Fine-Tuning Open Access LLMs for High-Precision NLU in Goal-Driven Dialog Systems

Lluís Padró, Roser Saurí

Computer Science Department
Universitat Politècnica de Catalunya
C. Jordi Girona 1-3 – 08034 Barcelona,
Catalonia, Spain
{lluis.padro, roser.sauri}@upc.edu

Abstract

This paper presents a set of experiments on fine-tuning LLMs to produce high-precision semantic representations for the NLU component of a dialog system front-end. The aim of this research is threefold. First, we want to explore the **capabilities of LLMs on real, industry-based use cases** that involve complex data and strict requirements on results. Since the LLM output should usable by the application backend, the produced semantic representation must satisfy strict format and consistency requirements. Second, we also want to assess the **language scalability** of the LLMs in this kind of applications; specifically, whether a multilingual model is able to cast patterns learnt from one language to other ones –with special attention to underresourced languages–, thus reducing required training data and computation costs. Finally, we want to evaluate **the cost-benefit of open-source LLMs**, that is, the feasibility of running this kind of models in machines affordable to small-medium enterprises (SMEs), in order to assess how far this organizations can go without depending on the large players controlling the market, and with a moderate use of computation resources. This work was carried out within an R&D context of assisting a real company in defining its NLU model strategy, and thus the results have a practical, industry-level focus.

Keywords: Large Language Models, Natural Language Understanding, Fine Tuning, NL assistants, Goal-Driven Dialog Systems, LLMs carbon footprint, Underresourced languages

1. Introduction

Many NLP applications demand a Natural Language Understanding (NLU) component able to transform language utterances into structured representations satisfying the requirements of the application backend. Some examples are database natural language interfaces, domotic assistants, voice-activated computer desktop assistants, and, in general, any goal-oriented dialog system beyond mere Q&A or recreational chatbots, aiming at helping the user to accomplish complex goals such as booking a flight or paying taxes. All these applications do not require a plausible textual response, but a highy precise set of complex arguments for the user intent (which taxes should be paid, from which bank account, which light at home should be turned off and when, which file should be moved to what folder and under what name, etc.)

In this study, we delve into a series of experiments on tuning Large Language Models (LLMs) for generating precise semantic representations within a dialog system. The research is structured around three primary objectives:

First, we investigate the potential of LLMs to handle complex, real-world scenarios in the industry. The aim is to ensure that the semantic outputs from the LLMs meet strict standards of format and consistency for seamless integration into application backends.

Secondly, we explore the scalability of LLMs across diverse linguistic landscapes, particularly their ability to support low-resourced languages. We aim to ascertain whether a multilingual model can transfer knowledge from well-resourced languages to those with fewer resources, thereby reducing the need for extensive training data and computational resources, favoring environmental and economic sustainability.

Lastly, a significant portion of our research is dedicated to evaluate the viability of leveraging open-source LLMs in a way that is economically and environmentally sustainable for small to medium-sized enterprises (SMEs). This involves exploring how these companies can use advanced language models without exacerbating environmental impacts or depending on large market-dominating corporations.

Conducted within a R&D framework aimed at assisting a start-up company in formulating its Natural Language Understanding (NLU) model strategy, this investigation offers insights with a practical, industry-oriented focus, highlighting the environmental impact and the challenge of inclusivity for low-resourced languages.

2. Background

Dialog systems, personal assistants, and other applications requiring precise understanding of user commands or queries have become ubiquitous in various sectors, including healthcare, customer service, and business, among others. NLU is a crucial component in these systems, responsible for transforming unstructured human language into a format that can be understood and processed by the application backend.

The recent launch of Large Language Models (LLMs) such as OpenAI GPT (OpenAI, 2023), Llama (Touvron et al., 2023), Falcon (Almazrouei et al., 2023), GPT-j (Wang and Komatsuzaki, 2021), GPT-neo (Black et al., 2021), Bloom (BigScience Workshop, 2022), or Mistral (Jiang et al., 2023), among others, has opened a large range of possibilities for all NLP applications. LLMs have shown to have powerful language “understanding” capabilities (Goldstein et al., 2023; Olney, 2023; Tsoutsanis and Tsoutsanis, 2024), being able to perform tasks such as entity recognition and classification (NERC), sentiment analysis, paraphrasing, summarization, or translation, with a quality close to human performance. Moreover, these models are able to generate syntactically (and often semantically) correct code in a variety of programming languages.

LLMs have been used as components in traditional NLP pipelines, proving able to perform NERC and relation extraction (Paolini et al., 2021; Ren et al., 2021), Semantic Role Labeling (Paolini et al., 2021), Coreference Resolution (Paolini et al., 2021), Event Extraction (Paolini et al., 2021; Du et al., 2021; Lu et al., 2021; Li et al., 2021), aspect-based sentiment analysis, or slot filling (Athiwaratkun et al., 2020; Rongali et al., 2020; Zhang et al., 2021). See (Min et al., 2023) for a detailed survey on the use of LLMs for NLP tasks.

However, using LLMs to perform partial NLP analysis has the same problems than traditional pipelines. On the one hand, the output is usually produced as annotated text, which requires a postprocessing step to integrate the relevant information. On the other, integrating the results from different stages into a unique semantic representation may lead to inconsistencies when outputs of different models are merged together.

LLMs’ natural environment is end-to-end tasks involving natural language in both the input and the output: machine translation, summarization, sentiment analysis, question answering, and, obviously, chatbots. However, the completion-like chatbots that LLMs can successfully produce are far from being able to satisfy the strict formatting and semantic constraints needed by the backend of goal-oriented dialog systems.

Existing research on LLMs has focused either on performing low-level NLP tasks (NERC, coreference resolution, parsing, slot-filling), or on high-level user-oriented language tasks (translation, summarization, information extraction, question answering, etc.), but fewer efforts have been devoted to making LLMs produce actionable output. Noteworthy approaches in this direction include the elaboration of plans to achieve a goal (Huang et al., 2022) or the translation of commands into API calls (Patil et al., 2023). In a line similar to the latter, we aim to use LLMs to produce structured complex semantic representations from text that are suitable to the requirements of an application backend in a real-world industrial scenario.

Although LLMs are able to generate code, and thus they can provide a well-formatted semantic structure for a sentence when requested to do so, the resulting structure will not necessarily match the constraints of the backend, neither the produced representation will be consistent between different requests. Yet, LLMs can be fine-tuned with a reasonable effort to produce, with high precision, a semantic structure matching the specifications of a dialog system or assistant backend. The tuned models (even “light” versions –with about 6B parameters) are able to create correct structures even for very complex utterances where any classical NLU pipeline would fail at some point.

3. Target Application

In this paper, we approach the usability of LLMs at the core of a user interface NLU component for an **office assistant** in charge of automating administrative and management tasks of different complexity degrees, like sending messages via various means (email, SMS, telegram, etc.), scheduling meetings, or managing files.

We focused on the 7 basic *intents* presented in Table 1 (intents i01 to i07). Most of them are instructions for *actions* for the system to perform (e.g., *send an email*), except for intents i01 and i04, which can only be *events* that the system must be sensitive to (e.g. in intent i01 the user cannot command the system to receive a message, but only to be aware of whether a message is received, i.e. as a trigger for some other action). On the other hand, intents i02 and i03 can be both *actions* for the system to perform and *events* to be sensitive to (e.g. the system can be instructed either to send an email message or to monitor whether the user does it herself).

We also experimented with composite intents (intents i08 and i09) and included also a set of random sentences to train the system to disregard unrelated content (intent i00).

4. Data

4.1. Semantic Representation

The JSON schema for the target semantic representation specifies: (a) the appropriate class for

ID	Intent	Process type	Example
Basic Intents			
i00	Intent non-related content	N/A	*They went looking for you several times. His brother came looking for us.*
i01	Receiving a message	Event	*An email message from pepe@gmail.com arrives to my outlook account with subject "invoice" and a PDF attachment.*
i02	Sending messages	Action	*Forward to my personal account any email from Lola arriving to my corporate account.*
		Event	*When anybody from my company replies with an attachment a message from a BSC employee*
i03	Creating calendar events	Action	*Invite Lola from BSC to a meeting called "weekly catchup" for every Wednesday at 9am, in office M3.*
		Event	*If I'm invited to a meeting on weekdays at 6pm on my Google Calendar*
i04	Scheduling a system action	Event	*Wait for 2 hours.*
		Event	*Every day at 3:30pm.*
		Event	*One week later.*
i05	Generating web forms	Action	*Create a form called "Personal information" asking for basic demographic and contact data, and email its URL to Lola.*
i06	Storing files	Action	*Store the new file in my cloud unit to folder MyDocs/customers/*
i07	Adding data to a spreadsheet	Action	*Add the values from the form fields "Name", "DOB" and "Email" as the last row in spreadsheet users-data.xslx*
Composite Intents			
i08	Combination of Intents i04 & i02	Event + Action	*Every Friday at 3pm, send a message to Lola with the file "summary.xls" attached and subject "weekly report"*
i09	Combination of Intents i03 & i02	Event + Action	*If Lola invites me to a meeting on Monday, morning, send her a message via Teams with text "Sorry, I can not make it".*

Table 1: Intents targeted by the NLU model

each intent, together with its relevant parameters; (b) the appropriate type for each of these parameters (string, integer, array, object); (c) any constraint on the possible values for these parameters (e.g., an integer value must be within a given range, an object value can only be of a certain class); and (d) the optionality for each parameter. The job of the NLU model is to identify the intent in the user utterance and convert it into a JSON structure compliant with the schema used by the assistant backend. For instance, Figure 1 shows the representation for the following sentence, which belongs to intent i09:

> *If Lola invites me to a meeting on Monday morning in room S1.207, send her a message via Teams with text "Sorry, I'm booked"*

The semantic representation for this sentence must be an object of class `CalendarEventAdded`, followed by an object of class `Send`. The former requires slots `process-type` and `event-object`. The latter is in turn instantiated by an object of class `CalendarEvent` with parameters `organizer`, `attendees`, `subject`, `location`, `start-time` (and optionally `end-time` and `duration`). Event parameters `organizer` and `attendees` are instantiated by objects of class `User`, and parameter `start-time` is instantiated by an object of class `TxSet`. The second `Send` object also has its own requirements on the expected parameters. Note how the coreference between *her* and *Lola* is resolved setting Lola as the recipient of the `TeamsMessage`.

Our schema manages 8 classes for modeling actions/events and 19 for entities of different sorts: messages, calendar events, users, forms, files, spreadsheet data, time expressions, etc. Some of those classes have also subclasses (e.g., class `Message` can be further specified into `EmailMessage`, `SMSMessage`, `TelegramMessage`, `TeamsMessage`, etc.). Finally, there are also a few classes that represent grammatical aspects of the

```
[{"class": "CalendarEventAdded",
  "process-type": "Event",
  "event-object": {
     "class": "CalendarEvent",
     "location": "room␣S1.207",
     "subject" : "_unknown",
     "attendees": [
         {"class": "User",
          "lemma": "me",
          "org-name": "_mine",
          "user-id": "_unknown",
          "user-name": "_me"
         }
     ],
     "organizer": {
          "class": "User",
          "lemma": "lola",
          "org-name": "_unknown",
          "user-id": "_unknown",
          "user-name": "Lola"
     },
     "start-time": {
          "class": "TxDateTime",
          "when": {
               "partofday": "MORNING",
               "weekday": "MONDAY"
          }
     }
   }
 },
 {
  "class": "Send",
  "process-type": "Action",
  "sent-object": [ {
     "class": "TeamsMessage",
     "subject": "Sorry,␣I'm␣booked"
     "recipient": [
         { "class": "TeamsUser",
           "lemma": "lola",
           "org-name": "_unknown",
           "user-id": "_unknown",
           "user-name": "Lola"
         }
       ],
     "sender": {
         "class": "TeamsUser",
         "lemma": ".implicit.",
         "org-name": "_mine",
         "user-id": "_unknown",
         "user-name": "_me"
     }
   }
  ]
 }
]
```

Figure 1: JSON representation for instruction: *If Lola invites me to a meeting on Monday morning in room S1.207, send her a message via Teams with text "Sorry, I'm booked"*

Process Classes			
None	663	ScheduledEvent	3758
ProcessCalEvent	3862	SendMessage	3971
ProcessSpreadsheet	579	SendForm	781
ReceiveMessage	1279	StoreFile	942
Entity Classes			
Attachment (ms)	1969	Form (fo)	781
CalendarEvent (ev)	3862	Message (ms)	6031
DatumField (sp)	300	DateTime (tx)	4195
DatumFrame (sp)	579	Duration (tx)	838
DatumLocation (sp)	579	DurationLen (tx)	838
DatumPosition (sp)	710	Set (tx)	3280
Field (fo)	1143	SetRepeat (tx)	3280
FieldValidation (fo)	1143	TxWhen (tx)	6807
File (fi)	4432	User (us)	19459
FileLocation (fi)	1316		
Classes for grammatical information			
CorefLocat (co)	926	CorefStep (co)	926
CorefObj (co)	1041	Cardinality (ca)	707

Table 2: Parent classes, their frequencies and the kind of information they encode. Legend: (ca) entity cardinality, (co) coreference, (ev) events, (fi) files, (fo) forms, (ms) messages, (sp) spreadsheet data, (tx) time expressions, (us) users.

utterance, namely coreference and entity cardinality. Table 2 presents the frequencies in the dataset of the top classes in the hierarchy.

Note that many intent parameters require a value that is an object, which, in turn, may also have parameters requiring other objects. Thus, the resulting semantic structures can be quite complex, with several nesting levels. As a result, the needed NLU model is multi-level: it not only must discern among the 8 types of basic intents –which would be a simple task for a classical ML classifier– but also to identify the relevant language fragments expressing their parameters, and properly combine the detected objects in compliance with the representation schema constraints.

4.2. Datasets

To train and test our models, we used sets of utterances expressing instructions for the intents presented above, together with their representation in JSON. That data is developed and owned by a startup company dedicated to build NLU-based office assistants. The dataset was created following a semi-automatic process that combines steps of manual curation and AI-based synthetic data augmentation, completed with a final phase for a fully manual check to ensure optimum data quality.

We used data in 3 different languages: Catalan, English, and Spanish. The number of total sentences used for each intent (train and test) is provided in Table 3. In addition, for evaluating the benefits of multilingual vs. monolingual models we also used a smaller subset with only English and Spanish data for intents i01 and i02. See Section

		Basic intents								Composite intents		
Language	Task	i00	i01	i02	i03	i04	i05	i06	i07	i08	i09	Total
Catalan	Train	0	56	71	58	704	37	33	112	46	56	1173
	Test	0	23	36	30	80	12	16	38	31	30	296
English	Train	270	529	597	625	829	308	397	252	507	481	4795
	Test	61	90	87	108	101	70	54	52	102	73	798
Spanish	Train	256	502	542	584	682	305	393	99	484	480	4327
	Test	76	79	107	107	90	49	49	26	99	107	789

Table 3: Number of sentences used for each intent and language (alphabetically sorted).

5.2 for a more detailed explanation.

5. Experiments and Results

We used the dataset described above to carry out different experiments aiming to shed light on three main questions: (1) whether LLMs are able to transform complex user instructions into a JSON semantic representation satisfying strict syntactic and semantic constraints required by the application backend, (2) whether a single multilingual model is better than tuning language-specialized models, (3) whether this multilingual model is able to process new languages with none or small data, and thus easing the support to under-resourced languages, and (4) whether existing open access LLMs are an effective alternative to existing proprietary LLM services, reducing the dependence on large models with large carbon footprint.
In all cases, JSON structures produced by the model where compared to a gold standard and evaluated both at the slot and sentence levels:

- For slots, we compute precision, recall, and F1. A slot is considered to be rightly extracted if it has the right value and it is in the right location inside the JSON structure.
- At sentence level, we compute the percentage of sentences with 100% accuracy (extracted JSON identical to the gold standard) and the percentage of sentences with an unusable output (non-parseable JSON).

The pre-trained LLMs that we analyzed include, on the one hand, five proprietary models owned by OpenAI: **Ada** (350M parameters), **Babbage** (1.3M parameters), **Curie** (6.7B), **Davinci** (175B) and **gpt-3.5-turbo** (20B)[1], and on the other hand, four open access LLMs: **GPT-j** (Wang and Komatsuzaki, 2021), **Falcon** (Almazrouei et al., 2023), **Mistral** (Jiang et al., 2023), and **Flor** (BSC, 2023)[2]. For all open access models, the version around 6-7 billion parameters was used.

[1]The size of gpt-3.5-turbo is not officially disclosed by OpenAI but it is assumed to be around 20B parameters.

[2]Flor is a Bloom version reinforced with additional Spanish and Catalan data.

5.1. Preliminary explorations

First trials involved using zero-shot and few-shot via prompting, where the model was asked to produce a JSON structure for a sentence after being given a few examples of the expected output.
As can be expected, the complexity of the required output structures and the variety of targeted intents is too wide for the models to grasp with only a few examples, and they behaved creatively with respect to which slots the JSON structure must contain and where to locate them, producing results unusable by the backend component.
Thus, fine-tuning was selected as the strategy to follow, since it allows to provide a larger number of examples and to adjust the model to the specific needs of the application.
Also, initial fine-tuning experiments with OpenAI proprietary models showed that Ada and Babbage had a performance under the minimum usability (under 70% F1 at slot level, under 50% sentences with perfect structure, over 10% sentences with invalid JSON output). Davinci had the best results, followed by Curie. Since the performance difference between them was under two percent points and Curie's economic cost was 10 times smaller, we chose Curie as our reference proprietary model. This allowed us to perform more thorough experimentation and to use larger tuning datasets. Later replacement of Curie and Davinci with gpt-3.5-turbo allowed as to include this newer model in the study.

5.2. Language Scalability

Firstly, we explored whether a multilingual model would be able to cast the patterns learnt from one language to another, or if instead a monolingual model for each target language was better. Table 4 shows the results of tuning different models for each target language versus tuning a single multilingual model. We ran that on the subset of English and Spanish data for intents i01 and i02.[3] We used Curie with 4 epochs, LR multiplier of 0.1, and default batch size (8).

[3]Since this piece of work was part of defining a language model strategy for the company, those were the only datasets available at that point.

Secondly, we explored if the inclusion of new languages into the system would benefit from the datasets for the already available languages, or would require extending the dataset. We speculated on such a feasibility due to the proximity of 2 of the languages involved: Spanish (already present in the multilingual model) and Catalan (the new language to be incorporated). We ran an experiment in which the new Catalan dataset was used only for testing, and a second one in which we split that dataset into 75% for training and 25% for testing. Results are presented in Table 5.

	P	R	F1	Perf	Fail
English					
Mono	88.0	88.1	88.1	38.4	0.0
Multi	92.4	92.6	92.5	42.0	0.0
Spanish					
Mono	91.6	91.2	91.4	37.9	0.7
Multi	92.3	93.0	92.6	38.6	0.0

Table 4: Results for fine-tuning with monolingual vs. multilingual models. Using English and Spanish data for intents 01 and 02.

	P	R	F1	Perf	Fail
No Catalan training data					
Catalan	88.0	87.2	87.6	28.0	0.4
English	94.8	94.7	94.8	61.6	0.0
Spanish	97.7	97.7	97.7	78.0	0.0
Some Catalan training data					
Catalan	96.5	96.8	96.6	61.2	0.0
English	95.2	95.4	95.3	63.4	0.1
Spanish	98.3	98.4	98.3	82.8	0.0

Table 5: Results of the multilingual model when fine-tuned only with English and Spanish data (top) or also including Catalan data – 11.4% of the total training dataset (bottom). Data for intents 01 and 02 is used.

5.3. Fine-tuning experiments

To compare proprietary and open access models, we tuned all of them with the same dataset and compared the results. Different combinations of learning rate, epoch number, and batch size were tried to select the best for each model.
Best overall results for each model are shown in Table 6. For each language, slot-wise precision/recall/F1 is reported, as well as percentage of perfect sentences and unusable JSON cases. Best parameterization for Curie is 4 epochs, 0.2 learning rate multiplier, and batch size 8. For the open access models, 2 epochs, 10^{-5} learning rate, and batch size 4 (for GPU memory limitations).
As shown in Table 6, Curie and Mistral obtain the best results. Curie is slightly better in English, and Mistral wins by a narrow margin in Catalan and Spanish. However, the difference is not statistically significant. The other open access models do not achieve the same performance and are all in a similar range of results.
It is noteworthy that despite being much larger than Curie and Davinci –and thus supposed to be a better model– gpt-3.5-turbo obtains results similar to those of the worst open access models. The reason seems to be that gpt-3.5-turbo is too oriented to chat and it tends to get too creative in the produced JSON structures and often fails to respect the output requirements. An second possibility could be that it has a stronger resilience to fine-tuning.
These results prove the ability of fine-tuned LLMs to produce strict constraint-compliant semantic representations of complex user utterances, therefore allowing to be used in an application backend such that for an advanced office assistant.
With regard to the usability of open access models, Table 7 shows performance results of the two best models (Curie and Mistral) detailed at intent level. Intent-wise, the differences are small in most intents, but in some cases (such as intents i03 and i04), there is a significant difference in either one or the other direction. Results for Spanish are better than for the two other languages because the Spanish dataset sentences are less complex.

6. Discussion

Given the results above, we can evaluate our initial questions: What are the capabilities of LLMs on real industry-based use cases requiring high precision NLU (Section 6.1); what is the model scalability to new languages (Section 6.2); and finally, what is the cost-benefit relation of comercial vs. open-source LLMs for SMEs (Section 6.3).

6.1. Usability of LLMs for high-precision NLU

As seen in Table 6, the average percentage of unusable output sentences (%Fail) per language is at most 0.3% for both Curie and Mistral; in fact, it is 0% for most intents in both cases. These are remarkably positive results considering the strict format required by the office assistant backend.
Moreover, the percentage of perfect sentences (%Perf) is reasonably acceptable, as it is around 65% for Catalan and English, and even in a much higher rate for Spanish: 79%. The fact that a sentence is not classified as perfect does not preclude the dialog system to process it. It just means that the JSON structure contains extra slots or misses some expected ones, which can often be managed by the backend or by the users interacting through the system's GUI. The difference between Spanish (79%) and Catalan and English (65-68%) has to do with the nature of the user sentences in our Spanish dataset, which are in general syntactically simpler and more homogeneous than the

	Catalan					**English**					**Spanish**				
model	**P**	**R**	**F1**	**Perf**	**Fail**	**P**	**R**	**F1**	**Perf**	**Fail**	**P**	**R**	**F1**	**Perf**	**Fail**
curie	96.3	95.7	96.0	65.2	**0.0**	**95.8**	**95.7**	**95.7**	62.5	**0.3**	97.5	97.2	97.3	78.6	**0.0**
gpt-3.5	90.4	88.0	89.2	58.4	7.8	93.6	90.4	92.0	51.3	4.6	95.5	94.0	94.7	71.2	3.7
Mistral	**96.7**	**96.7**	**96.7**	**67.9**	**0.0**	95.7	95.3	95.5	**65.2**	0.4	**97.6**	**97.7**	**97.7**	**79.3**	0.3
Falcon	89.6	89.5	89.5	39.5	0.3	92.5	92.3	92.4	48.7	0.8	95.4	95.2	95.3	67.3	0.4
GPT-j	90.6	89.9	90.2	42.6	0.3	92.5	92.6	92.6	51.5	0.8	95.2	94.7	94.9	66.5	0.6
Flor	95.6	94.5	95.1	61.8	1.4	94.1	89.3	91.6	53.6	3.5	96.4	90.1	93.1	70.5	4.3

Table 6: Results for different LLMs fine-tuned and evaluated on our target dataset. Curie and gpt-3.5-turbo are openAI proprietary models, and the rest are open access models. Columns P, R, F1 show slot-wise precision, recall and F1. Column *Perf* shows the percentage of senteces with perfect JSON. Column *Fail* shows the percentage of sentences with unusable ill-formed json.

Catalan										
	Curie					**Mistral**				
intent	P	R	F1	Perf	Fail	P	R	F1	Perf	Fail
intent 01	**94.9**	**94.9**	**94.9**	**73.9**	**0.0**	**94.9**	**94.9**	**94.9**	**73.9**	**0.0**
intent 02	95.4	95.4	95.4	66.7	**0.0**	**96.6**	**97.3**	**96.9**	**75.0**	**0.0**
intent 03	**98.0**	**97.0**	**97.5**	**80.0**	**0.0**	96.2	96.0	96.1	66.7	**0.0**
intent 04	93.5	94.2	93.9	70.0	**0.0**	**96.5**	**97.4**	**96.9**	**83.8**	**0.0**
intent 05	96.7	95.3	96.0	**58.1**	**0.0**	**97.7**	**97.3**	**97.5**	54.8	**0.0**
intent 06	97.1	96.3	96.7	43.3	**0.0**	**97.9**	**97.6**	**97.7**	**53.3**	**0.0**
intent 07	**99.6**	**99.6**	**99.6**	**91.7**	**0.0**	97.9	97.9	97.9	75.0	**0.0**
intent 08	98.0	**98.0**	**98.0**	68.8	**0.0**	**98.7**	96.3	97.5	81.2	**0.0**
intent 09	**94.4**	**94.6**	**94.5**	**50.0**	**0.0**	93.6	94.1	93.9	39.5	**0.0**
TOTAL	96.3	95.7	96.0	65.2	**0.0**	**96.7**	**96.7**	**96.7**	**67.9**	**0.0**

English										
	Curie					**Mistral**				
intent	P	R	F1	Perf	Fail	P	R	F1	Perf	Fail
intent 00	88.2	98.4	93.0	98.4	1.6	**100.0**	**100.0**	**100.0**	**100.0**	**0.0**
intent 01	**95.3**	**95.3**	**95.3**	**68.9**	**0.0**	91.1	91.1	91.1	58.9	**0.0**
intent 02	96.2	95.2	95.7	63.2	1.1	**97.7**	**97.7**	**97.7**	**70.1**	**0.0**
intent 03	91.9	92.5	92.2	47.2	**0.0**	**92.7**	**93.5**	**93.1**	**49.1**	**0.0**
intent 04	96.2	96.1	96.2	78.2	**0.0**	**98.2**	**98.7**	**98.4**	**89.1**	**0.0**
intent 05	**96.2**	**95.8**	**96.0**	**46.1**	**0.0**	95.5	94.2	94.9	42.2	2.0
intent 06	**97.9**	**97.7**	**97.8**	52.1	**0.0**	**97.9**	97.6	97.7	**58.9**	**0.0**
intent 07	**98.3**	**97.9**	**98.1**	**72.9**	**0.0**	98.1	96.8	97.5	71.4	1.4
intent 08	93.8	93.6	93.7	50.0	**0.0**	**94.3**	**94.0**	**94.1**	**57.4**	**0.0**
intent 09	96.3	95.8	96.1	55.8	**0.0**	**97.0**	**97.0**	**97.0**	**67.3**	**0.0**
TOTAL	**95.8**	**95.7**	**95.7**	62.5	**0.3**	95.7	95.3	95.5	**65.2**	0.4

Spanish										
	Curie					**Mistral**				
intent	P	R	F1	Perf	Fail	P	R	F1	Perf	Fail
intent i00	**97.4**	**99.1**	**98.3**	**98.7**	**0.0**	90.7	97.8	94.1	97.4	1.3
intent i01	97.9	97.8	97.8	82.3	**0.0**	**98.7**	**98.7**	**98.7**	**84.8**	**0.0**
intent i02	96.3	96.5	96.4	72.0	**0.0**	**96.4**	**97.2**	**96.8**	**75.7**	0.9
intent i03	98.0	**98.0**	**98.0**	**84.1**	**0.0**	**98.1**	97.9	**98.0**	**84.1**	**0.0**
intent i04	**98.7**	**98.5**	**98.6**	90.0	**0.0**	**98.7**	**98.5**	**98.6**	**91.1**	**0.0**
intent i05	97.4	96.9	97.1	**72.7**	**0.0**	**97.5**	**97.4**	**97.5**	70.7	**0.0**
intent i06	97.6	96.9	97.2	**65.4**	**0.0**	**98.0**	**97.5**	**97.8**	62.6	**0.0**
intent i07	98.9	**99.0**	98.9	**87.8**	**0.0**	**99.0**	**99.0**	**99.0**	**87.8**	**0.0**
intent i08	97.7	97.7	97.7	**77.6**	**0.0**	**98.3**	**98.3**	**98.3**	**77.6**	**0.0**
intent i09	92.0	94.5	93.2	34.6	**0.0**	**92.9**	**95.0**	**93.9**	**53.8**	**0.0**
TOTAL	97.5	97.2	97.3	78.6	**0.0**	**97.6**	**97.7**	**97.7**	**79.3**	0.3

Table 7: Results for Curie and Mistral on different languages. Columns P, R, F1 show slot-wise precision, recall and F1. Column *Perf* shows the percentage of senteces with perfect JSON. Column *Fail* shows the percentage of sentences with unusable ill-formed json.

sentences for the other two languages.

A qualitative analysis of the results shows that many of the errors concentrate in slots related to grammatical properties of the input sentences, such as properly identifying entity cardinality (e.g., *sending all the emails* vs. *3 emails* vs. *an email*) or representing coreference information (e.g. *the meeting that Pepa set up in the previous step* or *the email that I just sent*) so that the backend can retrieve the refered entity.

Another source of error involves time expressions (e.g., *within 2 hours*, *every Monday*, or *Wednesday at 15:30h*). Here, the most challenging language for both LLMs is Catalan, which suggests a scarce presence of Catalan time expressions in the pretraining data for both models.

Finally, a further area of error has to do with identifying named entities, both prototypical (people and organization names) and expressions such as names for Teams/Slack channel (e.g. *#dev-team*), folders (e.g., *MyDocuments/Invoices*) and drives (e.g., *the C unit*, *our cloud drive*, etc.).

6.2. Cross-language generalization

A second conclusion from our exploration is that the multilingual model takes advantage of cross-linguistic information, obtaining better results than the models tuned on single languages. Results in Table 4 show that the multilingual model yields an F1 between 1 and 4 points higher than separated monolingual models.

With regard to the extension to new languages, Table 5 (top) shows that the multilingual model delivers quite acceptable results for Catalan data when it is unseen in the fine-tuning data. However, these results for Catalan are still far from the great performance for English (around 94.7%) and especially from Spanish (around 97.7%), from which it should supposedly benefit the most, not to mention the poor score of only 28% Perfect parsed sentences for Catalan, as opposed to scores over 60% for the other two languages.

While it is obvious that the multilingual model is capable to generalize over a third unseen language, the advantage of the multilingual model over monolingual ones seems to be mainly due to the fact that it is fine-tuned with twice as much data. The benefit of larger datasets for fine-tuning can be also attested in the bottom half of Table 5. Note that the results for English and Spanish also slightly improve when an additional small dataset of Catalan training data is incorporated (containing 1173 datapoints, which amounts to only 11.4% of the multilingual training dataset).

6.3. Open-source LLMs as an alternative

Although the performance of Mistral in terms of output quality is comparable, or even slightly better than that of Curie, processing speed is another key issue to be considered when planning to develop an open access LLM-based app or service. Inference on Curie via OpenAI API runs at about 400 tokens/second, and processes one average utterance in 2.5 seconds, including network latency. By contrast, Mistral inference runs locally on a Nvidia RTX-3090 GPU (24Gb) at 17 tokens per second, with an average of 15 seconds per utterance, which is not suitable for real-time applications. However, the same Mistral model quantized to 4-bits, runs on the same RTX-3090 at a speed similar to that offered by OpenAI models, with a very small loss in performance, which definitely opens the door to in-house usage of open access LLMs in applications developed by start-ups and SMEs, enabling not only an economic cost reduction, but also a lighter carbon footprint.

7. Conclusions & Further Work

Our experiments point out that fine-tuned LLMs are a good choice for the NLU component of goal-driven dialog systems. Also, evaluated open access models are able to compete with proprietary models in output quality and speed. However, if a multi-user app or a SaaS application attending many customers simultaneously are envisioned, the cost of dedicated hardware may rise very fast and pay-per-use may be a cheaper option. Technological independence must also be taken into account. Big companies such as openAI not only have a larger carbon footprint, but also take strategic decisions that may negatively impact the performance of applications based on their models[4].

Finally, multilingual models can deal with unseen languages to an acceptable degree, although adding even a small amount of data for the new language contributes to an overall improvement.

Future lines of research include a wider exploration on quantization to increase speed and reduce carbon footprint, while maintaining as much quality as possible, as well as exploring new lighter open access models that may run locally or even in a phone or tablet (Google, 2023; Microsoft, 2023).

On the dataset front, we want to improve the degree of sentence heterogeneity, in particular concerning Spanish. A second line of data improvement has to do with incorporating more sentences displaying those features for which the models tended to performed the worst; in particular, entity cardinality, coreferences, and time expressions of different kinds. Last but not least, we plan to widen the range of supported intents by incorporating more office-related tasks, as well as to integrate a larger variety of languages.

[4]OpenAI recently deprecated Curie leaving gpt-3.5-turbo as the only available alternative, which in our case yields significantly worse results at a higher cost.

Acknowledgements

This research has been funded by the Spanish Science and Innovation Ministry, via grant TED2021-131257B-I00, supported by Next Generation European Funds. We thank the anonymous reviewers for their thorough and insightful comments.

8. Bibliographical References

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Merouane Debbah, Etienne Goffinet, Daniel Heslow, Julien Launay, Quentin Malartic, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. Falcon-40B: an open large language model with state-of-the-art performance. https://huggingface.co/tiiuae/falcon-40b.

Ben Athiwaratkun, Cicero Nogueira dos Santos, Jason Krone, and Bing Xiang. 2020. Augmented natural language for generative sequence labeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 375–385, Online. Association for Computational Linguistics.

BigScience Workshop. 2022. BLOOM (revision 4ab0472).

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow. https://doi.org/10.5281/zenodo.5297715.

BSC. 2023. FLOR-6.3B. https://huggingface.co/projecte-aina/FLOR-6.3B. Projecte AINA, Language Technology Unit, Barcelona Supercomputing Center. Barcelona, Spain.

Xinya Du, Alexander Rush, and Claire Cardie. 2021. Template filling with generative transformers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 909–914, Online. Association for Computational Linguistics.

Alon Goldstein, Miriam Havin, Roi Reichart, and Ariel Goldstein. 2023. Decoding stumpers: Large language models vs. human problem-solvers. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Google. 2023. Gemini. https://deepmind.google/technologies/gemini/.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. https://arxiv.org/abs/2310.06825.

Sha Li, Heng Ji, and Jiawei Han. 2021. Document-level event argument extraction by conditional generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 894–908, Online. Association for Computational Linguistics.

Yaojie Lu, Hongyu Lin, Jin Xu, Xianpei Han, Jialong Tang, Annan Li, Le Sun, Meng Liao, and Shaoyi Chen. 2021. Text2Event: Controllable sequence-to-structure generation for end-to-end event extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2795–2806, Online. Association for Computational Linguistics.

Microsoft. 2023. Phi-2: The surprising power of small language models. https://www.microsoft.com/en-us/research/blog/.

Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. 2023. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Comput. Surv.*, 56(2).

Andrew M. Olney. 2023. Generating multiple choice questions from a textbook: Llms match human performance on most metrics. *Workshop on Empowering Education with LLMs - the Next-Gen Interface and Content Generation at the AIED'23 Conference*.

OpenAI. 2023. Gpt-4 technical report. https://arxiv.org/abs/2303.08774.

Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Jie Ma, Alessandro Achille, Rishita Anubhai, Cicero Nogueira dos Santos, Bing Xiang, and Stefano Soatto. 2021. Structured prediction as translation between augmented natural languages. In *9th International Conference on Learning Representations (ICLR'21)*.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2023. Gorilla: Large language model connected with massive apis.

Liliang Ren, Chenkai Sun, Heng Ji, and Julia Hockenmaier. 2021. HySPA: Hybrid span generation for scalable text-to-graph extraction. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4066–4078, Online. Association for Computational Linguistics.

Subendhu Rongali, Luca Soldaini, Emilio Monti, and Wael Hamza. 2020. Don't parse, generate! a sequence to sequence architecture for task-oriented semantic parsing. WWW '20, page 2962–2968, New York, NY, USA. Association for Computing Machinery.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. `https://arxiv.org/abs/2302.13971`.

Panagiotis Tsoutsanis and Aristotelis Tsoutsanis. 2024. Evaluation of large language model performance on the multi-specialty recruitment assessment (msra) exam. *Computers in Biology and Medicine*, 168:107794.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. `https://github.com/kingoflolz/mesh-transformer-jax`.

Wenxuan Zhang, Xin Li, Yang Deng, Lidong Bing, and Wai Lam. 2021. Towards generative aspect-based sentiment analysis. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 504–510, Online. Association for Computational Linguistics.