MUAHAH: Taking the Most out of Simple Conversational Agents

Leonor Llansol
INESC-ID & Instituto Superior Técnico, Porto, Portugal

João Santos
INESC-ID & Instituto Superior Técnico, Porto, Portugal

Luís Duarte
CISUC, DEI, Universidade de Coimbra, Portugal

José Santos
CISUC, DEI, Universidade de Coimbra, Portugal

Mariana Gaspar
INESC-ID & Instituto Superior Técnico, Porto, Portugal

Ana Alves
CISUC & Instituto Politécnico de Coimbra, Portugal

Hugo Gonçalo Oliveira
CISUC, DEI, Universidade de Coimbra, Portugal

Luísa Coheur[a]
INESC-ID & Instituto Superior Técnico, Porto, Portugal

[a] Corresponding author

Abstract

Dialog engines based on multi-agent architectures usually select a single agent, deemed to be the most suitable for a given scenario or for responding to a specific request, and disregard the answers from all of the other available agents. In this work, we present a multi-agent plug-and-play architecture that: (i) enables the integration of different agents; (ii) includes a decision maker module, responsible for selecting a suitable answer out of the responses of different agents. As usual, a single agent can be chosen to provide the final answer, but the latter can also be obtained from the responses of several agents, according to a voting scheme. We also describe three case studies in which we test several agents and decision making strategies; and show how new agents and a new decision strategy can be easily plugged in and take advantage of this platform in different ways. Experimentation also confirms that considering several agents contributes to better responses.

2012 ACM Subject Classification Computing methodologies → Natural language processing; Computing methodologies → Multi-agent systems; Computing methodologies → Artificial intelligence; Information systems → Information retrieval

Keywords and phrases Dialog systems, question answering, information retrieval, multi-agent

Digital Object Identifier 10.4230/OASIcs.SLATE.2021.7

Supplementary Material *Software (Source Code)*: `https://github.com/leonorllansol/muahah`
Software (Source Code): `https://github.com/NLP-CISUC/muahah`

Funding This work was partially supported by: the project Flowance (POCI-01-0247-FEDER-047022), co-financed by the European Regional Development Fund (FEDER), through Portugal 2020 (PT2020), and by the Competitiveness and Internationalization Operational Programme (COMPETE 2020); the demonstration project AIA, "Apoio Inteligente a empreendedores (chatbots)", funded by the Fundação para a Ciência e Tecnologia (FCT), through the INCoDe 2030 initiative; and by national funds through FCT, within the scope of the project CISUC (UID/CEC/00326/2020) and by European Social Fund, through the Regional Operational Program Centro 2020.

10th Symposium on Languages, Applications and Technologies (SLATE 2021).
Editors: Ricardo Queirós, Mário Pinto, Alberto Simões, Filipe Portela, and Maria João Pereira; Article No. 7; pp. 7:1–7:12

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Recent technology advances have brought many frameworks for building conversational agents. Examples are Rasa[1] or Dialogflow[2] and they "are often designed with the tacit assumption that at any time, there is only one agent and one human" [11]. Behind such systems, there is usually a set of modules, specialized in different topics or types of dialog. Thus, each system is developed with a distinct task in mind, and, after understanding the user intentions, the most appropriate module is triggered, while the remaining are set aside. Yet, we can be faced with a scenario where (i) a new agent has to cover different domains or types of dialog, for which there is no training data nor resources for manually defining relevant entities and user intents; (ii) a set of independent agents is available, but they are only expert on more specific domains, possibly overlapping, or follow different techniques, and, again, there is no data to train a model for deciding which agent provides the answer to each user request.

In this paper, we tackle the aforementioned issue, and assume that *all agents can potentially answer all questions*. With this in mind, we implemented a new platform, in Python: the Multi-Agents Hand-in-Hand platform (dubbed MUAHAH), enables the integration of distinct agents, expert on different domains and/or with different levels of complexity, as well as decision strategies, possibly taking all the agents' answers into account. We present MUAHAH and evaluate it in three case studies. In the first two, we consider: (i) a panoply of retrieval-based agents, specifically, agents that get their answers from a given knowledge base with trigger/answer pairs, in which trigger is a user request (e.g., a question) and answer is the response to that request; (ii) two decision strategies, namely *Simple Majority*, which implements a majority vote between all the agents and picks the most voted answer, and *Priority System*, which prioritizes the answer of an agent over the others. Here, combining different agents indeed leads to more plausible answers for complex and out-of-domain requests. In the third case study, we integrated new agents in MUAHAH, based on retrieval and embedding methods, as well as a new decision strategy, *Borda Count*, to confirm that one can easily rely on MUAHAH for developing new dialog systems, and that combining ranks of answers by different agents leads to more accurate systems. Still, it should be clear that, although we have implemented several agents, we are not proposing a new way of creating them, but a plug-and-play framework to integrate previously created agents. In addition, we are not claiming a particular way of taking advantage of these agents, but, again, a framework that enables to customise how the responses of several agents are combined.

The paper is organized as follows: Section 2 briefly covers related work on chatbot development and architecture; Section 3 describes MUAHAH, the proposed platform; Sections 4, 5 and 6 describe the three case studies; finally, Section 7 discusses the main conclusions and directions for future work.

2 Related Work

Since the early rule-based systems such as the famous ELIZA [24] to the recent end-to-end dialog systems [25, 26, 22], conversational agents can be found in many different flavours. Also, some agents are task-oriented (e.g., Max [18] was a guide in the Heinz Nixdorf Museums Forum), while others target to engage in general conversations (e.g., SSS [1] is a retrieval-

[1] `https://rasa.com`
[2] `https://dialogflow.com`

based agent with a knowledge base of subtitles). As previously stated, we do not intend to contribute with an architecture for building agents, but with a platform that allows their integration.

We define *agent* as any piece of software that, upon receiving a user request, delivers one or more responses. In many systems, these agents are the modules that retrieve an answer about a specific topic. For instance, Gunrock [3], the winner of the 2018 Amazon Alexa Prize, maps user's input into one of 11 possible topic dialog modules (e.g., movies, books and animals). Here, we do not assume that an agent is specialized in a certain topic, but that we want a simple way to put all these systems together. A similar framework is MACA [23] that touches a number of common points with our work. However, MACA proposes several integrating processes, as, for instance, the re-usability of slots across different tasks. Here, even though the same knowledge sources can be shared between agents, each agent is independent of each other. Also, our agents are *slaves* of a coordinator, as there is no interaction between them.

An important module common to many chatbot architectures is the Dialog Manager, responsible for keeping track of the dialog state and using it to select a response [14]. Gunrock has a Dialog Manager that classifies the user's intent, and, based on this intent, selects the most appropriate module to forward the user request. Sounding Board [8], the winner of the 2017 Amazon Alexa Prize, uses a Dialog Manager that contains a set of "miniskills" for handling different topics, where only one miniskill is selected per turn, based in the topic of the user request. Both the aforementioned systems classify the intent and topic of the user request for selecting a single proper module that will handle it. NPCEditor's [14] Dialog Manager takes a list of responses and uses the dialog state to choose one of them. Here, we do not have a Dialog Manager, but: an Agent Manager, responsible for forwarding the user request to all the active agents and to send their answers to a coordinator; and a Decision Maker, that sends the agents' answers to a set of available decision methods and returns their answers to the coordinator. Unlike most Dialog Managers, these two modules are not trained.

Still on multi-agent scenarios, many other research questions are currently under study. For instance, Divekar et al. [5] target to determine which agent is being addressed, and Eisman et al. [6] train a multi-agent dialog model via reinforcement learning. In the latter scenario, agents learn with each other by interacting in natural language. Here, we assume that the agents are already trained or rely on unsupervised techniques.

3 Proposed Architecture

In this section, we describe the architecture of MUAHAH, depicted in Figure 1. The *Coordinator* is a central module that acts as an interface between the user, the *Agent Manager* and the *Decision Maker*.

The *Agent Manager* aims at providing an interaction point between MUAHAH and the provided agents. It is responsible for launching the agents, and for guaranteeing that the communication between them and the *Coordinator* is done correctly. When MUAHAH is booted up, the *Agent Manager* locates all the available agents through their configuration files and integrates an instance of each into the system. Upon receiving a request from the *Coordinator*, the *Agent Manager* sends it to the integrated agents, which provide their answers. Such answers are then sent back to the *Coordinator*, which finally sends them to the *Decision Maker*. Note that each agent is only responsible for outputting (at least) a response for the given request.

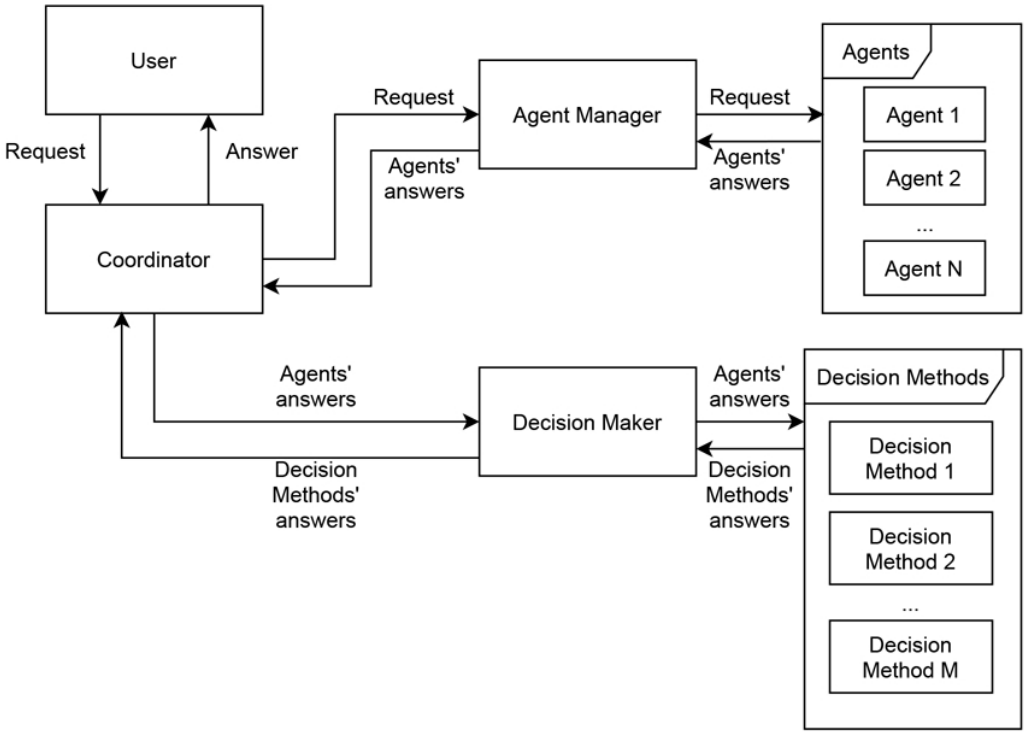

Figure 1 MUAHAH architecture.

The *Decision Maker* makes its decision based in one of the implemented decision strategies. These can consider different items, such as the answers of the agents, the agent who gave each answer, and the user query. But new decision strategies can be added to MUAHAH[3]. Each decision strategy has a weight, defined in the system's configuration file. The *Decision Maker* returns a set of answers, one given by each *Decision Method* to the *Coordinator*, which uses those weights for selecting the best answer and finally return it to the user. Originally, the *Decision Maker* includes two strategies: *Simple Majority* and *Priority System*. Yet, new methods can be integrated, as in our third case study (Section 6).

With *Simple Majority*, the *Decision Maker* will deliver the most frequent answer between the set of answers by the agents. In a scenario where a certain agent's answer would be accurate, even if it did not always deliver an answer to the given query, it would make sense for that agent to be prioritized. For that purpose, we developed the *Priority System*, a decision strategy that allows the developer to explicitly set priorities for the available agents. Therefore, when evaluating the answer each agent delivers to a given user request, MUAHAH verifies whether the prioritized agent was able to deliver a response to the user request: if so, that answer is deemed to be a plausible one and is returned to the user; if the prioritized agent is not able to answer the user request, the system considers the answers by each of the remaining agents and selects the final answer through *Simple Majority*, as described earlier.

[3] To do so, an abstract class, `Decision Method`, has an abstract method, `getAnswer`, that takes a set of answers and delivers the best, based on its heuristic. Hence, to add a new decision strategy, only a source file is needed, with the class that extends `Decision Method` and implements `getAnswer`. It is also necessary to add the new strategy's name and its weight to the system's configuration file.

Finally, in order to build a new agent for MUAHAH, the developer needs to customize a configuration file, which allows the agent to be called by the *Agent Manager*; it also allows the developer to set configurable parameters without directly interacting with the source files. In addition, the agent's source files must contain a class that extends the abstract class (`Agent`) and implements its own `requestAnswer` method, which takes a user request and returns a list of answers. Finally, since an agent can be active or inactive, it is also necessary to edit the system's configuration file, adding the name of the agent and whether it is active or not.

For a better understanding, we refer to the working examples in Sections 4 and 5.

4 Case study A: using the Simple Majority

We first evaluate the Simple Majority strategy in MUAHAH. Here, the Subtle corpus [2] was used as the knowledge base of nine agents. Subtle[4] contains about 3 million interactions from movie subtitles, i.e., trigger/answer pairs like "Trigger: I like the sea., Answer: me too". Due to the large number of included interactions, Subtle was indexed in a search engine, thus enabling that only a small set of interactions is retrieved as response candidates. In this case, 20 interactions are retrieved from Lucene[5]. Then, our agents compare the user request both with the trigger and with the answer, as both might be useful for finding an adequate answer. The difference between the agents is the algorithm for computing sentence similarity, always in the 0 to 1 range, which is either *Cosine* [20], *Jaccard* [13] or *Edit Distance* [15], and the weight given to the similarity between requests, triggers and their response. For such, three different agents, that extend the abstract class `Agent`, are added to our system: `Cosine-Agent`, `Jaccard-Agent` and `EditDistance-Agent`, each implementing the *requestAnswer* method according to their algorithm. Moreover, three agents were created of each type, with different similarity weights. Table 1 summarises the nine agents. For instance, agent C2 uses the cosine similarity and gives more weight to the similarity between the user request and the trigger than between the user request and the answer, in a relation of 75 to 25.

Table 1 Agent distribution according to the similarity computation and weights.

RequestSim/ResponseSim	50/50	75/25	100/0
Cosine	C1	C2	C3
Edit Distance	E1	E2	E3
Jaccard	J1	J2	J3

For each agent, sentences were lower-cased. Then, for computing the *Cosine Similarity*, punctuation was removed, stopwords were kept, and words were not stemmed. For the *Edit Distance*, both punctuation and stopwords were kept. Finally, for the *Jaccard Similarity*, both punctuation and stopwords were removed.

To test this configuration, we created a set of 100 simple questions (e.g., "What's your name?", "How are you?") and a set of 100 more complex questions (e.g., "What's your opinion on Linux-based platforms?", "If the world ended tomorrow, what would you do today?"). Then, we run both the resulting system and compared it to Say Something Smart (SSS) [16],

[4] A smaller version was used to reduce index creation time and is available from `https://github.com/leonorllansol/muahah/blob/master/corpora/1million.txt`

[5] `https://lucene.apache.org`

which has a single retrieval agent. For this, four annotators were given a sample of 25 simple and 25 complex questions with the answers by both systems, randomly selected, and asked to score each of the responses between 1 and 4, based on whether the answer:

4: Was plausible without additional context;
3: Was plausible in a specific context, or did not actively answer the query but maintained its context;
2: Actively changed the context (e.g., an answer that delivers a question to the user which does not match the initial query's topic), or contained Structural issues (even if its content fits in the context of the question);
1: Had no plausibility value.

For illustrative purposes, one response of each kind for the question "What is the price of olive oil in 7-Eleven?" would be:

4: "In 7-Eleven, the olive oil costs 3.00€."
3: "He knows what is the price."
2: "Olive oil."
1: "Eleven."

The mean and mode of all answer scores were computed for each system. Also, following the evaluation of other engines, such as AliMe [19], we considered that, to be discerned as acceptable, a given answer would need an average score of at least 2.75 between the four annotations (corresponding, e.g., to the case where three annotators score it 3 and the last one 2). Table 2 summarizes the obtained results.

Table 2 SSS system against our system when answering basic questions.

	Simple questions			**Complex questions**		
	Mean Score	**Answers $\geq$ 2.75**	**Approval**	**Mean Score**	**Answers $\geq$ 2.75**	**Approval**
SSS	2.68	23 / 50	46%	2.26	11 / 50	22%
MUAHAH	2.6	24 / 50	48%	2.6	22 / 50	44%

This instance of MUAHAH keeps up with SSS for basic questions and outperforms it with complex questions. We should add that, in what concerns simple questions, scores assigned to SSS were more polarized, with a special focus on scores of 2 and 4 to its answers, while scores of MUAHAH answers were more evenly distributed. With complex questions, 2 was the most common score for both systems, with a stronger preponderance for SSS to deliver implausible answers. This can be explained by the fact that SSS relies on a single agent to decide on all its answers, while the multi-agent system considers what possible answers are more common, from multiple points of view.

To better understand the use of the *Simple Majority* strategy, we present a running example, using the aforementioned agents, `Cosine-Agent`, `Jaccard-Agent` and `EditDistance-Agent`, and the Subtle corpus.

1. The user poses the query q to the system: "Quando começa o Verão?" (When does summer start?)
2. The *Coordinator* forwards q to the *Agent Manager*.
3. The *Agent Manager* instantiates the active agents: `Cosine-Agent`, `Jaccard-Agent` and `EditDistance-Agent`, and forwards q to them.
4. Each agent selects a response, based on the sentence similarity between the request and the retrieved candidates. Their selected responses are returned to the *Agent Manager*:

- A_{Cos}: "Assim que o senhor e a chuva desaparecerem." (As soon as you and the rain disappear.)
- A_{Jac}: "Assim que o senhor e a chuva desaparecerem." (As soon as you and the rain disappear.)
- A_{ED}: "Não te ouvimos." (We can't hear you.)

5. The *Agent Manager* sends the agents' answers to the *Coordinator*.
6. The *Coordinator* forwards the agents' answers, [A_{Cos}, A_{Jac}, A_{ED}], to the *Decision Maker*.
7. The *Decision Maker* sends the agents' answers to the active *Decision Method*: *Simple Majority*, with a weight of 1.
8. *Simple Majority* returns the more common answer, A_{Cos} and A_{Jac}, to the *Decision Maker*.
9. The *Decision Maker* returns the method's answer to the *Coordinator*.
10. The *Coordinator* returns the answer to the user: "Assim que o senhor e a chuva desaparecerem." (As soon as you and the rain disappear.)

5 Case study B: Priority System

The *Priority System* is a decision strategy that gives priority to a certain agent if that agent can deliver a response. To test this strategy, we use Edgar Smith [9], another retrieval-based conversational agent, this time on a specific domain. Edgar answers questions about the Monserrate palace, in Sintra, based on a corpus[6] of 1,179 interactions. Besides question/answer pairs about the palace, Edgar answers some out-of-domain (chit-chat) questions such as "What's your name?" or "How old are you?". However, Edgar will not be able to answer most out-of-domain questions. This is a significant issue, as users tend to get more engaged with conversational platforms when answers to chit-chat queries are delivered.

To address the aforementioned issue, we set up MUAHAH to use Edgar as a Priority Agent, with the nine agents described in case study A (Section 4) used as Edgar's back-up, in case Edgar is not able to provide an answer. If not backed-up, when Edgar's top-scored answer has a similarity score below a given threshold, defined in its configuration file, it says "I do not understand your question". We use again the *Jaccard Similarity* to compare the user request with the questions in Edgar's knowledge base, a threshold of 0.35, and the previous *Simple Majority* decision strategy for choosing an answer when Edgar is not able to provide one.

We created two sets of 100 questions each. The first with questions about the palace and the second with other kind of requests, such as personal and trivia questions (e.g., "What is your name?", "Do you like to sing?"). Similarly to case study A, four annotators were given a sample of 50 questions and answers from each set, and were, once again, asked to score each response between 1 and 4, according to the earlier described criteria. Table 3 summarizes the results.

As expected, in what concerns questions about the palace, Edgar achieved the best performance, but this MUAHAH configuration managed to keep up with minor accuracy costs, and both systems had most of their answers scored 4. On the other hand, regarding out-of-domain questions, MUAHAH beat Edgar by a comfortable margin.

To better understand the use of the *Priority System* strategy, we present a running example, using as prioritized agent the aforementioned Edgar, and also the remaining agents `Cosine-Agent`, `Jaccard-Agent` and `EditDistance-Agent` as backup. Note that Edgar has its own corpus, while the other agents use the Subtle corpus.

[6] Available from `https://github.com/leonorllansol/muahah/blob/master/corpora/edgar/edgar.txt`

Table 3 Edgar evaluated against our system.

	Questions about the palace			Out-of-domain Questions		
	Mean Score	**Answers $\geq$ 2.75**	**Approval**	**Mean Score**	**Answers $\geq$ 2.75**	**Approval**
Edgar	3.015	31 / 50	62%	2.37	18 / 50	36%
MUAHAH	2.87	29 / 50	58%	2.625	22 / 50	44%

1. The user poses query q to the system: "Quando foi construído o Palácio de Monserrate?" (When was the Palace of Monserrate built?)
2. The *Coordinator* forwards q to the *Agent Manager*.
3. The *Agent Manager* instantiates the active agents: Edgar, `Cosine-Agent`, `Jaccard-Agent` and `EditDistance-Agent`, and forwards q to them.
4. Each agent selects a response, based on the sentence similarity between the request and the retrieved candidates. Their selected responses are returned to the *Agent Manager*:
   - A_{Cos}: "Quando lhe telefonei a meio da noite." (When I called you in the middle of the night.)
   - A_{Edgar}: "O palácio foi construído entre 1858 e 1864, sobre a estrutura de um outro palácio." (The palace was built between 1858 and 1864, on the structure of another palace.)
   - A_{Jac}: "Meu Deus!" (My God!)
   - A_{ED}: "Quando lhe telefonei a meio da noite." (When I called you in the middle of the night.)
5. The *Agent Manager* sends the agents' answers to the *Coordinator*.
6. The *Coordinator* forwards the agents' answers, [A_{Cos}, A_{Edgar}, A_{Jac}, A_{ED}], to the *Decision Maker*.
7. The *Decision Maker* sends the agents' answers to the active *Decision Method*: *Priority System*, with a weight of 1.
8. *Priority System* returns the answer given by the prioritized agent Edgar, A_{Edgar}, to the *Decision Maker*.
9. The *Decision Maker* returns the method's answer to the *Coordinator*.
10. The *Coordinator* returns the answer to the user: "O palácio foi construído entre 1858 e 1864, sobre a estrutura de um outro palácio." (The palace was built between 1858 and 1864, on the structure of another palace.)

For queries out of Edgar's domain of expertise (e.g., "Quando começa o Verão?"), Edgar is expected to given a low similarity score to the top-ranked answer. If this score is below the set threshold, Edgar will return the default "no answer" message, which results in the application of *Simple Majority* to the remaining agents.

6 Case study C: Integrating new Agents and new Decision Strategy

The final case study was significantly different and tackled the integration of new agents and of a new decision strategy in MUAHAH. Briefly, we integrated three retrieval-based agents that, given a request, in natural language, retrieve the responses for similar requests in a knowledge base (KB). Each agent follows a different matching technique: (i) Agent-BM25 is based on the search engine Whoosh[7], used for indexing the KB and retrieving suitable candidates

[7] `https://whoosh.readthedocs.io/`

with the BM25 ranking function; (ii) Agent-W2V relies on a pre-trained word2vec [17] model for encoding the requests in a numeric vector, i.e., requests are represented by the average embedding vector of their words, and candidate requests in the KB are ranked according to their cosine similarity with the user request; (iii) Agent-BERT relies on a pre-trained BERT [4] model for encoding the requests in a numeric vector (i.e., the [CLS] embedding of the second to last layer), and ranks the candidate requests in the KB according to their cosine with the user request.

For integrating each agent, a sub-directory with the name of the agent was created under the directory managed by the Agent Manager. In this directory, a script with the directory name was added, implementing the `requestAnswer` method, where the retrieval logic resides. In this case, it has a string parameter with the request and returns a list of answers ranked according to their matching logic. In the same directory, a `config.xml` file was created with the name of the main class. Finally, in the main MUAHAH directory, the name of the agent was added to the main `config.xml` and set to active. We also inactivated all agents except the new three.

The integrated decision strategy was the Borda Count voting [7], which scores each candidate response according to their rank by different agents. This is different from the previous decision strategies, which rely exclusively on the first answer given by each agent. For this reason, Borda Count expects that the `requestAnswer` method of the agents returns a list of results. The higher the rank, the higher the score. More precisely, for each agent, the score of each candidate on a rank will be equal to the number of considered positions minus its position in the rank, with the latter starting in 0. For instance, if we consider the top-5 candidates, the first candidate gets 5 points and the fifth gets 1. The selected response will then be the one with the highest final score, obtained by summing all of its partial scores, in all the considered lists, in this case, retrieved by each search strategy.

For integrating Borda Count, a script was created with its name and added to the directory managed by the Decision Maker module. This script is based on a class that extends the `DecisionMethod` class and implements the method `getAnswer` with the logic underlying Borda Count.

To test the integration, we used the three agents for answering questions about entrepreneurship and economic activity in Portugal, for which there is a corpus of 855 Frequently Asked Questions (FAQs), in Portuguese, and their variations [10][8], i.e., paraphrases or related questions using other words, possibly omitting information, that simulate user requests. Some of the questions are quite complex, so our agents are not expected to answer every request successfully, especially considering that they rely on simple matching techniques and were not trained for the target domain. Examples of questions include "*É obrigatório declarar o exercício de uma atividade económica junto da Autoridade Tributária e Aduaneira?*" (Is it mandatory to declare the exercise of an economic activity with the Tax and Customs Authority?) or "*Qual o número máximo de crianças permitido por Ama?*" (What is the maximum number of children allowed by a nanny?), with variations like "*Como devo informar a autoridade tributária sobre a minha atividade*" and "*Quantas crianças uma ama pode acolher?*".

We note the natural language of the requests is irrelevant to MUAHAH. When necessary, specific natural language processing, namely language-specific resources and tools, have to be included in the agents. For instance, in this case study, where the corpus is in Portuguese,

[8] Available from `https://github.com/NLP-CISUC/AIA-BDE/`

Agent-W2V used a word2vec-CBOW model pre-trained for Portuguese, with 300-sized vectors available from the NILC embeddings [12] and Agent-BERT used BERTimbau [21] large, a pre-trained BERT model for Portuguese that encodes sentences in 1,024-sized vectors. Whoosh was used with no language-specific analyser.

The goal of this case study was twofold: (i) to test whether the agents, now in MUAHAH, could effectively respond to user requests, here simulated when matching variations with questions in the KB; (ii) to compare the performance of using all three agents, alone and in parallel, i.e., combined with SimpleMajority or Borda Count for selecting the most suitable question. The corpus contains variations produced by different methods, but we focused on two handcrafted types, namely: (i) VUC, 816, written by the creators of the corpus; (ii) VMT, 168, by contributors of the Mechanical Turk crowdsourcing platform. For each variation, agents ranked each of the 855 FAQs according to their suitability. Despite the availability of the agents described in Sections 4 and 5, we note that these three agents were the only effectively used for this purpose.

Since, for this case study, there were previously-created variations, simulating user requests, each one with a known answer, evaluation could be automatized. Table 4 has the accuracy of each agent and decision strategy, corresponding to the proportion of variations for which the correct question was ranked first, meaning that the correct answer would be given. Performances are modest, but they improve when the three agents are combined, either with Borda Count or SimpleMajority. This shows that, besides enabling the combination of different types of dialog or domains in the same platform (Sections 4 and 5), combining different agents for the same domain also leads to a more accurate system. The more complementary the answers of the agents are, the better, which is why we opted for significantly different retrieval techniques.

Table 4 Accuracy of integrated agents, alone and combined, when matching question variations with the correct questions.

	Agent-BM25	**Agent-W2V**	**Agent-BERT**	**SimpleMajority**	**Borda**
VMT	57.1%	65.5%	67.9%	74.4%	72.6%
VUC	55.9%	54.0%	56.7%	61.2%	63.0%

7 Conclusions and Future Work

We presented MUAHAH, a multi-agent platform for dialog systems that allows to easily integrate different agents and, in order to get suitable responses, different decision-making strategies that might consider the answers of all active agents. Such strategies are extremely useful when no data is available for training a model that decides on the best agent for each request. Instead, when targeting a specific domain or different types of dialog, they can often take the most out of an ensemble of simple agents, without requiring additional training or manual intent definition.

Some retrieval-based agents, such as those used in the case studies A and B, are already included in MUAHAH, for different purposes. The same for simple decision strategies. These include those of the first two presented case studies where, according to human users, different combinations of agents resulted in better responses. We further showed that MUAHAH is flexible enough for integrating additional agents and decision strategies, and thus adaptable to different domains. For this reason, we believe that it is a suitable alternative when developing new dialog systems. We thus make MUAHAH and its source code available to all of those

interested: MUAHAH is available from `https://github.com/leonorllansol/muahah`, and its fork including the new agents and new decision strategy, resulting from case study C, is at `https://github.com/NLP-CISUC/muahah`.

A critical aspect of MUAHAH are the decision strategies, where there is plenty of work to be done. For instance, the aforementioned Gunrock [3] takes into account a fact/opinion interleaving strategy, which is certainly interesting to explore. In particular, we believe that the system's results would greatly improve if it took user feedback into account, which would enable to learn specific weights for each agent, instead of a flat priority to a single (or to multiple) agents. Towards a more natural dialog, a module for dealing with context would also improve conversations. Such a model could keep track of concepts and entities referred and remember them in future interactions. This would enable the chatbots do deal with anaphora, and could also be exploited by pro-active chatbots.

References

1. David Ameixa, Luisa Coheur, Pedro Fialho, and Paulo Quaresma. Luke, I am your father: dealing with out-of-domain requests by using movies subtitles. In *Proceedings of the 14th International Conference on Intelligent Virtual Agents (IVA'14)*, LNCS/LNAI, Boston, 2014. Springer-Verlag.
2. David Ameixa, Luísa Coheur, and Rua Alves Redol. From subtitles to human interactions: introducing the subtle corpus. Technical report, Tech. rep., INESC-ID (November 2014), 2013.
3. Chun-Yen Chen, Dian Yu, Weiming Wen, Yi Mang Yang, Jiaping Zhang, Mingyang Zhou, Kevin Jesse, Austin Chau, Antara Bhowmick, Shreenath Iyer, Giritheja Sreenivasulu, Runxiang Cheng, Ashwin Bhandare, and Zhou Yu. Gunrock: Building a human-like social bot by leveraging large scale real user data, 2018.
4. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, pages 4171–4186, Minneapolis, Minnesota, 2019. ACL. `doi:10.18653/v1/N19-1423`.
5. Rahul R. Divekar, Xiangyang Mou, Lisha Chen, Maíra Gatti de Bayser, Melina Alberio Guerra, and Hui Su. Embodied conversational ai agents in a multi-modal multi-agent competitive dialogue. In *Proceedings of 28th International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 6512–6514. International Joint Conferences on Artificial Intelligence Organization, 2019.
6. Eduardo M. Eisman, María Navarro, and Juan Luis Castro. A multi-agent conversational system with heterogeneous data sources access. *Expert Syst. Appl.*, 53(C):172–191, July 2016.
7. Peter Emerson. The original Borda count and partial voting. *Social Choice and Welfare*, 40(2):353–358, February 2013.
8. Hao Fang, Hao Cheng, Maarten Sap, Elizabeth Clark, Ari Holtzman, Yejin Choi, Noah A. Smith, and Mari Ostendorf. Sounding board: A user-centric and content-driven social chatbot. *arXiv preprint arXiv:1804.10202*, 2018.
9. Pedro Fialho, Luísa Coheur, Sérgio Curto, Pedro Cláudio, Ângela Costa, Alberto Abad, Hugo Meinedo, and Isabel Trancoso. Meet EDGAR, a tutoring agent at MONSERRATE. In *Proceedings of 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 61–66, Sofia, Bulgaria, 2013. ACL.
10. Hugo Gonçalo Oliveira, João Ferreira, José Santos, Pedro Fialho, Ricardo Rodrigues, Luísa Coheur, and Ana Alves. AIA-BDE: A corpus of FAQs in portuguese and their variations. In *Proceedings of 12th International Conference on Language Resources and Evaluation*, LREC 2020, pages 5442–5449, Marseille, France, 2020. ELRA.

11 T. K. Harris, S. Banerjee, A. I. Rudnicky, J. Sison, K. Bodine, and A. W. Black. A research platform for multi-agent dialogue dynamics. In *RO-MAN 2004. 13th IEEE International Workshop on Robot and Human Interactive Communication (IEEE Catalog No.04TH8759)*, pages 497–502, 2004.

12 Nathan S. Hartmann, Erick R. Fonseca, Christopher D. Shulby, Marcos V. Treviso, Jéssica S. Rodrigues, and Sandra M. Aluísio. Portuguese word embeddings: Evaluating on word analogies and natural language tasks. In *Proceedings of 11th Brazilian Symposium in Information and Human Language Technology (STIL 2017)*, 2017.

13 Paul Jaccard. The Distribution of the Flora in the Alpine Zone. *New Phytologist*, 11(2):37–50, 1912.

14 Anton Leuski and David Traum. NPCEditor: A tool for building question-answering characters. In *Proceedings of 7th International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, 2010. ELRA.

15 Vladimir Iosifovich Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707, 1966.

16 Daniel Magarreiro, Luisa Coheur, and Francisco S. Melo. Using subtitles to deal with out-of-domain interactions. In *DialWatt – the 18th workshop on the semantics and pragmatics of dialogue*, SemDial Workshop Series, Edinburgh, 2014. Springer-Verlag.

17 Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of the Workshop track of ICLR*, 2013.

18 Thies Pfeiffer, Christian Liguda, Ipke Wachsmuth, and Stefan Stein. Living with a virtual agent: Seven years with an embodied conversational agent at the heinz nixdorf museumsforum. In *Proceedings of the International Conference Re-Thinking Technology in Museums 2011 - Emerging Experiences*, pages 121–131. thinkk creative & the University of Limerick, 2011.

19 Minghui Qiu, Feng-Lin Li, Siyu Wang, Xing Gao, Yan Chen, Weipeng Zhao, Haiqing Chen, Jun Huang, and Wei Chu. AliMe chat: A sequence to sequence and rerank based chatbot engine. In *Proceedings of 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 498–503, Vancouver, Canada, 2017. ACL.

20 Amit Singhal. Modern information retrieval: A brief overview. *IEEE Data Engineering Bulletin*, 24, January 2001.

21 Fábio Souza, Rodrigo Nogueira, and Roberto Lotufo. BERTimbau: Pretrained BERT models for Brazilian Portuguese. In *Proceedings of the Brazilian Conference on Intelligent Systems (BRACIS 2020)*, volume 12319 of *LNCS*, pages 403–417. Springer, 2020.

22 Zhiliang Tian, Rui Yan, Lili Mou, Yiping Song, Yansong Feng, and Dongyan Zhao. How to make context more useful? an empirical study on context-aware neural conversational models. In *Proceedings of 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 231–236, Vancouver, Canada, July 2017. ACL.

23 Hoai Phuoc Truong, Prasanna Parthasarathi, and Joelle Pineau. MACA: A modular architecture for conversational agents. In *Proceedings of 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 93–102, Saarbrücken, Germany, August 2017. ACL.

24 Joseph Weizenbaum. ELIZA – a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9:36–45, 1966.

25 Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. Neural generative question answering. In *International Joint Conference on Artificial Intelligence IJCAI*, pages 2972–2978, New York, 2016. IJCAI/AAAI Press.

26 Tiancheng Zhao and Maxine Eskenazi. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. In *Proceedings of 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 1–10, Los Angeles, 2016. ACL.