Intent Classification for a Management Conversational Assistant

Abdelrahman H. Hefny
Faculty of Computers & Information Technology
National Egyptian E-Learning University
Giza, Egypt
ahefny@eelu.edu.eg

Georgios A. Dafoulas
Computer Science Department
Faculty of Science and Technology
Middlesex University
London,United Kingdom
G.Dafoulas@mdx.ac.uk

Manal A. Ismail
Computer Engineering & Systems Department
Faculty of Engineering
Helwan University
Cairo, Egypt
mismaeel@eelu.edu.eg

***Abstract*—Intent classification is an essential step in processing user input to a conversational assistant. This work investigates techniques of intent classification of chat messages used for communication among software development teams with the aim of building an intent classifier for a management conversational assistant integrated into modern communication platforms used by developers. Experiments conducted using rule-based and common ML techniques have shown that careful choice of classification features has a significant impact on performance, and the best performing model was able to obtain a classification accuracy of 72%. A set of techniques for extracting useful features for text classification in the software engineering domain was also implemented and tested.**

Keywords—intent classification, chatbot, conversational assistant, natural language understanding, dialog act recognition

I. INTRODUCTION

Classifying the intent of user-generated text is essential for many information system applications such as search engines, e-commerce, question answering, and conversational agents [1]. Due to recent advances in artificial intelligence, conversational assistants are gaining attention as a way to increase productivity by automating complex processes that usually require human intervention [2]. Interaction with such assistants is mainly done using natural language via either speech commands or written text. Intent detection and classification, sometimes referred to as semantic utterance classification, is one of the basic steps of processing a user's utterance in a task-oriented conversational agent. During this step, a user utterance is classified to determine its intent from a list of predefined intents and redirect the conversation according to a specific scenario based on the detected intent. This computational task is often implemented as part of the natural language understanding (NLU) module in the conversational agent. A similar task, dialog/speech act recognition, is performed in non-task-oriented dialog systems, which classifies the dialog act (DA) of the user's utterance as a first step of processing [3, 4, 5, 6].

This work investigates techniques of intent classification of text chat messages used for communication among software development teams. The purpose of this work is to build an intent classifier for a management conversational assistant (chatbot) integrated into modern communication platforms used by software developers. The proposed chatbot will interact with developers via private channels and will also monitor and react to messages on multi-party chat rooms. Such chatbot would assist team management in various ways such as answering frequently asked questions or monitoring a task progress. This research investigates two main approaches for intent classification, a rule-based approach based on hand-crafted rules, and a statistical approach using well-known machine-learning (ML) algorithms.

Previous work in this area studied techniques for intent classification and dialog act recognition in several domains. Some recent studies investigated the use of conversational assistants in the field of software engineering. Applications include recommending experts [7], automating low-level workflows [8], resolving conflicts [9], and answering questions based on code repositories [10].

The major contributions of this work are:

1) A proposed list of intent categories for classifying software developers chat messages from a management perspective.
2) A set of techniques for extracting useful features for text classification in the software engineering domain.
3) Investigation of several text classification techniques for classifying intents of software developers' chat. All implemented code is also provided for reference via the project's website.[1]

The remainder of this paper is structured as follows. Section II provides a review on related work. Section III explains the research method and design. Section IV discusses experiments and results. Finally, section V presents the main conclusions.

II. RELATED WORK

The following sections describe related work in three main categories: A) intent classification techniques, B) applications of text classification in the software engineering domain, and C) conversational assistants in the software engineering domain.

[1] https://github.com/abdelrahman0101/MCA

A. Intent Classification Techniques

Almost all common text classification techniques were used in dialog act recognition or intent classification in conversational agents. These techniques vary from simple pattern matching with hand-crafted rules to advanced techniques that use deep neural networks.

Rule-based pattern matching techniques were traditionally used for utterance classification in chat-bot and question answering systems. Examples of chatbots with rule-based utterance classification include AutoTutor [11], a domain-portable intelligent tutoring system, and DBpedia chatbot [12], a chatbot designed to enhance interactions in the DBpedia community.

Another common approach for intent classification is the use of supervised ML techniques. This approach alleviates the need for writing and finetuning hand-crafted rules by training a statistical ML algorithm on a manually labeled set of data. Examples of ML algorithms used for this task are Support Vector Machine (SVM), Maximum Entropy (ME), and Naive Bayes (NB) [13, 14]. Recent research on intent classification made use of new advances in deep learning techniques [15, 16, 17, 18]. Deep neural networks can achieve higher performance than other statistical ML algorithms but usually require a much larger dataset.

B. Text Classification in the Software Engineering Domain:

Text classification has had several applications in the software engineering domain. Maalej, et al. [19] studied the use of simple string matching, Naive Bayes, Decision Trees, and Maximum Entropy for the classification of app reviews into bug reports, feature requests, user experiences, or ratings. Classification features included metadata such as star rating, length, verb tense, and sentiments in addition to text features.

Arya, et al. [20] studied the classification of issue discussions of open source software based on information type. Using qualitative content analysis, they identified 16 information types in issue discussions. Two supervised classifiers that use Logistic Regression and Random Forest were trained on a labeled dataset with textual and conversational features such as sentence location and length and participant role.

Wood, et al. [21] conducted a “Wizard of Oz” experiment to detect dialog act types in developer question/answer conversations during debugging. It involved 30 professional developers fixing bugs for 2 hours while using a simulated virtual assistant for help. The conversations were, then, annotated using open coding to identify 26 dialog act types. After solving inter-annotator disagreements, a Logistic Regression classifier was trained on the labelled data represented using binary Bag-of-Words with three shallow features; normalized length, words count, and number of seconds since previous message.

Classification of online discussions about software applications and services was studied by Ramirez, et al. [22]. They investigated the classification of text comments into enhancement requests vs other types of requests based on speech-act analytics. A dataset of sentences extracted from online discussions was first annotated with 20 speech act tags using a set of lexico-syntactic rules. The speech act tags are then used as a classification feature for classifying sentences into either enhancement requests or not, using three ML algorithms; J48, Sequential Minimal Optimization (SMO), and Random Forest.

C. Conversational Assistants in Software Engineering:

Several studies presented chatbots intended for supporting software engineers in their work. Cerezo et al. [7] designed a chatbot for recommending experts in the Pharo programming language community. The chatbot classifies a user message into one of 7 categories based on calculated term frequency, then it identifies names of source code artifacts (key concepts) and uses an expert recommender system to recommend an expert in the specified artifact.

Instead of manually building intent classifiers, some other studies made use of cloud-based chatbot services such as Google's DialogFlow, Amazon's Alexa, IBM's Watson Assistant, or Microsoft's LUIS [23]. These services allow designers to build a chatbot solution by providing example utterances and setting rules for triggering responses without having to manually code it in a programming language; thus, they provide an excellent choice for rapid prototyping and for non-programmers. Nonetheless, manually built solutions can, sometimes, outperform these black-box services in intent classification tasks [24].

Bradley et al. [8] used Amazon Alexa services to build a prototype for Devy, a conversational developer assistant that listens to developer's voice commands, infers her high-level intent, prompts her for any additional information, and automatically invokes a low-level actions workflow with the help of an automatically generated context model. Another work that uses a cloud-based service is Sayme by Paikari et al. [9]. Its goal is to detect and resolve conflicts between developers while working on the same project on collaboration platforms such as GitHub. They used DialogFlow and trained it on 28 to 45 phrases per intent. The chatbot also uses a Python backend with a MySQL DB to store needed data. DialogFlow was also used by Abdellatif et al. [10] to implement MSRBot, a chatbot to answer questions of software developers based on information extracted from software repositories. Questions shown are mostly related to code commits, bug tickets, and developer’s responsibility. While Devy, Sayme and MSRBot focus on helping developers perform their tasks by automating their low-level workflows, resolving conflicts, or answering questions based on code repositories, this work is focused on building a chatbot for tasks related to project management. In addition, the proposed chatbot will not only interact with developers via private channels but will also monitor and react to messages on multi-party chat rooms.

III. Proposed Classification Models

This research investigates two main approaches for intent classification; a rule-based approach based on hand-crafted rules, and a statistical ML approach. The proposed intent categories are based on analysis of an archived dataset of chat

messages. The following sections describe the dataset analysis and the classification techniques.

A. Dataset Analysis and Annotation

A dataset of archived chat messages was analyzed to determine the possible intent classes, and to specify the set of rules used in rule-based classification. This dataset was collected from a text chat platform used by multi-national university students during their work on course projects mostly related to software engineering.

Based on the preliminary analyses, the dataset was annotated using an open-coding methodology to indicate various topics and dialog acts. The result is the following list of 14 chat topics:

1. Greetings
2. Informal/off-topic talk.
3. Meeting schedule.
4. Individual expertise.
5. Platform problems.
6. Project/Task Requirements.
7. Project/Task schedule.
8. Task assignment.
9. Task implementation tools.
10. Task implementation details.
11. Task status or progress.
12. Feedback on submitted work.
13. General planning.
14. Coordination and organization of work.

Within each of these topics, chat messages may have different dialog acts. A simplified list of relevant dialog acts is used:

1. Information.
2. Suggestion.
3. Request.
4. Question.
5. Positive reply (agreement).
6. Negative reply (disagreement).
7. Partially positive/negative reply.

This work focuses on detecting the intent of a chat message to enable a conversational agent to participate in a group chat as an assistant to team management. For this purpose, a message intent is categorized into one of 13 categories by combining main topics and dialog acts. TABLE I describes each category.

In order to test the accuracy of the classification techniques and to train a supervised machine learning classifier, a labeled dataset is needed. Chat messages in the dataset were labeled to indicate the intent of each message based on the results of preliminary analysis and open coding. However, since careful labeling of the dataset requires much time and effort, it was applied on a smaller subset with only 8030 chat messages. In addition, the length of chat messages vary as some messages may contain several sentences with different indications and some sentences may span several chat messages. Those messages were manually split or joined so that most datapoints are meaningful and have single intent labels. During the labelling process, chat messages were also reviewed to correct basic spelling and grammatical errors resulting from low English language proficiency of some participants. A list of words that represent named entities such as persons or software tools were also collected and stored in database.

The number of messages in each category vary significantly with the largest class being “Others”. This is because many chat messages are simple short phrases that carry no specific intent related to the software engineering domain such as simple “yes”, “no”, or “ok” answers. Such messages are labeled with just a dialog act tag and the intent “Others”. Finally, since the performance of most ML classification algorithms is affected by the balance of the dataset, the dataset was augmented by a small set of 210 samples manually written by researchers to support low-frequency classes. The final frequency of each intent category is shown in Fig. 1.

TABLE I. INTENT CATEGORIES

#	Intent	Description
1	Greet	Greeting and introduction. Primarily used when opening a conversation.
2	Plan task	Task assignment and scheduling, and general work planning.
3	Query plan	Questions and requests for information on task plan.
4	Schedule meeting	Discussions of next meetings schedule.
5	Report progress	Report the status and progress of a specific task.
6	Query progress	Questions and requests for information on task status and progress.
7	Request feedback	Requests for feedback on completed work or proposed ideas.
8	Give feedback	Giving feedback on completed work or proposed ideas.
9	Discuss task	Discussions of technical details.
10	State rules	Information on work rules, such as the use of tools, file formats, and task submission.
11	Query rules	Questions and requests for information on work rules.
12	Report issues	Reporting problems and work blockers.
13	Others	Other or unclear intents.

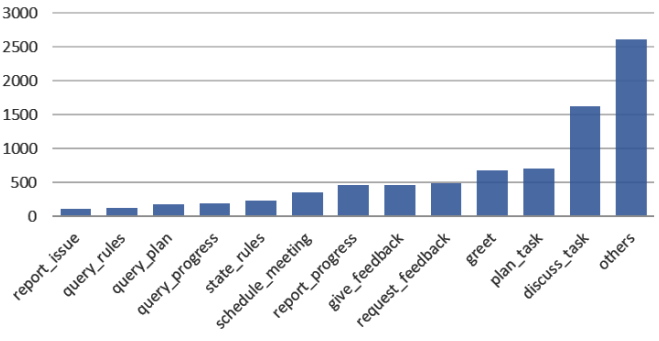

Fig. 1. Number of Messages in Each Intent Category

B. *Classification Techniques:*

This research investigates the performance of several models for intent classification of chat messages. A simple rule-based pattern-matching model, and supervised ML models using common classification algorithms. Prior to intent classification, input text passes through several steps for preprocessing, named-entity recognition, and feature extraction. Preprocessing involves segmenting the text into sentences and tokenizing each sentence into a list of tokens (words, numerals, symbols ... etc.) and normalizing alphabetic words by lemmatization. NLTK, a well-known Python library for natural language processing, is used for tokenization, normalization, and part-of-speech (POS) tagging. Named entity recognition processes the text to detect tokens that represent person names, dates and times, names of development tools, and software engineering artifacts. It uses a simple set of pattern-matching rules to detect those entities since the application is closed-domain and the number of possible values is limited. Feature extraction extracts a list of values used for representing each chat message as an input to a classification algorithm. Two types of features are used, textual features and numeric features. Textual features are representations of actual words and tokens in the text using term frequency-inverse document frequency (TF-IDF), while numeric features are calculated based on specific properties of the text. The choice of these numerical features is based on related literature and the analysis of data. Seven numeric features are used in this research representing the message length, number of verbs in past, present, and future tenses, and number of pronouns in first, second, and third person forms. In addition, five features represent numeric scores that indicate the existence of one of five basic dialog acts in the message: question, request, suggestion, agreement, and disagreement. These features are calculated based on simple syntactic pattern-matching rules. The complete NLU pipeline is shown in Fig. 2.

1) *Rule-based classifier:*

Rule-based classification uses a simple approach based on keyword matching and simple linguistic features such as words part of speech. In addition to using the detected named entities, it depends on a lexicon of words that have special semantics e.g. a common development activity.

2) *Machine Learning Classifiers:*

Four common ML techniques were used in this research. Namely: Naïve Bayes (NB), Support Vector Machine (SVM), Logistic Regression, and a Majority Voting ensemble based on the first three classifiers. Implementation of ML classifiers was done in Python using Scikit-learn, a widely used ML framework [25]. The four techniques were trained on the labeled dataset of chat messages. Each chat message is represented using textual and numerical features. After preprocessing and normalization, messages text is vectorized into a TF-IDF vector representing the weight of each. Other numerical features are normalized, and the complete set of features is used for representing each chat message during training.

IV. Experiments and Results

To measure the effectiveness of using the extracted features to discriminate various intent categories, dialog act detection functions for the five basic dialog acts were tested against the labeled dataset. TABLE II shows accuracy and macro average F1-score for the five dialog act detectors.

It is worth noting that dialog act detection accuracy was highly affected by simple DA clues such as question marks and the word “please”. Also, each message in the dataset had only one DA tag indicating the overall DA chosen by annotators, while it may contain several parts, each with its own DA, and the same utterance may also have more than one DA. For example, a request could be phrased in the form of a question, and hence, it should be classified as both request and question. This means that the actual performance of the five DA detectors is even better than what’s shown in TABLE II, which was confirmed by error analysis.

ML classification models were tested with various combinations of feature groups using textual features as a common ground. Textual features are represented using TF-IDF vectors with or without normalization and stop-words removal. Each ML model was also initially tested with different values of hyperparameters, and the best values were chosen for later comparisons. Detailed implementation of classification models and the chosen values of hyperparameters are provided via the project’s website.

Only 80% of the dataset was used for training the ML

TABLE II. Accuracy and Macro F1-Score for Dialog Act Detection

Performance Metric	Question	Request	Suggestion	Positive reply	Negative reply
Accuracy (%)	91.45	89.79	87.78	90.40	98.39
Macro F1-Score	0.87	0.73	0.67	0.59	0.73

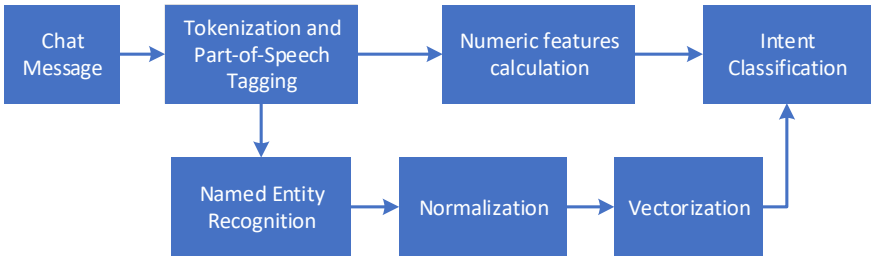

Fig. 2. NLU Pipeline

TABLE III. TEST RESULTS FOR CLASSIFICATION MODELS

Classifier				Accuracy	Weighted F1	Macro F1
Technique	***Textual Features***	***Normalization & Stop-words Removal***	***Numeric Features***			
Pattern Matching (Baseline)	Raw text	no	yes	0.45	0.43	0.38
Naïve Bayes	TFIDF	yes	yes	0.65	0.62	0.46
			no	0.63	0.61	0.47
		no	yes	0.68	0.66	0.52
			no	0.67	0.66	0.53
Support Vector Machine	TFIDF	yes	yes	0.67	0.67	0.56
			no	0.64	0.63	0.49
		no	yes	0.71	0.70	**0.62**
			no	0.70	0.70	0.60
Logistic Regression	TFIDF	yes	yes	0.68	0.68	0.58
			no	0.64	0.64	0.49
		no	yes	0.70	0.71	0.61
			no	0.70	0.70	0.61
Majority Voting Ensemble (NB+SVM+LogReg)	TFIDF	yes	yes	0.70	0.69	0.59
			no	0.67	0.66	0.52
		no	yes	**0.72**	**0.72**	**0.62**
			no	0.71	0.71	**0.62**

model, and the remaining 20% was kept for testing. TABLE III shows a summary of results obtained from testing the proposed classification techniques using accuracy, weighted average F1-score, and macro average F1-score. The best performing model was the majority-voting ensemble trained on TF-IDF and numeric features without normalization nor stop-words removal.

Overall results indicate that both normalization and removal of stop words had a negative impact on the performance. This is justifiable, since normalization and stop-words removal may remove some useful information that would help improve intent detection. The use of numeric meta features, on the other hand, had a positive effect that becomes very clear when normalization or stop-words removal is applied.

TABLE IV shows detailed test results for every intent category using the majority voting ensemble. The best performance was in the *greet* intent category, while the worst performance was in *report issue*. The low value for the macro average F1-score in most models was due to the very low frequency of some categories resulting in a very small number of samples in the training set compared to other categories.

TABLE IV. DETAILED RESULTS FOR MAJORITY VOTING ENSEMBLE

intent	precision	recall	f1-score	samples
discuss_task	0.79	0.80	0.80	357
give_feedback	0.66	0.48	0.55	84
greet	0.93	0.86	0.89	143
plan_task	0.57	0.68	0.62	120
query_plan	0.58	0.50	0.54	28
query_progress	0.62	0.54	0.58	28
query_rules	0.67	0.40	0.50	20
report_issue	0.40	0.33	0.36	24
report_progress	0.75	0.66	0.70	100
request_feedback	0.61	0.67	0.64	86
schedule_meeting	0.73	0.69	0.71	64
state_rules	0.55	0.39	0.46	46
others	0.72	0.79	0.75	496

V. CONCLUSIONS

The results show that intent classification of software developers chat messages from a management perspective is possible using common ML algorithms. Experiments conducted on different classification models have shown that careful choice of classification features has a significant impact on performance. A set of techniques for extracting useful features for text classification in the software engineering domain was implemented and tested.

Future work will include further investigation of deep learning techniques for intent classification on a larger dataset, testing the proposed classification models on datasets of industrial software engineering projects, and the integration of intent classification into a complete conversational assistant for software engineering team management.

VI. REFERENCES

[1] M. Hamroun and M. S. Gouider, "A survey on intention analysis: successful approaches and open challenges," *Journal of Intelligent Information Systems,* 2020.

[2] P. B. Brandtzaeg and A. Følstad, "Why People Use Chatbots," in *Internet Science*, Cham, 2017.

[3] H. Chen, X. Liu, D. Yin and J. Tang, "A Survey on Dialogue Systems: Recent Advances and New Frontiers," *SIGKDD Explor. Newsl.,* vol. 19, p. 25–35, 11 2017.

[4] S. Hussain, O. Ameri Sianaki and N. Ababneh, "A Survey on Conversational Agents/Chatbots Classification and Design Techniques," in *Web, Artificial Intelligence and Network Applications*, Cham, 2019.

[5] K. Ramesh, S. Ravishankaran, A. Joshi and K. Chandrasekaran, "A Survey of Design Techniques for Conversational Agents," in *Information, Communication and Computing Technology*, Singapore, 2017.

[6] A. Sameera, D. J. Abdul-Kader and Woods, "Survey on Chatbot Design Techniques in Speech Conversation Systems," *International Journal of Advanced Computer Science and Applications(ijacsa),* vol. 6, 2015.

[7] J. Cerezo, J. Kubelka, R. Robbes and A. Bergel, "Building an Expert Recommender Chatbot," in *2019 IEEE/ACM 1st International Workshop on Bots in Software Engineering (BotSE)*, 2019.

[8] N. Bradley, T. Fritz and R. Holmes, "Context-Aware Conversational Developer Assistants," in *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, 2018.

[9] E. Paikari, J. Choi, S. Kim, S. Baek, M. Kim, S. Lee, C. Han, Y. Kim, K. Ahn, C. Cheong and A. van der hoek, "A Chatbot for Conflict Detection and Resolution," in *2019 IEEE/ACM 1st International Workshop on Bots in Software Engineering (BotSE)*, 2019.

[10] A. Abdellatif, K. Badran and E. Shihab, "MSRBot: Using bots to answer questions from software repositories," *Empirical Software Engineering,* vol. 25, p. 1834–1863, 2020.

[11] A. Olney, M. Louwerse, E. Matthews, J. Marineau, H. Hite-Mitchell and A. Graesser, "Utterance Classification in AutoTutor," in *Proceedings of the HLT-NAACL 03 Workshop on Building Educational Applications Using Natural Language Processing - Volume 2*, USA, 2003.

[12] R. G. Athreya, A.-C. Ngonga Ngomo and R. Usbeck, "Enhancing Community Interactions with Data-Driven Chatbots–The DBpedia Chatbot," in *Companion Proceedings of the The Web Conference 2018*, Republic and Canton of Geneva, CHE, 2018.

[13] M. Y. H. Setyawan, R. M. Awangga and S. R. Efendi, "Comparison Of Multinomial Naive Bayes Algorithm And Logistic Regression For Intent Classification In Chatbot," in *2018 International Conference on Applied Engineering (ICAE)*, 2018.

[14] C. Chelba, M. Mahajan and A. Acero, "Speech utterance classification," in *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03).*, 2003.

[15] S. Ravuri and A. Stoicke, "A comparative study of neural network models for lexical intent classification," in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2015.

[16] G. Tur, L. Deng, D. Hakkani-Tür and X. He, "Towards deeper understanding: Deep convex networks for semantic utterance classification," in *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, 2012.

[17] J. Gao, M. Galley and L. Li, "Neural Approaches to Conversational AI," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, New York, NY, USA, 2018.

[18] S. Ravuri and A. Stolcke, "Recurrent neural network and LSTM models for lexical utterance classification," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[19] W. Maalej, Z. Kurtanović, H. Nabil and C. Stanik, "On the automatic classification of app reviews," *Requirements Engineering,* vol. 21, p. 311–331, 2016.

[20] D. Arya, W. Wang, J. L. C. Guo and J. Cheng, "Analysis and Detection of Information Types of Open Source Software Issue Discussions," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, 2019.

[21] A. Wood, P. Rodeghero, A. Armaly and C. McMillan, "Detecting Speech Act Types in Developer Question/Answer Conversations during Bug Repair," in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, New York, NY, USA, 2018.

[22] I. Morales-Ramirez, F. M. Kifetew and A. Perini, "Speech-acts based analysis for requirements discovery from online discussions," *Information Systems,* vol. 86, pp. 94-112, 2019.

[23] D. Braun, A. Hernandez Mendez, F. Matthes and M. Langen, "Evaluating Natural Language Understanding Services for Conversational Question Answering Systems," in *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, Saarbrücken, Germany, 2017.

[24] J. Schuurmans and F. Frasincar, "Intent Classification for Dialogue Utterances," *IEEE Intelligent Systems,* vol. 35, pp. 82-88, 1 2020.

[25] P. Fabian, G. Alexandre, M. Vincent, T. Bertrand, I. Parietal, G. Olivier, P. Peter, W. Ron, V. Jake and B. Mikio, *Escikit-learn: Machine Elearning in Python.*