A Comparison of Natural Language Understanding Services to build a chatbot in Italian*

Matteo Zubani[1,2], Serina Ivan[1], and Alfonso Emilio Gerevini[1]

[1] Department of Information Engineering, University of Brescia, Via Branze 38, Brescia 25123, Italy
[2] Mega Italia Media S.P.A.,Via Roncadelle 70A, Castel Mella 25030, Italy
{m.zubani004,ivan.serina, alfonso.gerevini}@unibs.it

Abstract. All leading IT companies have developed cloud-based platforms that allow building a chatbot in few steps and most times without knowledge about programming languages. These services are based on Natural Language Understanding (NLU) engines which deal with identifying information such as entities and intents from the sentences provided as input. In order to integrate a chatbot on an e-learning platform, we want to study the performance in intent recognition task of major NLU platforms available on the market through a deep and severe comparison, using an Italian dataset which is provided by the owner of the e-learning platform. We focused on the intent recognition task because we believe that it is the core part of an efficient chatbot, which is able to operate in a complex context with thousands of users who have different language skills. We carried out different experiments and collected performance information about F-score, error rate, response time and robustness of all selected NLU platforms.

Keywords: Chatbot · Cloud platform · Natural Language Understanding · E-learning.

1 Introduction

In the last decade more and more companies have replaced their traditional communication channels with chatbots which can satisfy automatically the users' requests. A chatbot is a virtual person that can talk to a human user using textual messages or voice.

Given the high demand for this technology, leading IT companies have developed cloud-based platforms which offer NLU engines and fast-developing tools,

* Supported by Mega Italia Media S.P.A

with the purpose to allow the user to build virtual assistants only providing a bunch of examples without knowing anything about NLU algorithms.

This paper comes from the need of the private company Mega Italia Media S.P.A., to implement a chatbot in their messaging system. We realised the lack of a systematic evaluation of cloud base NLU services for the Italian language, moreover research papers which present real cases of use of these systems do not explain why one service was preferred over another.

This research aims to evaluate the performance of leading cloud-based NLU services in a real context using an Italian dataset. Thus, we used users' requests provided by Mega Italia Media S.p.A collected through their chat system and we designed four different experiments in order to compare the selected platforms from different points of view such as the ability to recognise intents, response time and robustness to spelling errors.

2 Related Works

NLU algorithms can be used in complex and critical contexts such as the extraction and the classification of drug-drug interactions in the medical field [10][12], the classification of radiological reports [3][11] or named entity recognition (NER) task[9][8]. These works do not use commercial NLU platforms that are only appropriate for not critical and general-purposes like chatbots or Spoken Dialogue Systems (SDS).

Several publications exist which use the cloud-based platforms to create virtual assistants for different domains, e.g. [13] uses DialogFlow to implement a medical dialogue system, [4] presents a virtual teacher for online education based on IBM Watson and [5] shows the use of chatbots to the Internet of things. One of the architectural elements of these virtual assistants is a cloud-based NLU service. However, none of these papers discuss how they chose a particular service instead of another or propose an in-depth analysis concerning the performance of their system.

Since 2017 different papers have analysed the performance of the leading platforms on different datasets; e.g. [1] builds and analyses two different corpora and [2] presents a descriptive comparison (offering a taxonomy) and a performance evaluation, based on a dataset which includes utterances concerning the weather. While in 2019 [6] proposed a comparison concerning the performance of 4 NLU services on large datasets over 21 domains and 64 intents, and [14] presents a less extensive and deep comparison than what is shown in this paper on the same domain.

All the platforms are evolving year after year then also the performance could change compared to what previous papers presented. Furthermore, to our knowledge, a comparison of a dataset which is not in English does not exist; this pushed us to study if the performance in Italian is still the same as in latest research.

3 Natural Language Understanding Cloud Platforms

The principal goal of NLU algorithms is the extraction of useful and structured information from a natural language input which by its nature is unstructured. The structures obtained by the NLU service are two, *intents* and *entities*.

- **Intent:** the services should understand and classify what the user means in his sentence, which is not necessarily a request or a question but can be any user's sentence.
- **Entity:** instead of dealing with the overall meaning of the user's sentences, the entities extraction tries to identify information and parameter values inside the sentence itself.

Watson (IBM): It is a complete NLU framework which allows developing a chatbot using a web-based user interface or to exploit different SDKs, in order to build a virtual assistant fast and with different programming languages. The platform is easy to integrate with a variety of communication services and provides complete support to Text-to-Speech and Speech-to-Text technology. Watson allows recognising intents and entities, and some models concerning general topic are provided by the platform, while for specific domains the users have to create custom models giving a quite small set of examples to train the NLU engine. The context allows, once intents or entities are recognised, to store data and to reuse them in following dialogue interactions. A Dialogue frame is developed through a tree structure which allows designing deep and articulate conversation.

Dialogflow (Google): Previously known as Api.ai it has recently changed its name into Dialogflow and it is a platform which develops virtual textual assistants and quickly adds speech capability. This framework creates chatbots through a complete web-based user interface, or for complex projects, it uses a vast number of APIs via Rest (there are also many SDKs for different programming languages). Different intents and entities can be created by giving examples. In Dialogflow the context is an essential component because the data stored inside it allow developing a sophisticated dialogue between the virtual assistant and the user.

Luis (Microsoft): Similar to the previous platforms, Luis is a cloud-based service available through a web user interface or the API requests. Luis provides many pre-built domain models including intents, utterances, and entities, furthermore, it allows defining custom models. The management of dialogue flow results to be more intricate than other products. However, it is possible to build a complete virtual assistant with the same capabilities offered by other services. Different tools, which facilitate the development of a virtual assistant, are supported (such as massive test support, Text-to-Speech and sentiment analysis).

Wit.ai (Facebook): The Wit.ai philosophy is slightly different compared to other platforms. Indeed, the concept of intent does not exist, but every model in Wit.ai is an entity. There are three different types of "Lookup Strategy" that distinguish among various kinds of entities. *"Trait"* is used when the entity value is not inferred from a keyword or specific phrase in the sentence. *"Free Text"*

is used when we need to extract a substring of the message, and this substring does not belong to a predefined list of possible values. *"Keyword"* is used when the entity value belongs to a predefined list, and we need substring matching to look it up in the sentence. Wit.ai has both a web-based user interface and APIs furthermore there are several SDKs.

4 Case of Study

The Italian company Mega Italia Media S.P.A. acting in the e-learning sector is the owner of DynDevice, a cloud-based platform which provides online courses about "occupational safety". Every day they supply courses to thousands of users. Therefore, they want to introduce an artificial assistant on their platform to respond to the frequently asked questions. Among different types of virtual assistants, they chose to implement a chatbot because they already have a messaging system on their platform and users are accustomed to using this kind of interface, but currently, only human operators can respond to users' requests. Aiming at suggesting the best cloud-based NLU service that fits the company requirements, we decided to analyse the performance of different platforms exploiting real data which the company has made available in the Italian language.

4.1 Data Collection

In order to undertake the study proposed in this paper, we use data provided by the company concerning the conversations between users and human operators occurred between 2018-01-01 and 2019-12-31. We used only data from the last two years because the e-learning platform is continuously developing, and also the requests of the users are evolving. Therefore, introducing an additional feature in the platform may produce a new class of requests from the users; moreover, bugs resolution can cause the disuse of one or more class of requests.

4.2 Data Classification

Obviously, NLU services have to be trained, thus it is necessary to create a training dataset which includes labelled data. The requests collected were not classified, so we had to label them manually. For the classification phase, we developed a simple program that randomly extracts 1000 requests and then shows them one by one to the operator who is assigned to the classification. The operator, through a command-line interface, assigns the label which describes better the intent of request and then saves the tuple $< request, label >$ in the dataset. All the sentences extracted from conversations are made anonymous, and we do not treat them in any other way such as removing special characters and punctuation marks or correcting spelling mistakes. This anonymising step is necessary to satisfy the privacy policy of the company.

It is necessary to highlight that 33.1% of the total amount of requests was rejected because they were meaningless or because inside of these requests there

were references to emails or phone calls occurred previously between user and operator. Therefore, only a human operator can solve this type of requests. Some meaningless requests occurred because some users wrote a text in the chat interface to try it, with no needs. All 669 classified requests are split into 33 different intents; the major part of these had just one or few instances associate to them. So, we selected only the most relevant intents among the entire set of detected intents. How we can see in table 1, the examples belonging to the subgroup of principal intents are 551, and they cover the 82.4% of all examples classified. A small part (37) of these are not compliant with the constraints imposed by platforms. The actual number of examples that can be used to build the training set and the test set is 514. In table 2 of section 5.1, we show the distribution of the 551 classified sentences over the selected intents which range from a minimum of 22 up to 190.

Table 1. Requests made to e-learning platform between 2018-01-01 and 2019-12-31

Total	Analysed	Meaningless	Classified	Classified in main intents	discarded	Used
5089	1000	331	669	551	37	514

Below, we present the English translation of two requests about the same intent *"Expired course"*, one simple and another one more complex in order to show the variety and the complexity of the sentences coming from a real case of study:
1. "Course expired; is it possible to reactivate it?"
2. "Good morning, I sent a message this morning because I can't join "Workers - Specific training course about Low risk in Offices", probably because it has expired and unfortunately I didn't realise it. What shall I do? Thank you"

5 Evaluation Experiments

We selected only services with Italian language support: Watson, Dialogflow, Luis, Wit.ai. We chose the free version for all NLU services analysed because there is only a constraint about the number of APIs calls that can be made daily or monthly, while the NLU engine capabilities are not limited. We tested the capability of recognising correctly the underlying intent of every single message sent by users, in the real case described in section 4. To evaluate the results as thoroughly as possible, we designed four different experiments. The experiment (A) aiming at evaluating the performance on the whole extracted training set, in other words, we use all the examples available to train the NLU platform. In experiment (B) we study the performance of the different systems at the increase of the number of examples provided to train the platforms, while the test set remains fixed. While the experiment (C) tests the response time of each platform. Finally in experiment (D) we test the robustness of the analysed services, so we built different test sets with an increasing number of misspellings contained in each example and then we calculate the error rate of every single test set.

5.1 Training And Test Sets

To carry out the first experiment (A), we use classified data divided by main intents. We extract the training set composed of 75% of entire examples collection and test set made of the remaining elements (which corresponds to 25% of the whole dataset). In table 2, we present an overview of how the different datasets are made up for each intent.

Table 2. Composition of training set and test set divided by intents

	I1	I2	I3	I4	I5	I6	I7	I8	I9	Total
Dataset	45	22	52	88	26	36	26	27	190	514
Training set	33	16	39	66	19	28	19	20	142	382
Test set	12	6	13	22	7	10	7	7	48	132

In experiment (B), we use a fixed test set, and we build nine training subsets of the training set previously defined for the experiment (A). The first training subset is created by extracting 10% of elements relating to every single intent of the initial training set. The second one comprises 20% of the examples of the initial training set, keeping all instances which are included in the first training subset. This process is repeated increasing by 10% of examples for each intent until we reach the entire initial training set. We show the composition of training subsets and test set in table 3.

Table 3. Number of elements for each training subset and test set

	DS 10%	DS 20%	DS 30%	DS 40%	DS 50%	DS 60%	DS 70%	DS 80%	DS 90%	DS 100%
Training subset	39	76	116	153	192	228	265	305	342	382
Test set	132	132	132	132	132	132	132	132	132	132

In order to carry out the experiment (C), we select the dataset built for the experiment (A). The training set remains unchanged while the test set is expanded replicating elements in it until the number of elements reaches 1000.

We select the dataset built for the experiment (A) also for the experiment (D). We use the training set as in experiment (A) to train the services; from the test set of experiment (A), we extracted the 25 examples which intents were recognised correctly by all the platforms and then we create 30 new different test sets: in the first new test set, we change randomly a character for each examples belonging to the test set. In order to build the second test set, we use the test set just create and we change another character randomly for each example belonging to the test set. In this way, we create new 30 test sets, where the last test set has for each example 30 characters changed compared to the initial test set.

5.2 Experimental Design

The number of examples for some intents is quite small, and it is not possible to build a k-Fold Cross-Validation model. Thus, we define ten different training sets and corresponding test sets with the structure illustrated in section 5.1 experiment (A), making sure to extract the examples randomly as an alternative to Cross-Validation. To evaluate if the differences between the results of the NLU platforms are statistically significant we use Friedman test and then we propose the pairwise comparison using Post-hoc Conover Friedman test as shown in [7].

To analyse Experiment (B), we take among previously mentioned datasets the one that has the F-score closest to the average of the F-score on all datasets, and we create nine training subsets, as reported in section 5.1.

For experiment (C) we use a training set of experiment (A) and we built a test set with 1000 elements (as mention in section 5.1). In order to analyse the response time of the different services we send to the NLU engines each example in the test set independently, and we collect the time lapses between sending the message and receiving the reply. We summarise the measured times in a report where one can find average, standard deviation, highest and lower response time. We repeat this experiment three times in different hours of the day because the servers are located in different places of the world and the load of each server may depend on the time when we perform the test.

In experiment (D) we chose a training set used in experiment (A) and we built 30 different test sets as reported in section 5.1. We examine all the test sets and for each one, we collect how many times the service recognises the correct intent and then we calculate the error rate on all test sets analysed.

We developed a Python application which uses the SDKs afforded by the owners of each platform with the aim to train, test and evaluate the performance of each service. Wit.ai supports Python programming language, however, the instructions set is limited and to overcome this deficiency, we wrote a specific code that allowed us to invoke a set of HTTP APIs. The program receives two CSV files as input, one for the training set and one for the test set, and then it produces a report. The application is divided into three modules completely independent that means each one can be used individually.

Training module: for each intent defined, all examples related to it are sent to the NLU platform, and then the module waits until the service ends the training phase.

Testing module: for each element in the test set, the module sends a message containing the text of the element. The application waits for the NLU service reply. All the analysed platforms produce information about the intent recognised and the confidence associated with it. The module compares the intent predicted and the real intent, and then it saves the result of the comparison in a file. We want to underline that the application sends and analyses every instance of test set independently. In this module, there is an option that allows testing the response time using a timer which is activated before sending the message and it is ended when the service reply arrives.

Reporting module: this is responsible for the reading of the file saved which

contains results of the test module and then calculates the statistical indexes, both for each intent and the entire model. The indexes calculated are Recall, Accuracy, F-score and Error rate. This module also allows saving a report about response time where one can find average, standard deviation, maximum and minimum response time.

6 Results And Discussion

As mentioned in section 5, we create 4 different experiments. In experiment(A) we studied the performance on the training set with all elements available, in experiment (B) instead, we evaluated the results with the increase of training set instances, in the experiment (C) we measured the response time of all selected platforms and in the experiment (D) we evaluated the robustness of services.

Experiment (A) What we expected in the first experiment is that the performance might not be uniform among different NLU platforms, but we supposed that the outcomes would be relatively constant on different datasets randomly built as described in section 5.1 (from here called D1, D2...D10).

Table 4 presents the performance for each service in terms of *Error rate* and *F-score* [3] and the last two columns provide the average and standard deviation on the ten different datasets.

Table 4. Results in terms of F-score (F1) and error rate (ER) for all ten datasets

		D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	AVG	σ
Watson	ER	0.136	0.144	0.091	0.121	0.114	0.174	0.152	0.136	0.167	0.144	0.138	0.023
	F1	0.858	0.851	**0.908**	0.878	**0.887**	0.822	**0.848**	0.863	0.833	0.851	0.860	0.024
Dailoflow	ER	0.129	0.144	0.144	0.106	0.121	0.152	0.159	0.106	0.136	0.144	0.134	0.017
	F1	**0.877**	**0.852**	0.856	**0.894**	0.875	**0.857**	0.844	**0.894**	**0.871**	**0.863**	**0.868**	**0.016**
Luis	ER	0.174	0.197	0.144	0.167	0.159	0.174	0.182	0.167	0.144	0.182	0.169	0.016
	F1	0.825	0.785	0.853	0.823	0.840	0.828	0.815	0.831	0.851	0.819	0.827	0.019
Wit	ER	0.227	0.273	0.280	0.273	0.265	0.235	0.311	0.258	0.273	0.318	0.271	0.027
	F1	0.778	0.716	0.692	0.708	0.706	0.757	0.671	0.743	0.718	0.656	0.715	0.036

As we had supposed the general performance among platforms is different. The average of F-score is roughly equal between Dialogflow and Watson over 0.86; however, the latter has a more significant standard deviation. Luis' results are slightly worse than the other two with 0.82, while Wit.ai is the worst with an average just above 0.71. So to confirm this, we can look at table 4 where the best F-score for each dataset is bold, and in no case, Luis or Wit.ai have managed to overcome Dialogflow or Watson. These considerations are evident in figure 1 (a), where the median of Dialogflow is the highest while that of Wit.ai is the

[3] We used python function f1_score belonging to the sklearn module. Being dataset unbalanced, we set the parameter "average" equal "weighted".

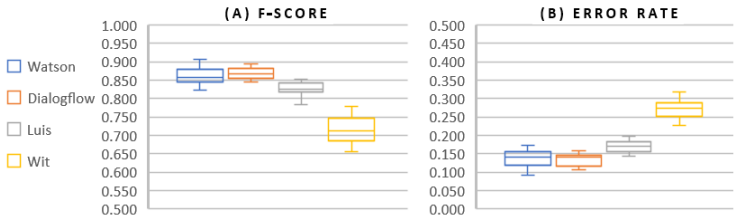

Fig. 1. Boxplots about (a) F-score, (b) Error rate for all platforms

lowest. We can see almost all the lengths of boxes are quite short, this indicates low dispersion, and it means that the performance of services is stable moving through the ten different datasets.

The number of examples associated with each intent is not balanced. Consequently, the performances on the single intent can be various, so in figure 2 we decided to break down the F-score outcome in every single intent. The legends show the intent ID and number of examples used to train the services. Looking at the diagrams, we notice that intent I9, which uses the highest number of elements, has one of the highest median and lower dispersion for all services. I2 is trained with the lowest number of examples, and its performances are quite variable, but in all platforms, the median is far below 0.8, and the boxes are very stretched. Dialogflow has better results in general, and the boxes are shorter than others, but there is an anomaly, in fact, I2 is significantly worse than Watson and Luis with a very high index of dispersion. The outcomes of Watson and Luis are pretty good; indeed, the median is always over 0.7. To confirm previous analysis on table 4 Wit.ai does not perform well, it has two intents under 0.6 and only two over 0.8, also for some intents the dispersion is very high.

Figure 3 presents the result of Post-hoc Conover Friedman test and it shows if the difference between results produced by all the NLU platforms analysed is significant with different p values. We can observe that with $p < 0.001$, Watson and Dialogflow perform better than Luis and Wit.ai and we can also assert that the difference between Dialogflow and Watson is not statistically significant with the same p value.

Experiment (B) In this experiment, we selected the first dataset DS1, and we split it into nine training subsets (as described in section 5.1), while the test set remains the same. We expected that increasing the size of the training set corresponds to an increase of performance until they reach the same F-score obtained on the entire training set. The graph (a) in figure 4 confirms our assumption that F-score of all four platforms starts low and then increases. What we can see on the same graph is that the curves of Watson, Dialogflow and also Wit.ai grow quite fast until 40% and then fluctuate or grow slowly, while Luis rises steadily up until it reaches its maximum. Watson and Dialogflow are significantly better than Luis and Wit.ai with small training sub-datasets which

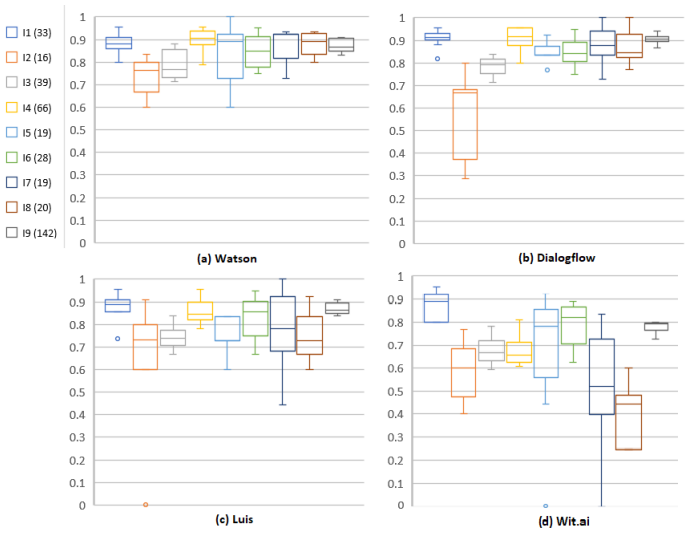

Fig. 2. Boxplots using F-score results on all ten different datasets divided by intents. (a) Watson, (b) Dialogflow, (c) Luis, (d) Wit.ai

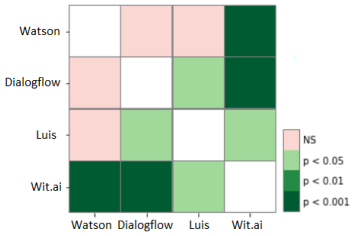

Fig. 3. Post-hoc Conover Friedman Test (NS represents not significant)

is showed by the graph (b) in figure 4 where the error rate, especially on the left, for the first two services is significantly lower than the other two.

Experiment (C) The response time of chatbots is generally not a critical value. In some specific applications such as a Spoken Dialogue Systems (SDS), the time of intent recognition task should be short so that increase the speaking experience when a user tries to use the SDS.

In this experiment, we want to understand if the response time is very different between the various platforms. It is not possible to place the servers in the same location, so we execute the test three times in three different moments of the day, then we calculate the average. In figure 5 each column is the average of the response times on the entire test set, while the yellow line represents the average of the three executions during the day. The results are expressed in milliseconds and we use Rome Time Zone (CEST) to express times. Luis is

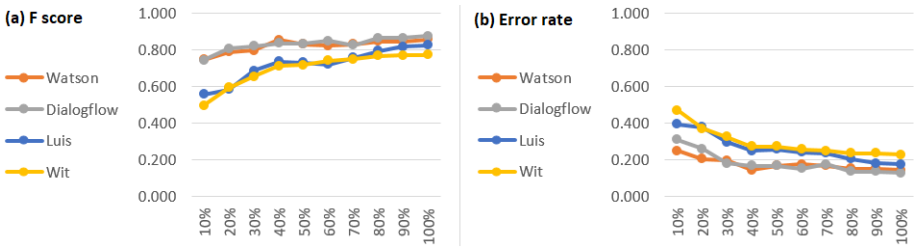

Fig. 4. Trend of (a) F-score and (b) Error rate for each platform while the size of the training set increases

the fastest platform while Watson and Wit.ai are the slowest. Watson presents similar response times during the day. Wit.ai's performance seems to suffer some excessive servers load during particular times of the day, in fact, it has excellent results in the experiment performed at 11 p.m. while in the remaining two tests it shows the slowest response times. It should be noted that all platforms have average response times for each moment of the day under 400 ms and this is a reasonable response time for a messaging system.

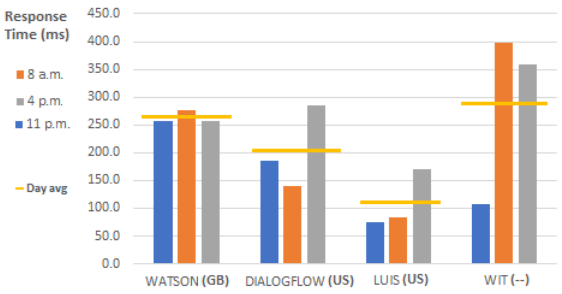

Fig. 5. Response time in three different moments of the day. The average of the day is indicated in yellow and the server location is next to the platform name.

Experiment (D) We described in section 5.1 how the test sets for this experiment are constructed. We want to evaluate the spelling errors robustness of NLU services; in order to do so, we provide 30 datasets with an increasing number of errors inside of the examples.

As shown in figure 6 Watson is the most robust platform, it maintains an error rate below 0.1 with less than 10 misspellings and error rate below 0.4 with 30 spelling errors. The other three platforms have similar trends. Luis maintains an almost constant error rate close to 0.5 when the spelling errors are between 20 and 30. Wit.ai and Dialogflow appear to be the least robust platforms in this experiment, in fact, their error rate exceeds 0.6.

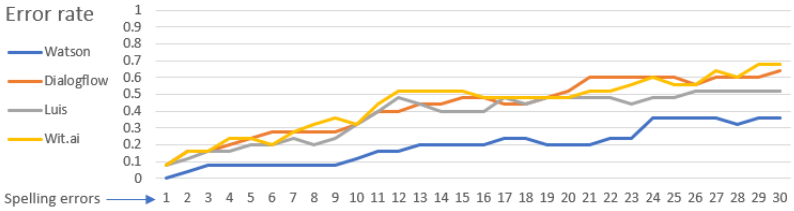

Fig. 6. Error rate of all platforms while the number of spelling errors increases

7 Conclusion

In this paper, we present four different experiments which allow comparing from different points of view the ability of different cloud based NLU platforms to recognise the underlying intent of sentences, belonging to an Italian dataset, coming from a real business case. The idea that pushed us to compare various services is to create the best chatbot which fit the company's needs. The first step in order to implement a chatbot is to choose the best platform on the market through an accurate and severe comparison.

In our analysis, Dialogflow shows the better overall results, however, Watson achieves similar performance and in some cases, it overcomes Dialogflow. Luis also has good performance, but in no case, it provides better results than services already mentioned. In our experiment, Wit.ai provides the worst results, and we cannot rule out this is due to the language used. Experiment (B) presents a quite impressive result, and that is Watson and Dialogflow achieve excellent outcomes, using just 40% of the whole training set. Experiment (C) shows that Luis is the fastest platform to recognise the intent associated to an input sentence. All platforms reply with an average of response times less than 400ms, this can be considered acceptable for a chat messaging system. In experiment (D) we notice that Watson is the most robust service when the sentences contain spelling errors.

To build a chatbot able to answer questions in Italian on an e-learning platform, the best NLU services are Watson and Dialogflow. In the specific case of Mega Italia Media S.p.A, the users of their e-learning platform are students with different language skills and someone has just a basic level or little knowledge of the Italian language. In this context, Watson is probably the best choice because it is the most robust service and the performance difference in term of F-score is not statistically significant compared to Dialogflow as shown in section 6.1.

As future work, we plan to increase the entire dataset size and add more intents, aiming to build a more deep and robust analysis. We plan to define a deep linguistic analysis of our dataset and a comparison of performance on other domains. And finally, to have a complete performance evaluation, we want to analyse not only the ability to recognise intents but also the ability to identify entities.

References

1. Braun, D., Hernandez Mendez, A., Matthes, F., Langen, M.: Evaluating natural language understanding services for conversational question answering systems. In: Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue. pp. 174–185. Association for Computational Linguistics, Saarbrücken, Germany (Aug 2017)
2. Canonico, M., Russis, L.D.: A comparison and critique of natural language understanding tools. pp. 110–115. CLOUD COMPUTING 2018 : The Ninth International Conference on Cloud Computing, GRIDs, and Virtualization (2018)
3. Gerevini, A.E., Lavelli, A., Maffi, A., Maroldi, R., Minard, A., Serina, I., Squassina, G.: Automatic classification of radiological reports for clinical care. Artif. Intell. Medicine **91**, 72–81 (2018). https://doi.org/10.1016/j.artmed.2018.05.006
4. Goel, A.K., Polepeddi, L.: Jill watson: A virtual teaching assistant for online education (2016)
5. Kar, R., Haldar, R.: Applying chatbots to the internet of things: Opportunities and architectural elements. International Journal of Advanced Computer Science and Applications **7**(11) (2016)
6. Liu, X., Eshghi, A., Swietojanski, P., Rieser, V.: Benchmarking natural language understanding services for building conversational agents. CoRR **abs/1903.05566** (2019)
7. Mehmood, T., Gerevini, A., Lavelli, A., Serina, I.: Leveraging multi-task learning for biomedical named entity recognition. In: Alviano, M., Greco, G., Scarcello, F. (eds.) AI*IA 2019 - Advances in Artificial Intelligence - XVIIIth International Conference of the Italian Association for Artificial Intelligence, November 19-22, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11946, pp. 431–444. Springer (2019). https://doi.org/10.1007/978-3-030-35166-3_31
8. Mehmood, T., Gerevini, A., Lavelli, A., Serina, I.: Multi-task learning applied to biomedical named entity recognition task. In: Bernardi, R., Navigli, R., Semeraro, G. (eds.) Proceedings of the Sixth Italian Conference on Computational Linguistics, Bari, Italy, November 13-15, 2019. CEUR Workshop Proceedings, vol. 2481. CEUR-WS.org (2019), http://ceur-ws.org/Vol-2481/paper47.pdf
9. Mehmood, T., Gerevini, A.E., Lavelli, A., Serina, I.: Combining multi-task learning with transfer learning for biomedical named entity recognition. Procedia Computer Science **176**, 848 – 857 (2020), knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 24th International Conference KES2020
10. Putelli, L., Gerevini, A., Lavelli, A., Serina, I.: Applying self-interaction attention for extracting drug-drug interactions. In: Alviano, M., Greco, G., Scarcello, F. (eds.) AI*IA 2019 - Advances in Artificial Intelligence - XVIIIth International Conference of the Italian Association for Artificial Intelligence, November 19-22, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11946, pp. 445–460. Springer (2019). https://doi.org/10.1007/978-3-030-35166-3_32
11. Putelli, L., Gerevini, A.E., Lavelli, A., Olivato, M., Serina, I.: Deep learning for classification of radiology reports with a hierarchical schema. Procedia Computer Science **176**, 349 – 359 (2020), knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 24th International Conference KES2020
12. Putelli, L., Gerevini, A.E., Lavelli, A., Serina, I.: The impact of self-interaction attention on the extraction of drug-drug interactions. In: Bernardi, R., Navigli, R., Semeraro, G. (eds.) Proceedings of the Sixth Italian Conference on Computational Linguistics, Bari, Italy, November 13-15, 2019. CEUR Workshop Proceedings, vol. 2481. CEUR-WS.org (2019), http://ceur-ws.org/Vol-2481/paper61.pdf

13. Rosruen, N., Samanchuen, T.: Chatbot utilization for medical consultant system. 2018 3rd Technology Innovation Management and Engineering Science International Conference (TIMES-iCON) pp. 1–5 (2018)
14. Zubani, M., Sigalini, L., Serina, I., Gerevini, A.E.: Evaluating different natural language understanding services in a real business case for the italian language. Procedia Computer Science **176**, 995 – 1004 (2020), knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 24th International Conference KES2020