Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 176 (2020) 995–1004

Procedia Computer Science

www.elsevier.com/locate/procedia

24th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

Evaluating different Natural Language Understanding services in a real business case for the Italian language

Matteo Zubani[a,b,*], Luca Sigalini[b], Ivan Serina[a], Alfonso Emilio Gerevini[a]

[a]*Department of Information Engineering, University of Brescia, Via Branze 38, Brescia 25123, Italy*
[b]*Mega Italia Media S.p.A.,Via Roncadelle 70A, Castel Mella 25030, Italy*

Abstract

In the last decade, the major private IT companies have developed lots of cloud platforms for Natural Language Understanding (NLU), that are widely used for research and commercial purposes. One of the main reasons for the success of NLU platforms is because they allow to simplify the Chatbots or Spoken Dialogue Systems (SDS) development and, in many cases, without knowledge about programming languages.

In this paper, we present a general description and a taxonomy that brings together features and constraints of different cloud-based NLU services available on the market. Furthermore, we provide an evaluation and a comparison concerning the ability to recognise the underlying intents of different sentences. The sentences used are a collection of requests made by users in Italian on an e-learning platform. This analysis wants to help the company, owner of the e-learning platform, to understand in which NLU platform it is better to build a chatbot for the Italian language aiming at answering the most frequently asked questions.

Keywords: Chatbot; Cloud platform; Natural Language Understanding; E-learning;

1. Introduction

Over twenty years, the study concerning NLU has been intensive, and it has led to remarkable results. In the last five years there has been a growing use of this research in private business, and one of the most notable examples is the development of "chatbots". A chatbot is a program that allows to interact with users via textual methods, in order to provide the correct answers to users' questions through artificial intelligence.

The core of a chatbot is the NLU engine which allows to deduce the underlying intents of users' utterances, regardless of the way they express them. An NLU engine allows to recognise two different concepts in a statement: an *intent* which is a mapping between the user's utterance and a specific class that allows the virtual assistant to decide

* Corresponding authors reference:
E-mail address: m.zubani004@unibs.it

1877-0509
10.1016/j.procs.2020.09.095

which answer or action should be chosen, and the *entities* which represent essential information (such as locations, times and roles). The combined use of intents and entities allows to build a detailed and precise response to satisfy the users' needs.

Given the high demand for this type of technology, leading IT companies have developed cloud-based platforms which offer NLU engines and fast-developing tools, with the purpose to build virtual assistants without knowing anything about NLU algorithms, and the user only has to provide a bunch of examples.

The aim of this paper is to evaluate the performance of leading cloud-based NLU services in a real context. Thus, we used users' requests collected through the chat service provided by Mega Italia Media S.p.A. The utterances were not treated; therefore, we did not remove any special character, punctuation or accents; we just focused on anonymising the sentences by excluding personal names and sensitive data. Furthermore, we want to investigate the performance using a dataset collected in a language which is not English (in our case Italian) and to understand if the results remain the same as in other related works presented in section 2.

The paper follows this structure: in section 2 we analyse related works while in part 3 we describe the platforms taken into account and we suggest a taxonomy to be able to compare them quickly, from a technical point of view. In section 4, we describe the case of study, where the data came from, and how we classify and define the datasets. Then in chapter 5, we present how the analysis was conducted, and in section 6, we show the results. Finally, in section 7, we put forward our conclusion and future works.

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 [8] or the classification of radiological reports [3]. These works do not use commercial NLU platforms that are only appropriate for not critical and general-purposes like chatbots or SDS. Indeed, several publications exist which use the cloud-based platforms to create virtual assistants for different domains, e.g. [9] 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.

All the platforms are evolving year after year then also the performance could evolve compared to what was presented in previous papers. 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 a foreign language (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. In general, 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 user's sentences, the entities extraction tries to identify information and parameter values inside the sentence itself.

To better clarify this concept, figure 1 shows the output of IBM Watson after providing the sentence "Do you want to go out tomorrow?" as input. In figure 1, labelled with number 1, the platform identifies the intent "hang_out", and this is the general meaning of the utterance inserted by the user. The platform recognises the intent correctly because it

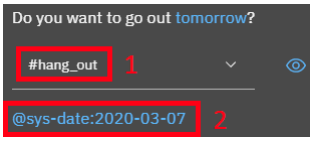

Fig. 1: Output of a NLU elaboration with IBM Watson

was trained previously with a few labelled examples. In point 2 of figure 1, the NLU engine finds the word "tomorrow" which is recognised like entity, and its type is "date". The platform automatically provides the conversion from the word "tomorrow" to an accurate date in a structured format. Below, we describe the main characteristics concerning five of widely used NLU platforms on the market.

Watson (IBM): This platform became famous in 2011 because it won a competition against the human champions of the jeopardy game. Watson is a complete NLU framework which permits to develop 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 to recognise 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 to design deep and articulate dialogues. Watson allows to define different skills, and each of them recognise just a group of specific intents and entities; skills can be connected and disconnected aiming at adding or removing the virtual assistant capabilities in different domains.

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, uses 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 to develop 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 to define custom models. The management of dialogue flow results to be more intricate than for 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): Facebook bought the startup Wit.ai in 2015 and allowed it to be used freely by the users. 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. Therefore, it is the correct choice if we want to create entities with the same meaning of intents present in other platforms.*"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 API support; furthermore, there are several SDKs, but at present, they cover only a small features subset of API.

Lex (Amazon): We want to mention Lex, which was not taken into account in this paper analysis because, at the time of producing our research, the framework supports just the English language.The technology behind Lex is the same as Amazon Alexa, it uses the intent idea like other products, but the entities are called slots. The platform allows to create complex dialogue flow which can be integrated with vocal assistants or instant messaging systems.

3.1. NLU Taxonomy

In table 1, we present a taxonomy which shows an overview from a technical point of view about the limits of different platforms. Two types of limits exist, the first is general, and it is the most evident, e.g. supported languages or SDKs available. The second type of constraints involves the vital elements of NLU service such as intents and entities, e.g. the maximum length of examples provided or the maximum number of intents (or entities) recognisable. The latter type of constraints (on intents and entities) is significant because when datasets are created in a real context, we have to be sure that the elements inside the training set and test set are compliant with constraints imposed by the platforms. If our goal is to create a virtual assistant, probably, we want to take care of other parameters like the presence of pre-built entities and intents in different domains, this too is present in the taxonomy.

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 use 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 in the Italian language, which the company has made available.

Below, we show two examples of requests about the same intent, 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. "Corso scaduto è possibile riattivarlo?" - "Course expired; is it possible to reactivate it?"
2. "Buongiorno ho inviato stamattina un messaggio perche non riesco ad accedere al corso "Lavoratori - Formazione specifica Basso rischio Uffici" probabilmente perchè è scaduto e purtroppo non me ne sono reso conto. Cosa posso fare? Grazie" - "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"

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 decided to use only data from the last two years because the e-learning platform is continuously evolving, and also the requests of the users are evolving. Therefore, the introduction of a new 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, and consequently, it is necessary to create a training dataset which included labelled data. The requests collected were not classified, so the data have been labelled 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. Through the program mentioned above, when the text of request shown to operator contains sensitive data such as the name of the person who makes the request, the operator has to replace the data with another one that does not allow to identify who made the request.

It is necessary to highlight that 33.1% of the whole 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

Table 1: Taxonomy for all NLU cloud platforms examined

Platform	Languages	SDKs Supported	Pre-trained intents	Pre-trained entities	Intent limits	Entity limits	Utterance limits
Watson	**12**	**10** (Android, Go, Java, Node.js, Python, Ruby, .NET, Salesforce, Swift, Unity) and HTTP API	**95** Intents concerning **7** different domains	From **10** to **150** contextual entities (It depends on the plan subscribed)	**Number intents:** From 100 to 2,000 intents for each skill (It depends on the plan subscribed) **Maximum number of examples:** for each skill is 25,000 **Minimum number of examples:** recommended at least 5 examples for each intent **Maximum example length:** 1024 characters	**Number entities:** from 25 to 1000 (It depends on the plan subscribed) **Values and synonyms:** 100,000 values and 100,000 synonyms for each skill **Maximum value length:** 64 characters **Maximum synonym length:** 64 characters **Patterns limitations:** Maximum 5 different patterns limited at 512 characters for each entity	**Maximum string length:** 2048 characters **Characters limitation:** This string cannot contain carriage return, newline, or tab characters.
DialogFlow	**20** (32 incl. Dialects)	**7** (C#, Go, Java, Node.js, PHP, Python, Ruby) and HTTP API	**11** Follow-up intents **8** prebuilt agents which include specific intents	**60** in total but it depends on the language	**Number intents:** 2,000 **Maximum number of examples:** 2,000 for each intent (100,000 per agent) **Maximum example length:** 768 characters	**Number entities:** 250 **Maximum number of examples:** 30000 **Maximum example length:** 512	**Maximum string length:** 256 characters
Luis	**13** (15 incl. Dialects)	**5** (C#, Java, Go, Node.js, Python) and HTTP API	11 Prebuilt domains which include specific intents	**12** System prebuilt entities **11** prebuilt domains which include specific entities	**Number intents:** 500 per application (499 custom intents, and the required None intent) **Maximum number of examples:** 15,000 per application - there is no specific limit per intent **Maximum example length :** 500	**Number entities:** Parent: 50, child: 20,000 **Constraints of regular expression:** 20 entities and maximum of 500 characters per regular expression.	**Maximum string length:** 500 characters
Wit.ai	More than **100**	**3** (Python, Ruby, Node.js) and HTTP API	NO	More than **30** System entities	Do not exist the concept of intent	**Maximum value length :** 280 characters	**Maximum string length:** 280 characters
Lex	**1** (English)	**11** (Android, JavaScript, iOS, Java, .NET, Node.js, PHP, Ruby, Python, Mobile Web, React Native) and HTTP API	**9** categories which include specific intents	Alexa Skills Kit slot types + **8** built-in slot types	**Number intents:** 100 for each bot **Maximum number examples:** 1,500 for each intent **Minimum number of examples:** 1 for each intent **Maximum example length :** 200 characters **Maximum number of characters:** 200,000 character for all intents	**Number slots:** 50,000 **Values and synonyms:** 10,000 **Maximum example length:** 140 characters	**Maximum string length:** 1024 characters

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, without any 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 decided to select only the most relevant intents among the whole set of detected intents. How we can see in table 2, the examples belonging to the subgroup of principal intents are 551, and they cover the 82.4% of all examples classified. A small part (31) of these are not compliant with the constraints presented in table 1 and to conduct the experiment we had to discard them. The actual number of examples that can be used to build the training set and the test set is 514.

In table 3, we show the total of examples available for every single intent which range from a minimum of 22 up to 190.

Table 2: Requests made to e-learning platform between 2018-01-01 and 2019-12-31

Total requests	Analysed	Meaningless	Classified	Classified in main intents	Out of limits	Entered in dataset
5089	1000	331	669	551	37	514

Table 3: Classification of the elements according to their intent

I1	I2	I3	I4	I5	I6	I7	I8	I9
45	22	52	88	26	38	26	27	190

5. Evaluation experiments

As already mentioned in section 3, 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 evaluated the capability of recognising correctly the underlying intent of every single message sent by users, in the real case described in section 4. We decided to focus on intent recognition because we believe that 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. To evaluate the results as thoroughly as possible we split the experiment in two parts: (a) the first one aims to understand the performance on the whole extracted training set, in other words, we use all the examples available to train the NLU platform. (b) The second one studies the different systems performance at the increase of the number of examples provided to train the platforms, while the test set remains fixed.

5.1. Training and test sets

To carry out the first part (a) of the experiment, we use classified data divided by main intents (table 3). 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 4, we present an overview of how the different datasets are made up for each intent. For the second part of the experiment (b), we use a fixed test set, and we build nine training subsets of the training set previously defined for the case (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 consists of 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. The composition of training subsets and test set is shown in table 5.

Table 4: 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

Table 5: 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

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 (part 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 the second part (b) of the experiment, 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 (part b).

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 to respond. All the analysed platforms produce information about the intent recognised and the confidence associated to it. The module compares the intent predicted and the real intent, and then it saves the result of comparison in a file. We want to underline that the application sends and analyses every instance of test set independently.

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.

6. Results and discussion

As mentioned in section 5, we separated the experiment into two parts. In the first part we studied the performance on the training set with all elements available, in the second part instead, we evaluated the results with the increase of training set instances.

What we expected in the first part of the 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 6: Results in terms of F-score and error rate for all ten datasets created

		D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	**AVG**	**STD. DEV.**
Watson	Error rate	0.1364	0.1439	0.0909	0.1212	0.1136	0.1742	0.1515	0.1364	0.1667	0.1439	0.1379	0.0234
	F score	0.8575	0.8509	0.9079	0.8778	0.8874	0.8220	0.8479	0.8629	0.8325	0.8571	0.8604	0.024
Dailoflow	Error rate	0.1288	0.1439	0.1439	0.1061	0.1212	0.1515	0.1591	0.1061	0.1364	0.1439	**0.1341**	0.0173
	F score	0.8767	0.8515	0.8559	0.8939	0.8750	0.8567	0.8441	0.8935	0.8711	0.8629	**0.8681**	**0.0161**
Luis	Error rate	0.1742	0.1970	0.1439	0.1667	0.1591	0.1742	0.1818	0.1667	0.1439	0.1818	0.1689	**0.0159**
	F score	0.8254	0.7846	0.8527	0.8234	0.8397	0.8278	0.8150	0.8309	0.8508	0.8195	0.8270	0.0185
Wit	Error rate	0.2273	0.2727	0.2803	0.2727	0.2652	0.2348	0.3106	0.2576	0.2727	0.3182	0.2712	0.0271
	F score	0.7779	0.7164	0.6917	0.7083	0.7063	0.7566	0.6709	0.7432	0.7181	0.6557	0.7145	0.0355

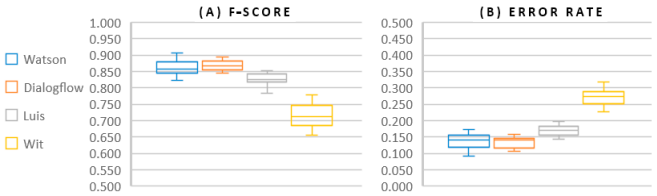

Fig. 2: Boxplots about (a) F-score, (b) Error rate for all platforms

Table 6 presents the performance for each service in terms of *Error rate* and *F-score* [1] on the ten different datasets and sideways there is average and standard deviation.

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 the table 6 where the best F-score for each dataset is underlined, and in no case, Luis or Wit.ai have managed to overcome Dialogflow or Watson. These considerations are evident in figure 2 (a), where the median of Dialogflow is the highest while that of Wit.ai is the 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 3 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 6 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 4 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.

In the second part of the 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

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

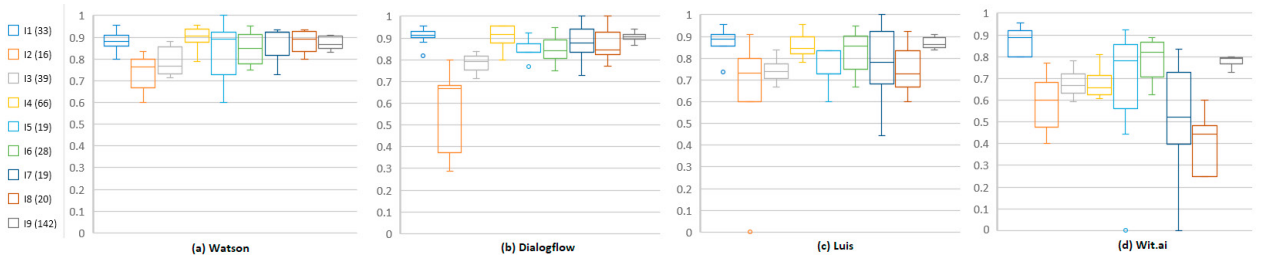

Fig. 3: Boxplots using F-score results on all ten different datasets divided by intents. (a) Watson, (b) Dialogflow, (c) Luis, (d) Wit.ai

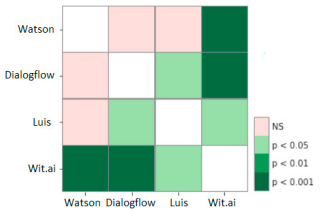

Fig. 4: Post-hoc Conover Friedman Test (NS represents not significant)

graph (a) in figure 5 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 is showed by the graph (b) in figure 5 where the error rate, especially on the left, for the first two services is significantly lower than the other two. Figure 6 presents how many intents are correctly identified (ok), how many are incorrectly identified (ko) and how many are not found (these elements are considered like incorrectly identified (ko) in Error rate calculus). Watson and Luis always try to provide a classification of intent, while Wit.ai and Dialogflow do not provide any classification under a certain threshold of confidence. The graph also shows a decrease of the number of "not found" while the training set size increases.

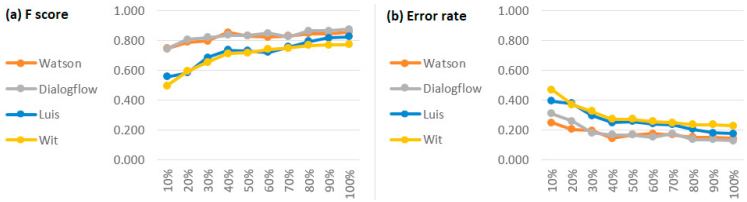

Fig. 5: Trend of (a) F-score and (b) Error rate for each platform while the size of the training set increases

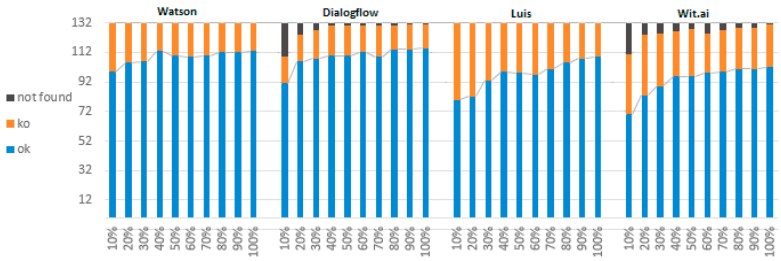

Fig. 6: Number of correct identifications, incorrect identifications and identifications not provided, while the size of the training set increases

7. Conclusion

In this paper, we present 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 that responds to the needs of the company, which provided us the data. The first step in order to implement a chatbot is to choose the best platform on the market through an accurate and severe comparison.

We have to take into account the constraints presented in table 1 because they affect the implementation choices. The limitations about the number of entities and intents that can be trained are similar among the platforms, and then they do not have much weight in the choice. The number of ready to use pre-trained intents models is useful to reduce the development time of the chatbot, only Wit.ai does not provide pre-trained intents models, and therefore it may not be the best choice in a project with limited time resources. The constraint that can mainly influence which platform should be chosen is the maximum length of text that can be submitted to the NLU engine. DialogFlow allows the processing of requests up to 256 characters, and this caused the rejection of 37 elements corresponding to 6.7% of the entire dataset. To address the limitation imposed by this constraint, during the development of a chatbot, the programmer has to build controls that limit the length of users' request. If working on long sentences is a requirement in a chatbot project, probably Watson is the best choice because it permits to elaborate sentences up to 2048 characters.

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. The second part of the experiment presents a quite impressive result, and that is Watson and Dialogflow achieve excellent results, with just 40% of the whole training set. To build a chatbot able to answer questions in Italian on an e-learning platform, the best NLU services are Watson and Dialogflow. We can not assert that one is certainly better than the other because the performance difference is not statistically significant.

In the specific case of Mega Italia Media S.p.A probably Watson is the best option because it allows to analyse longer requests than Dialogflow.

As future work, we plan to increase the entire dataset size and add more intents, to build a more deep and robust analysis. In order to have a complete performance comparison, 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., 2017. Evaluating natural language understanding services for conversational question answering systems, in: Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue, Association for Computational Linguistics, Saarbrücken, Germany. pp. 174–185.*

[2] *Canonico, M., Russis, L.D., 2018. A comparison and critique of natural language understanding tools, CLOUD COMPUTING 2018 : The Ninth International Conference on Cloud Computing, GRIDs, and Virtualization. pp. 110–115.*

[3] *Gerevini, A.E., Lavelli, A., Maffi, A., Maroldi, R., Minard, A., Serina, I., Squassina, G., 2018. Automatic classification of radiological reports for clinical care. Artif. Intell. Medicine 91, 72–81. doi:10.1016/j.artmed.2018.05.006.*

[4] *Goel, A.K., Polepeddi, L., 2016. Jill watson: A virtual teaching assistant for online education .*

[5] *Kar, R., Haldar, R., 2016. Applying chatbots to the internet of things: Opportunities and architectural elements. International Journal of Advanced Computer Science and Applications 7.*

[6] *Liu, X., Eshghi, A., Swietojanski, P., Rieser, V., 2019. Benchmarking natural language understanding services for building conversational agents. CoRR abs/1903.05566. arXiv:1903.05566.*

[7] *Mehmood, T., Gerevini, A., Lavelli, A., Serina, I., 2019. 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, Springer. pp. 431–444. doi:10.1007/978-3-030-35166-3_31.*

[8] *Putelli, L., Gerevini, A., Lavelli, A., Serina, I., 2019. 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, Springer. pp. 445–460. doi:10.1007/978-3-030-35166-3_32.*

[9] *Rosruen, N., Samanchuen, T., 2018. Chatbot utilization for medical consultant system. 2018 3rd Technology Innovation Management and Engineering Science International Conference (TIMES-iCON) , 1–5.*