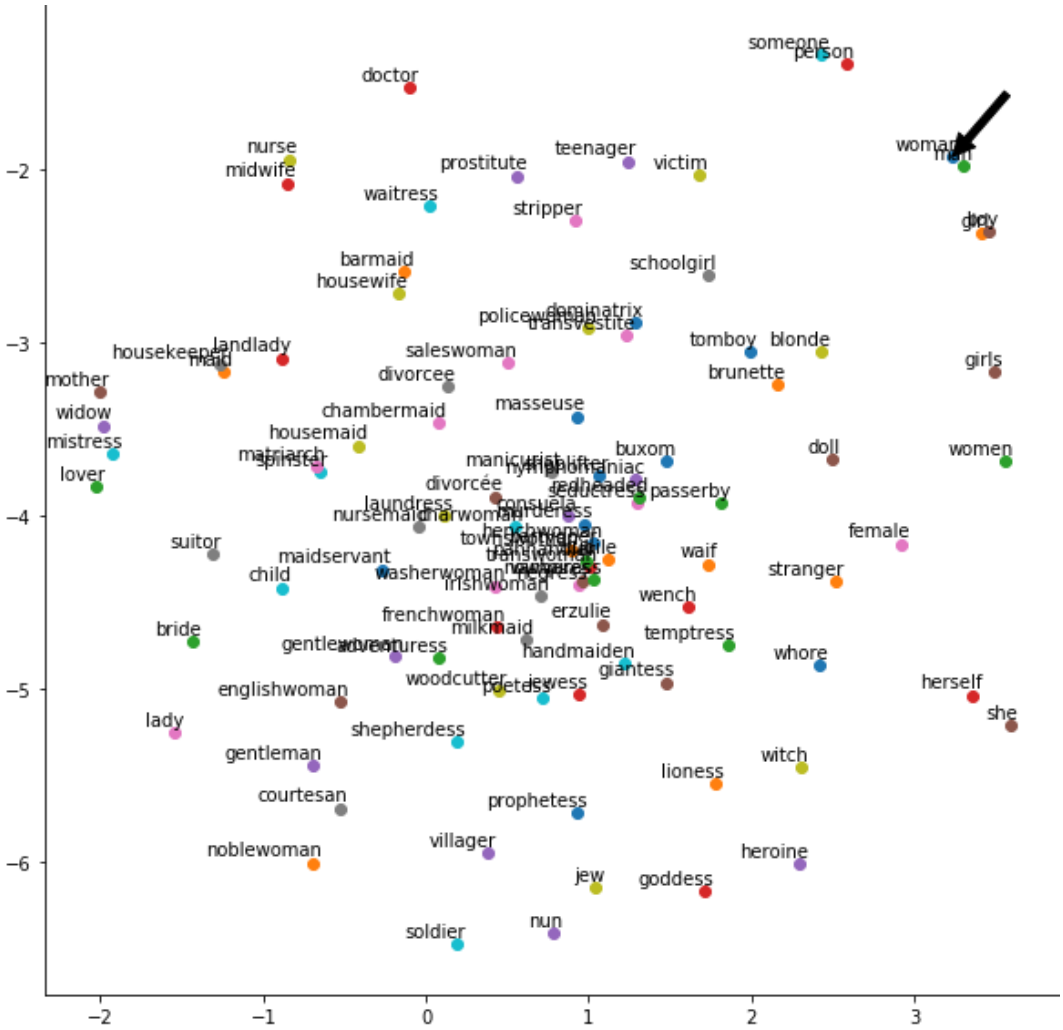


Figure 6.2: Top 100 T-SNE representation of similar words vectors to Woman

The idea was that, to make a chatbot proactive, it needs to be able to generate sentences out of a meaningful context. Moreover, a solution was to make a sentence generator of facts, based on real proverbs and facts such as: *Better late than never*, *There is no place like home* or even *the financial capital of the world is wall street*.

However, the quality of the results was intuitively not excellent and hard to quantify, but as a conclusion, it was found that the most impactful operations are the additions and subtractions.

6.4.2 Abstract Analogies

Another idea to make proactive chatbots was to exploit the analogy capabilities to generate abstract analogies such as: *What is the capital of science?*. As it is, and at least with the Wikipedia Word2Vec model is not possible to have this layer of attraction; indeed, words are bonded to contexts, and abstractions are equivalent to random operations. However, as a solution to bypass the limitations would be to create an algorithm able, out of the similarities, to find contexts in common and

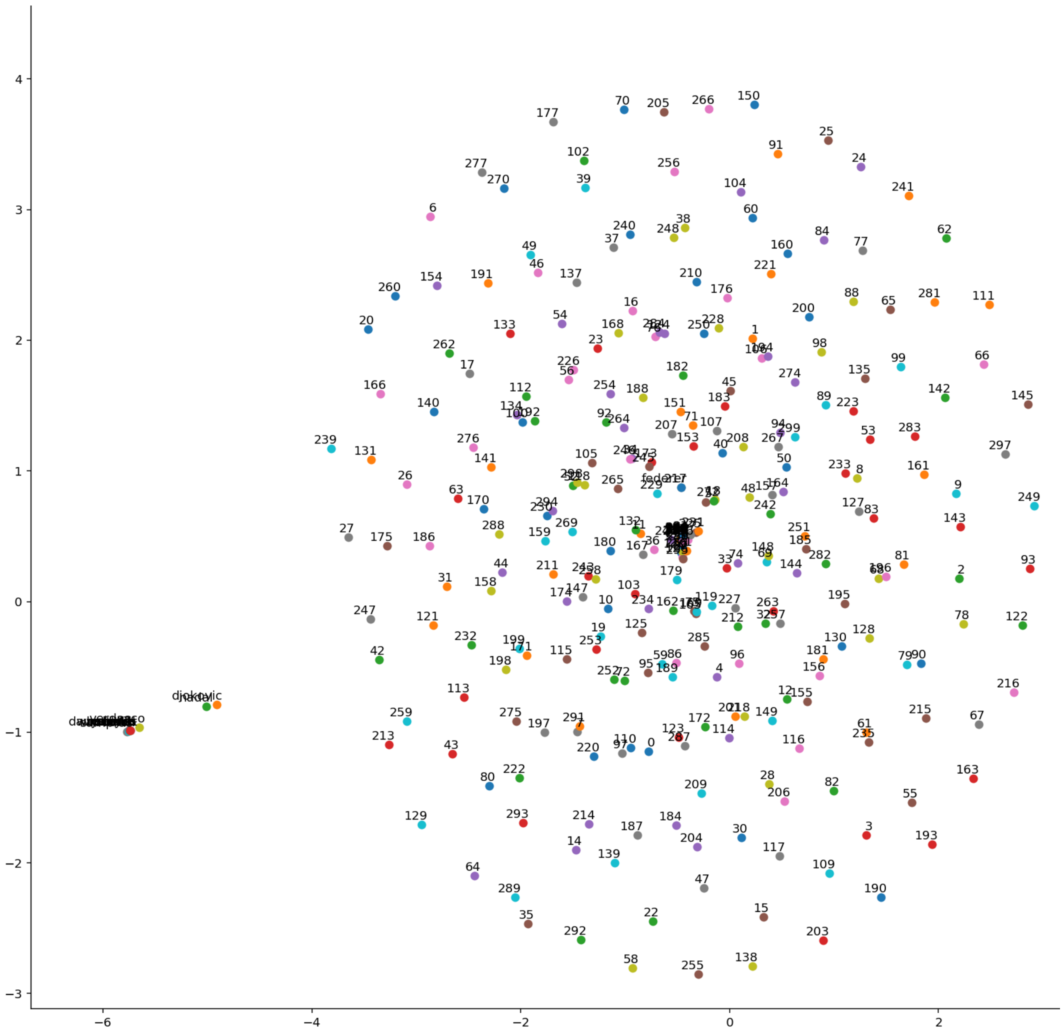


Figure 6.3: Top 5 T-SNE of similar words vectors to the word Federer with altered vector elements

then apply indirect analogies.

Notebook in Appendices

- pa-w2v-sentence-generator

6.5 Chatbot

Sadly for this last section, the time was missing to make a correct implementation. However, some research has been done in order to build a chatbot using a Recurrent Neural Network (RNN) such as LSTM.

6.5.1 Concept of the Chatbot

With the idea to use the Wikipedia Word2Vec model, in a meaningful way, the author wanted to make a Chatbot using the Doc Embedding Seq2Seq and based on a retrained Word2vec model combining Wikipedia and the corpora from the author Edgar Allan Poe.

6.5.2 The current missing pieces in the model

It appears that Gensim, while building the Word2Vec model on the Wikipedia Corpus, was not using punctuations. Indeed, in the context of word similarities, it does not make sense to keep track of the punctuations in the model. However, in the generation of NL texts, punctuations are essential. Intending to achieve a Chatbot talking like Edgar Allan Poe implies to add the punctuation to the vocabulary.

Chapter 7

Discussion

7.1 Next steps?

Comparing the initial speculative plan [3.2] and the effective plan [3.3], it is observable that the aiming was made toward the expertise acquisition of the Word Embedding technology Word2Vec. Even if much knowledge has been gathered during the DP, there is always more to be learned and experimented as a way to dive further into the Word2Vec expertise.

7.1.1 Memoir

One of the motivations for this DP was to make a premise to the author's memoir, which will be using word embedding as a foundation. Indeed, the understanding of Word2Vec will help the incoming design thinking to make a deep retrieval chatbot using topic extractions.

7.1.2 AGI

Another motivation of this project was to explore and understand Word2Vec extensions initiating General Chatbots [4.1.10] such as the recent Contextual Embedding [4.6.2] and the predicted Though Embedding [4.6.1]. Even if the author did not have the time to get into details of those promising evolutions, he would be keeping an eye on those, and even maybe be able to explore further during his memoir.

7.1.3 DL

As discussed briefly during the state of art chapter, Word Embedding is an alien in the current ML world, where almost everything is made out of DNN [4.5.3]. However, the claim that the *standard* nn from Word Embedding is more performant than a DNN will be quickly updated. Indeed, with the current progress in Sequence to Sequence (Seq2Seq) due to a market need, frameworks like Keras which are backed by big companies such are Google is providing out the box indirectly Word Embedding as it is required for Seq2Seq. Word Embedding is a solid foundation for incoming new types of *Embeddings*, and its performance will more than probably increase over time with new algorithms able to catch better contextual information about words.

7.1.4 Benchmarking

The most regretted author's section, the lack of time could not allow diving into benchmarking. Indeed, various models have been computed [.2] to evaluate and compare them to each other. Comparing Wikipedia models to find out which parameters are providing the best accuracy [5.1.13], and how it compares to public pre-trained models from Google, for instance.

Chapter 8

Conclusion

During this DP, much knowledge has been acquired by the author via Research and Experimentation. This section is meant to summarise and conclude the 180 hours over 15 weeks allocated for this project.

8.1 DP

As expected, the initial plan [3.2] was not followed. However, the effective plan [3.3] resulted in a project focusing on the Word Embedding concept, and more specifically, the Word2Vec technic. Indeed, taking a step back on the 15 weeks spent on the project, the way that the DP is designed, the project was meant to go in this direction. Hours are flying during research and experimentation, and unexpected situations are part of the game when working with unknown technologies.

8.1.1 Results

The outcome of the project can be summarized as follow:

- The author gained expertise into the Word Embedding field, and the Word2Vec technic.
- A research-oriented document has been produced.
- Experiments have been implemented.

8.1.2 Time spent

The initial naive plan to split the required 180 hours across the 15 weeks was to perform 12 hours per week. The effective hours sums up to 191h. A delta of 11h is fairly acceptable for a semester project. It is difficult to not regret of not providing more hours into the DP, but the workload of other classes is not providing many flexibilities.

- Research: 72h
- Development: 48h
- Meeting: 11h
- Redaction: 60h

8.2 Objectives

As a wrap up, let's conclude on the red line [2.4].

8.2.1 Word Embedding: Word2Vec

Using the Gensim Framework, the author was able to experiment and understand the Word2Vec technic with details. The difficulties encountered while building his own Wikipedia model forced to dive into the technicalities related to the nature of how Word2Vec works, which increased the expertise and resulted in state of the art solutions.

8.2.2 Word2Vec Chatbots

Making Word2Vec only chatbots is not possible; however the output model is already used in advanced DL chatbots using LSTM or even Transformers [4.6.2]. Word Embedding provides the foundation for the next generation chatbots with Thought Embedding, getting every step closer to the General Chatbots [4.1.10].

8.2.3 Word2Vec for Proactivity

Proactivity as explored in this DP is not able to provide proactivity only based on the Word2Vec technic. Indeed, more experiments must be done in order to be convinced of its feasibility. An idea of a solution has been briefly described with abstract analogies [6.4.2], which could provide an indirect layer of abstraction while staying with Word2Vec low-level operations.

Lausanne, June 3, 2019

Romain Claret

Bibliography

[1] A. M. Turing. *COMPUTING MACHINERY AND INTELLIGENCE*. [Online; accessed 26-May-2019]. 1950. URL: https://www.csee.umbc.edu/courses/471/papers/turing.pdf.

[2] Martín Abadi et al. “TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems”. In: (2015). Software available from tensorflow.org. URL: https://www.tensorflow.org/.

[3] Aelu013. *File:CBOW eta Skipgram.png*. [Online; accessed 26-May-2019]. 2018. URL: https://commons.wikimedia.org/wiki/File:Colored_neural_network.svg.

[4] Tim Salimans Alec Radford Karthik Narasimhan and Ilya Sutskever. “Improving Language Understanding by Generative Pre-Training”. In: (2018). URL: https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf.

[5] Amazon. *Amazon EC2 Pricing*. [Online; accessed 26-May-2019]. 2019. URL: https://aws.amazon.com/ec2/pricing/on-demand/.

[6] Amazon. *EC2 Service Limits*. [Online; accessed 26-May-2019]. 2019. URL: https://eu-central-1.console.aws.amazon.com/ec2/v2/home?region=eu-central-1#Limits:.

[7] Amazon Lab126. *Amazon Alexa*. [Online; accessed 26-May-2019]. 2014. URL: https://developer.amazon.com/alexa.

[8] Apple. *Siri*. [Online; accessed 26-May-2019]. 2010. URL: https://machinelearning.apple.com/2018/04/16/personalized-hey-siri.html.

[9] Sergey Bartunov et al. “Breaking Sticks and Ambiguities with Adaptive Skip-gram”. In: *arXiv e-prints*, arXiv:1502.07257 (2015), arXiv:1502.07257. arXiv: 1502.07257 [cs.CL].

[10] Bilby. *File:Turing Test version 3.png*. [Online; accessed 26-May-2019]. 2008. URL: https://commons.wikimedia.org/wiki/File:Turing_Test_version_3.png.

[11] Piotr Bojanowski et al. “Enriching Word Vectors with Subword Information”. In: *arXiv e-prints*, arXiv:1607.04606 (2016), arXiv:1607.04606. arXiv: 1607.04606 [cs.CL].

[12] Chatbots.org Team. *Chatbots.org*. [Online; accessed 26-May-2019]. 2005. URL: https://www.chatbots.org/.

[13] Creative Labs. *Dr. Sbaitso*. [Online; accessed 26-May-2019]. 1991. URL: http://ccftp.creative.com/manualdn/Manuals/TSD/2389/sblive.pdf.

[14] Hannah Devlin. *Google a step closer to developing machines with human-like intelligence*. May 21, 2015. URL: https://www.theguardian.com/science/2015/may/21/google-a-step-closer-to-developing-machines-with-human-like-intelligence (visited on 05/26/2019).

[15] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *arXiv e-prints*, arXiv:1810.04805 (2018), arXiv:1810.04805. arXiv: 1810.04805 [cs.CL].

[16] Facebook. *Bots for Messenger*. [Online; accessed 26-May-2019]. 2016. URL: https://developers.facebook.com/blog/post/2016/04/12/bots-for-messenger/.

[17] Futurism, LLC. *The History of Chatbots Infographic*. [Online; accessed 26-May-2019]. 2016. URL: https://futurism.com/images/the-history-of-chatbots-infographic.

[18] Gensim. *models.word2vec – Word2vec embeddings*. [Online; accessed 26-May-2019]. 2019. URL: https://radimrehurek.com/gensim/models/word2vec.html.

[19] Glosser.ca. *File:Colored neural network.svg*. [Online; accessed 26-May-2019]. 2013. URL: https://commons.wikimedia.org/wiki/File:Colored_neural_network.svg.

[20] Google. *Colaboratory is a free Jupyter notebook environment that requires no setup and runs entirely in the cloud.* [Online; accessed 26-May-2019]. 2019. URL: https://colab.research.google.com/notebooks/welcome.ipynb#.

[21] Google. *Google Assiante*. [Online; accessed 26-May-2019]. 2012. URL: https://assistant.google.com.

[22] Google. *Google Colaboratory Timout*. [Online; accessed 26-May-2019]. 2019. URL: https://cloud.google.com/automl-tables/docs/notebooks.

[23] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[24] IBM. *Watson*. [Online; accessed 26-May-2019]. 2007. URL: https://researcher.watson.ibm.com/researcher/files/us-heq/W(3)%20INTRODUCTION%2006177724.pdf.

[25] Ken Colby. *PARRY: Paranoia mental hospital patient*. [Online; accessed 26-May-2019]. 1995. URL: https://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/classics/parry/0.html.

[26] Ryan Kiros et al. "Skip-Thought Vectors". In: *arXiv e-prints*, arXiv:1506.06726 (2015), arXiv:1506.06726. arXiv: 1506.06726 [cs.CL].

[27] Jey Han Lau and Timothy Baldwin. "An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation". In: (2016), 78–86. URL: http://www.aclweb.org/anthology/D14-1162.

[28] Quoc V. Le and Tomas Mikolov. "Distributed Representations of Sentences and Documents". In: *arXiv e-prints*, arXiv:1405.4053 (2014), arXiv:1405.4053. arXiv: 1405.4053 [cs.CL].

[29] Laurens van der Maaten and Geoffrey Hinton. "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9 (2008), pp. 2579–2605. URL: http://www.jmlr.org/papers/v9/vandermaaten08a.html.

[30] Machine Learning Plus. *Gensim Tutorial – A Complete Beginners Guide*. [Online; accessed 26-May-2019]. 2018. URL: https://www.machinelearningplus.com/nlp/gensim-tutorial/.

[31] Michal Wallace & George Dunlop. *Eliza, the Rogerian Therapist*. [Online; accessed 26-May-2019]. 1999. URL: http://psych.fullerton.edu/mbirnbaum/psych101/Eliza.htm.

[32] Microsoft. *Cortana*. [Online; accessed 26-May-2019]. 2014. URL: https://www.microsoft.com/en-us/research/group/cortana-research/.

[33] Microsoft. *Develop and run code from anywhere with Jupyter notebooks on Azure.* [Online; accessed 26-May-2019]. 2019. URL: https://notebooks.azure.com.

[34] Tomas Mikolov et al. "Efficient Estimation of Word Representations in Vector Space". In: *arXiv e-prints*, arXiv:1301.3781 (2013), arXiv:1301.3781. arXiv: 1301.3781 [cs.CL].

[35] Min. *A Lazy Girl's Guide to Setting Up Jupyter on EC2 for Deep Learning*. [Online; accessed 26-May-2019]. 2017. URL: https://kenophobio.github.io/2017-01-10/deep-learning-jupyter-ec2/.

[36] Jacob Goldberger Oren Melamud and Ido Dagan. "context2vec: Learning Generic Context Embedding with Bidirectional LSTM". In: (2016), 51–61. URL: https://www.aclweb.org/anthology/K16-1006.

[37] Hamid Palangi et al. "Deep Sentence Embedding Using Long Short-Term Memory Networks: Analysis and Application to Information Retrieval". In: *arXiv e-prints*, arXiv:1502.06922 (2015), arXiv:1502.06922. arXiv: 1502.06922 [cs.CL].

[38] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "GloVe: Global Vectors for Word Representation". In: (2014), pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

[39] Peter Lee. *Learning from Tay's introduction*. [Online; accessed 26-May-2019]. 2016. URL: https://blogs.microsoft.com/blog/2016/03/25/learning-tays-introduction/.

[40] Prakash Gupta. *Google Colab FAQs*. [Online; accessed 26-May-2019]. 2019. URL: https://help.clouderizer.com/running-on-cloud/google-colab/google-colab-faqs.

[41] Radim Řehůřek. *Making sense of word2vec*. [Online; accessed 26-May-2019]. 2014. URL: https://rare-technologies.com/making-sense-of-word2vec/.

[42] Radim Řehůřek. *RaRe Consulting*. [Online; accessed 26-May-2019]. 2011. URL: https://radimrehurek.com.

[43] Radim Řehůřek. *RaRe Technologies*. [Online; accessed 26-May-2019]. 2013. URL: https://rare-technologies.com.

[44] Radim Řehůřek and Petr Sojka. "Software Framework for Topic Modelling with Large Corpora". English. In: (May 2010). http://is.muni.cz/publication/884893/en, pp. 45–50.

[45] Richard Wallace. *A.L.I.C.E. A.I Foundation*. [Online; accessed 26-May-2019]. 1995. URL: http://www.alicebot.org/.

[46] Robert Hoffer, Timothy Kay and Peter Levitan. *SmarterChild*. [Online; accessed 26-May-2019]. 2001. URL: https://patents.google.com/patent/EP1767015A1/da.

[47] Rollo Carpenter. *Jabberwacky*. [Online; accessed 26-May-2019]. 1997. URL: http://www.jabberwacky.com.

[48] Xin Rong. "word2vec Parameter Learning Explained". In: *arXiv e-prints*, arXiv:1411.2738 (2014), arXiv:1411.2738. arXiv: 1411.2738 [cs.CL].

[49] Adriaan M. J. Schakel and Benjamin J. Wilson. "Measuring Word Significance using Distributed Representations of Words". In: *arXiv e-prints*, arXiv:1508.02297 (2015), arXiv:1508.02297. arXiv: 1508.02297 [cs.CL].

[50] Ashish Vaswani et al. "Attention Is All You Need". In: *arXiv e-prints*, arXiv:1706.03762 (2017), arXiv:1706.03762. arXiv: 1706.03762 [cs.CL].

[51] Šimon Pavlík. *Gensim word2vec on CPU faster than Word2veckeras on GPU (Incubator Student Blog)*. [Online; accessed 26-May-2019]. 2016. URL: https://rare-technologies.com/gensim-word2vec-on-cpu-faster-than-word2veckeras-on-gpu-incubator-student-blog/.