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

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.

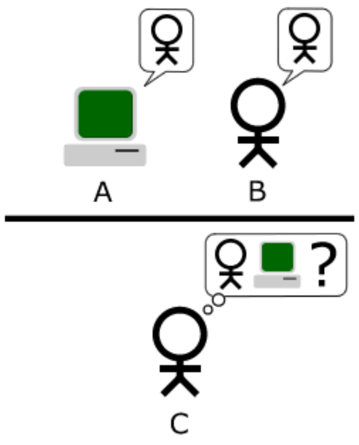


Figure 4.1: The "standard interpretation" of the Turing Test, in which player C, the interrogator, is tasked with trying to determine which player - A or B - is a computer and which is a human. The interrogator is limited to only using the responses to written questions in order to make the determination.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

- 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.

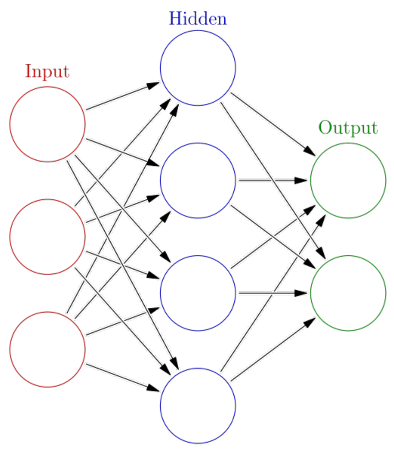


Figure 4.2: Artificial neural network with layer coloring [19]

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.

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.

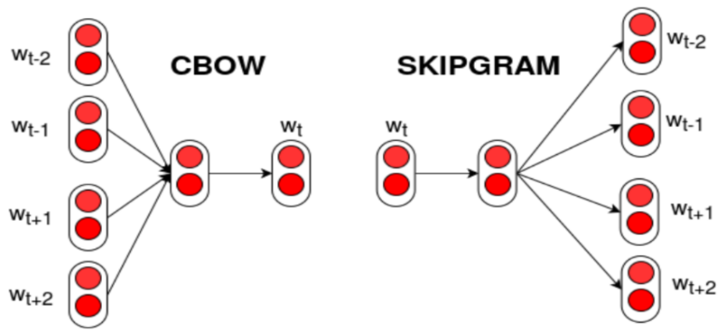
The concept behind CBOW is to take, for a given word vector, the *n* neighboring word vectors as the context, then predict the given word vector, as seen on the figure 5.1.

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

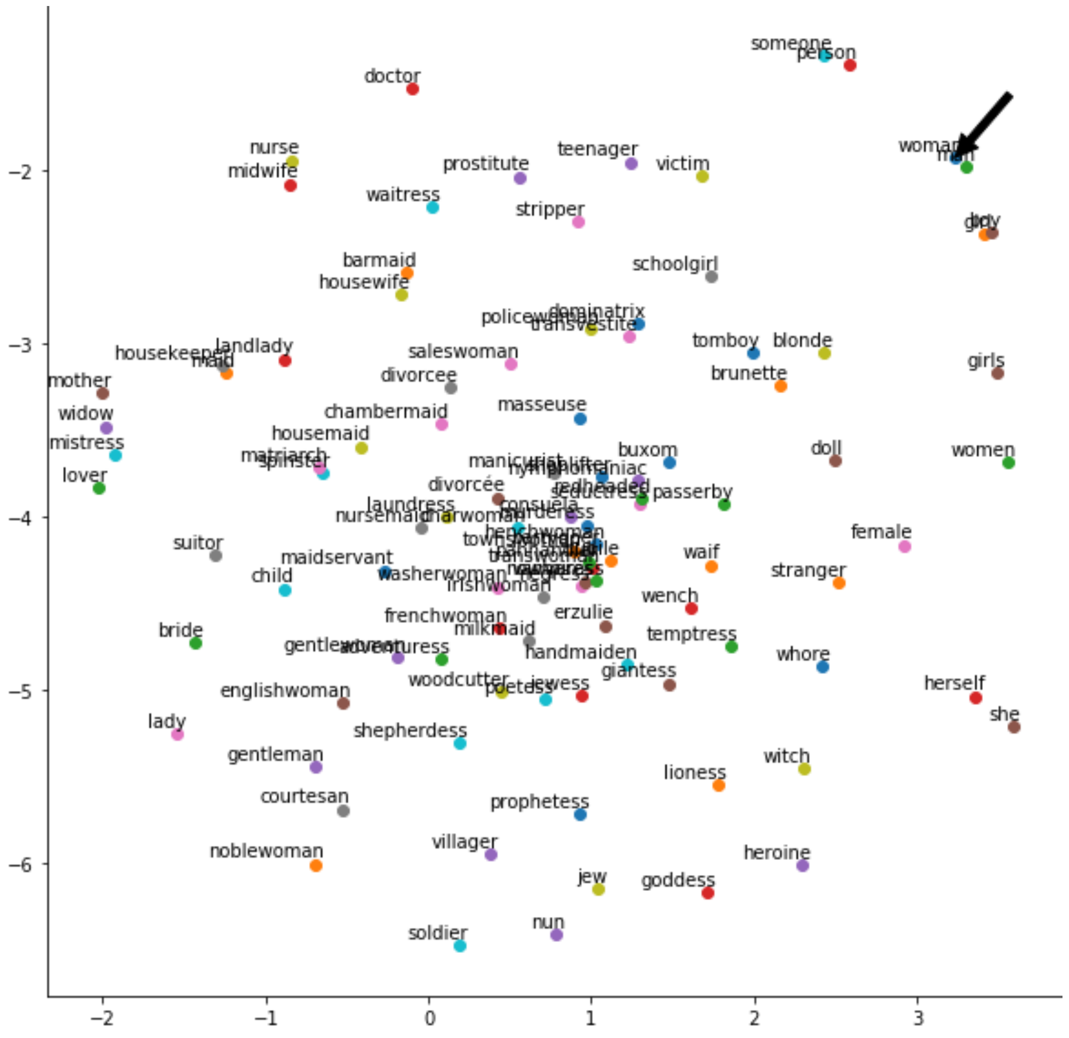


Figure 6.2: Top 100 T-SNE representation of similar words vectors to Woman

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

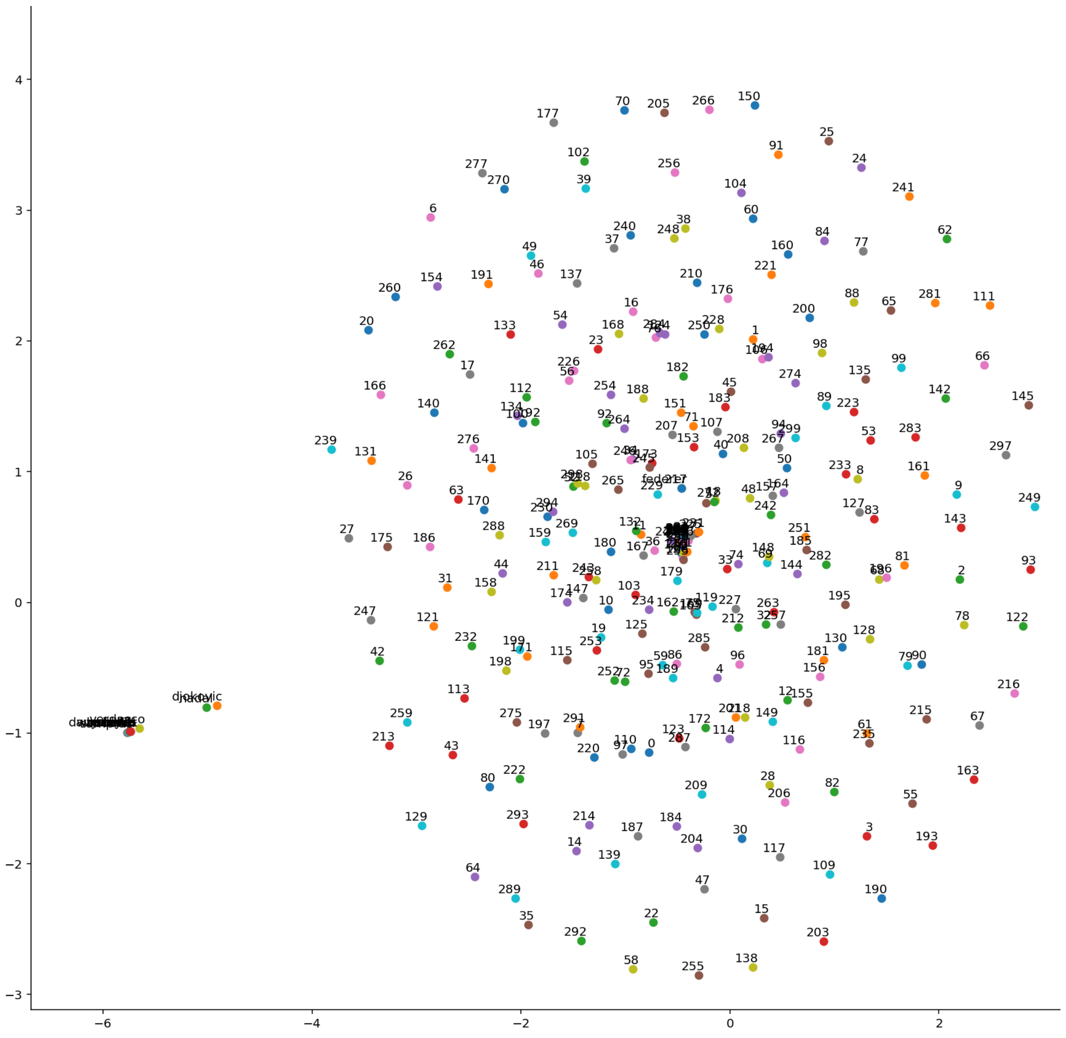


Figure 6.3: Top 5 T-SNE of similar words vectors to the word Federer with altered vector elements

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