Chatty: A multi-modal digital assistant

Department of Data Science and Knowledge Engineering
Maastricht University

2020/2021

William De Clercq, Adele Imparato, Alexei Roșca, Ngoc Hoang Trieu, Nico Grassetto

Abstract—**This paper investigates the implementation of a multimodal digital assistant designed to assist DSAI students. Its aim is to discuss the robustness of the chatbot achieved by using several Image Video Processing (IVP) and Natural Language Processing (NLP) techniques. While using Context-Free Grammar to store different skills that the chatbot can use, other advanced techniques have been added to improve its robustness. These techniques include an auto-correction system, a bag-of-words approach, and some neural network models to allow the user's query to be more flexible. Moreover, a face detection system was implemented and tested with two techniques, namely, skin colour detection based on a color spaces model, and Haar Cascade classifier. Later, the results of the aforementioned techniques were compared on a benchmark dataset to find out that the Haar Cascade classifier achieved higher detection accuracy. A face recognition system was also implemented with the use of Principal Component Analysis (PCA). Finally, an experiment was conducted and an optimal number of 80 eigenvectors was found to be the best compromise between accuracy and computation time.**

I. Introduction

A. *Problem description*

Currently, chatbots are used in various fields; their main task is to assist customers/users. Many services have been replaced by a conversation (a chat) between the customer and a bot. Depending on the context, the bot's goal is to support the customer as much as possible. In the future, these interactions will increase and replace multiple assistant jobs. The objective here is not to replace workers, but to help customers more expeditiously and efficiently.

This paper describes the implementation of a multi-modal digital assistant (a chatbot) called Chatty. Its goal is to assist Data Science and Artificial Intelligence (DSAI) students in their daily student life by interacting with them within a chat window. However, in order to start the chat, a camera must detect if there is a human in front of the screen. Moreover, the user has the possibility to integrate new skills via a skill editor. Similar work has already been performed, the most famous being Apple Siri, Amazon Alexa and Google Home. Nevertheless, there are many other chatbots that are used in different companies as web assistants for their customers.

This paper aims to develop the methods used for the functioning of this digital assistant. First, it will describe the methods used for the NLP part: CFG, CNF and CYK. Based on these techniques, an explanation about the heuristic used for the bot to answer queries will be provided. Second, this paper will describe the techniques used for the IVP part: Haar Cascade classifier, Skin Color Detection and eventually the Principal Component Algorithm. These algorithms will be used for face recognition. Eventually, some experiments comparing the performances of the described algorithms will be performed and discussed. The main goal of this paper is to answer these research questions below:

B. *Research questions*

1) *Is it possible to implement a skin color model that is going to perform as well as a Haar Cascade Classifier?*

2) *What is the optimal number of eigenvectors to successfully recognize faces using PCA (Principal Component Analysis) with satisfactory accuracy and computational speed?*

3) *What is the highest accuracy that can be reached for recognizing utterances from the CFG, and what model achieves this accuracy?*

II. Methods for Natural Language Processing (NLP)

To answer the aforementioned research questions, one needs to focus on the implementation of the chatbot and its functionalities. Below, the algorithms used to build Chatty's abilities are described in detail. The first important part is the NLP one, which concerns the written communication between the bot and the user.

A. *Context-Free Grammar (CFG)*

The information needed for the chat bot to answer basic questions are stored in the form of Context-Free Grammar language, where the starting rules yields the variable "<ACTION>" which, in turn, yields every skills available at our disposition. A CFG works with a set of rules that can entail terminal variable, for instance concrete word such as "which lecture do we have". They are called terminal since they can't be replaced by anything. The other type of variable are called non-terminal. In their cases, they have to be replaced by another variable in order to form a concrete sentence. As long as there remains any non terminal variable, it is possible to replace it with anything that is yielded by that rule. More often than not, rules often yields a mix of both terminal and non-terminal variables. It is possible for a rule to yield more than one possible variable, in which case they are separated by "|".

B. *Chomsky Normal Form (CNF)*

In order to be used further in the process, a phrase and its placeholders in context-free grammar needs to be converted to Chomsky Normal Form. It is therefore used as a pre-processing step to be further used by the CYK algorithm as this one works with CNF's. Moreover, it enables a polynomial algorithm to decide whether a string belongs to a grammar. The process works in four steps:

1) First, we remove null productions.
2) Second, unit productions are removed. That is, we remove the productions wherein one non-terminal symbol gives one non-terminal symbol.
3) Third, we replace each production where n > 2 (n being the number of symbols in the right hand side)
4) Fourth, for every production we repeat the following step: if the production is of the form $A \to \alpha B$ we replace by $A \to XB$ and $X \to \alpha$

See Algorithm 2 in Appendix.

C. *The Cocke–Younger–Kasami-Algorithm (CYK)*

In order to confirm that a query from the user belongs to the Context Free Grammar, there is no better performing algorithm than the CYK. It however requires the CFG to be in Chomsky Normal Form. The time complexity of the algorithm is $O(n^3|G|)$, where n is the length of the input string, and $|G|$ is the size of the grammar in CNF. Since the size of the grammar is constant, the complexity simplifies to $O(n^3)$.

The algorithm traverse each word from the user's query, to verify if any rule from the CNF yields that word. If so, the boolean array is set to *true*. This is the first iteration of the algorithm, looking at each word individually. The next steps involve checking multiple words at the same time, traversing again all rules, in order to see if any of them yields the current set of words that are being checked.

Little by little, the algorithm checks a bigger portion of the query, each time looking at what rules yield the smaller part of it. At the end, if it determines that the starting rule yields the whole sentence, it simply means that the input belongs in fact to the CFG language, otherwise it does not.

The boolean array represents the rules that yields a specific portion of the query. The first dimension represents the starting index of the portion of the query currently being checked. The second dimension represent how many words it is looking at. And finally, the last dimension represents the rule that yields (if true) or not (if false) the specific portion of the input.

Here is the pseudocode, first published by Itiroo Sakai in 1961[9].

See Algorithm 1 in Appendix.

D. *Answering mechanism*

During the CYK algorithm, variables are saved in a list, in order to retrieve the appropriate answer from the user input, if the skill list permits to do so. Since the CFG was converted to CNF, it is natural to convert it back to the original context-free grammar language.

It is easily achieved as the additional variable of the CNF have all a specific pattern that is never seen elsewhere. While this pattern is present in the variables from the list, we replace each term containing the specific pattern to its true value.

A simple traversal of the actions file containing the different answers while comparing the list is sufficient to retrieve the response. At the end of the file, if no significant match is found, the bot will simply answer "I don't know".

E. *Heuristic-based Question Answering System*

For benchmark sake the decision was taken to implement a context-free chatbot. The context-free chatbot does not take into account the sequences of words nor the context in which the words are placed. It builds a probability that a question is in the database by using the following formula that was created to that purpose:

$$\frac{P_{wordsincommon} + P_{letterateachindex}}{2} \tag{1}$$

where $P_{wordsincommon}$ builds a probability of match between the user's question and a given question in the database based on the number of words in common. And $P_{letterateachindex}$ builds a probability of match between the user's question and a given question (coming from the database).

The probability is based on the number of letters at the correct index. By correct index, we mean the index where the letter in the user's question is the same as the letter at the same index in the database question. That probability is then "relaxed" by being more lenient when the letter that is not at the right index is replaced with a letter than is one of its direct neighbours on a QWERTY keyboard layout.

Do note that even though the responsible class can support multiple keyboard layouts, we took the decision to use a QWERTY layout as reference as it is the most used layout among our target audience. Therefore, the algorithm is also robust to different keyboard layouts as long as those are provided in the following .txt format: <keycap symbol>|<neighbour1>, <neighbour2>, ... For example, one might want to insert the neighbours of Q on the keyboard like so: q|w, a, s and the same logic applies to other keyboard's keycaps.

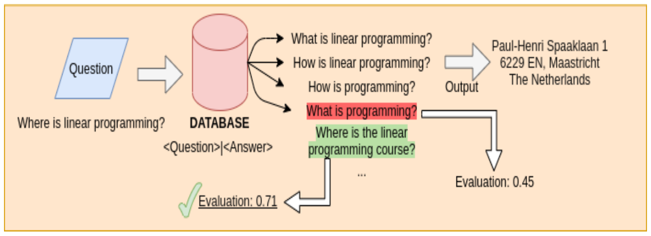

Figure 1: Heuristic-based question answering system

1) Weaknesses: A heedful reader might have noticed some inherent weaknesses to this approach. This, not to mention that the algorithm assumes a sentence in plain English rather than in CNF doing so limiting its effectiveness.

Secondly, besides not taking into account context, the algorithm does not recognise nor adds weights to questions with the same meaning but a totally different structure.

2) Strengths: Be this as it may, this approach is easy to implement and does not require any form of training which makes it a good benchmark for more complex approaches.

F. *Advanced techniques*

Whilst it is certainly interesting to know if a specific query belongs to the language of a given CFG, it is somewhat lacking in flexibility. That approach does not take into account if a user forgets how to ask a specific question. Neither does it take into account the different ways of asking for the same task. Spelling mistakes can also be a hindrance to the performance of the system.

1) Auto-correct: To handle misspellings, one may implement an auto-correct algorithm that stores a domain specific corpus related to DSAI in a Trie data structure. Following this idea, a word is represented by the sequence of characters going from a root node to an end node (a node that has been defined as the end of a real word). The algorithm takes into account edit distance/Levenshtein distance (deletion, transposition, alteration, insertion) to return the likeliest word. To break ties, having also stored the words' frequencies (amount of times it appeared in the corpus) in the trie, the algorithm returns the most used word (further ties are determined alphabetically). This solution is effective as long as the corpus it is trained on captures all possible inputs. Otherwise it will force interpretations (attribute words inside the corpus to words that are not within it).

2) Bag of words (BoW): What if the trained corpus does not capture all possible inputs ? One simple solution to that is to use the Bag of Word approach. The robustness is further improved by adding the above mentioned auto-correct on the user's query. The query is then compared to all possible sentences generated by the CFG language. It then returns the answers to the question which matches best the input query, if the similarity is above a certain threshold. This threshold play a key role in the viability of this approach, as it is not desired to match any query, nor to be too strict either.

The problem with a Bag of Words classifier is that it does not handle sentences with similar meanings that are not produced by the CFG. It can also return false positives on invalid sentences if the latter have sufficient words in common with a valid sentence. A schema detailing the system with this enhancement can be found in the appendix 8.

3) Convoluted Neural Network (CNN): This approach relies on training a model on the different sentences generated by the CFG, and see if the algorithm is able to recognise queries from the language, and sentences that are not generated by the context-free-grammar.

A critical step in this approach is to convert the user's input into a Word2Vec representation as it is important that each word correctly reflects its context/meaning. For this purpose, a Word2Vec model was trained on a corpus of one million sentences from diverse DSAI resources (student handbook, machine learning books, etc.) as well as sentences generated by the CFG.

The context-free grammar holds 52 different "skills", or set of phrases that treat the same theme. For instance, one could ask "What class do I have on Monday?" as "Which lecture is there on Monday?" and ideally, those two sentences would be recognized as the same skill, and both would belong to the CFG. It requires however, at the time of creation of the skill to be hand coded in the application. For each of these skills, the number of possible sentences varies from half a dozen to a hundred. Do bear in mind that the significant difference is due to the fact that some skills benefit from variables with a high number of available possibilities whereas other are treating a very specific theme. Using all combinations of available variables, from those 52 skills, around 1400 context-free grammar sentences can be generated.

These sentences were used to train a CNN to recognise a query from a specific skill. The data was split into around 80-20 for each skill and the testing was done on occurrences never seen in the training set. The model was provided by Alex Black [2]. The structure of the CNN is composed of 3 convoluted layers with filter width of 3,4 and 5 as per Kim paper [5]. The aggregation layers uses Max-pooling to extract 100 features from each filter. The concatenation uses marge vertex which performs depth concatenation. The input layer uses a leaky Relu activation whereas the

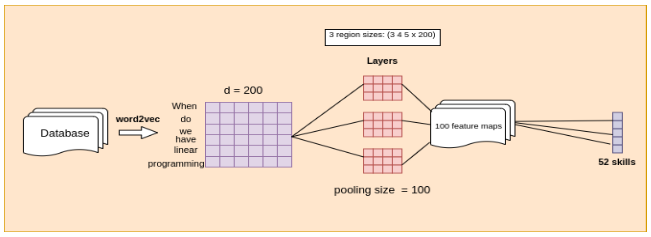

Figure 2: Convolutional neural network

output uses SoftMax in order to have probabilities. For the training, the complex optimizer Adam was used, the learning rate was "0.01" and an $l2$ normalizer was applied.

Using this method, it is possible to determine whether a query has a high probability of belonging to our CFG-based language. If so, the next steps involve using the Bag of word approach with a low threshold to search for the closest actual query from the CFG in order to provide an appropriate response.

4) Recurent Neural Network (RNN): RNN's are meant to be more effective than CNN's as they keep track of the sequential order of sentences globally, whereas the latter have only local context. However, there are two major problems in implementing any NN model. The first one is a lack of data. Indeed, to produce an effective model, sufficient variations of possible queries is necessary. Yet, no such database exists for DSAI. For this reason, one had to find ways to bootstrap variations from sentences produced by the CFG by settling on producing variations by misspelling words and transposing parts of sentences. The idea being to create a Long-Short-Term Memory (LSTM) RNN where each character is a data point in the sequence, and every subsequent character is a label for the previous character, with the final character being the label for the entire sequence. In the appendix, a diagram detailing the structure of the flow of the algorithm can be found 8.

*5) **Data Augmentation**:* In order to improve the accuracy of this learning algorithm, several data augmentation techniques can be used. Including EDA that was first proposed by Jason Wei, and Kai Zou [4].

The proposed method augments the data by:

1) Replacing words by a synonym given a certain probability.

2) Swapping words without altering the correctness and the sense of the phrase.

3) Deleting words without altering the correctness and the sense of the phrase.

4) Adding words without altering the correctness and the sense of the phrase.

This approach, despite its obvious simplicity, shows on most NLP applications significant results improvements.

There exists a famous state-of-the-art data augmentation library that is well known for its flexibility and its simplicity of use (**nlpaug**). This library provides several functions that each generates new sentences based on a given sentence. After thorough investigation of the library and its key functions,one can set up two strategies:

Augmentation via back translation: It appears that there was not much difference between the original dataset of sentences (thus, including training dataset and testing dataset) and the output using back translation. This is probably due to the size of each sentence. Each sentence being relatively small, it is hard not to say almost impossible with modern translators to diverge

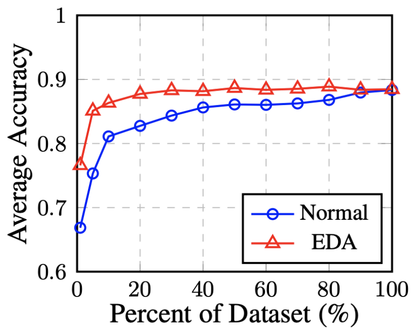

Figure 3: Performance boost [11]

from the original structure. The original structure would be altered for text corpora of medium to big size. As a result, this approach can be abandoned.

Contextual Word Embeddings Augmentation: A word augmentation technique was used to augment the data with word substitutions, deletions, additions using BERT and wordnet. As mentioned earlier the new words were placed in a folder called "unknown" to test our model on completely new cases wherein some sentences were not completely correct from an English standpoint, doing so testing the robustness of this strategy.

III. Methods for Image & Video Processing (IVP)

A. *Skin Color Detection*

Since one of the best approaches to finding and extracting human faces is considered to be skin color, a skin color model was implemented. The primary color spaces used to detect skin tones are RGB (red, green, blue), YCbCR (luminance, chrominance), and HSV (hue, saturation, value). For linear separation of the pixels, the rules proposed by See et al. [1] were used. In the RGB color space, a distinction must be made between uniform daylight illumination and flash or lateral illumination. For uniform daylight illumination, the following rule [6] is used:

$$\begin{aligned}&R > 95, G > 40, B > 20,\\&(Max\{R,G,B\} - min\{R,G,B\}) > 15,\\&|R - G| > 15, R > G, R > B\end{aligned} \tag{2}$$

while the skin tones under flashlight or lateral daylight illumination are given by

$$\begin{aligned}&R > 220, G > 210, B > 170,\\&|R - G| \leq 15, B < R, B < G\end{aligned} \tag{3}$$

Finally, to combine both of these rules a logical 'OR' is used.

$$\textbf{Rule RGB} = Rule2 \cup Rule\,3 \tag{4}$$

The five bounding rules which are combined using a logical 'AND' in the YCbCr subspace formulated from its 2-D subspace distribution surrounding the skin color regions are as follows:

$$\begin{aligned}&\textbf{Rule YCbCr} =\\ &Cr \leq 1.5862 \times Cb + 20 \cap\\ &Cr \geq 0.3448 \times Cb + 76.2069 \cap\\ &Cr \geq -4.5652 \times Cb + 234.5652 \cap\\ &Cr \leq -1.15 \times Cb + 301.75 \cap\\ &Cr \leq -2.2857 \times Cb + 432.85\end{aligned} \tag{5}$$

In the HSV space space, the hue values provide the best separation of skin tones from non-skin tones. The skin tone hue values are estimated as follows:

$$\textbf{Rule HSV} = H < 25 \cup H > 225 \tag{6}$$

Now, any pixel that satisfies the requirements for the three rules in the RGB, YCbCr, and HSV color spaces is considered a skin tone region. To correctly detect faces in large groups of people, a morphological opening was also used. The problem lies in the fact that two faces close enough to each other in an image could be considered as one person, so the opening aids us to pull apart connected regions.

B. ***Haar Cascade Classifier (HCC)***

Haar Cascade is a feature-based classifier. It is first trained on a large dataset of positive images (with faces) and negative images (without faces). Second, the features of these images, which act similar to convolution kernels, are extracted.

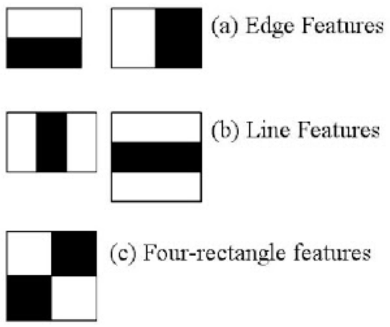

Figure 4: Haar features [7]

Every single feature is a value derived from the subtraction of the sum of pixels under the white rectangle from the sum of pixels under the black rectangle, as it can be visualized in Figure 4. Subsequently, these kernels are applied to all possible locations and sizes of an image. This makes the process quite slow, and to accelerate its speed, integral images come in handy. Instead of computing a value for each pixel, we compute only the values of sub-rectangles that resemble a region.

Considering that there are many computed features, we need to find the best possible ones. This is achieved by using **Adaboost** [10], where all features are applied to all training images and the ones with the lowest error rates are selected. These features are considered "weak classifiers" because they cannot classify an image by themselves, but in combination with the other features they can. The final classifier is a weighted sum of these classifiers, which alone cannot classify an image.

The concept of **Cascade of Classifiers** is introduced. Rather than applying all features to a window, features are grouped into stages of classifiers and applied sequentially. If a window fails a stage, it is marked as a non-skin region.

C. ***Principal Component Analysis (PCA)***

The PCA algorithm is used for face recognition. The main idea behind it is to reduce a large data space, such as the observed variables of a face, into a smaller feature space containing only the principal components of it. The simplest and one of the most effective techniques for recognizing a face using PCA is the eigenface approach. It operates as follow: to recognize a face, one has to compute the covariance matrix from the initial training set of face images, obtain the eigenvalues of the covariance matrix as well as the eigenvectors. The whole process can be divided into two main phases [8]: the training phase and the recognition phase.

1) ***Training phase:*** The first step of the training phase is to acquire a training set of images consisting of different person faces. To this end, online databases can be accessed or even taking a lot of pictures of different people. Next, one computes the eigenfaces by keeping the eigenvectors corresponding to the highest eigenvalues of the covariance matrix. Eigenvectors of the covariance matrix form the orthonormal basis for the eigenspace in which most of the image data can be represented. Note that we are only interested in keeping the largest ones because they reflect the greatest variance in the images. Then, for every person in the training set, his or her faces are projected onto the eigenspace to compute the distribution.

2) ***Recognition phase:*** To avoid errors, the input image is first tested with the Haar Cascade Classifier. If a face is successfully detected, the recognition process is started. This is done by projecting the input image onto each of the eigenfaces in the eigenspace. An example of eigenfaces can be visualised in Figure 5. Basically, the distance between the input image and the stored faces is calculated. The minimum distance is selected and if it is less than a certain predefined threshold, the label corresponding to the face with the minimum distance is assigned to the input image. However, if the minimum distance is greater than the threshold value, the input image is classified as "unknown face".

IV. **Experiments**

A. ***Face Detection Performance Comparison***

The goal of this experiment is to find out which is the best algorithm for face detection and whether it is possible to achieve the accuracy of the Haar Cascade Classifier with a skin color model. To accomplish this task, three variants of skin color model algorithm were used namely:

RGB model, RGB-HSV model, RGB-HSV-YCbCr model as well as Haar Cascade Classifier. The dataset used to conduct the experiment was the "CalTech Faces 1999" [3] dataset. It contains 450 images of about 27 people under different lighting, expressions, and backgrounds, which would be a good way to test the skin color models since the results vary depending on the lighting. Since PCA is used exclusively for face recognition, it was not included in this experiment as it would not be possible to compare it with the other algorithms under the same conditions, because it requires a training set of images.

Two metrics were used to evaluate the performance of the algorithms, False Detection Rate (FDR) [1], which is the number of false detections (face detected even though it is not present) divided by the total number of detections, and Detection Success Rate (DSR), which is the number of correctly detected faces divided by the total number of faces.

B. *Optimal Eigenvectors PCA*

The idea behind this experiment is to find the optimal number of eigenvectors/eigenfaces to keep and use later in the recognition phase. Since the number of eigenvectors can greatly affect the time required to recognize a face as well as the training time of the algorithm, we will try to find the best tradeoff in terms of accuracy and computation time. The dataset used here is the same as for the experiment above, namely "CalTech Faces 1999". The images are divided into two sets, 400 of which belong to the training set and 50 to the test set. As for the performance measure, **DSR** is going to be used. So, the algorithm will try to predict the label of the images from the test set, and if the predicted label matches the actual one, it is considered as a successful recognition. To compare the different configurations, the total time taken to train the PCA algorithm and predict the 50 labels is also used. Image dimensions for both sets are 896x592 pixels.

C. *Performance Analysis of the context free spellchecker and the Machine Learning based systems*

To be able to test the effectiveness of this approach, it is important to use sentence never seen in the training set, as well as sentences which cannot derived from our CFG language. Separating the whole set of sentence generated by the CFG language is the obvious approach to create a test set. In order to create phrases not yield by the CFG, which have to be comprehensible and syntactically correct in an semi-automated way was done by the step "Data Augmentation".

Using this method, it is possible to generate at least five sentences that are somewhat grammatically correct, as well as consistent with the meaning of the current skill. The number of sentence should be balanced amongst each of these skill, which is not the case with the original test set. This help to counter balance the fact that the instances with higher impact on the training would also be the kind of sentence that are easily recognize, as well as the fact that they appear more often in the test set, and therefore influence the F-score the most. Keeping a set of "unknown" utterance of 5 per skill will help in making a more accurate evaluation. These unknown sentences can be classified with the correct skill as well as positive instances from the CFG.

Another set of sentence was created for the purpose to have negative sentences. It was hand-made and consists of 50 sentences completely random, and 50 sentences using more specific words coming from the CFG.

V. **Results**

A. *Face Detection Performance*

As can be seen in the following table, the algorithm with the best performance when it comes to face detection is Haar Cascade Classifier scoring a DSR of 99.33%, directly followed by the RGB-HSV-YCbCr model with a DSR of 81.33%.

Algorithms	FDR (%)	DSR (%)
Haar Cascade Classifier	18.82	99.33
RGB-HSV-YCbCr	**20.26**	**81.33**
RGB-HSV	26.72	73.55
RGB	37.84	64.22

Table I: *Face detection performance among algorithms*

B. *Face recognition accuracy*

When testing the performance of the PCA algorithm, there was no improvement in accuracy after 80 eigenvectors were used to predict the label of the training image. However, there was a significant increase in training/prediction time.

Eigenvectors Count	DSR (%)	Total Time (s)
10	36	66.2
20	42	70
40	46	71.5
80	**52**	**76.9**
160	52	116.9
All	52	212

Table II: *Face recognition results based on a different number of eigenvectors*

C. *Machine learning based approach*

For the CNN model, the evaluation on the test set has an F-score of 0.9678, where the incorrectly classified instances have a tendency to belong to classes with fewer training examples. More details can be found in III.

Number of class	52
Accuracy	0.9895
Precision	0.9828
Recall	0.9641
F1 Score	0.9678

Table III: Test on unseen instances belonging to the CFG

Evaluating the CNN model on the generated sentences by the data augmentation step yields an F-score of 0.75, which is lower than the bench-mark approach. The table IV shows the numbers with more details.

Number of class	52
Accuracy	0.7462
Precision	0.8289
Recall	0.7462
F1 Score	0.7591

Table IV: Test on unknown utterances

For the RNN, the evaluation was only done using instance of CFG. It obtains a poor F-score of 0.035.

Number of class	52
Accuracy	0.1956
Precision	0.1956
Recall	0.0192
F1 Score	0.035

Table V: Test on RNN

For the BoW approach, recognizing unknown sentences that are some kind of variation of the CFG-based language is fairly high with 94.62% of correctly identified instances, however the true negative are as low as 52.48%:

- Valid sentences score: 0.9461538461538461
- Invalid sentences score: 0.5247524752475248

Number of class	2
Accuracy	0.8283
Precision	0.8367
Recall	0.9462
F1 Score	0.8881

Table VI: F-score for BoW

For the CNN model, it recognizes 63.08% of unknown utterances correctly and reject correctly 68.32% of the sentences that are outside of the CFG skills' scope.

- Valid sentences score: 0.630769230769230
- Invalid sentences score : 0.683168316831683

Number of class	2
Accuracy	0.6454
Precision	0.6308
Recall	0.8367
F1 Score	0.7193

Table VII: F-score for BOW

VI. **Discussion**

A. Face detection performance

Based on the results seen in Table I, we can safely say that a skin color model is significantly worse than the HCC, with a higher FDR and a lower DSR in every possible configuration of the skin color model. The results were also worse when the skin color model that used three color spaces was reduced to only two or one color space. We found that the skin color model usually performed slightly better when used on images with good lighting, but the dataset used in the experiments contained images of individuals in very different illumination settings. However, we were still interested in finding the algorithm that produced the best results without being influenced by external factors. Since the detection of a face with the skin color model relies heavily on pixel intensity values, it often detects a face if part of the background in the test image contains something resembling a skin tone, such an error can be seen in Figure 7. To counter this, we tried using the ratio of the dimensions of a face, but this only helped us slightly reduce the FDR of the algorithm.

B. Face recognition accuracy

When considering the optimality of the number of eigenfaces used to predict a label, we must appropriately choose the number that gives us high accuracy while keeping the computation time relatively low. Since the Total Time in Table II represents the time it takes to train the model on 450 images and then predict the labels for 50 images, we decided that the best number of eigenvectors is 80, because the difference in computation time will not really affect Chatty users. The accuracy may seem a bit on the low side, but this is due to the dataset used for the experiments. It is known that PCA performs slightly worse when the training images have different illumination than the test images, and the dataset has different illumination for almost every image of the same person. In addition, the images from the training and test sets were not aligned according to face coordinates, which also affects the accuracy of the algorithm, but the purpose of this experiment was to find the optimal number of eigenvectors rather than trying to achieve the highest possible accuracy.

C. The different approaches for NLP

The CNN-based model performs fairly well on the task of recognizing instances of the CFG language that was unseen in the training set. It was expected since the structure of the sentences from the CFG are all the same: the few mistakes made are from skills where few variations were possible, due to the lack of content variety. For instance, "Which lectures do we have on Monday at 9" can be changed in the CFG thanks to the day of the week, the pronoun, and even the time. The mistakes appear when no such variations are possible, and therefore little data was available.

When faced with the transformed phrases by BERT, the F- score drops significantly, showing the limitation of such a model. As the score for classifying different sentences into the different skill is significantly reduced, we assume that using this classifier to recognize unknown utterance based on the probability output by the model unreliable. We suspect that the model overfits the training set, which is too small to begin with.

For the RNN, unfortunately, we were unable to produce a neural network that could train a meaningful model. This could have been due to our data augmentation technique but it could also be due to failure in engineering a proper network. Indeed, we chose DeepLearning4J as our deep learning library and we found the documentation to be lacking (deadlinks, errors in documentation). The library has not been updated in over a year, and its developing community is essentially inactive. It is our opinion that we should have gone with TensorFlow, or Pytorch and used a Java wrapper.

The results on classifying unknown utterances for the BoW approach reflects the obvious weakness of such a method: 50% of the data were made using words coming from the CFG. The high false positive rate is proof that it was enough to throw off the performance of this model. However, the CNN approach isn't much better as it recognize with high confidence around 63% of the "unknown" positive utterances, but has a lower rate of false positive (around 30%).

It is important to note that finding a way to deal with synonym was a major roadblock for us. As mentioned in the data augmentation part, the results were sometimes, at most, barely comprehensible. Some synonyms also were not interesting to us as it doesn't necessarily fit in our context. For instance, in "Which course should I take...", the word "course" could also mean "meal", and doesn't belong in the CFG skills set. As to a way to deal with synonym as input, Word2Vec couldn't solve this issue, as this representation is influenced by the context of other meanings of the synonyms. Rephrasing sentences while keeping the same general idea could be considered a project of its own right.

It was therefore decided to use the BOW approach for the implementation of Chatty due to the poor results of the above mentioned models, and the even lower F-score than the baseline approach.

VII. **Conclusion**

The conception of Chatty involves a NLP component where a context free grammar was used as a baseline implementation. Further improvement involved enhancing the robustness of the chatbot, by using a spell checker to handle spelling mistakes, and some Machine Learning based techniques to handle synonym and different sentence structures not belonging to the language.

One of the most important hurdle was mainly the lack of data. Using varying methods to augment the data available, BERT, while not perfect, stood out to be the most effective. The sentences ended up being used to test the machine learning based models. The CNN model showed a significantly lower score when predicting the class of the newly made data. Another problem was the lack of relevant documentation using the library deeplearning4j, which hindered the conception of a valid RNN model. The better option would have been to use any python library and a Java wrapper.

In the end, Chatty uses the Bag of Word approach to improve the performance of the chat functionality. While not optimal, it is now possible to formulate question in other ways to get an answer from the chatbot.

As for the IVP part, after rigorous testing of various configurations of the skin color model, with many methods to try to make it better, we could not get its DSR close to that of the HCC. Therefore, the default option when using Chatty for face detection is HCC, with the possibility to change it to Skin Color Detection based on user preferences.

If the user desires Chatty to recognize his face, he can simply add his face to the database using the software. Later, the PCA with 80 eigenfaces will be used to recognize the user as there is no difference in the accuracy of the predictions with a higher eigenface count. The prediction time will also be reasonable as the actual dataset used in the software is much smaller than that of the experiments.

References .

[1] Nusirwan Anwar bin Abdul Rahman, Kit Chong Wei, and John See. "Rgb-h-cbcr skin colour model for human face detection". In: Faculty of Information Technology, Multimedia University 4 (2007).

[2] Alex Black. Convolutional Neural Networks for Sentence Classification. https : / / github . com / breandan / deep - learning - samples / blob / master / dl4j - examples / src / main / java / org / deeplearning4j / examples / convolution / sentenceclassification / CnnSentenceClassificationExample.java. 2017.

[3] CalTech. Faces 1999 (Front). http://www.vision.caltech.edu/archive.html. 1999.

[4] Kai Zou Jason Wei. "EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks". In: Computer Science, Cornell University 4 (2019).

[5] Yoon Kim. "Convolutional Neural Networks for Sentence Classification". In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1746–1751. DOI: 10.3115/v1/D14-1181. URL: https://www.aclweb.org/anthology/D14-1181.

[6] J. Kovac, P. Peer, and F. Solina. "2D versus 3D colour space face detection". In: Proceedings EC-VIP-MC 2003. Vol. 2. 2003, 449–454 vol.2. DOI: 10.1109/VIPMC.2003.1220504.

[7] OpenCV. Face Detection using Haar Cascades. https://docs.opencv.org/master/d2/d99/tutorial_js_face_detection.html.

[8] Liton Chandra Paul and Abdulla Al Sumam. "Face recognition using principal component analysis method". In: International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) 1.9 (2012), pp. 135–139.

[9] Itiroo Sakai. "Syntax in universal translation." In: International Conference on MTL and ALA. Vol. 4. Her Majesty's Stationery Office., 1961, pp. 593–608.

[10] Robert E Schapire. "Explaining adaboost". In: Empirical inference. Springer, 2013, pp. 37–52.

[11] Jason Wei. EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. https://github.com/jasonwei20/eda_nlp. 2019.

Appendix

Algorithm 1 CYK Algorithm

```
Require: $query$ input string
Require: $CNF$ contains unit rules $R_1, ..., R_r$
 1: $n = length(query)$ where length is the number of words
 2: $P[n, n, r] \leftarrow false$
 3: for $w = 1$ to $n$ do
 4:     for all unit production $R_v \rightarrow query_w$ do
 5:         $P[1, w, v] \leftarrow true$
 6:     end for
 7: end for
 8: for $l = 2$ to $n$ do
 9:     for $s = 1$ to $n - l + 1$ do
10:         for $p = 1$ to $l - 1$ do
11:             for all unit production $R_a \rightarrow R_b R_c$ do
12:                 if $P[p, s, b]$ and $Pl - p, s + p, c$ then
13:                     $P[l, s, a] \leftarrow true$
14:                 end if
15:             end for
16:         end for
17:     end for
18: end for
19: if $P[n, 1, 1]$ is true then
20:     query is member of language
21: else
22:     query is not member of language
23: end if
```

Algorithm 2 Conversion to Chomsky Normal Form

```
Require: a string w
 1: start symbol $S_{start}$
 2: for all $unitsproduction$ do
 3:     if $S_{start} atRHSofproduction$ then
 4:         $S_{new} \rightarrow S$
 5:     end if
 6: end for
 7: for all elements in the grammar do
 8:     remove $\epsilon$
 9: end for
10: for all $productionsP$ do
11:     if $IsUnitProduction$ then
12:         $remove(P)$
13:     end if
14: end for
15: for all $productionsP$ do
16:     if $form(P) = A \rightarrow \alpha B$ then
17:         $A \rightarrow \alpha B := A \rightarrow XB$
18:     end if
19: end for
```

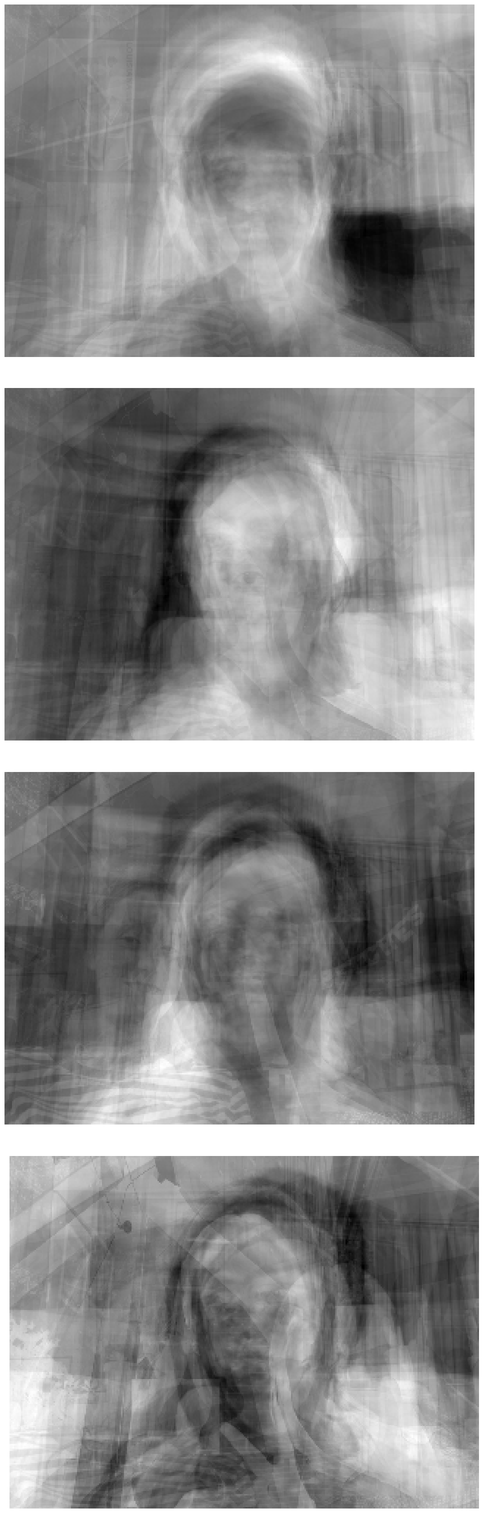

Figure 5: Eigenfaces example trained on our faces

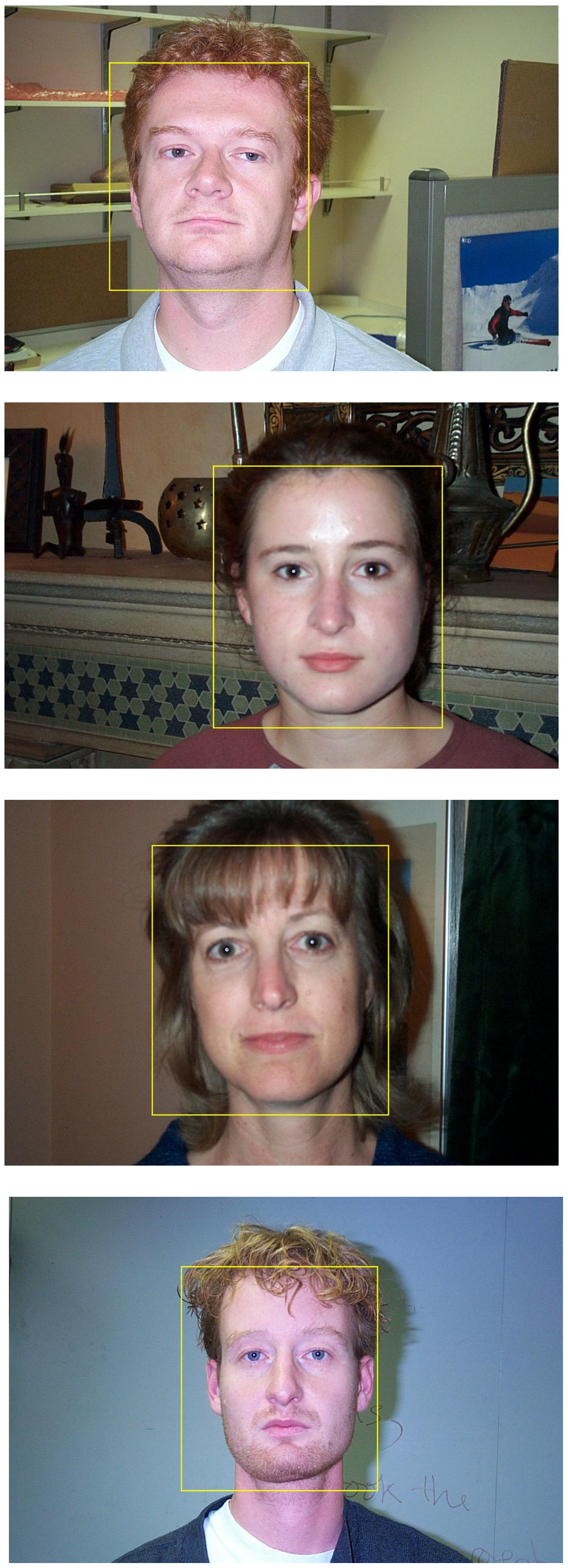
Figure 6: Face detection using HCC on CalTech dataset

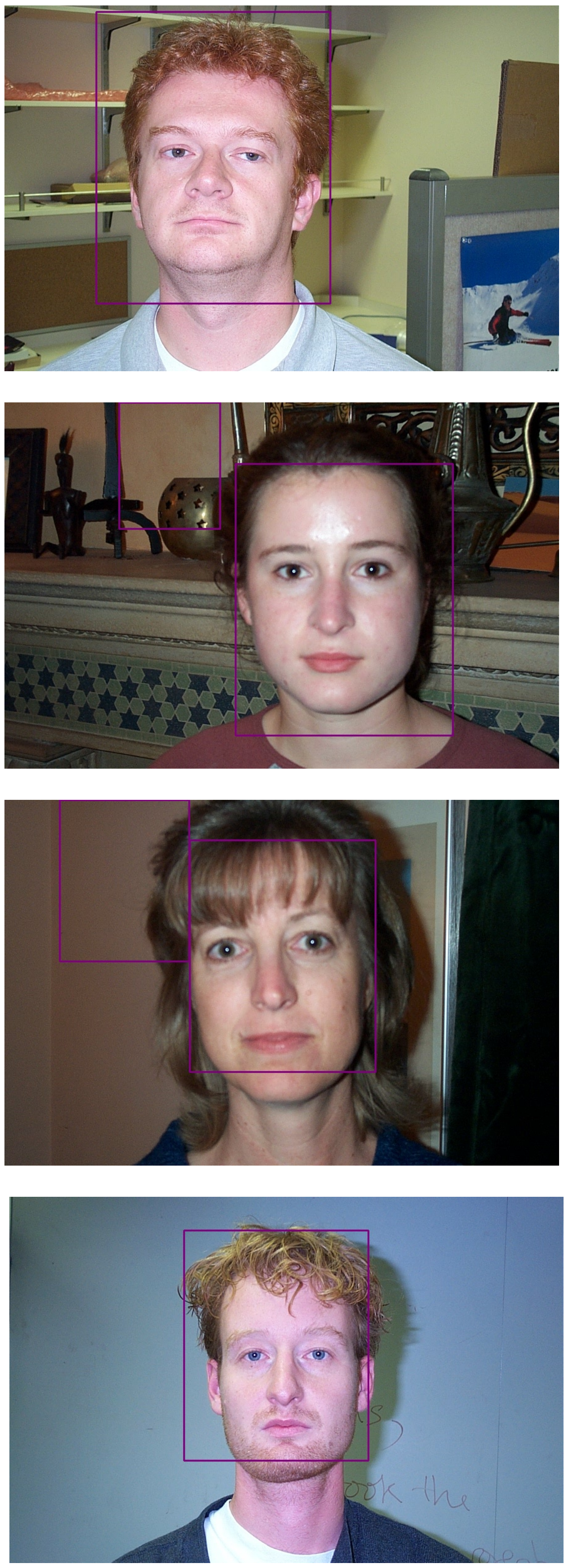
Figure 7: Face detection using the RGB-HSV-YCbCr model on CalTech dataset

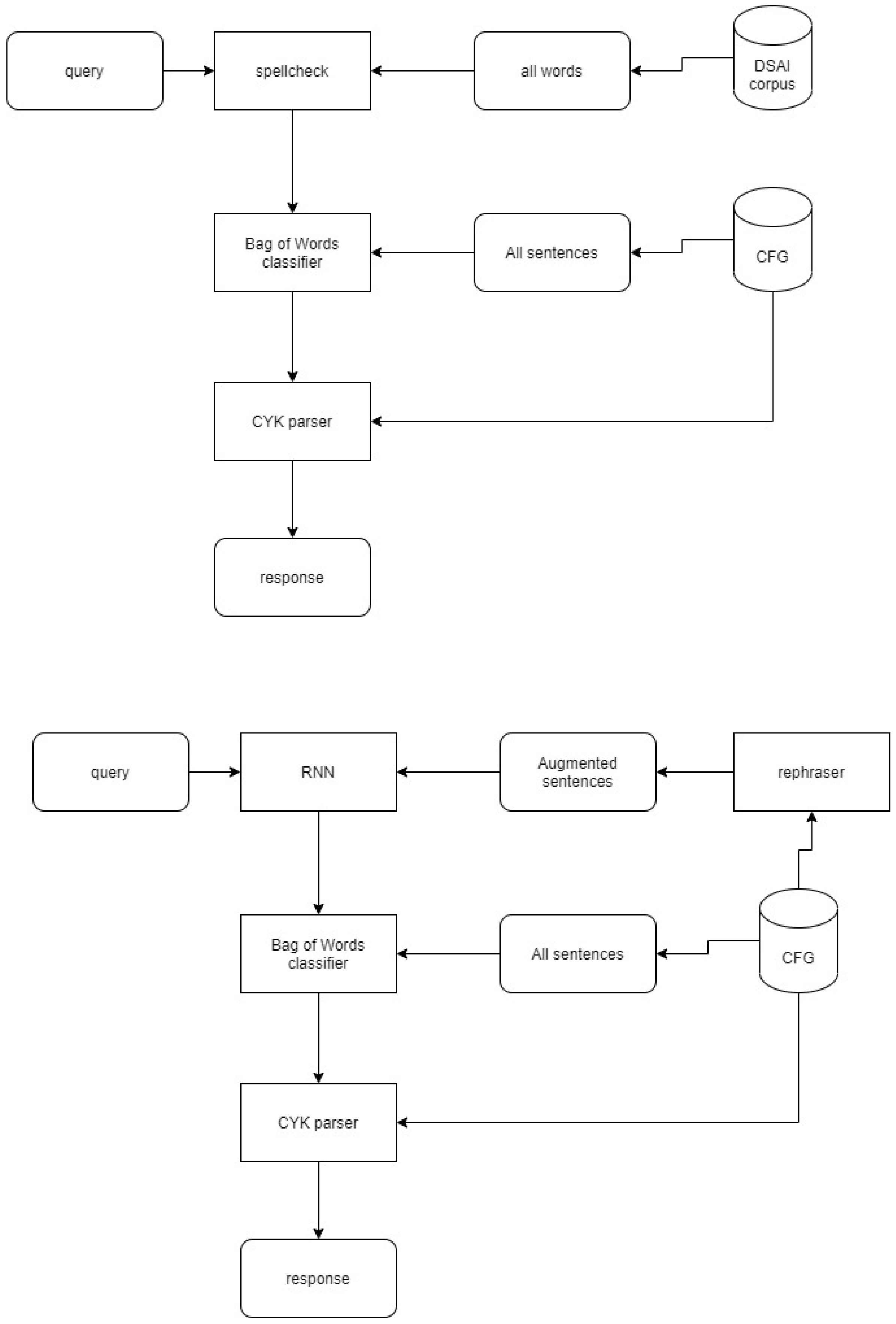

Figure 8: Schema BOW and RNN