Improving NLP Applications with Neural Networks and Genetic Algorithms: A Travel Assistant Chatbot Use Case

Ioannis Gkourtzounis
Department of Computing, The University of Northampton,
Park Campus, Boughton Green Road, NN2 7AL, Northampton, UK
ioannisgk@live.com

***Abstract*—Natural Language Processing applications are increasing rapidly. From chatbots and personal assistants to sophisticated software that analyzes and processes natural language, machines seem to better understand our speech and our texts. Artificial Intelligence offers Machine Learning techniques that have produced great results, and a convergence is inevitable. In this paper we examine how Neural Networks and Genetic Algorithms can improve Natural Language Processing applications. We design, implement and test a Travel Assistant Chatbot application. Throughout the development, we explain in what way we use a Neural Network and a Genetic Algorithm to achieve better text classification results. Finally, we present our findings and discuss about the strengths and weaknesses of our approach. And as Sophia the Robot said, "I can understand speech and meaning behind words [...] but I am still learning a lot".**

***Keywords*—Natural Language Processing, Artificial Intelligence, Machine Learning, Neural Networks, Genetic Algorithms, Chatbots.**

I. Introduction

Today, chatbots and personal assistant applications are increasing rapidly. Natural Language Processing provides software engineers with the techniques needed to develop such applications. Written and spoken language is parsed, words and sentences are analyzed and transformed. Those systems present quite impressive results when Machine Learning methods are incorporated.

Some examples that motivated us to look deeper into the fields of Natural Language Processing and Machine Learning, are personal assistants like Google Now and Siri, machine translation apps like Duolingo, and image classification applications that use deep Neural Networks. Siri can identify unknown callers from emails, while Google Now automatically pulls flight data and offers relevant information. Duolingo predicts word strength and finds sentences that help practicing weak words. Those examples clearly show that text can be broken down to words and their syntactic role and semantic meaning can be recognized. Image classification implementations reveal the power of Machine Learning and how Neural Networks can be trained in order to offer greater efficiency. Innovations like these led our research, providing valuable feedback.

In this paper we examine Natural Language Processing and how Neural Networks and Genetic Algorithms can be used to improve related applications. After exploring the background theory, we focus on a Travel Assistant Chatbot use case. We design, implement and test our application and in the last section we present our findings and discuss about the strengths and weaknesses of our approach.

II. Theoretical Background

A. Natural Language Processing

Speech and language technology relies on formal models, or representations of knowledge of language at the levels of phonology and phonetics, morphology, syntax, semantics, pragmatics and discourse (Jurafsky, 2007; Pustejovsky and Stubbs, 2012). Natural Language Processing (NLP) is a field of linguistics and computer science that focuses on processing natural language. Its main goal is to convert spoken or written human language into formal data that can be further processed by computers. Methods that range from assigning probabilities to words or sequences of words, to full-scale transformation of sentences into new sentences, are used to analyze, modify, augment, and generate human language. Most NLP systems process input via statistical language models that are trained on observations of natural language using Machine Learning techniques. For example, word prediction and word completion models are often developed by collecting large text corpora[1] with millions of words (Higginbotham et al., 2012).

The natural language processes include some standard tasks: Part-Of-Speech tagging (POS), chunking (CHUNK), Named Entity Recognition (NER) and Semantic Role Labeling (SRL) (Collobert et al., 2011). POS tagging labels each word with a unique tag that indicates its syntactic role. Parsing or chunking, labels segments of a sentence with syntactic constituents such as noun or verb phrases. NER labels atomic elements in the sentence into categories, such as "PERSON" or "LOCATION". SRL aims at giving a semantic role to a syntactic constituent of a sentence. Another task is stemming, where words are mapped to some base form, and comes in two

[1] A corpus/corpora is a representative sample of machine-readable texts that have been produced in a natural communicative setting.

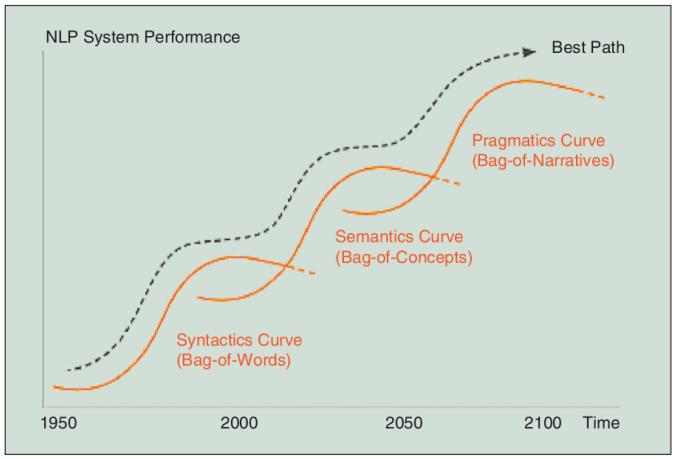

Fig. 1. Evolution of NLP research through three different eras or curves (Cambria and White, 2014).

methods: linguistic/dictionary-based stemming, and Porter-style stemming. The former has higher accuracy, but higher implementation and processing costs, and lower coverage.

The evolution of NLP research can be interpreted as the intersection of three overlapping curves: Syntactics, Semantics, and Pragmatics curve (Fig. 1). While intelligent behaviors emerged like Google Now, Watson from IBM and Siri from Apple, user generated content helped in the performance of those systems. But with the explosion of this content, web trolling and opinion spam (Ritter et al., 2011) caused standard NLP algorithms to be increasingly less efficient. Such systems will stop relying on word-based techniques and will start exploiting semantics in order to make the leap from the Syntactics Curve to the Semantics Curve. Semantics include common sense and common knowledge, reusable by machines. These factors will enable NLP systems to properly deconstruct natural language text into sentiments according to different contexts. The jump from the Semantics Curve to the Pragmatics Curve will happen when computational models become more adaptive, open-domain, context-aware, and intent-driven (Cambria and White, 2014).

NLP has become increasingly powerful and that makes it possible to capture sentiments more accurately, and semantics in a more nuanced way. Naturally, many applications seek improvements by adopting cutting edge NLP techniques, like financial forecasting (Xing, Cambria and Welsch, 2017). Revolutionary applications of speech and language processing are used every day, in areas like information extraction, question answering and summarization, dialogue and conversational agents, and machine translation (Jurafsky, 2007; Pustejovsky and Stubbs, 2012).

B. Neural Networks

Machine Learning is the field of Artificial Intelligence concerned with the development of algorithms which learn or improve their performance from experience or previous encounters with data. There are three major types of Machine Learning algorithms, that are also used in NLP: supervised learning where input data are labeled, unsupervised learning that uses unlabeled data, and semi-supervised learning. As Goldberg (2016) describes, the training networks in Machine Learning resemble the brains computation mechanism and are called Neural Networks (NN). A neuron in NN is a computational unit that has scalar inputs and outputs, and each input has an associated weight. The neuron multiplies each input by its weight, and then sums them. It applies a non-linear function to the result, and passes it to its output. The neurons are connected to each other, forming a network. Such networks were shown to be very capable computational devices. If the weights are set correctly, a neural network with enough neurons and a non-linear activation function can approximate a very wide range of mathematical functions (Goldberg, 2016).

Training a NN is done by trying to minimize a loss function over a training set, using a Gradient Descent[2] method. By repeatedly computing an estimate of the error over the dataset, computing the gradient with respect to the error, and then moving the parameters in the opposite direction of the gradient, the loss is reduced (Gurney, 2014; Goldberg, 2016). One of the most common techniques to reduce the error in multi-layer NN is the Back Propagation, where the weights of each neuron are recalculated. A system based on Machine Learning, improves its knowledge about the dataset, in order to achieve better results in the future. There are many software libraries for Machine Learning and deep NN research available to the public, like TensorFlow by Google Brain Team (Zaccone, 2016).

There are two major types of NN architectures, that can be mixed and matched: feed-forward networks and recurrent/recursive networks. Feed-forward networks include networks with fully connected layers, such as the Multi-Layer Perceptron (MLP), which includes one or more hidden layers (Fig. 2), as well as networks with convolutional and pooling layers (Mehrotra, Mohan and Ranka, 1997). In a feed-forward NN each input of a neuron is attached to each output in the next layer. Convolutional NN use convolutions over the input layer to compute the output. Recurrent/recursive networks accept continuous input over time and the weights are shared along the length of the sequence (Hassan and Mahmood, 2018).

MLPs give great results and can be applied successfully in fields like financial sentiment analysis (Akhtar et al., 2017), medical decision support systems for heart disease diagnosis (Yan et al., 2006) and epilepsy treatment (Orhan, Hekim and Ozer, 2011). Collobert and Weston (2008) implemented a deep NN architecture for NLP, that was trained with huge databases (e.g. 631 million words from Wikipedia) and demonstrated that simultaneously learning tasks can improve the generalization performance[3]. Furthermore, recursive deep learning models can solve multiple language tasks involving word and sentence-level predictions of both continuous and discrete nature (Majewski and Zurada, 2008; Socher, 2014). Various

[2]Gradient Descent is a first-order iterative optimization algorithm for finding the minimum of a function, in this case minimizing the error.

[3]Generalization refers to the ability of an algorithm to be effective across a range of inputs and applications.

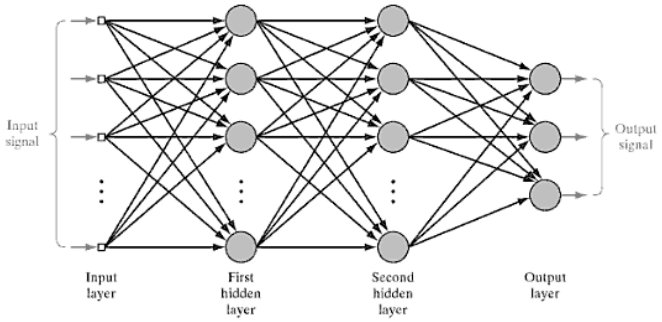

Fig. 2. Architectural graph of a Multi-Layer Perceptron with an input signal, two hidden layers and an output signal.

deep models have become the new state-of-the-art methods for NLP problems, while supervised learning is the most popular practice in recent deep learning research for NLP (Young et al., 2017).

C. Genetic Algorithms

Evolutionary algorithms (EAs) provide a framework for effectively finding solutions to applications where the search space is exponentially proportional to the problem dimensions. The main types of EAs are the Genetic Algorithms (GAs), Evolutionary Strategies (ES) and Evolutionary Programming (EP). They are broadly similar but with some differences, like the selection type and the mutation parameters. EA techniques can easily and broadly be applied to specific problems.

GAs are search methods based on principles of natural selection and genetics. GAs encode the potential solutions of a search problem into finite-length strings (chromosomes) of alphabets (genes) of certain cardinality (Herrera, Lozano and Verdegay, 1998). Chattoe-Brown and Edmonds (2017) present the main flow of GAs as depicted in Fig. 3: (1) we represent potential solutions to the problem as data structures (population), (2) we generate a number of solutions, (3) we evaluate the fitness of each solution, (4) we apply genetic operators (crossover, mutation) to chosen solutions (parents) and create new solutions (children) to evolve the population, (5) we repeat steps 3, and 4 until the current population satisfies a termination condition. Some popular selection methods are roulette-wheel, stochastic universal selection, tournament and truncation. The main types of the crossover operator are one-point, two-point and uniform crossover (Sastry, Goldberg and Kendall, 2013).

One of the most important advantages of GAs is that they are parallel in their nature. Other algorithms explore the solution space in one direction at a time and if the solution is not suitable, they have to abandon all work previously completed and start over. However, since GAs have multiple offspring, they can explore the solution space in multiple directions at once an if they find an unsuitable solution, they can easily continue to work on more promising avenues, giving them a greater chance each run, for finding the optimal solution (Meghna and Jyoti, 2010). On the other hand, GAs have some important limitations, like defining a representation for the problem. The language used to specify candidate solutions must also be robust and tolerable to random changes.

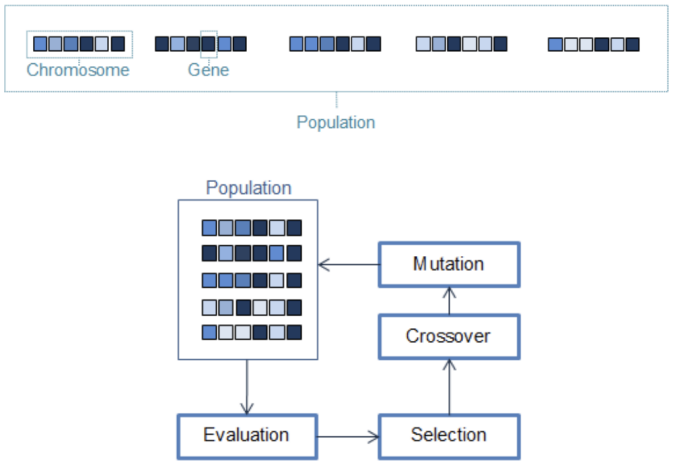

Fig. 3. How population, chromosomes, genes and genetic operators are used in Genetic Algorithms.

As Bungum and Gambck (2010) show in their study, GAs can be used in text summarization to explore a large search space with an optimal combination of a statistical sentence scoring rank. In language learning, context-free grammar strings are evaluated on the ability to generate new strings based on a training set. GAs have been used as a tool for grammar development by inducing a grammar from the training set and then using it to evaluate a test set. Furthermore, these algorithms have been applied to machine translation systems to find the optimal alignment between two sentences. Other NLP related applications based on GAs are document classifiers (Diaz et al., 2018), summarization systems (Peyrard and Eckle-Kohler, 2016), dialogue systems and language generation (Lempa, Ptaszynski and Masui, 2016). Additionally, Manurung, Ritchie and Thompson (2012), with their implementation showed that meaningful poetic text can be achieved with the help of GAs. And if we optimize NN and GAs, by applying a GA for feature or parameter selection, or by optimizing the connection weights, we can have even more impressive results.

III. Chatbot Application Use Case

Chatbots have evolved from semi-intelligent agents, to modern intelligent personal assistants. They take advantage of Artificial Intelligence and Machine Learning, and their knowledge base and conversational properties are comparable to humans (Mittal et al., 2016). The latest NLP techniques made them more effective and human-like (Hirschberg and Manning, 2015; Abdul-Kader and Woods, 2015; Shah, Lahoti and Lavanya, 2017). They use NLP tagging like POS, CHUNK, NER and SRL to fill slots[4], and they are trained with a large enough dataset (corpus) in order to be able to categorize user input (Jurafsky and Martin, 2017).

[4]The set of slots specifies what the system needs to know, and the filler of each slot is constrained to values of a particular semantic type.

One of the most impressive developments in NLP applications and Chatbots is Sophia[5], a social humanoid robot developed by the Hong Kong-based company Hanson Robotics. Sophia tries to simulate human conversation utilizing NLP, Machine Learning, cloud and blockchain technologies.

A. Problem Statement

Focusing on how NLP is used on Chatbot applications, we will apply the techniques discussed earlier on our Travel Assistant Chatbot use case. We assume that a client requests a web application that processes the input text of a user and produces a suitable answer, according to the semantic category that the input is assigned to. There will be 10 categories, such as ”General Travel Support”, ”Booking a Flight Online” and ”Complaining about a Tour”. The system should parse the user input, analyze the text and decide which label/category is the most appropriate. Furthermore, the Chatbot should have the ability to fill ”slots”, so that it can remember the name of the user, the requested destination and the duration of his trip and ask him only the missing information. When all information are gathered, the user will be presented with the details of his trip. The slot filling process will only be valid if the user requests for flight or hotel, and not when he is complaining about a tour (business logic or criteria).

B. Analysis & Design

The requirements for the Travel Assistant Chatbot are as follows: (1) the system should categorize user input, depending on its semantic content, (2) the system should recognize information that can fill predefined slots, like name, destination, duration, etc., (3) the system should ask the missing information, but present booking confirmation only if the user input falls into a specific set of categories. In order to tackle the first requirement, which is essentially a text classification problem, we used a NN in order to train the system on a training dataset, that is assigned to the predefined categories. We also used a GA to determine the optimal settings for training the NN, like the number of hidden layers, the number of neurons for each layer and the number of iterations. For recognizing specific tags on the user input, we performed basic NLP tasks like POS and NER tagging. For the last requirement, we combined all the information and configured the business logic (e.g. when to show the booking details) in our algorithm.

For the development of our web application we chose the Java Spring Framework, a well established application framework fully compatible with the Model-View-Controller (MVC) pattern. The user interface consists of two web pages. On the main page, the user can select specific settings to train the NN. He can also execute the GA, which is responsible for finding the optimal settings for training the NN, and he can also compare the results of the GA and the manually trained network. The chat page will provide the results on the test dataset of the NN, which are the overall probability (the certainty of the system that the test data is assigned to the correct category), and the percentage of the correct predictions.

[5]http://www.hansonrobotics.com/robot/sophia/

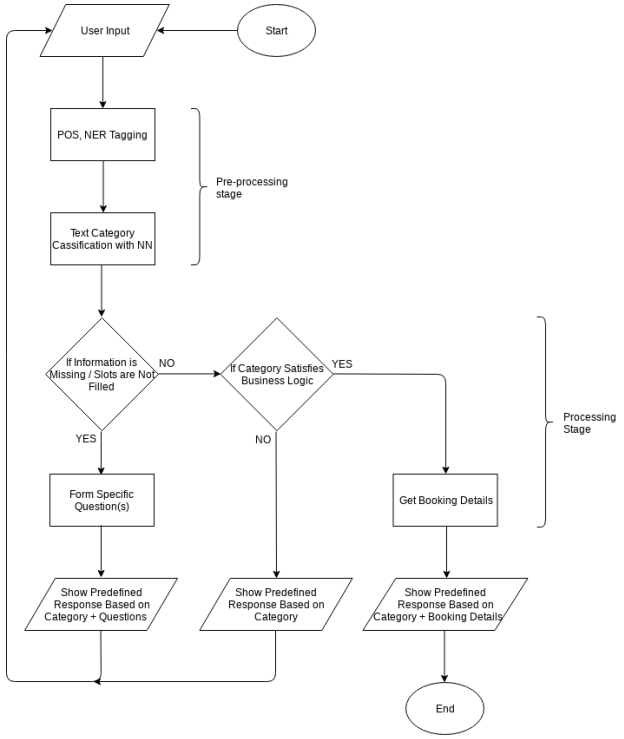

Fig. 4. Travel Assistant Chatbot flowchart with the pre-processing stage and the processing stage, where the system applies the business logic or criteria.

When the user inputs his messages on the chat page, the system will tag the input and save the following information: name, location, date, duration and budget. The NN will also classify the input according to a set of predefined categories. This is the pre-processing stage. In the processing stage, if some information is missing, a question (or more) will be formed to request this missing information. If all slots are filled, then the system will present the booking confirmation only if the user input falls into a specific set of categories, and not if, for example, the user is complaining about a tour. Finally, the Chatbot will respond according to a set of predefined responses, assigned to the input category, and it will also add either the question formed earlier, or the booking details, depending on the business logic (Fig. 4).

In order to develop the application according to Spring MVC standards, we created 4 packages: Controllers, Repositories, Services and Utils. The Controllers package includes the HomeController class which processes, saves data to the model and opens jsp views. It also gets the appropriate POS and NER tags and creates, trains and tests the NN with the use of the appropriate services classes. The DatasetDAO class in the Repositories package contains the hardcoded datasets for training and testing the NN, along with the dialogue responses and the predefined categories. Of course in a production application, all data should have been parsed and saved to a database first. In the Services package the DatasetService gets the training set and test set from the DatasetDAO, and creates the corpus and binary arrays that will be the input for the NN.

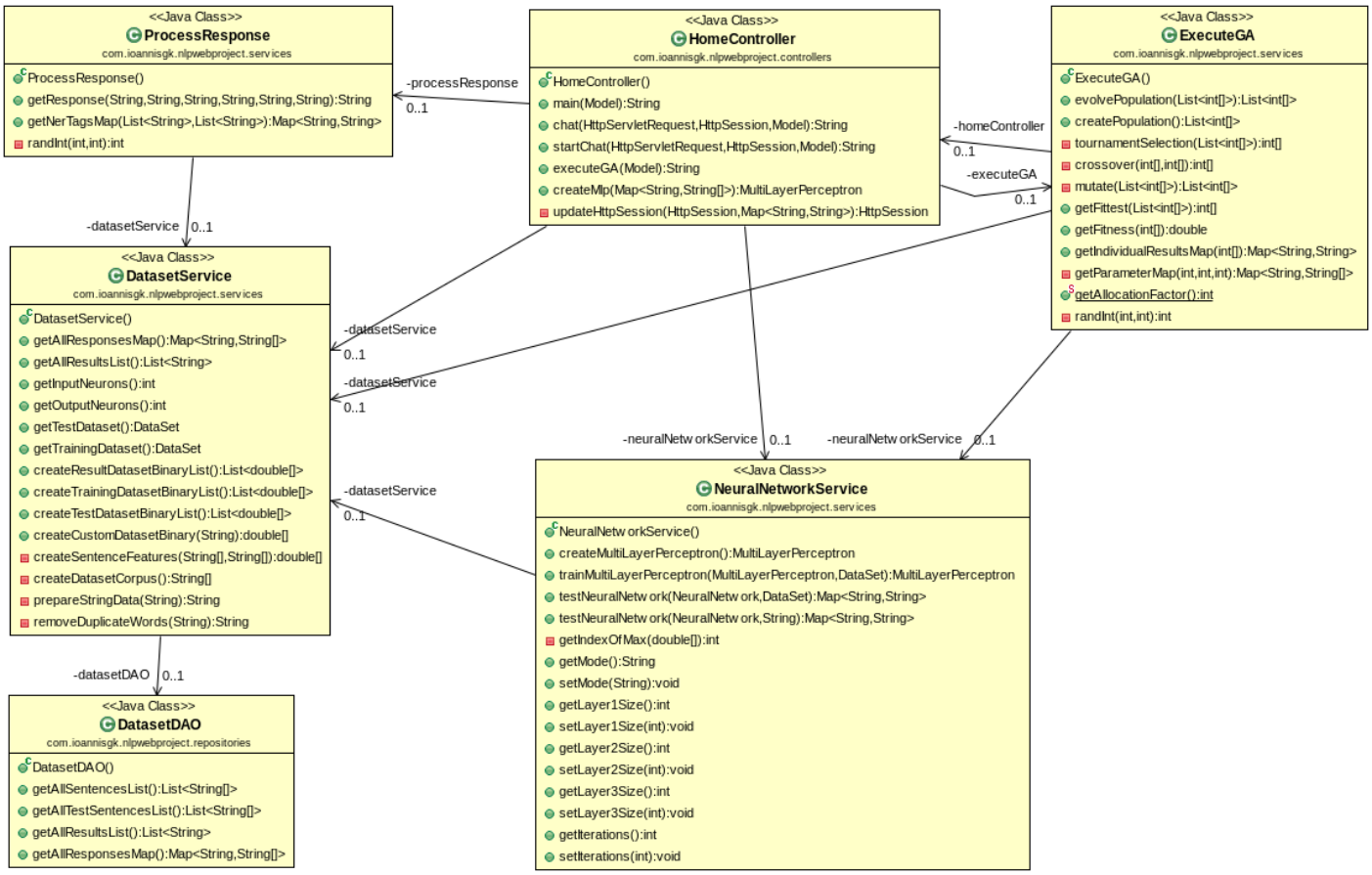

Fig. 5. Travel Assistant Chatbot class diagram.

The NeuralNetwork service creates an MLP network, trains it and tests the test dataset. The ExecuteGA class uses a GA to find the optimal settings for training the MLP, like the number of hidden layers, the number of neurons for each layer and the number of iterations. The ProcessResponse class executes the business logic and creates the response, and the Stemmer class in Utils package is used to perform linguistic stemming on input sentences (Fig. 5). Fully commented source code can be found in Appendix section A.

C. Implementation

One very important implementation requirement is recognizing the words from the user input and the POS and NER tagging tasks for each word. For this purpose we used Stanford CoreNLP (Manning et al., 2014), an integrated suite of NLP tools for Java that includes all standard tasks and has been used numerous times in NLP applications (Cabrio et al., 2012; Chen and He, 2013). In the HomeController class we create a Sentence object, and with the methods of CoreNLP library[6] *words*, *posTags* and *nerTags*, we save the words with NER tags related to person, location, date, duration and budget, into HTTP session variables for later use in the ProcessResponse class.

The NN will be trained and used for classifying the text data. We used Neuroph (Sevarac, 2018), a lightweight Java framework for developing standard types of NN architectures (Dogaru and Dogaru, 2013; Valkov and Zdravkova, 2016). The class NeuralNetworkService uses the *createMultiLayerPerceptron* method to create an MLP, based on the class attributes that configure the number of hidden layers and the number of neurons per layer. We set the Sigmoid[7] as the activation function of each neuron in order to produce an output between 0 and 1. We also set the Back Propagation as the learning rule for the MLP to calculate the gradient, estimate the error and reduce it. The *testNeuralNetwork* overloaded methods are used on a test dataset and on the user message, to assign them to a suitable category. The HomeController class creates the MLP, trains it and saves it as a file for later use, by calling methods from the service and from the Neuroph library[8]. The controller handles the user choices, saves them in the HTTP session and uses them to set the attributes on the NeuralNetworkService class, so the MLP is configured according to the user settings.

[6]https://github.com/stanfordnlp/corenlp

[7]A Sigmoid function is a mathematical function having a characteristic S-shaped curve or sigmoid curve.

[8]https://github.com/neuroph/neuroph

Our conversation dataset with travel related dialogues comes from ESL Fast website[9], which contains many written and recorded conversations as a learning resource, and we found that it fits greatly to the objectives of the Travel Assistant Chatbot. Our dataset consists of 150 sentences assigned to 10 travel themed categories, and we divided them into the training set (67%) and the test set (33%).

The *createDatasetCorpus* method in the DatasetService class gets all the sentences of our initial dataset, removes duplicates and performs stemming in order to create the bag-of-words, a simplified representation model. This is the corpus for our MLP and in order to create the input arrays, we do the following: (1) we remove duplicates and perform stemming on each sentence, (2) we now have a bag-of-words for that sentence, and we check if each word exists in the corpus bag-of-words, (3) if a word exists, the specific index of the word in the input array of the sentence gets the value of 1, if it does not exist, it gets the value of 0. Each input in the MLP is a binary array with a length equal to the number of the words in the corpus. The number of output neurons are 10, as the sum of the categories, and they get a value between 0 and 1. The neuron with the higher value points to the assigned category for the current input sentence. Other functionalities of the DatasetService class include the creation of the training set and test set binary arrays.

The ExecuteGA class is responsible for finding the optimal settings for the MLP. The GA generates random settings, executes the MLP and tests it with the test set. The fitness of each set of settings is calculated as the percentage of the correct predictions against the test set. The set of settings is encoded into a series of 0 and 1. The first 2 characters are the binary equivalents of the number of hidden layers, the next 4 characters represent the number of neurons of each layer and the final 4 characters represent the number of iterations. First, the population is initialized with individuals that contain the encoded 10 character string, with random values of 0 and 1. For each generation, we calculate the fitness of each individual and we apply elitism to keep the best individual. Then, we select parents with the tournament selection method and apply the crossover genetic operation to create children. Mutation comes to add the needed randomness, in order to explore the solution space and find better solutions. After 10 generations, the population evolves and the fittest individual is decoded and gives us the best set of settings for our MLP. We executed the GA for 10 generations and it produced the following optimal settings: Layers: 1, Neurons: 120, Iterations: 96, with an estimated fitness of 74.0% as seen in Fig. 6.

Finally, the *getResponse* method of the ProcessResponse class checks if some information is missing and a question is formed. If all slots are filled and the input message is assigned to a specific category (business logic), the Travel Assistant Chatbot presents the booking details to the user.

D. Testing

We performed black box, white box and experience based testing to evaluate Travel Assistant Chatbot. Our main target was functional testing and we focused on creating test cases specifically designed for NLP tagging, text classification, GA and MLP functionality, and response processing. Test cases were designed to exercise 100% decision coverage and we also performed exploratory testing in order to find hidden weaknesses. Our test cases revealed that the system performed very well if the GA was executed with the following parameters: at least 5 generations, a population of 8 individuals, a mutate rate of 0.2 and elitism set to *true*.

[9]https://www.eslfast.com/robot/topics/travel/travel01.htm

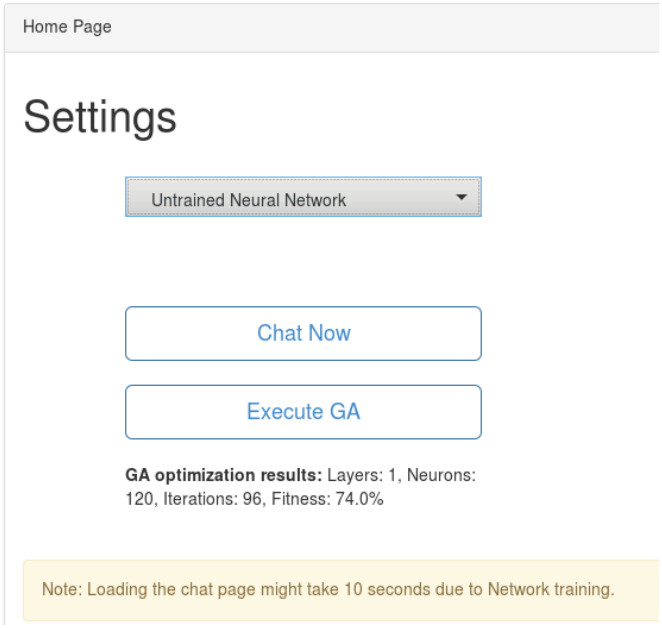

Fig. 6. Optimal settings for training the MLP, after executing the GA: 1 hidden layer with 120 neurons and 96 iterations have a 74% chance of making correct predictions on the test set.

We created, trained and tested the MLP with the optimal settings generated by the GA, but we also tried different number of neurons and iterations for 1, 2 and 3 hidden layers. An untrained MLP gave the worst percentage in correct predictions which was expected, and generally more layers and number of neurons caused an increase in that ratio, but that was not always the case. There were times that a specific combination in MLP settings produced better results. Our tests confirmed that the highest percentage in correct predictions was 74% when applying the settings generated by the GA.

NLP tagging with the Stanford CoreNLP suite performance was excellent and the system successfully recognized all NER tags related to person, location, date, duration and budget. Fig. 7 shows the questions generated and the messages flow. The user says that he wants to travel to ”Sydney, on the 1st of June”. The system classifies that sentence in the ”Buying a Plane Ticket” category and answers with a message assigned on that category ”Would you rather fly in the morning or later in the day?”. This means that the system has recognized some of the required NER tags but some slots are still missing, like the duration and budget, so it adds the appropriate questions ”How long will you be staying? How much money is in

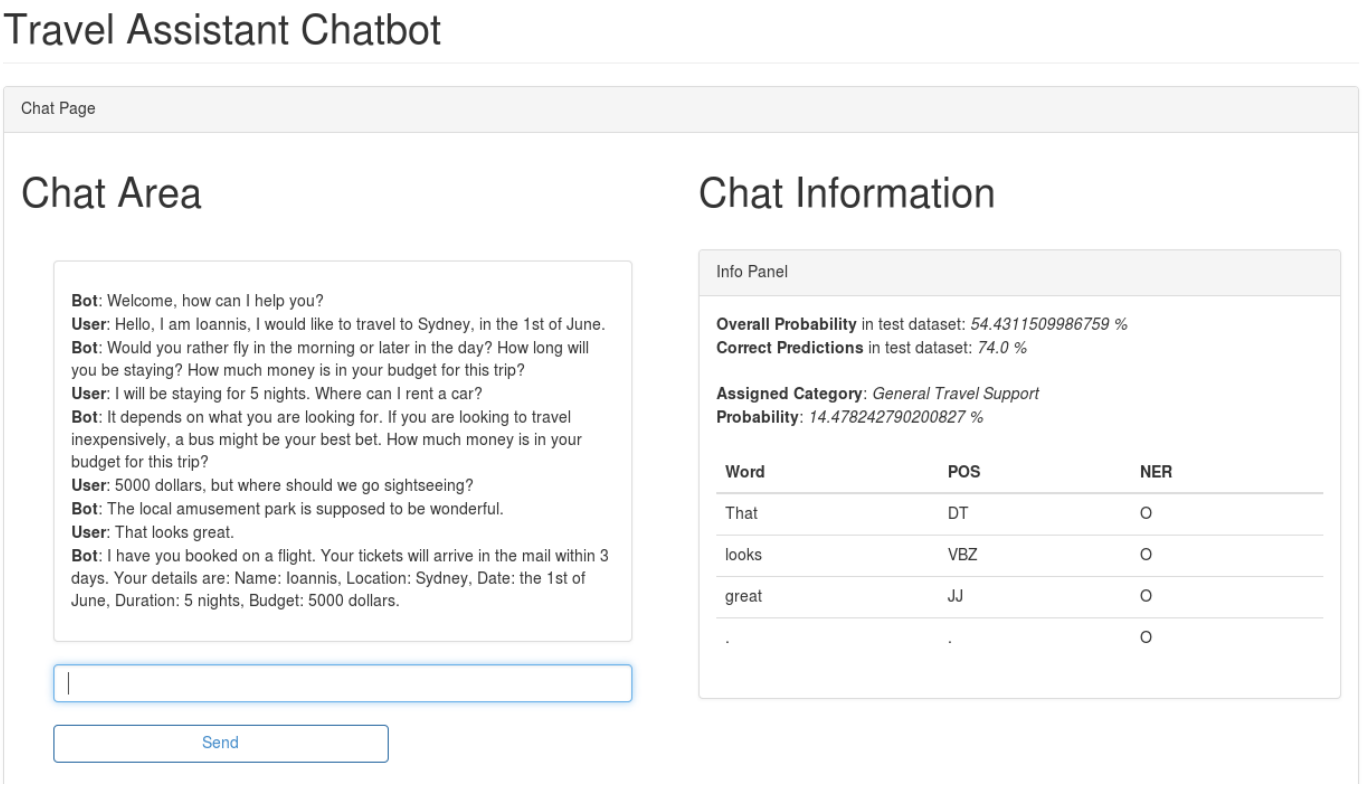

Fig. 7. Travel Assistant Chatbot in action, showing the questions generated and the messages flow. On the right side we can see the overall probability and the correct predictions made on the test set, the assigned category of the current user message and the probability of the current assigned category, as well as the POS and NER tags.

your budget for this trip?". The user enters the duration, the budget is still missing and the system asks about the budget again. Now, the answer from the user is "5000 dollars, but where should we go sightseeing?" which is assigned to the "Sightseeing" category. All slots have been filled, but because of the business logic, the booking details are not presented yet, and a sightseeing related message is displayed. The last message of the user is assigned to "General Travel Support" and now the system can present the booking information recognized during the conversation by the NER tagging task.

Exploratory testing showed that input sentences that were quite similar with the training dataset showed a correct prediction rate of about 60% - 75%. The input of the user is natural language, which itself is a vast search space and text classification can not be thoroughly tested. Whatever method we use, we will not be able to test all possible inputs. The GA has produced good results that indeed optimized the MLP and the text classification procedure, but there is a lot of room for experimenting with different parameters and settings. Acceptance tests by a large amount of beta testers would give a more objective overall evaluation, since the success of the Chatbot depends heavily on the classification results.

IV. Conclusions

NLP applications are increasing every day, making easier for various platforms to process written and spoken natural language. Artificial Intelligence has evolved and now NLP converges with known techniques used in Machine Learning and Evolutionary Algorithms. In this paper we examined how NN can be used to train an NLP system for classification purposes and how a GA can be used to generate the optimal settings to train the NN. GAs can also optimize NN in different ways, by generating new sets of feature selections or by optimizing the connection weights between neurons.

From a theoretical perspective, we saw that NN give great results when properly configured and trained. They enhance the performance of various NLP systems, related to question answering, summarization, dialogue and conversational agents, and machine translation. Deep learning models can solve multiple language tasks, while supervised learning is the most popular practice in recent NLP research (Young et al., 2017).

We analyzed, designed, implemented and tested a Travel Assistant Chatbot web application, where the system recognizes user input based on its semantic content and assigns it to specific categories. The system also recognizes words that provide the required information for slot filling, asks for missing information, and answers with predefined sentences. For the development of the application we used Stanford CoreNLP library for POS and NER tagging, and implemented an MLP by utilizing the Neuroph library. We developed a GA algorithm to provide us the optimal settings for training the

MLP, such as the number of hidden layers, the number of neurons per layer and the number of iterations.

From a practical perspective, implementing an MLP optimized with a GA, gave us the best results on text classification of the user input. Despite the fact that we used a relatively small dataset of 150 sentences and 10 categories for training and testing the MLP, the system performed well when the user input sentences were similar to the training set. A correct prediction rate of about 60% - 75%, is impressive for such a small dataset and just 10 generations of the GA. Fully commented source code can be found on Appendix and on the following link on GitHub: *https://github.com/ioannisgk/nlp-webproject*.

Although the results seem quite promising, we need to note some important weaknesses of our approach. Firstly, a GA procedure requires the training and testing of a NN many times in each stage of the evolution of the populations. This means that we need extended computational resources and time for the GA execution to complete. In our tests, we needed more than 4 hours of processing in a desktop computer with the latest I7 CPU and 16GB of DDR4 RAM, for 10 generations to be evolved, in order for the best individual to get selected and decoded. This time would be reduced if we had a fast server, but we also had to test the MLP and the GA with different parameters and settings and this brings us to the second weakness, which is time for experimentation to find the best settings. Furthermore, testing an NLP application requires results from a lot of users to get a more objective overall evaluation, as the natural language as input, is enormous in terms of a search space.

Concluding, techniques based on NN and GAs can certainly improve NLP applications through optimized NN configuration and deep learning. Indeed, NLP and Machine Learning have already produced great results, like Sophia, and with the rapid technological advances in CPU and GPU processors, the computational costs are reducing, allowing more time for experimentation. Companies can now train NN faster and easier with open source platforms like TensorFlow, and create more effective and human-like chatbots and personal assistants.

REFERENCES

Cambria, E. and White, B. (2014). Jumping NLP Curves: A Review of Natural Language Processing Research [Review Article]. *IEEE Computational Intelligence Magazine*, 9(2), pp.48-57.

Xing, F., Cambria, E. and Welsch, R. (2017). Natural language based financial forecasting: a survey. *Artificial Intelligence Review*.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K. and Kuksa, P. (2011). Natural Language Processing (Almost) from Scratch. *The Journal of Machine Learning Research*, 12(1532-4435), pp.2493-2537.

Higginbotham, D., Lesher, G., Moulton, B. and Roark, B. (2012). The Application of Natural Language Processing to Augmentative and Alternative Communication. *Assistive Technology*, 24(1), pp.14-24.

Jurafsky, D. (2007). *Speech and language processing*. 2nd ed. New York: Prentice Hall.

Pustejovsky, J. and Stubbs, A. (2012). *Natural Language Annotation for Machine Learning*. O'Reilly.

Ritter, A., Clark, S., Etzioni, M. and Etzioni, O. (2011). Named Entity Recognition in Tweets: An Experimental Study. *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pp.15241534.

Goldberg, Y. (2016). A Primer on Neural Network Models for Natural Language Processing. *Journal of Artificial Intelligence Research*, 57, pp.345-420.

Collobert, R. and Weston, J. (2008). A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. *Proceedings of the 25th international conference on Machine learning - ICML '08*.

Hassan, A. and Mahmood, A. (2018). Convolutional Recurrent Deep Learning Model for Sentence Classification. *IEEE Access*, pp.1-1.

Socher, R. (2014). Recursive Deep Learning for Natural Language Processing and Computer Vision.

Young, T., Hazarika, D., Poria, S. and Cambria, E. (2017). Recent Trends in Deep Learning Based Natural Language Processing. *Computing Research Repository*. Zaccone, G. (2016). *Getting Started with TensorFlow*. Birmingham: Packt Publishing.

Gurney, K. (2014). *An Introduction to Neural Networks*. Hoboken: CRC Press.

Mehrotra, K., Mohan, C. and Ranka, S. (1997). *Elements of artificial neural networks*. Cambridge, Mass.: MIT Press.

Majewski, M. and Zurada, J. (2008). Sentence recognition using artificial neural networks. *Knowledge-Based Systems*, 21(7), pp.629-635.

Akhtar, M., Kumar, A., Ghosal, D., Ekbal, A. and Bhattacharyya, P. (2017). A Multilayer Perceptron based Ensemble Technique for Fine-grained Financial Sentiment Analysis. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Yan, H., Jiang, Y., Zheng, J., Peng, C. and Li, Q. (2006). A multilayer perceptron-based medical decision support system for heart disease diagnosis. *Expert Systems with Applications*, 30(2), pp.272-281.

Orhan, U., Hekim, M. and Ozer, M. (2011). EEG signals classification using the K-means clustering and a multilayer perceptron neural network model. *Expert Systems with Applications*, 38(10), pp.13475-13481.

Chattoe-Brown, E. and Edmonds, B. (2017). Evolutionary Mechanisms. *Understanding Complex Systems*, pp.525-566.

Sastry, K., Goldberg, D. and Kendall, G. (2013). Genetic Algorithms. *Search Methodologies*, pp.93-117.

Meghna, N. and Jyoti, K. (2010). Genetic Algorithms and Evolutionary Computation. *IJCSNS International Journal of Computer Science and Network Security*, 10(12).

Herrera, F., Lozano, M. and Verdegay, J. (1998). Tackling Real-Coded Genetic Algorithms: Operators and Tools for Behaviour Analysis. *Artificial Intelligence Review*, 12(4), pp.265-319.

Diaz, A., Rios, A., Barron, J., Guerrero, T. and Elizondo, J. (2018). An automatic document classifier system based on genetic algorithm and taxonomy. *IEEE Access*, pp.1-1.

Peyrard, M. and Eckle-Kohler, E. (2016). A General Optimization Framework for Multi-Document Summarization Using Genetic Algorithms and Swarm Intelligence. *Proceedings of COLING 2016, the*

26th International Conference on Computational Linguistics, pp.247257.

Lempa, P., Ptaszynski, M. and Masui, F. (2016). A Survey on the Use of Genetic Algorithms in Natural Language Processing. *The Association for Natural Language Processing*.

Bungum, L. and Gambck, B. (2010). Evolutionary Algorithms in Natural Language Processing. *Proceedings of the Second Norwegian Artificial Intelligence Symposium*, pp.7-18.

Manurung, R., Ritchie, G. and Thompson, H. (2012). Using genetic algorithms to create meaningful poetic text. *Journal of Experimental Theoretical Artificial Intelligence*, 24(1), pp.43-64.

Mittal, A., Agrawal, A., Chouksey, A., Shriwas, R. and Agrawal, S. (2016). A Comparative Study of Chatbots and Humans. *International Journal of Advanced Research in Computer and Communication Engineering*, 5(3).

Hirschberg, J. and Manning, C. (2015). Advances in natural language processing. *Science*, 349(6245), pp.261-266.

Shah, R., Lahoti, S. and Lavanya, K. (2017). An intelligent chat-bot using natural language processing. *International Journal of Engineering Research*, 6(5), p.281.

Jurafsky, D. and Martin, J. (2017). *Speech and Language Processing*. 3rd ed. New York: Prentice Hall.

Abdul-Kader, S. and Woods, J. (2015). Survey on Chatbot Design Techniques in Speech Conversation Systems. *International Journal of Advanced Computer Science and Applications*, 6(7).

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. and McClosky, D. (2014). The Stanford CoreNLP Natural Language Processing Toolkit. *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp.55-60.

Sevarac, Z. (2018). *Java Neural Network Framework Neuroph*. [online] Neuroph.sourceforge.net. Available at: http://neuroph.sourceforge.net/ [Accessed 19 Mar. 2018].

Valkov, N. and Zdravkova, E. (2016). Implementation of a Neural Network Using Simulator and Petri Nets. *International Journal of Advanced Computer Science and Applications*, 7(1).

Dogaru, I. and Dogaru, R. (2013). JLCNN: An object-oriented Java package for low complexity neural networks. *2013 4th International Symposium on Electrical and Electronics Engineering (ISEEE)*.

Chen, H. and He, B. (2013). Automated Essay Scoring by Maximizing Human-machine Agreement. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp.17411752.

Cabrio, E., Cojan, J., Aprosio, A., Magnini, B., Lavelli, A. and Gandon, F. (2012). What's Cookin'? Interpreting Cooking Videos using Text, Speech and Vision. *Proceedings of the 2012th International Conference on Posters & Demonstrations Track*, 914, pp.9-12.