Human-Machine Dialogue Final Project Report

Steve Azzolin
DISI, University of Trento
steve.azzolin@studenti.unitn.it

1. Summary

This document represents the final report for the project of the AIS course Human-Machine Dialogue.

2. Introduction

This works presents FIN, a conversational agent built with the Rasa framework for solving several tasks in the financial domain (FINancial chatbot). Specifically, the general goal of the system, along with answering to some fundamental questions regarding companies and their stock value, is to give investment suggestions based on user preferences. So, the overall mission of the system is to support a user in its daily routine of keeping up with the trends of the stock market, together with looking for new investment opportunities. The system was finally deployed as an Amazon Alexa Skill, and tested using the online Alexa Developer Console.

The target user group of the proposed agent is constituted by both expert and non-expert finance users. To achieve this, the system does not adopt a strict financial jargon. For the rare occasions in which a proper financial jargon is employed (e.g. shorting), the user can ask clarifications at any time about the meaning of the word. The reason for building the agent in this way resides in the intent to assume as little information as possible about the final user. More details about the design of conversations will be given in Section 3.

The proposed system constitutes a mixed initiative task-based conversational agent which is also capable of interacting in a multi-modal manner by sending plots and hyperlinks to the user via a Telegram bot, upon explicit user request. An example of such an interaction is given in Figure 1. As will be described in more detail in Section 4, all information related to the stock value, textual description of companies, and financial news are provided by external APIs, such as Yahoo Finance[1] and The Guardian[2].

3. Conversation Design & Financial Data

Before defining in detail the scope of the agent, an interview with a domain expert was carried out following the principles of the Human-Computer Interaction design lifecycle. This initial confrontation was helpful to frame which are the most important components that a financial chatbot should aim for. After this initial framing, and after several iterations of refinement, the tasks that the agent can perform can be summarized into:

Giving Suggestions based on user preferences: User queries like *Can you give me some suggestions on what to buy?* or *I don't know what to buy* make the bot following this conversational path. The system, then, asks for some slot filling in order to understand which are the preferences of the user in terms of which category of company he/she is interested in (like sport, hardware, entertainment, automotive, software, medicine) and which type of investment he/she is willing to pursue (shorting or long term investment). Based on these preferences, the system then picks a suitable company adhering to the mentioned preferences. Given the difficulty of providing reliable and accurate stock market predictions, and given that the main topic of the project is the Human-Machine conversation, the choice of the company is implemented as a simple random selection between a pool of pre-selected companies divided into the aforementioned categories. A similar intent is automatically fired every time the user asks for information on a company which is not supported by the system (i.e., it is not in the database of supported companies), resulting in a dialogue like the following *U: I would like some news about DeepMind; M: I didn't manage to find DeepMind in my database, please try with another company. Based on your previous searches, I may suggest you to check out Microsoft*. To accomplish this, the previous user's preferences are saved and carried over the conversation.

Searching for stock updates, recent trends, news, and information about companies: A fundamental part of virtually any financial chatbot is the ability to directly provide the current stock value, to provide the latest news, and to succinctly describe a certain company. These three are implemented in FIN by retrieving data from Yahoo Finance for the stock value, from the The Guardian APIs for the latest news, and from the Knowledge Graph Search API[3] for the description of companies. In addition to that, FIN is also capable of sending via Telegram a plot showing the trend of the stock value for the requested company. The data to build such plot, which can be in the form of a simple line plot or a more typical candlestick plot[4], comes from the Stooq APIs[5]. The user preferences for the kind of plot to use are asked the first time and then saved for the rest of the conversation, upon an explicit user request asking to change the plot type.

Searching the best and the worst index: Questions like *Which is the best index?* or *Which index is performing the worse?* bring the proposed agent to search for respectively the best and the worst index of the market. This information, together with some slightly variations like providing

[1] https://it.finance.yahoo.com/
[2] https://open-platform.theguardian.com/
[3] https://developers.google.com/knowledge-graph
[4] https://en.wikipedia.org/wiki/Candlestick_chart
[5] https://stooq.com/

the best/worst category, can naturally fit into a broader conversation flow in order to both properly inform the user about the current status of the market, and to better suggest a suitable novel investment. For the sake of this project however, and in accordance with the lab tutors, this intent was not further developed in favor of a richer mixed initiative conversation. From the implementation side, in order to avoid a full lookup of the market and to avoid complex technicalities like caching or multi-threaded requests, a static dumping of the NASDAQ stock market[6] was performed. So, the best/worst index are simply taken as the best/worst index in this static CSV table.

User Feedbacks: To allow progressive refinements of the developed system also after deployment, and to favor the realization of a complete design lifecycle, the system can gather direct user feedbacks in such a way to constantly monitor the usage and the gaps of the proposed system. Such intent can be fired by requests like: *I want to give a feedback*, *I found a problem*, or *Can you add a feature?*

The aforementioned conversation patterns were also influenced by the data collection phase following the best practice of iterative refinements, which will be described in Section 4. Overall, given that the system can provide suggestions and actively contribute to the dialogue's prosecution, the interaction type can be classified as mixed initiative. In addition to this, the system adopts an implicit confirmation strategy in most of the cases, since the task is non critical and allows for more compact turns. For what concern errors instead, the system implements: *i)* a two-stage fallback policy, which permits the disambiguation of the user's message by asking clarifying questions; *ii)* an unexpected policy (UnexpecTEDIntentPolicy), which allows the model to recognize unlikely intents which happen in unexpected stages of the conversation; and *iii)* a RulePolicy, which enables the system to recognize when the Core module is not able to confidently predict the next action to take, firing an automatic mechanism asking for rephrasing. Nevertheless, in situations in which the system is not performing well the user can always start a new conversation by uttering, for instance, *"Stop"* which fires the *reset* intent.

4. Data Description and Analysis

Thanks to the initial conversation design with a domain expert as discussed in Section 3, it was possible to also draft the first set of happy paths, which was subsequently enriched by hand-defined most probable unhappy paths. During the development lifecycle, revisions were made in order to fix conversation flaws and to better capture the different possible unhappy paths. For achieving this, a total of five users were involved. All users, three males and two females, have a NLP background and the majority of them also have a background with the Rasa framework. In addition, they are all non-native English speaker, yet with a good knowledge of it. A total of 53 stories were collected, from which 8 user stories were left as test stories to test the generalization abilities of the system. The stories involving

[6]https://www.nasdaq.com/market-activity/stocks/screener

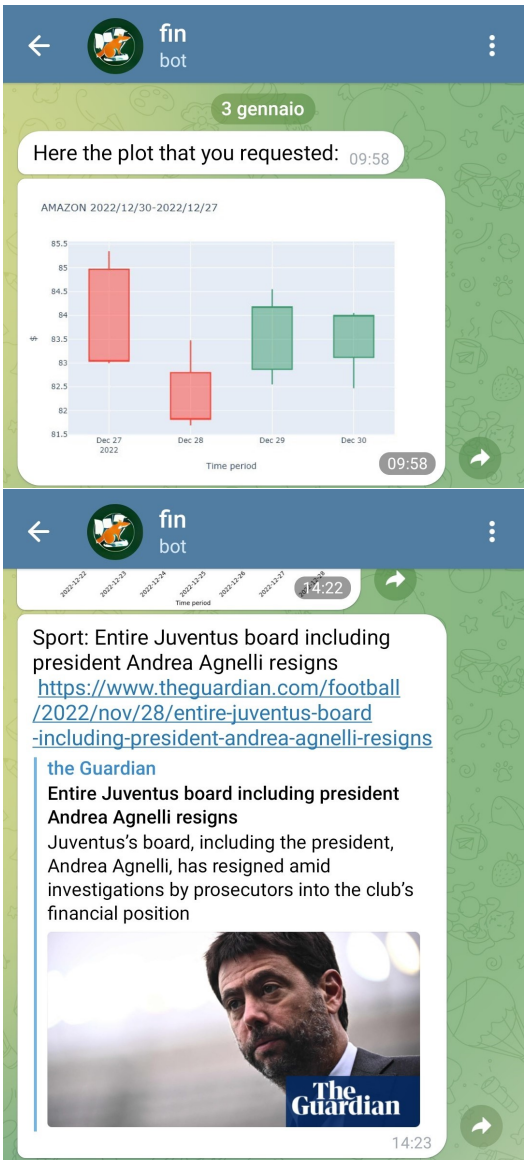

Figure 1: Two examples of the proposed system interacting in a multi-modal manner with the user via a Telegram Bot. The Telegram Bot is named after the name of the proposed system, id est, FIN. In addition, a Bot profile picture was set as an artificially created image by means of DALL-E mini[8](Ramesh et al., 2021) using as prompt *"logo with a fox in a circle holding papers"*

the main task of asking the system for suggestions are composed, on average, by 8 turns, whereas other more tailored intents contain a fewer number of turns.
For what concern the NLU language data instead, the system is equipped with:

- An average of 16 examples for every of the 18 intents. The most critical intents, like the one having entities, clearly have a greater number of examples, for a maximum of 52. Whilst easier intents, like asking for help or for leaving feedback, have a smaller amount of data
- An average of 33 for every of the 6 entities, with a maximum of 134 examples for *company* and a minimum of 8 for *plot_type*

- A total of 11 slots constitute the memory of the system. They are used, for example, to keep track of the company the user wants information about.
- A total of 6 forms in order to fill-in the slots necessary to carry out the defined actions.
- A synonym mapper to help the system to deal with different ways of referring to the same company (Meta=Facebook, Juventus=Juve, software=SW, etc..).
- A local database listing the supported companies along with their associated category (Meta=software, Tesla=cars, etc..). For the sake of simplicity, this database is implemented as a simple Python dictionary. In total, 15 companies were available for 5 different categories.

A detail worth of notice is that during the entire phase of data generation/gathering the guiding principle has been to collect the smallest amount of data covering as much as possible all the required aspects for a fairly good performing model. Indeed, as will be described in Section 7, the training of the system on my laptop was extremely slow and in order to maintain a fairly contained training time, allowing for a reasonable debugging lifecycle, I opted for limiting on purpose the amount of data diversity. This, of course, somehow limits the generalization ability of the proposed system. However, please note that increasing the amount of training data, by means for example of Chatette[9] or any other hand-designed data generator script, would be straightforward given the availability of a faster calculator. This choice was supported by the simplifying assumptions of a clean communication channel and of a close company domain, id est, the list of available companies is fixed.
In the proposed system, despite the potential advantage of making the training data stories modular, *checkpoints* were not used. The specific reason is that, as already mentioned, the training data was aimed to be the smallest amount possible yet resulting in a good performing model. Thus the effect of a more modular set of stories was less effective. In addition, as outlined by the official Rasa documentation[10], checkpoints should be used with care and may slow down the training. So, I decided to do not use them at all.
The source of the financial information was already hinted along with its relative use cases, in Section 3. A complete description of the data format is out of the scope of this report, since they often contain a multitude of information often related to specific aspects of finance, which are not taken into account in the proposed agent.

5. Conversation Model

As already indicated, the agent was built via the Rasa framework. The computational pipeline for the NLU Rasa module is comprised by: *i)* WhiteTokenizer for splitting the sentence into tokens by simply looking at white spaces; *ii)* Featurizers (RegexFeaturizer, LexicalSyntacticFeaturizer, and CountVectorsFeaturizer) in order to convert a word into a set of features which can be processed by the successive steps; *iii)* DIETClassifier (Bunk et al., 2020) for performing both intent and entity classification; *iv)* EntitySynonymMapper to map different synonyms to a unified entity; *v)* FallbackClassifier, which maps any ambiguous predicted intent to the default *nlu_fallback* for handling the potential misinterpretation.
For what concern the CORE module instead, the system is composed by: *i)* MemoizationPolicy which learns the training stories and tries to replicate the conversational path at inference time; *ii)* RulePolicy which enables the system to handle fixed conversational patterns and to recognize when the Core module is not able to confidently predict the next action to take; *iii)* UnexpecTEDIntentPolicy which enables the system to detect unexpected intents; *iv)* TEDPolicy (Vlasov et al., 2019) for next action prediction and entity recognition.
Please, note that the aforementioned pipeline resembles the standard pipeline offered by Rasa. In Section 6 a comparison of different pipelines will be provided, where it will be shown that even this simple pipeline performs almost on par with the others.
Further details about specific hyper-parameters can be found in the attached repository.

6. Evaluation

6.1. Intrinsic Evaluation

The Rasa framework provides several built-in tools for performing a complete intrinsic evaluation of the system. Many different evaluation strategies are possible, and among those Table 1 shows the 3-fold cross-validated F1-scores and the test F1-score for several architectural configurations of the NLU component. Specifically, both intent prediction and entity prediction performances seem to be on par. A detail worth of notice is that the validation score is significantly lower compared to the train and test one. The reason can be presumably found in the way 3-fold cross-validation works. By excluding 1/3 of the available training data, the model is no longer able to properly generalize, resulting in lower performances on the held-out validation set (*cv val* column of Table Table 1). On the contrary, when evaluating the test performances of the model trained over the entire training set (*test* column of Table 1), the performances highlight a better model. Thus, the amount of training data represents indeed the minimum amount of data assuring a good model, as aimed in Section 4.
Table 2 instead, presents the ablation study over different components of the CORE pipeline. Results show that the default configuration, yet with a properly tuned *max_history* hyper-parameter (the agent seems to benefit from a larger context), achieves satisfactory conversation prediction accuracy and action prediction F1-score.
Of course, several other configurations may be tested[11], along with measuring the performances under different ratios of training data. However, given the default configuration of Rasa performing fairly well, and given that the

[9] https://github.com/SimGus/Chatette
[10] https://rasa.com/docs/rasa/2.x/stories#checkpoints-and-or-statements
[11] https://rasa.com/blog/rasa-nlu-in-depth-part-3-hyperparameters/

Tokenizer	Featurizer	Intent Clf.	Entity Clf.	Intent Prediction			Entity Prediction		
				cv train	cv val	test	cv train	cv val	test
WhiteSpace	LexicalSyn. + CountVec.	DIET	DIET	0.99	0.83	0.99	0.98	0.88	0.99
WhiteSpace	DistilBERT	Sklearn	CRF	0.97	0.79	0.67	1.0	0.83	0.66
SpaCy (md)	SpaCy + CountVec.	DIET	DIET	0.99	0.79	0.99	0.96	0.87	0.99
SpaCy (md)	SpaCy + CountVec.	Sklearn	CRF	0.96	0.74	0.62	0.99	0.88	0.80

Table 1: Ablation study over different components of the NLU pipeline.

Policies			Max.	Action Pred.		Correct Stories	
Rule	TED	Memoization	History	train	test	train	test
✓	✓	✓	9	0.99	1.00	0.97	1.00
✓	✓	✓	3	0.99	0.95	0.97	0.33
✓	✓		9	0.92	0.98	0.69	0.77
✓			9	0.16	0.35	0.00	0.00

Table 2: Ablation study over different components of the CORE pipeline.

amount of training data already represent a lower bound as discussed previously, a more extensive tuning is not needed.

6.2. Extrinsic Evaluation

Before testing the system with real users with the goal of collecting feedback, the proposed system was deployed as an Amazon Alexa Skill. In total, 5 users were asked to interact with the agent by means of the Alexa's vocal module and to rate their interaction in a scale from 1 to 5. Specifically, the following questions were asked to users: *i)* "How likely are you to interact with the agent again?" with an average score of 3.4; *ii)* "Did you feel that the system properly understood your questions?" with an average score of 3.2; *iii)* "Were the answers appropriate?" with an average score of 3.5; *iv)* "How close was your interaction to a human-human interaction?" with an average score of 2.6. The results highlight an overall satisfaction with the ability of the system to answer to the posed questions. Nonetheless, the systems still presents some flaws which should be fixed by adding more data, better if from real conversations. In addition to that, users sometimes felt some discomfort when asking questions that were not supported by the system, leaving precious suggestions for an eventual expansion of the work. On the same vein, the limitedness of the supported companies represented another major concern of users. The Extrinsic Evaluation also shed light over common miss-spelling by Alexa, especially for finance-specific terms, like *shorting* being spelled as *short thing* or *shortening*. Those examples were added to the training data.

7. Technical Issues and Limitations

One of the major technical difficulties related to the development of this project was constituted by the Rasa's computational burden. Indeed, on my ASUS Vivobook laptop the command *rasa train* was taking several minutes for completion, even for the agent at the early stages. On the same vein, and arguably more severely, each turn during the interactive learning sessions was taking about 10 seconds, which resulted in a system difficult to be debugged. I later realized that this behaviour was influenced by the operative system in use. Indeed, by simply switching to Ubuntu, the time necessary for processing a single turn during the interactive session was greatly reduced. Unfortunately, the Ubuntu-based environment was provided in the form of a Virtual Machine, thus resulting in an even slower training time given the reduced set of hardware resources available. An additional source of discomfort was related to the fact that Rasa X was solely working on my Ubuntu virtual machine. All these issues constitute the motivations behind the *low-data regime* described in the previous Sections.

Another source of technical problems was related to the APIs. Financial data and financial news are an extremely valuable resource for many enterprises. However, different services provide free APIs access, yet with a limited number of requests. During the development of the system such limit was reached several times, especially during intense periods of debugging. So, unfortunately I can not ensure that in the case of an evaluation testing the API module will work as expected.

8. Conclusions

In this work a Rasa-based AI-powered conversational agent was developed for the financial domain. The system was tested by real users as being able to answer questions regarding the stock value of some pre-selected companies, along with searching for information and latest news. The *main task* that was highlighted for the system was to provide suggestions about which company to look for based on user preferences. As a side, the system is also able to interact in a multi-modal manner with the user by sending hyper-links and plots via a Telegram Bot. The evaluation showed that the proposed system exhibit fairly good performances, yet with some flaws that should be accounted before an eventual real deployment. The vocal interaction highlighted a non-robust Alexa's vocal module which in many cases miss-spells words, especially for financial-specific terms.

9. References

Bunk, T., Varshneya, D., Vlasov, V., and Nichol, A. (2020). Diet: Lightweight language understanding for dialogue systems.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. (2021). Zero-shot text-to-image generation.

Vlasov, V., Mosig, J. E. M., and Nichol, A. (2019). Dialogue transformers.