A Software Agent for Social Networks using Natural Language Processing Techniques

Drazen Draskovic, *Member*, *IEEE*, Vidor Gencel, Slavko Zitnik, *Member*, *IEEE*, Marko Bajec, *Member*, *IEEE*, and Bosko Nikolic, *Member*, *IEEE*

***Abstract* — Machine-learning techniques are widely used in the computer processing of natural language. Software agents are programs that use machine learning and natural language processing to communicate with users and to perform certain tasks or provide specific information. This paper provides an overview of basic software agents and describes the implementation of an intelligent software agent for social network Facebook. Finally, the results of the research propose potential improvements of implemented system.**

***Keywords* — machine learning, natural language processing, social networks**

I. Introduction

Machine learning belongs to a branch of Artificial Intelligence that studies algorithms, which may provide answers to previously unseen questions without explicit programming. Learning is carried out on examples defined in advance, acquired experience or explicit instructions.

Software agents are computer programs that communicate with users and execute certain tasks or provide certain pieces of information by utilizing machine-learning techniques. Software agents are being applied to a large extent in automatization of diverse processes. Agents are characterized by basic properties: adaptability, independence, ability to reach conclusions, infer and react.

Natural language processing (NLP) is a process whereby computers analyze, understand and infer meaning from the text written in the natural language. Programmers can organize the data and acquired knowledge by using these techniques thereby enabling the creation of a wide range of applications such as: translators, text entity recognition systems, paper reference recognizers, message classification. This paper focuses particularly on part of speech tagging and time adverb detection [1].

The recommended communication flow between a user and an agent would be the following:

1) Utilizing certain interface, a user sends a message to an agent.
2) The agent receives the message and forwards it into the NLP module.
3) After the initial message has been processed, it is sent to the control part where the fulfillment of certain task required by the user is tried to be achieved.
4) The agent responds to the user message by means of an adequate answer.

Since nowadays, the number of social network users is greater than 2 billion and these networks are a good starting point to end-users. The interface through which users communicate with an agent is particularly important. It has to be accessible for different device types and also intuitive for a developer. The Facebook Messenger platform has been selected for the purpose of this paper with the aim of developing a personal assistant that enables scheduling a reminder. Subsequently, an assistant should perform a task and at a given point remind a user about the task inserted at scheduling a reminder.

Section II analyzes machine-learning issues, represents the basic software agent classification and puts forward concrete examples. Ideas and functionalities carried out within this system have been described in Section III. Certain implementation detail has been provided in Section IV. The last part summarizes the research and software agent operation, and also proposes the plausible enhancement.

II. Problem Analysis

Machine learning has facilitated the following issues: classification, regression, clusterization and rule inference. A problem of text message classification has been also dealt with in this paper. Message classification comprises two steps: 1) extracting characteristics from messages and 2) characteristic-based classification.

A. Extracting characteristics from messages

If we consider one sentence as input data, then characteristics must be extracted from the sentence. Characteristics should have a certain structure in order to represent valid classifier entry. This aspect of linguistic analysis has utilized the n-gram model. If the given sentence S is regarded as a set of characters $s_1s_2...s_n$ n-gram

This research was conducted within the bilateral project "Open extraction of information for Slovenian and Serbian language" between the School of Electrical Engineering, University of Belgrade, Serbia, and the Faculty for Computer and Information Science, University of Ljubljana, Slovenia.

Drazen D. Draskovic (corresponding author), Vidor A. Gencel and Bosko D. Nikolic are with the School of Electrical Engineering, University of Belgrade, Bul. kralja Aleksandra 73, 11120 Belgrade, Serbia (e-mails: drazen.draskovic@etf.bg.ac.rs, gv163235m@student.etf.bg.ac.rs, bosko.nikolic@etf.bg.ac.rs).

Slavko Zitnik and Marko Bajec, are now with the Faculty for Computer and Information Science, University of Ljubljana, Volgina 15, SI-1000 Ljubljana, Slovenia (e-mails: {slavko.zitnik, marko.bajec}@fri.uni-lj.si).

sequences of a sentence S are a set of all subsequences of adjacent distance symbol n.

B. *Characteristic-based classification*

The aim of statistical classification is to identify the class to which the object belongs by means of object characteristics. There are many types of classifiers based on diverse models and algorithms. This paper will observe maximum entropy classifier (MaxEnt) as the most convenient one for textual message classification.

The MaxEnt classifier belongs to the exponential model class. Contrary to "Naïve Bayes" classifiers, the MaxEnt does not presuppose the mutually interdependence of characteristics. It is based on the maximum entropy principle and it selects the ones with the highest entropy from all models pertaining to a training set. The MaxEnt classifier is utilized for solving wide spectrum of text classification problems [2] such as: language detection, general classification, sense analysis, etc.

Due to its property to make minimal assumptions, the MaxEnt classifier is used when nothing is known about characteristics distribution and reliable assumption cannot be inferred. In addition to this, the MaxEnt classifier is utilized when conditional characteristic independence cannot be foreseen. It takes much more time for training in case of the MaxEnt classifier than in the case of "Naïve Bayes" classifier due to a problem of optimizing the way of solving certain equations so as to obtain model parameters in a more efficient manner. The Stanford NLP library has been used because of good classifier performances, as well as possibility of executing diverse classification optimizations [3].

C. *Part-of-Speech Tagging in NLP*

Part-of-speech tagging is a process whereby each word within the text is assigned a certain part of speech. In the course of word tagging it is necessary to take the definition and context in which the word is found into consideration. A problem in part of speech tagging occurs when one word can be assigned several different parts of speech. In that case, various methods for solving conflicts are implemented.

The part of speech marker is a computer program that deals with these issues. The markers utilize several knowledge sources: dictionaries, lexicons, rules, etc. Dictionaries contain categories of each word and if a word cannot be places in several categories the markers use the probability-based methods in order to resolve a conflict.

There are two categories of taggers: 1) grammar-based taggers, which rely on dictionaries and lexicons in order to determine the possible word categories; 2) stochastic taggers based on the hidden Markov models, which have tendency to maximize the likelihood of words belonging to certain parts of speech based on the whole overview.

Part-of-speech tagging process consists of the following parts: tokenization, conflict search and conflict solving.

When parts of sentence are marked, it is easy to extract necessary information by means of the formal language. In this research, the Stanford NLP library [4] was used for part-of-speech tagging. The library was designed as a pipeline. Each pipeline degree contains an annotator that adds annotation for the input data. The following annotators were used: the tokenization device (which divides the input text into tokens), annotator that groups the input tokes into sentences, and annotator what tags parts of speech. At the pipeline end, it is possible to see parts of speech for each input token and thereby carry out certain transformations on the input text. Parts of a sentence as well as certain constructions can be extracted by means of special regular expressions.

D. *Time adverbs tagging*

Time adverbs tagging is a process whereby phrases with temporal meaning within the context of a larger document is sought. Temporal normalization is a process of textual phrase mapping that describes the time, date and duration within a given temporal structure (e.g., java.util.Date). Development techniques can be: linguistic, grammar- and rule-based, statistical, knowledge-based on already annotated data training [5].

In this research, the SUTime and Natty libraries have been explored. The SUTime [4] is a library for recognition and normalization of temporal expressions. It is available within the Stanford NLP pipeline, as a separate annotator and can be used for tagging documents with temporal information. An advantage of this library lies in its ability to easily integrate in addition to the existing Stanford NLP library, and a disadvantage is that it returns the temporal information in the TIMEX3 format.

Natty [7] is a date parser from texts written in natural languages. Natty applies standard techniques of language recognition and translation of a given text containing temporal expression thereby yielding a date list from a given text. In the background, the Natty uses the ANTLR parser generator for tokenization and parsing the input data into abstract syntactic tree, which afterward is referred to in search of temporal references. Conflicts arising in polysemous temporal references are resolved statistically. The supported time adverbs include: formal (2017-04-22) and relaxed dates (e.g. jan 1st, new years eve), relative dates (e.g. 3 days from now, next week), prefixes (e.g. 2 fridays before), exact time in different formats (e.g. 5, 6am, 18:00), relative time (e.g. in 5 minutes).

III. System Operation Description

The fulfilled intelligent agent has the following functionalities: scheduling a reminder, reminding a user, display of all reminders, deleting reminders, and replying to generic messages.

A. *Scheduling a reminder*

Each reminder has two items: reminder topic and reminder time. Reminder topic is a piece of information about the reminder that the user wants to be informed about. Reminder time can be defined in different ways, as was described in the previous sections. In order to define the message input, lexical analysis of tasking a reminder ought to be taken into consideration. Two imperative sentences will be analyzed:

- Remind me to walk my dog tomorrow.
- Wake me up tomorrow.

Immediately, it is noticed that there are two basic types of scheduling a reminder, one being the instance when a topic is being explicitly defined in the sentence, the other wherein the topic is inherited from the imperative of a verb. Also, reminders with incomplete information are possible. The existence of various classed can be determined by analyzing the following sentences:

- Remind me to walk my dog tomorrow.
- Remind me tomorrow.
- Remind me to walk my dog.
- Remind me.

We conclude that there are following message classes: messages with complete information, which contain time and reminder topic, and messages with incomplete information. Messages with incomplete information can contain only reminder time, or only reminder topic, or ought not to contain any information except the intention of creating a reminder. Furthermore, we will analyze the sentences in which the reminder topic has been inherited from the imperative of a verb:

- Wake me up.
- Wake me up tomorrow morning.

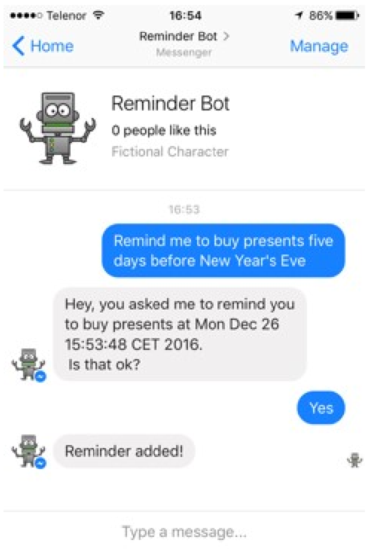
We have concluded that there are the following classes of messages with the imperative: messages with complete information, which contain the imperatives of verbs from which the reminder topic follows, but do not contain reminder time. The analysis can be shown by simple state automata, where the state transitions are caused by receiving the message of certain content, and states denote the data being defined while creating the temporary reminder. It was necessary to conduct this analysis in order to create a message set which would train the classifier thereby creating and enabling the appropriate state automata assigned to each user. Due to plausible error in time parsing, additional step has been introduced which entails that having collected data, the agent asks a user if the data are correct.

When a user sends a message to an agent with a scheduled reminder, the agent will first parse the message, determine a reminder topic, time and ask the user for final confirmation. The user can confirm or reject the reminder's suggestion by selecting the button "Yes" or "No", or by manually entering the message expressing satisfaction or dissatisfaction concerning the required data that the agent has read from the primary message. In case of confirming the reminder accuracy on the part of the ser, the reminder is being added the base, and the user is being informed about that, as is shown in Fig. 1.

If the user tried to schedule a reminder with incomplete data, the agent detects this and asks the user for time and topic of a reminder. The agent retains every client response connected with reminder time or topic in the database. After entering necessary information, the agent summarizes all data in a message and asks the user if he/she would like to add such a reminder.

B. *Reminding the user*

When the time has come to remind a user, an agent sends a message with a reminder topic. Here, attention should be paid to the person of a pronoun in the reminder topic. For example, in the assigned reminder "Remind me to walk my dog tomorrow", the agent reminder should sound "Hey, you asked me to remind you to walk your dog". The analysis for this problem has been carried out only for the English language, because this is somewhat more complex in Serbian and Slovenian, because there is a case in addition to the person. We have noticed that every pronoun in the first person singular or the first person plural should be substituted by an appropriate pronoun of the second person singular or plural. Even though this is not evident at first, pronouns in the English language are the closed word class, which implies that all pronouns belong to one finite set, which is never expanded.



Remind me to buy presents five days before New Year's Eve

Hey, you asked me to remind you to buy presents at Mon Dec 26 15:53:48 CET 2016.
Is that ok?

Yes

Reminder added!

Fig. 1. An example of scheduling the reminder with full information, where a topic has been explicitly defined in the message

C. *Displaying all reminders*

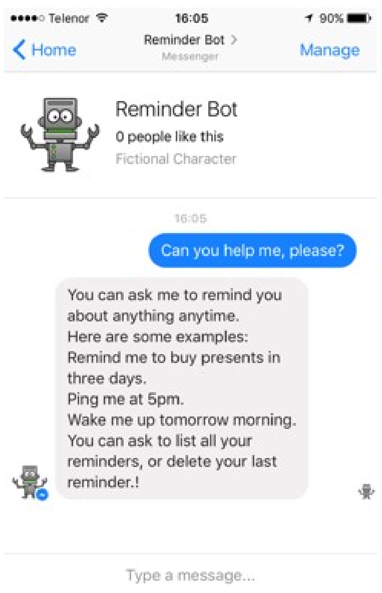
The agent must respond to the message by which a user requires displaying all current reminders should in the appropriate way. In addition to solely text messages by which the required information can be displayed to the user, the Messenger application supports different message types. Each reminder will be displayed to the user in a horizontal sliding dialogue, wherein at one time point only one reminder will be displayed.

D. *Deleting reminders*

The agent should react to a user's message which demands that it should be deleted from the reminder list and accordingly delete such a reminder. Deleting the last entered reminder has been introduced as a separate type of deleting. A dialogue has been introduced as an additional deleting step within which the user must confirm it if he is sure that he wants to delete the reminder.

E. *Responding to generic messages*

Generic messages have been divided into two categories: searching for help and greetings. In the course of searching for help, the agent sends a message with its capabilities, i.e. ways of usage to the user. When the agent receives a wrong type message, it will warn the user that this is not a message type, which was expected in such a state. An example of message for whose content the user asks for help in application usage is shown in Fig. 2. The agent responds to this message with the possible reminder formats, as well as possibilities of listing and deleting reminders.



Can you help me, please?

You can ask me to remind you about anything anytime.
Here are some examples:
Remind me to buy presents in three days.
Ping me at 5pm.
Wake me up tomorrow morning.
You can ask to list all your reminders, or delete your last reminder.!

Fig. 2. An example of the answer to a message with searching for help

When the user sends the message containing "Hello bot", he will receive a message containing greeting with naming the user who has sent the message. Upon the receiving the message, the agent has the possibility of receiving the following user data through Messenger API: name, surname, user profile photo address, user application set language, time zone and gender.

IV. Software Agent Implementation

A. Database design

The MongoDB was used in system implementation. The MongoDB is an open code database that stores the data such as JSON dynamic scheme documents. The MongoDB is one of representatives of NoSQL non-relational databases. Collections are used instead of tables, documents are used instead of table rows, fields are used instead of columns. The database consists of merely two collections, State and Reminders, the description of which has been provided in Table 1.

Table 1: The appearance of collection 'reminders'

Field	***Description***
_id	Internal Identifier (MongoDB)
user_id	User Identifier (Facebook API)
type	Type of reminder that is being scheduled
date	Reminder Time (Unix timestamp)
task	Reminder Topic
state	Reminder State

B. Communication with message platform

Communication with message platform is reduced to receiving and sending messages. Receiving messages is carried out in such a way that the platform for messages sends an HTTP request to the corresponding URL agent. In its body, the request contains all data on the received message in JSON format. Due to this, it is necessary for an agent program to contain a web server that can receive HTTP requests, as well as the deserialization library, which can convert JSON into Java objects. Spark was utilized as a web server agent in this research, while Google Gson library was used for deserialization.

Types of received messages that are distinguished are the following ones: text message, quick answer and feedback message. Two class types were utilized for representing messages in our system: contract classes whose each field is mapped into one field in the JSON body of the received message, and model classes, which are used in program code to render the whole specialization and deserialization logic abstract.

An additional problem in using the message platform is the problem of limiting the URL agent, because it is possible to enter one URL that must contain an SSL certificate. Since it would be unacceptable to deliver a new code on a server after every change, a small proxy has been made. The message platform sends messages to proxy and then they are being forwarded to all entered addresses.

V. Conclusion

A realization of the intelligent software agent has been described in this paper. The research comprised the analysis of existing libraries, modeling, implementation and testing the completed agent. This system implementation phase has been completed only for the agent for the English language, but it is also expected to complete the agents for Serbian and Slovenian in next phases. The application of agent for different localizations would imply deep lexical analysis of each language and application of the corpus of the respective language.

The classification would be considerably more precise, if the neural network would be used instead of static classifier. Also, it is possible to replace the static classifier with a dynamic classifier. This could be completed by storing all user messages and dynamic participation in classifier characteristic transformation.

References

[1] E. Cambria, B. White, "Jumping NLP Curves: A Review of Natural Language Processing Research," IEEE Computational Intelligence Magazine, vol. 9, issue 2, May 2014, pp. 48-57.
[2] M. Kanakaraj, R.M.Guddeti, "NLP based sentiment analysis on Twitter data using ensemble classifiers," 3rd International Conference on Signal Processing, Communication and Networking, IEEE, Chennai, India, March 2015.
[3] M. Marneffe, C. Manning, "The Stanford typed dependencies representation," Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation, Pages 1-8, 2008.
[4] The Stanford NLP group, http://nlp.stanford.edu/software/, accessed on October 9th, 2016
[5] P. Liang, "Learning executable semantic parsers for natural language understanding," Magazine Communications of the ACM, vol. 59, issue 9, Sept. 2016, pp. 68-76.
[6] Natty, http://natty.joestelmach.com, accessed on Sept. 15th, 2016