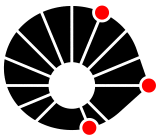
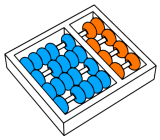




Active Learning for Natural Language Data Annotation

A. G. M. Purim *J. C. dos Reis*

Relatório Técnico - IC-PFG-23-55
Projeto Final de Graduação
2023 - Dezembro

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Active Learning for Natural Language Data Annotation

Andreis Gustavo Malta Purim Julio Cesar dos Reis

01.12.23

Abstract

Developing task-oriented conversational systems requires substantial annotated data, posing a challenge in Natural Language Processing (NLP). Manual annotation is time-consuming and error-prone, hindering progress for smaller AI teams. This work presents a novel Dialog Annotation Methodology and a ready-to-use adaptable software tool offering automatic annotation. The automatic annotation model is based on a cascade of Machine Learning and Large Language Model annotation to annotate entities and intentions in natural language dialogs.

1 Introduction

The Natural Language Processing (NLP) is an area of research in Artificial Inteligence (AI) and computer science focused on the processing of natural human languages, such as Portuguese or English. This processing typically involves translating natural language into numbers that computers can use to interpret it [1]. In this context, language models are often applied to interpret written text. Therefore, NLP can be understood as an aspect of AI that helps computers understand, interpret, and use human languages. NLP enables computers to communicate with people using human language and allows computers to read text, listen to speech, and interpret it [2].

NLP is used to understand the structure and meaning of human language by analyzing different aspects such as syntax, semantics, pragmatics, and morphology. Computer science then transforms this linguistic knowledge into rule-based machine learning algorithms that can solve specific problems and perform desired tasks [3]. Language models quantify the probability of a sequence in a text, allowing them to compute the likelihood of a sequence of words in a text being plausible. Such models have various applications, such as automatic translators, speech recognizers, and dialogue systems [4]. In addition to the mentioned applications, these models can be used for constructing datasets through textual annotation.

In particular, virtual customer service through AI chatbots is a rapidly growing area for businesses [5]. It requires large amounts of structured dialogue data [6]. However, annotating this data is typically one of the most exhaustive, time-consuming, and financially demanding tasks for companies and research teams.

Generating these annotated datasets can involve human annotators (an assisted approach) or be done automatically based on data characteristics (an unassisted approach). If done by more than one person, it can result in disagreements in the annotation pattern (e.g., annotating an entity or intention) [7]. Additionally, studies indicate a lack of precise guidelines adapted to the peculiar human-AI nature of conversation system interaction [8], resulting in discrepancies and inconsistencies [9].

This study emphasis the annotation of Entities and Intentions. An entity refers to a specific object, person, location, or concept that is recognizable and distinguishable within a given tex - while intentions refer to the underlying purpose or goal behind a particular statement or action. For instance, in a customer service chatbot, identifying the entity “product” and the intention “purchase” would enable the system to provide relevant assistance when a user wants to buy a specific item.

Existing tools that support annotation are not user-friendly, often requiring technical knowledge, lacking quick adaptability, and having non-intuitive interfaces [10]. They rarely support structured data like a conversation between two or more agents. Moreover, there is no standardization and interoperability between data structures among tools, and there is no ease for cooperative annotations (one or more users annotating the same data simultaneously).

Previously, we introduced a new online dialogue annotation tool called Assis [11], aiming for ease of use and quick adaptability and integration with data generation tools. The main contribution of this current research is expanding the tool into being capable of fully automatic annotation, reducing the main bottleneck in annotation time. The system assisted in annotation in previous versions, but the human operator was still responsible for the bulk of annotation.

In this investigation, the new, improved Automatic Learning method, with Active Learning, can now change the bulk of annotation to a cascade of Machine Learning, Deep Learning, or Large Language Models, with the human operator responsible solely for checking the correctness of annotation results.

The methodology employed in this study involved a systematic approach to addressing user needs in the annotation process. Initially, users' common challenges and requirements during annotation tasks were identified. Then, tools were implemented as an annotation software to mitigate these challenges and improve overall efficiency. To assess the effectiveness of the software, we conducted a quantitative test, comparing the time required for manual annotation without the aid of tools versus the time taken when utilizing the newly implemented tools and a qualitative survey t the users. The objective was to quantify and evaluate the impact of tool integration on annotation speed and accuracy.

The study produced a new online natural language processing annotation tool with an active learning module, designed to expedite annotation times. Quantitatively, the tool demonstrated faster annotation speeds. Additionally, a qualitative analysis verified user satisfaction with the provided tools. The results collectively highlight the efficiency gains and user approval of the developed annotation tool.

This report is organized as follows: Section 2 presents works related to this study. Section 3 describes our annotation methodology of the work while 4 presents the software. In 5 We present the organized experiments and strategies for experimental evaluation nd the obtained results, while in Section 6 develops a discussion on the results. Finally, we present a conclusion in Section 7.

2 Literature Review

Online Annotation Tools. There is no shortage of tools for annotating textual data: Stav[12], Brat[13], WebAnno[14], WordFreak[15], Palladin[16], Incepttion[17], Doccano[18], Pal[7], and Yedda[19].

A comparative analysis among existing tools and the desired behavior for dialogue annotation revealed several limitations. Table 1 presents a comparative analysis of the available tools, which have some limitations, in which the need for installation and the impossibility of annotating *mobile* stand out.

In the context of the Table 1, we separate annotation (whose definition will finally be formalized in detail in Section 3) as *Manual*, *Automatic* and *Distributed*, but in short: *Manual* annotation refers to the ability of users to annotate by clicking and manipulating tools to manually label entities and intentions; *Automatic* annotation refers to automatic tools available to the user to annotate parts of the text, either using Machine Learning models or simple heuristic programs. *Collaborative* refers to the capacity of many users to edit the same document without causing conflict, either by separating the document in batches to different users and joining them togheter after, or by allowing an online, in-real-time, multiuse of the tool.

No-deploy refers to tools that are available completely online without the need for configuration. In most cases, this means the user has to install the software, run a package manager and possibly configure his system to run - which is one of the major burdens to most users, especially those without technical backgrounds. As can be seen, no tool in the current literature review allows it.

Vocabulary Suggestion means either that the tool can import annotation vocabulary (entity or intentions) from previous texts, or that a smart module in the tool is capable of suggesting new entities or intentions to appearing texts.

Table 1: Comparative analysis of available annotation tools

Tool	Annotation			No Deploy Needed	Vocabulary Suggestion	Documented Methodology	Language
	Manual	Automatic	Collaborative				
Stav	x			x			Java
Brat	x	x			x		Java
WebAnno	x	x					Java
WordFreak	x	x					Java
Palladin	x	x	x				Java
Incepttion	x	x	x		x		Java
Doccano	x	x	x				Python
Pal	x	x				x	Python
Yedda	x		x		x		Python

Tools with an intuitive interface. Stav provides excellent annotation visualization but is restrictive in the accepted data format [12]. Brat offers powerful annotation functions and rich visualization capabilities but lacks result analysis integration [13]. WebAnno can compare annotation discrepancies for each sentence but is not ideal for dialogues [14].

Tools with active learning. WordFreak adds an annotation recommendation function and integrates active learning to classify unannotated sentences based on recommended confidence, but lacks post-annotation analysis support [15]. Palladin integrates active and proactive learning for annotation but requires other active/proactive learning algorithms for large-scale annotation [16].

Fast annotation. Inception generally supports extension annotation use cases but needs performance improvements for large-scale tasks [17]. Doccano provides annotation features for text classification, sequence labeling, and tasks but does not use a standardized data structure [18]. Pal can significantly reduce annotation time but, like other tools, lacks active learning [7]. Finally, Yedda lacks an active learning strategy for a supervised sequence labeling model [19].

- All tools allow entity marking for NER (Named Entity Recognition) tasks but lack the annotation of intents in individual messages—only for the entire text—compromising chatbot AI learning.
- None of the tools presented focus on dialogues, rarely supporting structured data like messages in a chat between two agents (messages separated by turns). This often requires running a script to format messages as a single text, causing information loss during annotation.
- Installation or deployment is required, typically hindering use by companies or teams lacking extensive deployment knowledge.
- Non-standardized data structure: each tool has its particular way of importing, storing, and exporting data, requiring a custom conversion script for users employing various tools.
- Vocabulary suggestions and pre-annotated data import exist only in three of the analyzed tools (WebAnno, Inception, Yedda).
- None of the tools support annotation on mobile devices. Although not the primary use of the tools, it could be a necessary feature for users without access to personal computers.
- Few tools integrate with automatic annotation/Artificial Intelligence tools, and if they do, implementation is typically challenging.

The evaluation is given qualitatively and interpretively, based on the user's journey and experience in various real-world case studies, assessed through a satisfaction questionnaire and interviews with users. This is not only because it is a tool based on Human-Machine Interactions but also because the final result—a chatbot with AI—is unsupervised. Thus, the "correctness" of messages is perceived through naturalness and user satisfaction.

NLP Machine Learning Methods. Bidirectional Encoder Representation from Transformers (BERT), one of the basis of the current research, has a transformer-based architecture and can be pretrained on specific datasets [20]. BERT, designed by Google, has surpassed boundaries in various NLP areas. The pretraining of the language model has proven effective in enhancing many NLP tasks. It includes sentence-level tasks, such as natural language inference [21], which aims to predict relationships between sentences holistically, as well as token-level tasks like named entity recognition and question answering, where models need to produce token-level outputs [22]. It is also used in various languages, including Portuguese [20]. Originally pretrained on English texts, recent work by Brazilian researchers resulted in pretraining BERT on Portuguese texts. This Portuguese pretrained model, named BERTimbau, extends advancements achieved for the English language to Portuguese and is specifically tailored to Brazilian culture [23]. Because of the familiarity of many of the main users of the *Assis* tool, the BERTimbau model was chosen to be a compoment of the Active Learning module.

BERTimbau has important applications for corpus construction. As it has been pre-trained on large amounts of Portuguese texts, it is highly applicable to processing conversations and messages in this language, making it suitable for tasks like textual annotation of datasets [23]. In the context of tasks performed by BERT, two approaches are highlighted: the feature-based approach and the fine-tuning-based approach. These approaches differ in how the model is used and updated but can be applied to the same task.

Feature-based tasks involve applying the pretrained model to a textual dataset to generate a vector representation (embedding) of the text. The generated vectors (features) are then used in models specialized for specific tasks, such as classification. In this case, the BERT model's parameters are not altered, as it is used only for text embedding. On the other hand, the fine-tuning approach involves updating the entire model. This method adds a final classification layer to the model, and during training, all parameters are refined together. In this case, the entire pretrained version of the model is adjusted to perform the specified task. For the task of constructing a corpus from dataset annotation, both approaches can be used, as annotation can be seen as a text classification task.

Annotation Methodology. We understand as a Annotation Methodology (AM) any instructions and procedures describing how data should be annotated. In the context of NLP, an annotation methodology is a comprehensive framework that delineates the systematic process of annotating various aspects of linguistic data, encompassing intentions, entities, and other relevant linguistic features.

Remarkably, to the best of our knowledge, the development of a comprehensive annotation methodology for NLP data has not yet been explored in the broader literature. While existing methodologies address specific aspects such as geo-related semantic annotation [24] or sentiment annotation in texts [25], a general annotation methodology that encompasses the diverse facets of NLP remains conspicuously absent. This underscores the need for further research and exploration in developing a unified annotation methodology that can enhance the quality and consistency of annotated NLP datasets across a range of applications, which is one of the contributions of this research.

3 Our Annotation Methodology

Tools such as stav [12], claim to be "fast", or Paladin [16], that claims to be "faster" than other tools in time of annotation. However, these papers do not define - for example - what is considered the time of annotation. Should the download and configuration of a tool be considered at the time of annotation? Should the user's

time to navigate the menus be part of the annotation time?

We aim to create a cohesive definition for many of the terms mentioned but not defined by published Annotation Tools to serve as a basis for developing our solution.

3.1 Definitions

Consider the following definitions:

- A *word* is a sequence of *characters*, *numbers* or *symbols* concatenated in order. *Words* are indivisible, and are separated from other *words* by punctuation marks or spaces (including line breaks). A priori, the methodology does not concern with semantic cohesion, that is, if the *words* "makes sense" - however, we assume *words* carry certain relations to one another, thus, *words*, *sentences* and *texts* carry meaning in relation to one another (even if said meaning is a simple statistical correlation).

- A *sequence of words* is an ordered finite sequence of *words* such that $W = \{w_0, w_1, w_2, ...w_n\}$, where W is the sentence, w_i a word and n the number of words, these words are separated by a punctuation marks or spaces which are not relevant to the methodology. We assume the order of words is still important because they may carry semantic meaning, thus $\{w_1, w_0, w_2\} \neq \{w_0, w_1, w_2\}$. This definition differs slightly from linguistics, which would define clauses, phrases and sentences as different and would attribute value to ideas such as verbs and subjects being present.

- For a more concise notation, hereinafter we will refer to *sentences* as any large *sequence of words*, or simply $S = \bigcup_{i=1}^{k} W_i$.

- An *agent* is any being (human or otherwise) capable of producing sentences, sentences created by agents are marked as S_a.

- A *text* is an ordered finite sequence of sentences such that $T = \{S_0, S_1, S_2, ..., S_m\}$ separated by punctuation marks and spaces. Note that texts can contain sentences from multiple agents. Texts created by a single *agent* are called a *monologue* and texts with two or more *agents* are called a *dialogue*. Per our definition, there can be no text without an *agent*.

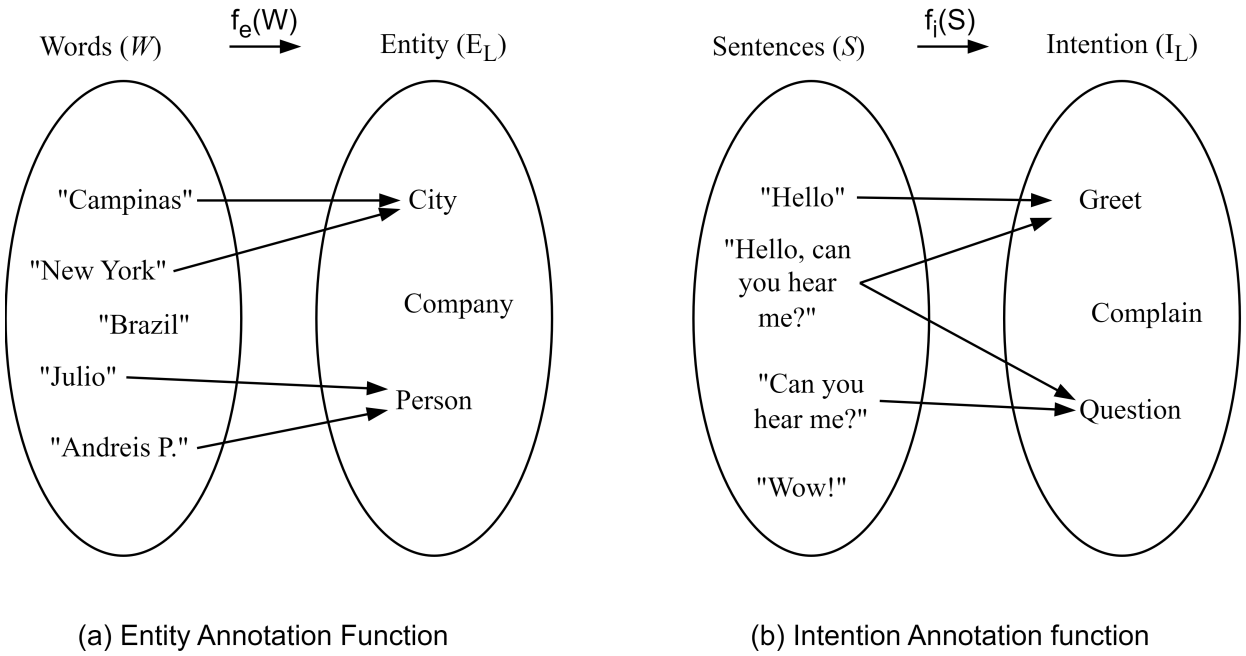
- A *message* is an uninterrupted, ordered finite sequence of *sentences* in a *text* $T = \{S_0, S_1, S_2, ..., S_m\}$ created by the same *agent*. In monologues, we have that there is a single message $M = T$, in dialogues, however, messages from different agents take the form of $M_a = \{S_{i_a}, S_{i+1_a}, ..., S_{k_a}\}$ where $0 \leq i \leq k$ and $i \leq k \leq m$.

With these definitions in mind, we can now help define what annotation is. For the purposes of limiting our scope, we will focus on the *entity*, *intention* and *topic* annotations. An *annotation* is the act of mapping a *word*, *sentence* or *message*, depending on the situation, to an equivalent semantic label - such as shown in Figure 1. Note that these functions/relations are not necessarily injective or surjective.

Intention Annotation: An *intention* refers to the underlying goal or purpose expressed by any *agent* in a *sentence* or *message*. In a semantic analysis, intents are usually correlated to action verbs, such as "question", "greet", "inform". We will call the specific label that defines the a certain purpose as intention label. *Intent Annotation* is the act of mapping any *sentence* or *message* S to one or more *intentions* i in a list of I_L labels using an mapping relation $f_{intent}(S, I_L) : S \rightarrow i$. Note that $f_{intent}(S)$ is a relation, not necessarily a function, because a single sentence can map to multiple intents.

Finally, $Intent(T, I_L)$ is an unordered set of tuples that contain the a sentence S_i and a intention label i_j in a text T. Take the following example of a dialogue between two agents:

- S_0: [Agent 1] "Hello. Can you hear me?"



(a) Entity Annotation Function

(b) Intention Annotation function

Figure 1: Example of Entity Annotation and Intention Annotation.

- S_1: [Agent 2] "Hello. Yes."
- S_2: [Agent 1] "Can you close the door?"

And the the following intents:

- i_a: Greet
- i_b: Inquire
- i_c: Answer

In this case, the final unordered set of tuples representing the messages and its corresponding intents is $Intent(T, I_L) = \{(S_0, i_a), (S_0, i_b), (S_1, i_a), (S_1, i_c), (S_3, i_b)\}$.

Annotation systems that allow for a single intention per sentence/messages are called *single-intention annotations*, while annotation systems that allow for a *sentences* or *messages* to have multiple intents are called *multi-intention annotations*. As are all the tools in the literature review and the tool developed in this project, most systems available are multi-intention.

Entity Annotation: A *named entity*, or simply *entity*, is any specific type of object, concept, or "thing" that is mentioned and identifiable in a *sequence of words*. In most cases, entities are nouns. Conversely, we can state that a single *entity* label is a concept that can group multiple disparaging semantic words. Thus, an *Entity Annotation* is the act of mapping a word or a short sequence of words $W = w_a, w_{a+1}, ...w_b$ within a sentence to the corresponding Entity e of a list of E_L labels using an mapping function $f_{entity}(W, E_L) : W \rightarrow e$.

Finally, $Entity(S, E_L)$ is an unordered set of tuples that contain the each sequence of words W_i and a entity label e_j in a sentence. Take the following sentence S:

- w_0: "Julio"

- w_1: "went"
- w_2: "to"
- w_3: "São"
- w_4: "Paulo."

Consider that for convenience, we can join w_3 and w_4 in the sequence of words $W_{3,4} = \{w_3, w_4\}$. Now, consider the following entities:

- e_a: Person
- e_b: City

Thus, we have that $Entity(S, E_L) = \{(w_0, e_a), (W_{3,4}, e_b)\}$. For convenience, we may also simply write $Entity(S) = \{(\text{"Julio"}, Person), (\text{"São Paulo"}, City)\}$.

Topics Annotation: A textual *topic* is a the underlying goal or context of a group of *texts*, which can be from one or many *agents*. *Topics* are a species of grouping of *intents* in sequential *messages*. *Topics* represent, for example, a certain part of a conversation that refers to a specific context that may not comprise the entire text: two users talking about the climate, before changing the conversation to their plans to the next day. Per our definition, we also consider that entities are consistent inside a topic - that is, if a certain noun is tagged to an *entity*, every following *message* in the topic will know that if the *entity label* is mentioned, it may content the semantic context of the noun. This is a powerful concept for annotation methodologies because in NLP we may train language models to "remember" facts and information inside a context.

We call *vocabulary* $V(T)$ the list of lists of *entity labels, intention labels*, and *topic labels* of a given text. In our current definition. $V(T) = (I_L, E_L, T_L)$.

Disconsidering topics (for a briefer notation), with this definitions, we can create the notation $Annotate(T, V)$, that is an unordered list of two sets: all mappings of every sentence to the existing intents, and every sequence of words that represents an entity to the existing entities. Mathematically:

$$Annotate(T, V) = \left(\bigcup_{S \in T} Intention(S, I_L), \bigcup_{S \in T} \left(\bigcup_{W \in S} Entity(W, E_L) \right) \right) \tag{1}$$

An *annotator* can use the mapping functions/relations to annotate texts. When the *annotator* is human, we refer the the act of *annotation* as *manual annotation*. When there are multiple human *annotators* capable of *annotating* the same *text* in an incremental, automatic manner (that is, there is no need for a human operator to solve conflicts and merge annotations), it is called *collaborative annotation*. If there is one or more human *annotators* and one or more computer scripts (thus, machine *annotator* - be they simple codes to Large Language Models) annotating togheter, but the bulk of *annotations* is made by the humans, it is called *assisted annotation* - if the bulk of *annotations* is made by the computer scripts, it is called *automatic annotation*.

Annotation time, error and fatigue: Consider that to complete an *annotation* task, an *annotator* needs to take a n number of actions a, such as opening a program, clicking a button, writing in his keyboard, etc... Consider $A(T)$ the set of actions needed to fully annotate a text.

The main objective of any annotation tool is reduce the number of actions taken by a human *annotator* as much as possible. That is, the objective of annotation tools is always to reduce the burden of annotation on human users. Thus, we can finally call a tool a more *efficient* than tool b when $|A_a| < |A_b|$.

The most simple and practical way of calculating a tool efficiency is by simply running a background script to count the number of clicks from the beggining of an annotation to its end.

Now, consider that *total annotation time*, or simple *total time* T, is the measure of time from the moment a user starts an annotation task to its end. It includes time to configure a tool (if needs be), open a text, create entities and intentions. Depending on the current need, different methodologies may create different definitions: for example T_d for download and configuration time, T_v for the creation of vocabulary, T_a for annotation time, etc...

Finally, we can say that a tool 1 is *faster* than another tool 2 if $T_1 < T_2$.

Now, consider that given a vocabulary $V(T)$ of a text T, we have always the same complete output/mappings. That is, we can say that a *completely* annotated text is that which all mappings have been done.

We define a *annotation error* when an *annotation* maps incompletely T. That can happen, for example, when an entity was not mapped to its label, or a sentence was not labelled to its intent, or when two annotations of the same vocabulary and same text yield different results. We may define $Error$ between two anotations a and b as the different between unordered sets:

$$Error = Annotate_a(T,V) - Annotate_b(T,V) \tag{2}$$

If $Error = \varnothing$, we can simply say there is no error/difference in annotations.

Finally, consider the concept of *annotation fatigue:* We propose the more actions A that human *annotators* has to execute, the more propense an *annotation* is to have *errors*:

$$|A_1| \leq |A_2| \Rightarrow |Error_1| \leq |Error_2| \text{ and } |A_1| \geq |A_2| \Rightarrow |Error_1| \geq |Error_2| \tag{3}$$

3.2 Dialog Annotation Blueprint (DAB) Methodology

The literature review addresses the need to annotate large volumes of dialogues and highlights the lack of accessibility and standardization in existing annotation tools. Here, we introduces a unified methodology called Dialog Annotation Blueprint (DAB) based on various case studies. Furthermore, this methodology has served as the foundation for developing two free software tools designed for creating and annotating data in dialogues. This aims to streamline the annotation process and promote greater automatic integration between systems, making it more accessible to non-technical users.

As defined previously, we define as a *annotation methodology* the detailed procedures and steps of completely annotating a given text. DAB serves as one of the possible methodologies that could be created with this definition. DAB is divided into macrosteps, As shown by Figure 2.

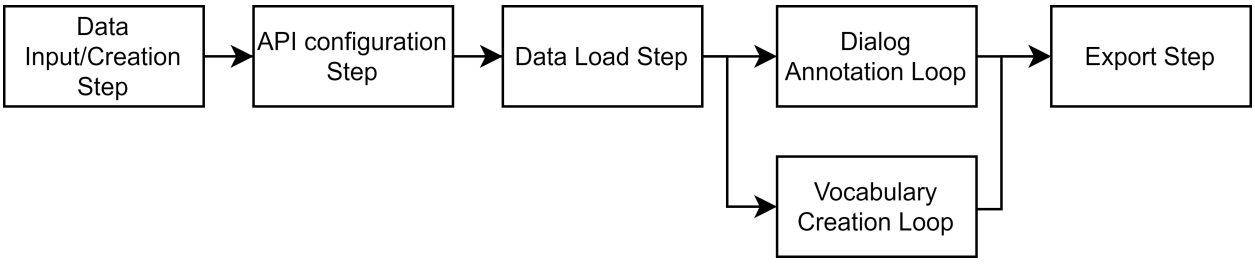


Figure 2: Macrosteps of the Dialog Annotation Process in the DAB methodology.

The Data Input/Creation step is responsible for connecting to data creation tools such as MCCD [26], as shown in Figure 3. At first, it reqquests from the user if they would like to configure a Data Input Endpoint (DIE). If added, the system will make a test request. If the test request has worked, the next step in the annotation process is to retrieve data schemas, configurations, metadata, and vocabularies from the endpoint.

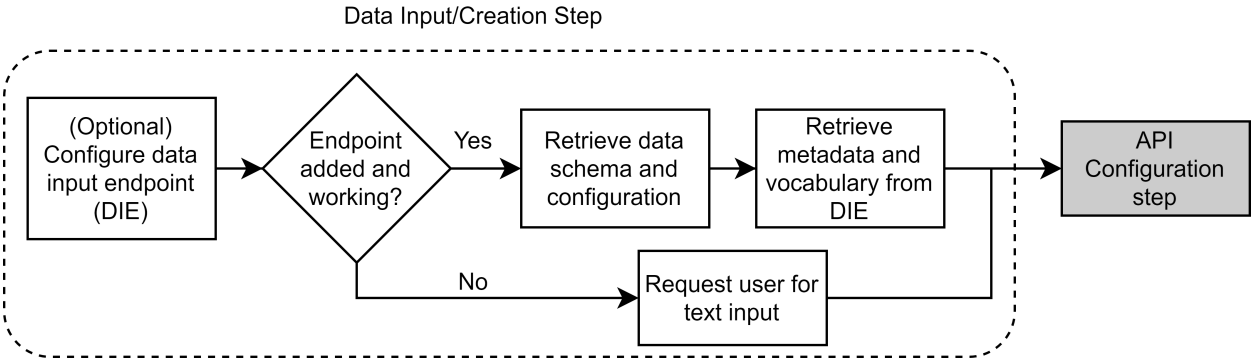


Figure 3: Data Input/Creation Step

Figure 4 shows the second step: the API Configuration Step. This step is important for any tool using the methodology to configure automatic annotation, server imports, and dialogs data.

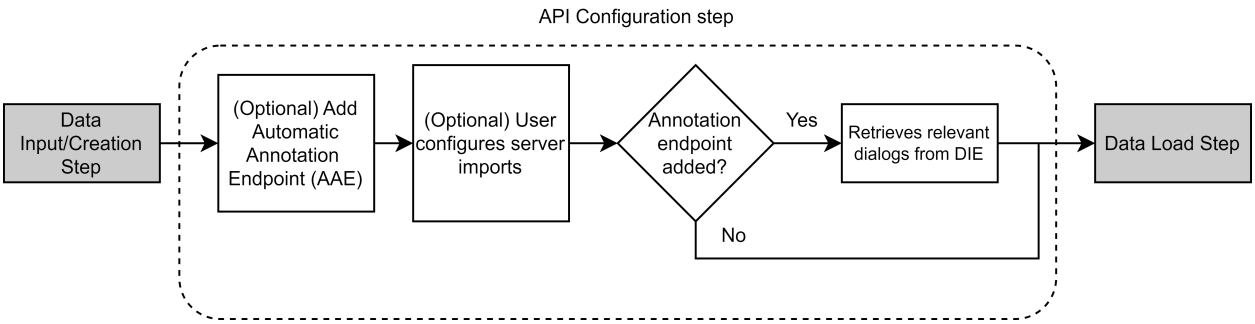


Figure 4: API Configuration step

Figure 5 represents the Data Load step: It is a *sanity check* of the loaded dialog data and configuration of vocabulary. It should also be in this step that any visual interface is set with the loaded data, and the dialog control data (i.e., the annotation *metadata*) is created.

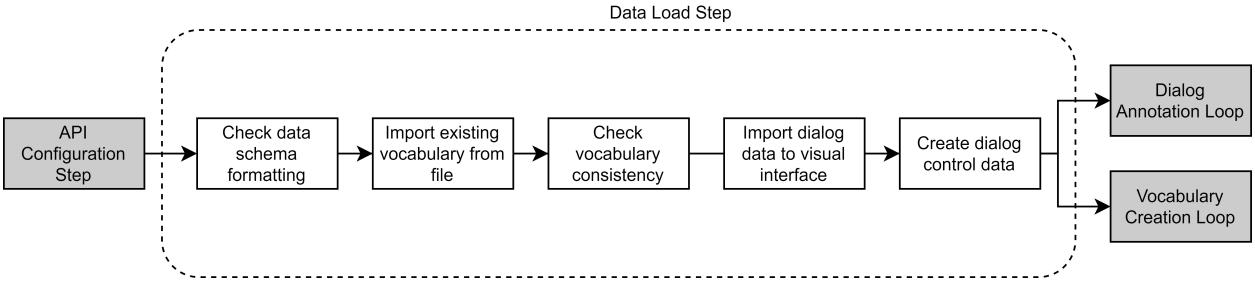


Figure 5: Dialog Load Step

The Vocabulary Creation Loop is shown in Figure 6. This loop begins after loading the dialog data and remains in loop until the dialog is marked as finished by the user. This is the step which creates and manages the *vocabulary* (intents, intentions, topics). When finished, it should automatically save and export vocabularies.

Figure 7 presents the Dialog Annotation Loop, which contains the Manual Annotation and Automatic

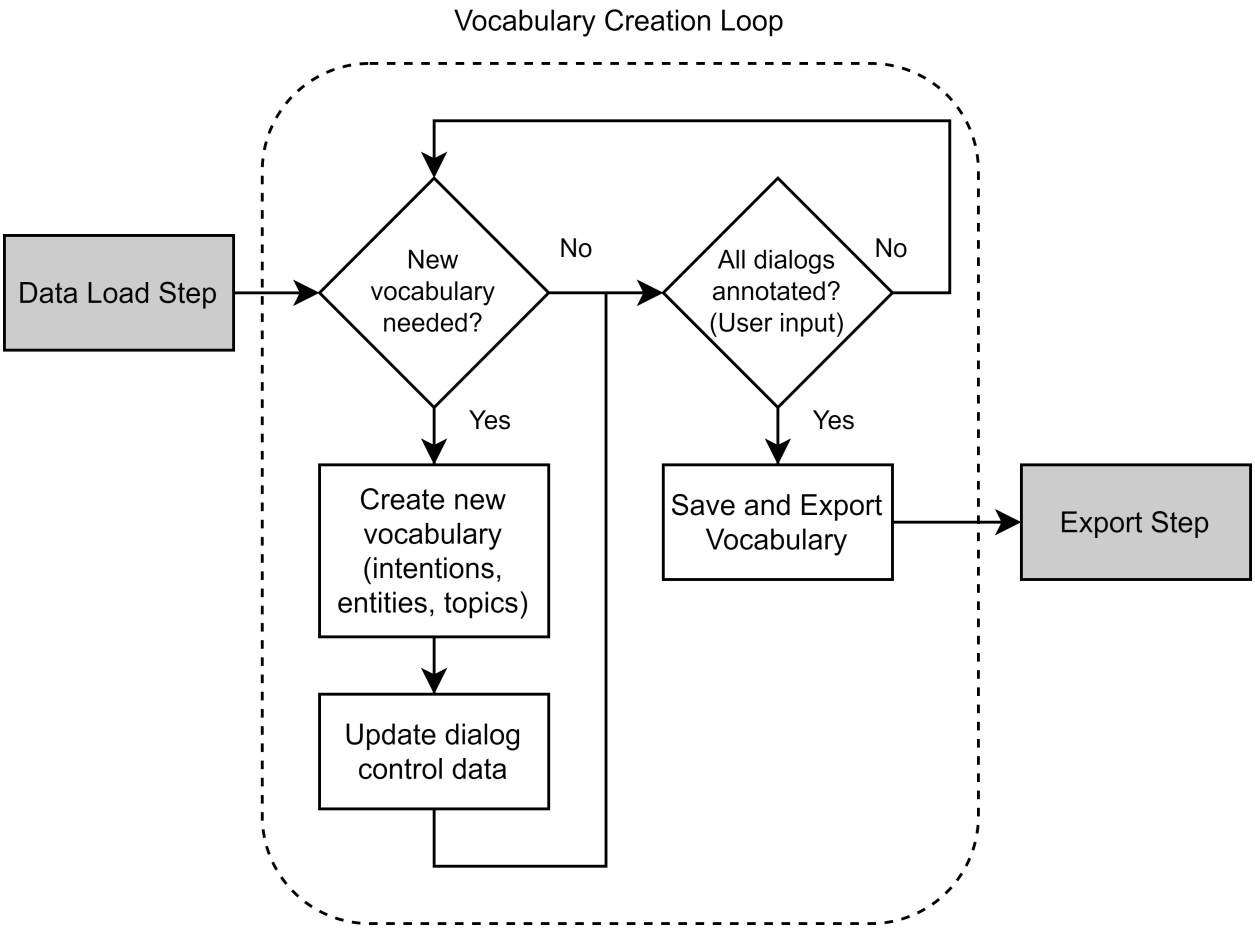


Figure 6: Dialog annotation step/loop

Annotation Modules. This loop repeats until the user marks the dialog as finished. The manual annotation module should be capable of accepting User input in marking the dialog with the available vocabulary. Meanwhile, the Automatic Annotation Module is responsible for the Machine Learning/LLMs that annotate dialogs in the Automatic Annotation Endpoint (AAE). The tool then returns the annotation data to the Manual Annotation Module.

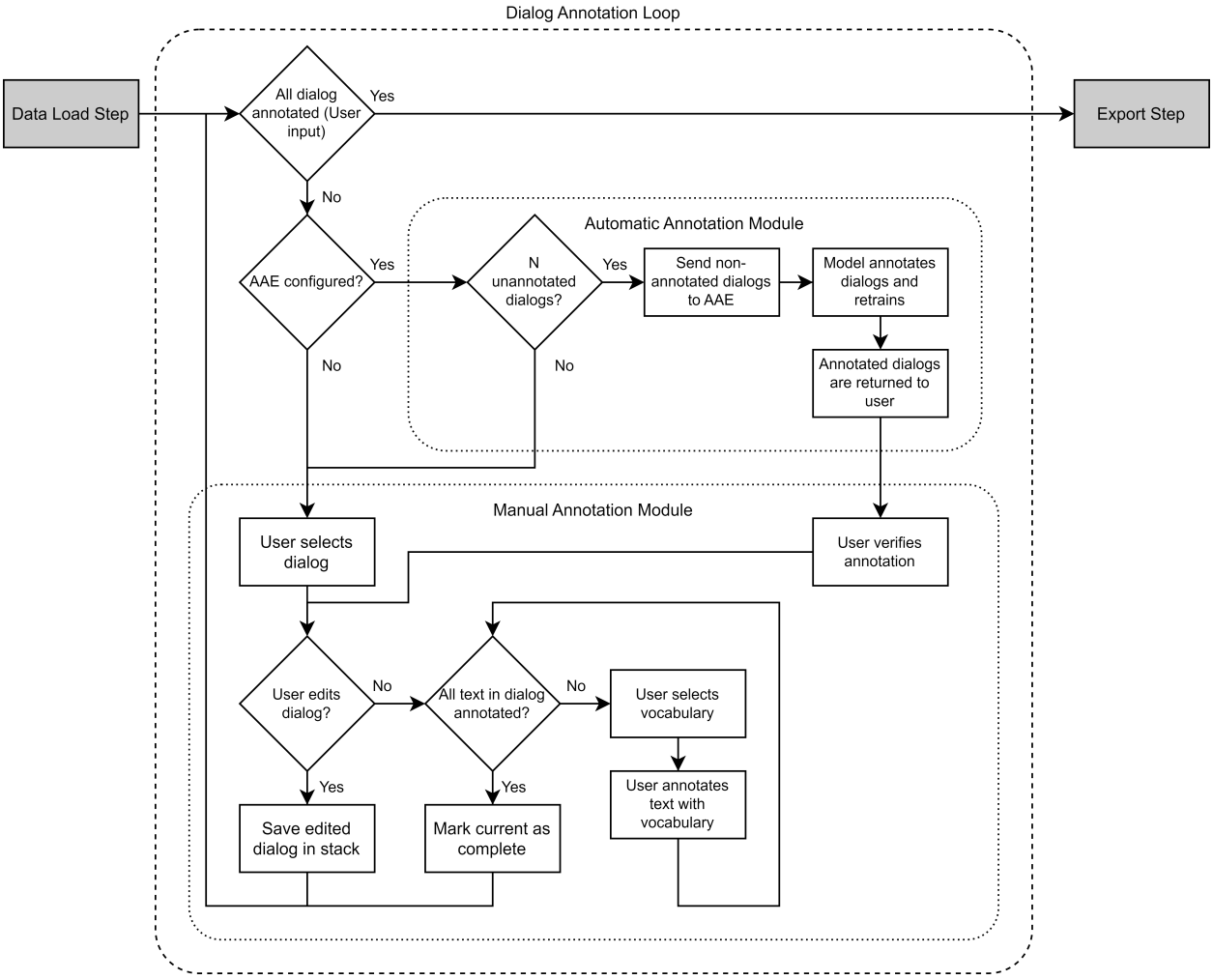


Figure 7: Dialog annotation step/loop

Finally, the Export Step is presented in Figure 8. The export step is responsible for exporting files and saving configured data and metadata in the endpoints.

4 Assis Software Tool

Standardized Formatting with MultiWoZ Data

The *MultiWoz* dataset stands out as a reference in the field of textual data due to the quantity and diversity of texts, dialogues, and domains. This dataset consists of ten thousand fully annotated dialogues distributed across seven different domains inserted in a general context of customer service. In addition to providing a significant amount of structured and annotated data on a larger scale than previously available, it establishes a useful structure of organization and annotation for conversations that can be applied in the construction of

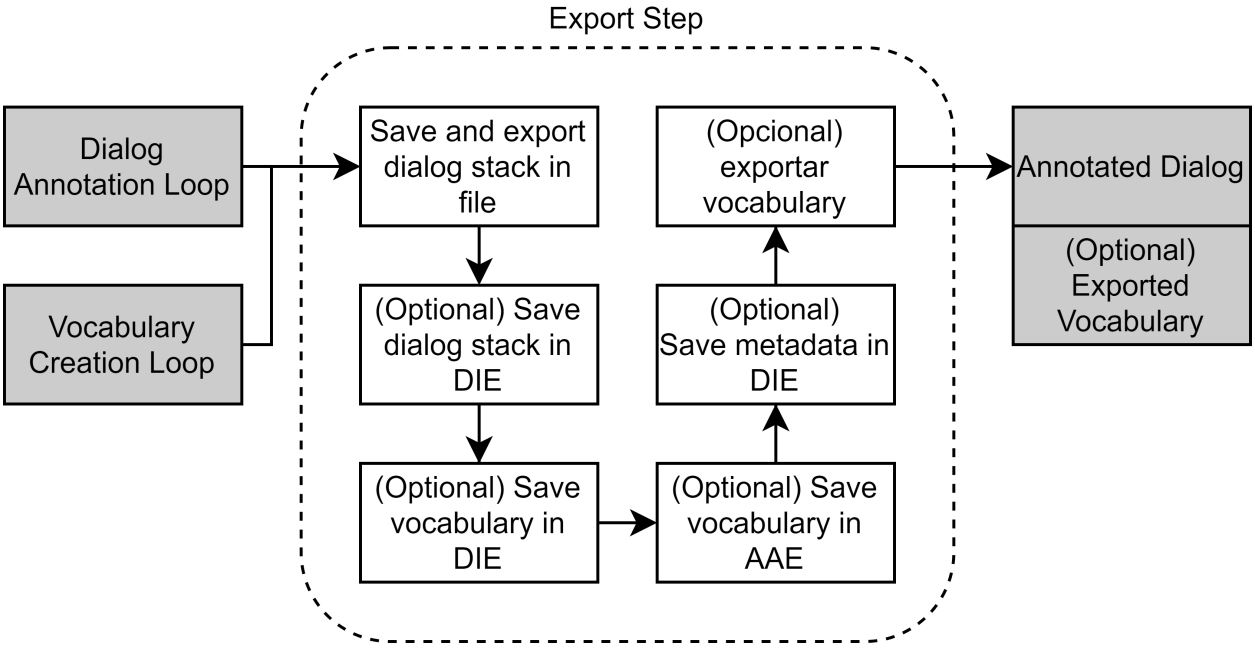


Figure 8: Export Step

other datasets [27]. Therefore, the Assis tool was designed in such a way that the entire process uses the JSON format structure of MultiWoz.

In summary, the JSON data format of MultiWoz contains two main types of data: vocabulary (often referred in by MultiWoz as "ontology"), which functions as a dictionary of all types of intentions and entities (and their respective values); and dialogues, a set of all dialogue data. Each dialogue represents a conversation between a user and the system (AI or Oz); it is identified by an *id* and the domain it implies (what the conversation was about). Each exchanged message is called a turn. The property of dialogues in the JSON file is the part of the data that will be annotated, filled, and manipulated, while the ontology will present the annotations created in a categorized manner.

This approach not only eliminates the need for conversion scripts but also allows interoperability between tools since it is a well-known and well-defined format - enabling other dialogue generation or AI tools to be easily integrated with Assis. Another advantage created by this usage is the ability to import data since the ontology serves as a dictionary of possible annotations.

In general, the Assis annotator allows tolerance in the format of input data. Although the model is structured with inspiration from the MultiWoZ format, a JSON file that has the dialogue format but lacks all the properties will still be accepted (for example, ids, agent/user identifiers, and ontology may be missing without major consequences in the system).

Text Editing, Topics, and Entity Propagation

Chatbot AIs are typically oriented towards two tasks: identifying the user's message intentions (whether to demand or provide information, notify events, among others), and identifying the entities contained in the message (companies, organizations, people, and other Figures that can be described by their name or value). The MultiWoZ format for dialogues already provides for both types of annotation.

However, one of the innovative concepts implemented by the Assis Tool is the concept of topics. Topics exist to describe a broad part of a dialogue and help developers choose only relevant parts for their AI learning: It is a group of turns and functions as a general intention. For example, a long dialogue may sometimes cause the user and the system to switch between various topics (i.e., asking for basic information,

inquiring about specific products, and asking about delivery times).

Assis has a non-invasive architecture for manipulating MultiWoz data, allowing the user to use topics or not. Another consequence of this implementation and a unique feature of Assis is allowing the user to make changes to the text (via the pencil icon, next to the name) and restore the original version if there are errors. The editing tool is capable of maintaining the already annotated intentions and topics in a message.

Finally, another innovation of Assis is entity and intention propagation, created after verifying this need during use cases. Propagation means that when exporting files, the user can allow certain entities or intentions to "appear" in later messages. That is, if in the first message the customer provides, for example, the CPF (considered an entity), it is useful for this data to be stored so that in later messages the AI model still knows the given value.

Figure 9 shows an example of annotated dialogues in the tool with entities, intentions, and topics.

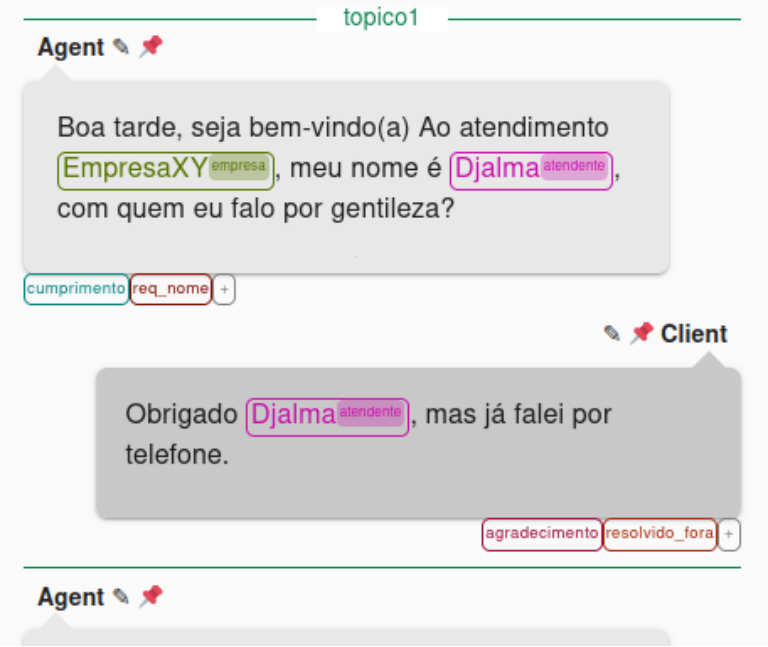


Figure 9: Example with entities: company and attendant; intentions: greeting, req_name, resolved_outside, and farewell; and topics: topic1

Integration with Other Tools

Figure 2 also presents another important feature of the Assis tool: by being designed with an interoperable methodology in mind, Assis can be integrated with other tools. The main goal is to allow companies and users to create or adapt tools as they see fit, avoiding one of the main problems of related works: the difficulty of adapting/integrating all desired functionalities into the tools.

The architecture of Assis is designed in two separate modules: the frontend (which includes manual annotation and management tools with other tools) and the backend (with automatic annotation tools using BERT), communicating through APIs.

Since the input and output of both parts of the tool are APIs, they can be used separately. One case study was the creation of a dialogue data generation tool that allowed the Assis frontend to automatically collect data via API. The same applies to automatic tools: just configure the endpoint on the tool's options page for Assis to start sending data for automatic annotation (Figure 10 shows an example of using the implemented BERT from Assis (Vanilla) and a specialized BERT from an example company: 'CompanyXY').

Easy Access

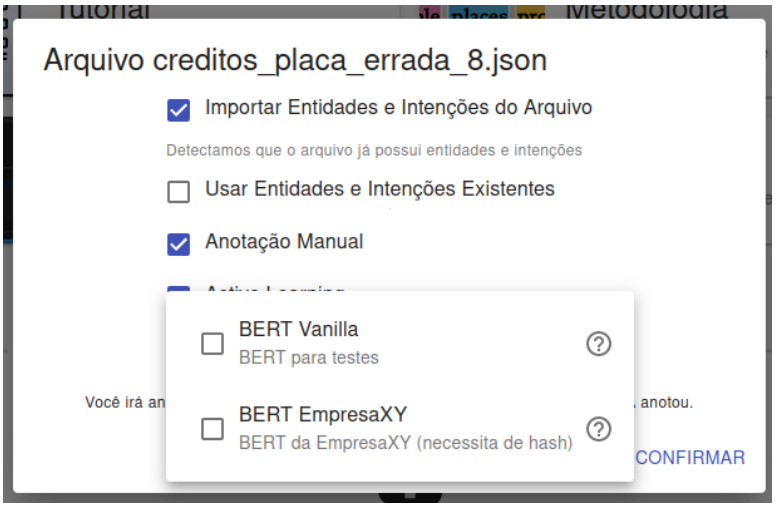


Figure 10: Example of configuration of endpoints in the tool

One of the major problems identified with other tools is the lengthy process of installing the program and/or deploying it on the company's own server. This means that even small use cases will take a considerable amount of time and cannot be done by teams without technical knowledge.

The Assis tool circumvents this problem by having its file management module and manual annotation tools in the frontend module, programmed in React (JavaScript) for use as a static page. This means that the main functionalities (except those involving artificial intelligence) are available for online use in the current version of the tool. Users can also use the project on their own GitHub with just a few button clicks, allowing for a high level of modifications and different uses.

Following the philosophy of single-page application with React Hooks, each item on the screen is managed by JS functions, allowing a developer to use the project as a base and add a new page or a new annotation button without having to change the rest of the system. The tool has a significant number of fail-safes to ensure the continuous execution of the program, even in case of failures (e.g., in the JSON format).

Quick and Intuitive Visual Annotation

The purpose of annotation tools is to reduce the exhaustive annotation process - which is usually the bottleneck of Natural Language Processing technology development. Therefore, a fast, intuitive UI that facilitates visual annotation is essential.

The tool was based on the philosophy of the MUI (Material UI) library, used by various companies such as Spotify, Netflix, and NASA. The focus of the tool is to create an easy and agile interface for the user, allowing high freedom, speed, and configurability (it is the only tool so far capable of being used entirely in mobile applications without visual bugs or compromising usage time, as shown in Figure 11). The user can create and configure the vocabulary created. By clicking next to the name of the speaker (Agent or Client), the user can edit the text in the message or add a topic/domain. The user is free to navigate between dialogues.

The annotation tool in the frontend (in its current version) has a floating action button for dialogue management (select for annotation, mark as deleted, mark as complete). And three floating action buttons for each annotation vocabulary, domains (or topics), intentions, and entities. It is the only tool where the user can select all three at the same time and annotate them in parallel (since each annotation has a different scope, the tool knows which one to use), meaning the user does not need to open the menu to re-select an entity or intention.

Import and Export of Vocabulary

Another major contributor to the excessive data annotation time is the vocabulary creation phase (re-

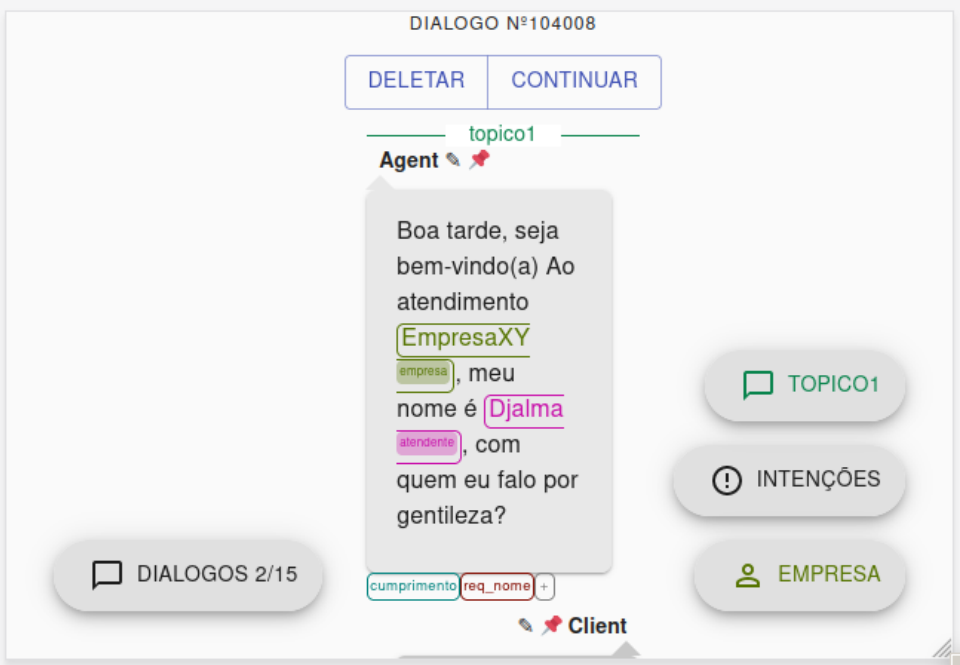


Figure 11: Example of the tool on a tablet screen

ferred to in the MultiWoZ format as "ontology"). Every time there is a new file to be annotated, the user needs to create the entities and intentions they want to use - even if they have used them before. This also tends to cause annotation errors since different users (working under the same file) may name variables differently (e.g., "*accepts_card*" and "*ac_card*"), causing incorrect learning for AI. In general, the larger the file, the more time will be spent creating entities.

Following the idea of tool inter-compatibility and the MultiWoz format, the tool has a function to import the ontology field (vocabulary) from MultiWoz - typically complete with all possible entities and intentions for a given topic. This data can come from the MultiWoz dataset, created by the client, imported by JSON, or an endpoint from a server. The tool has an agnostic input approach: if the ontology is empty or does not exist, it will create the vocabulary based on existing annotations. Similarly, the tool by default will always export the ontology in the MultiWoz format, allowing annotated files to be reused as vocabulary for new files.

The backend module can also store a copy of the vocabulary used. It is also possible to limit users to only use existing vocabularies on the server. The tool is also capable of annotating metadata for annotation (extra variables in the MultiWoz format that are normally used by the tool to store less important information, such as the color of each entity or intention).

4.1 Active Learning and Backend Module

As explained in the integration with other tools, each person or organization can implement their own module on a server and communicate through API calls with the module. Consider a dialogue with size $|D|$, that is, the $|D|$ all the sentences in a dialogue, and N represents the number of sentences sent to an computer agent to be annotated.

Per our previous definition in Section 3.2, a fully manual annotation, is one that does not have a configured endpoint, and all annotation will be done by the user ($N = 0$). Semi-automatic a backend which only annotates small parts of the text (a conversation or a message) whenever there is input from the user to do

so ($0 \leq N \leq |D|$).

On the other hand, a fully automatic server is one which already receives the entirety of data ($N = |D|$) when after the user has uploaded, and starts the annotation process using computer scripts and machine learning models. By contrast, a fully automatic server doesn't require initial user input to start annotating, and will try to annotate as much data is possible. The annotated data is returned to the user, which will now verify and correct the annotation. The final corrected annotated text is returned to the server.

Both semi-automatic and fully-automatic options can have multiple pre-trained models, such BERT, BERTimbau, or even simpler models such as Multilayer Perceptrons. In our experiments, we used BERTimbau for entity annotation and a MLP for intention annotation. These models are stored and saved in internal folders, and will be loaded and be ready to use after initialization. Note that for the annotation studied by this work, both entity and intention models are always Machine Learning Classifiers.

The proposed "standard" backend server was developed in Python with language models like BERT using the Flask microframework, which allows web development in python with a simple but extensible core. The active learning approach [28] employed in the developed backend module allows the models to function, initially, as annotation assistants and later, when performing well enough, as complete annotators. This approach takes into account continuous feedback of dialogues annotated by humans in the model and consists of three stages. When an API endpoint address is configured by the frontend module on the API configuration step (Figure 4 and Figure 10), the following actions are taken:

1. Frontend sends a GET request to address index.
2. Backend returns metadata regarding current run (which model is loaded, how many data has been annotatted, vocabulary size, and others) and a dictionary of actions and endpoints.
3. Frontend sends a GET requesting the vocabulary and checks if there is any discrepancy (different sizes).

The frontend module can request a few actions to the backend, a few are:

- **Load Model.** When initialized, the backend server already loads an annotation model. This request exists if the the frontend/user desires another model or a new instance of the model.
- **Save Model.** This requests flags the server to save the current model being executed in the model folder. This action exists should some users desire to "overwrite" older models, or save the models with retrained data.
- **Annotate Data.** This action sends a number of dialogs (as explained previously, a fully automatic server will request from the frontend ALL available dialog data) to the server, which will now be delivered to the Machine Learning models to be annotated. The resulting annotated data is returned to the frontend.
- **Retrain Model.** This action sends a number of annotated dialogs to be used to retrain the models. Depending on how the Machine Learning model is implemented, either the server will retrain using only the new dialogs, or using the new dialogs and old stored dialogs. This is the action that defines the Active Learning loop.
- **Request Metadata.** This request demands for all the available metadata in the server (current loaded model, existing saved vocabulary, and others), and servers mainly to inform the user of the backend status via the frontend module.

- **Request Vocabulary.** This request demands for the saved vocabulary in the server. This is used to verify if the vocabulary has been changed, or if the imported vocabulary from the file matches the vocabulary existing in the server.

- **Upload Vocabulary.** This action uploads a new vocabulary to the server, overwriting the previous one. This is used when new entities, intentions or topics are created.

As seen previously in Figure 7, the **first** stage is model training, where they receive dialogues with their respective annotations and learn recognized annotation patterns in these examples. The **second** stage is semi-automatic annotation of dialogues by models (recommendation of entities and intentions), where they receive non-annotated dialogues and, based on previous training, predict possible annotations. This annotation is more prone to errors, so it is treated as a suggestion to be confirmed or corrected by an annotator, i.e., a person. The **third** stage is refinement of manual annotation. In this stage, an annotator must receive dialogues with suggestions for annotation made automatically by models and correct or accept such suggestions.

Each server can have a configured "minimum confidence" treshold, that warns the user when an annotated sentence has an a low annotation confidence. For BERT, models, for example, the return of a certain entity is always an array of the probability of each entity. We assume the entity with the highest probabiity is the correct one, and should this probability still be lower than the treshold, the frotend can display a small warning sign at the side of the dialog, giving the user the visual indication to verify annotation. This is important for selecting useful data for the active learning procedure.

It is also possible to separate the N annotated messages in smaller annotation "batches" of size x, this reduces serve roverhead and creates a more iterative process. A part of x will be sent back to the application for the user to check. The user's correction and verification are then sent back to the AI and used for learning. In the case of the proposed BERT, the recommended values for the batch is 5 dialogues.

The chosen method for training annotator models takes into account a technique of incremental and active learning. This means that the models are constantly classifying dialogues regarding their annotations and training. Learning is incremental because training examples are provided gradually in each cycle and is active because it takes into account continuous human action in the teaching and improvement process of the models. A pseudocode of this process is shown in Algorithm 1

It is also possible to created Large-Language Model annotation modules for the backend, changing the traditional backend model for a LLM model such as ChatGPT. The gain created by such a module is eliminating the need to train and retrain models, and the high accuracy provided. However, a disadvantage of doing so is increasing the time and cost (if using a paid API) of automatic annotation. In this configuration, it is necessary to create a "Staging Server", which manages the Frontend calls and prepares the prompts for the Large Language Model. An abstraction of these two possible approaches are shown in Figure 12.

One of the ideas explored by Assis is combining the LLM automatic annotation module with the traditional automatic annotation module as shown in Figure 13. In this case, the Selection Server is responsible for sending all the dialog to the automatic annotation, and instead of returning the low confidence annotations directly to the user, the selection server sends them to the LLM server, for correction before returning to the user. This mixed approach combines the speed and low-cost of the automatic annotation and the high accuracy of the LLM annotation.

Finally, an approach that maximizes accuracy (but increases annotation time and cost) is found in Figure 14. This is an approach with multiple annotation modules working in parallel, and for each sentence, the staging server selects the annotations in a "majority voting" procedure, where the annotations that match the most with others are selected. While this experimental procedure was tested and is now documented, the time consuming parallel annotations proved to be too high to be practical at the moment.

Algorithm 1 Dialog Annotation and Model Update

```
Load classifier from models folder, pre-trained with previous data
Load the new dialog/batch data
while the dialog has not been marked as finished do
    Select sentence and annotate with entities and/or intentions
    Calculate maximum probability for the annotation
    if maximum probability < minimum confidence threshold then
        Mark sentence as low confidence
    end if
end while
Return the annotated dialog to the user
for every dialog do
    if user changed dialog in frontend module after annotation then
        Replace changed dialogs in backend
    end if
end for
Re-train the model
Re-calculate model accuracy
```

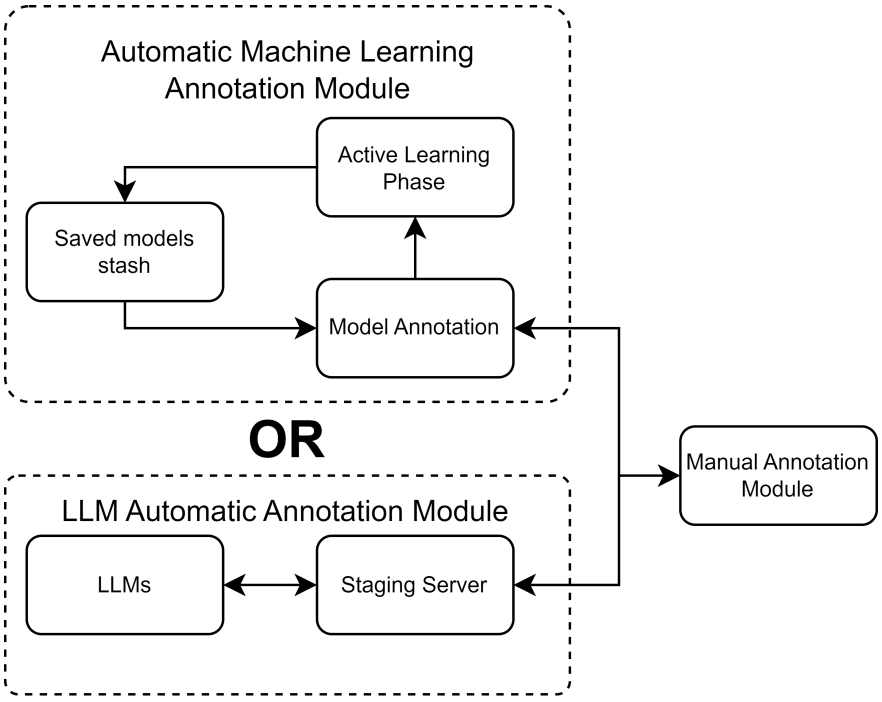


Figure 12: Flow diagram of the interaction betwen the Manual Annotation Module and two possible modules, LLM and Automatic ML module.

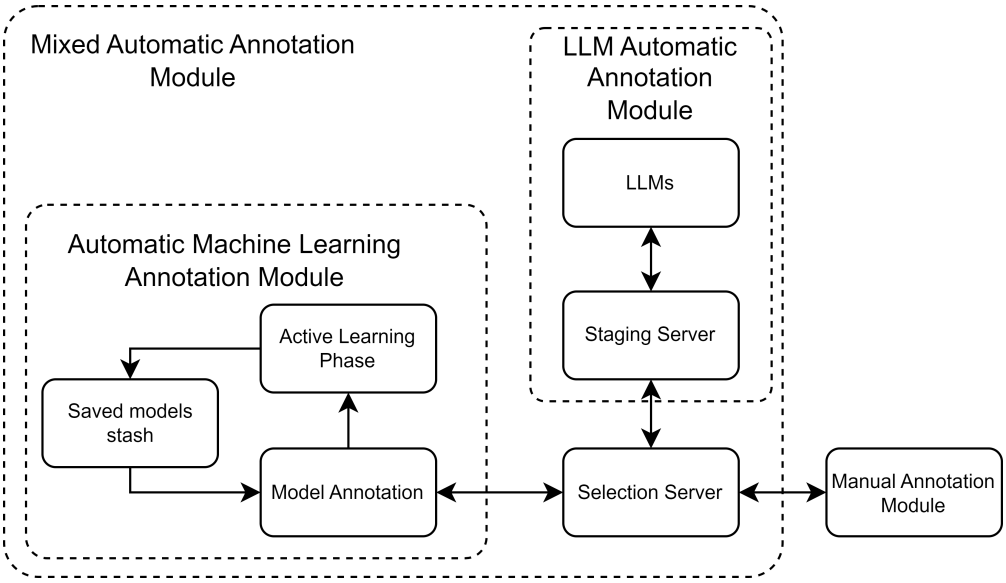


Figure 13: Example of a low-cost low-confidence mixed backend.

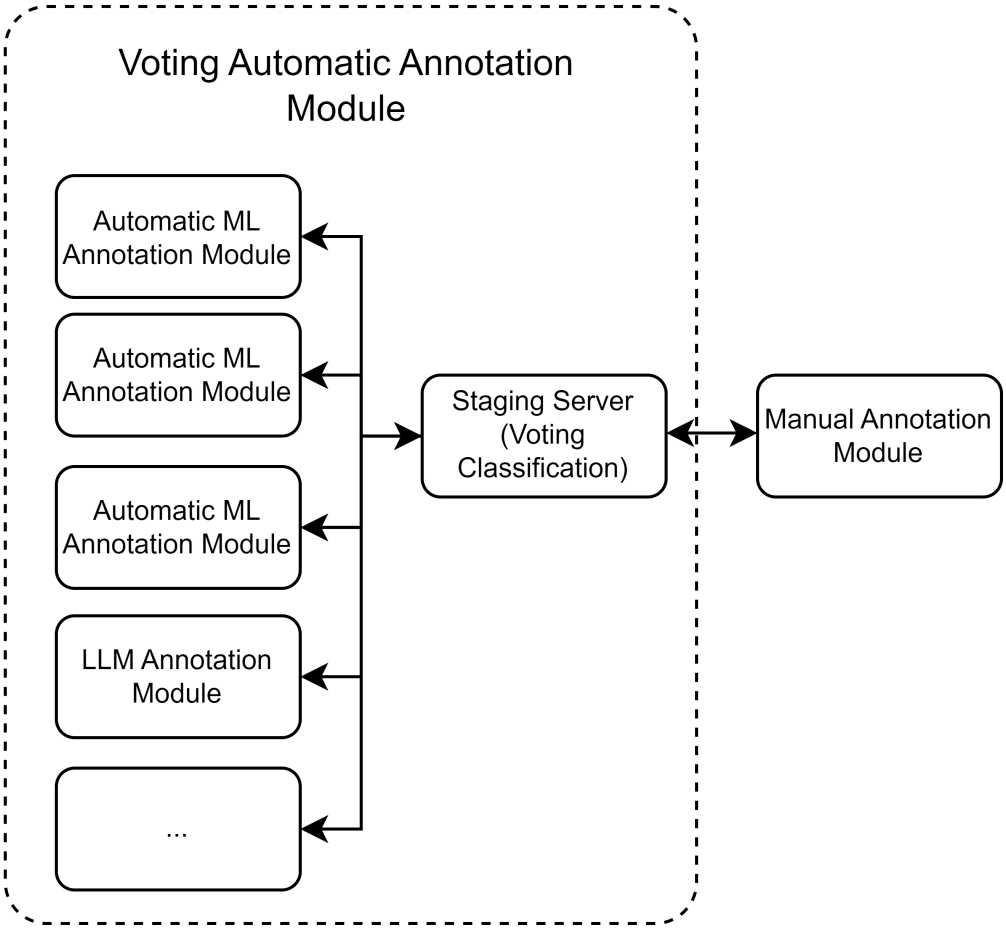


Figure 14: Flow diagram of a possible (but process-heavy) backend of voting classifiers.

5 Evaluation

The Assis tool was used during its development to annotate real datasets from companies, where its utility as a dialogue annotation accelerator was confirmed. Two case studies and a third validation battery were conducted, annotating data in three different ways: using Brat[13], Manual Annotation, and Assis, totaling over 96 dialogues with an average of 7 turns each, comprising almost 800 annotated messages from clients and agents.

- **Case Study 1 (toll company):** 13 company partner users. 6 university users. All familiar with Information Systems (working or researching in the field). In this first study, only 1 declared having basic experience with annotation tools before.
- **Case Study 2 (customer service):** 15 company partner users, 8 of them from the previous team. 10 university users, 6 of them from the previous team, now with basic experience with annotation tools.
- **Validation:** After the case studies were completed, to evaluate the metrics, two sets of dialogues from the first case study and 2 sets from the second case study were anonymized (had information such as names, addresses, monetary values, locations replaced with other fictional information) and left as test data, denoted as A1, A2, B1, B2. Then, 26 volunteers were called among university students. None of them had previous experience, with 11 having basic knowledge in Information Systems, 8 with medium knowledge, and 7 with advanced knowledge (having previously developed tools in I.S.).

The 26 volunteers were placed in an online call with an evaluator (from the tool development team to observe the usage times of each tool but without the power to intervene in the user's journey) and with access to the 4 anonymized sets and a list of intentions and entities.

They had the tasks of:

1. Access the tools (in the case of Assis, connect to the website; in the case of Brat, connect to the website and install it; and in the case of manual annotation, choose the preferred native tool).
2. A first test was performed, and the task was to open a dialogue set in the tool.
3. Annotate the dialogues with intentions and entities (in this case, there is no evaluation of the "correctness" of the annotation. The user decides when they believe they have finished annotating). Repeating the same process for the three tools, varying which tool would be the first and which the last (to mitigate the speed gain that users have by "getting used to" the package data). The 4 sets were also presented in different orders to users (for example [A1, B1, A2, B2] or [B1, B2, A1, A2], etc.), to avoid favoring a tool with an "easier" package.
   (a) The first set was done in the following order: Manual Annotation, Brat, Assis.
   (b) The second set was done in the order: Brat, Assis, Manual Annotation.
   (c) The third set was done in the order: Assis, Manual Annotation, Brat.
   (d) The fourth set was done in the order: Manual Annotation, Assis, Brat.

   The end of annotation was recorded by the user notifying the evaluator, for example, by opening the microphone and saying "finished."

To evaluate the time gain, the following tasks were used:

- Installation/opening time: Time from task 1 to the end of task 2.

- Annotation time: Time from uploading/opening a dialogue in the tool until the end of annotation.

While the average time to open files was 25 seconds for both Assis and Manual Annotation, the average time for users with Brat was 36 minutes - since users who are not used to running Python files (especially the 11 volunteers with little knowledge in I.S.) need to read the instructions manual, look for how to open the command prompt, try to understand errors until they can execute.

Table 2: Average annotation time for dialogue sets

	Assis (min)	Manual (min)
Set A1	13:12	26:09
Set A2	21:29	34:05
Set B1	19:11	39:32
Set B2	31:26	45:16

Table 2 demonstrates that the results of both Brat and Assis tools are much faster than manual annotation. A satisfaction questionnaire with 11 questions was applied for the three case studies, evaluating user experience aspects, with 37 responses. While Brat and Assis obtained similar results in certain questions (For *"This tool helps reduce annotation time significantly"* and *"This tool helps reduce the number of annotation errors"*, both tools received about 90% of *"yes"* responses).

On the other hand, all evaluated users (37) stated that they would use Assis for daily work if needed for annotating data, while only 18 claimed the same for Brat. Finally, out of the 37 users, 33 preferred Assis over all the analyzed tools.

6 Discussion

The target users of Assis are those who would typically configure a similar annotation tool, so the technical skills required to use Assis should not be a problem for its target users. There are other annotation tools that incorporate active learning, providing a user interface through which settings can be changed and actively selected batches of annotations can be generated [29]. A study states that user and system action annotations provide a semantic representation of their respective utterances. Simple checks on data annotation help prevent typographical errors and value paraphrase issues, requiring the creation of an ontology or schema before data collection, listing the interface for all domains and APIs. This form should identify categorical slots, which have a fixed set of possible values, and the annotation interface should enforce the accuracy of these slots [30]. [31]

Pre-annotation has been shown to increase annotation speed without introducing bias [32], and data annotation where pre-annotation was successfully applied [33]. There are also cases where high-quality pre-annotations have proven generally successful, with an increase in annotation speed and overall annotation quality, but where pre-annotations reduced annotator attention and resulted in bias. The contribution of this research establishes prerequisites and prepares for the creation of fully annotated datasets that can be used in training NLP models in different contexts and languages.

One of the main limitations in the study is the lack of a more robust evaluative methodology. One existing bias is the selection bias since a function (propagation of entities/intentions) was created between case studies 1 and 2, based on the feedback from case study 1. Another existing bias is the fact that Assis is a tool with support in Portuguese, so even though all volunteers declared being comfortable using a tool in English - the ease of the native language ends up giving an advantage in understanding the instructions' time. Another problem is that the metrics/tasks used (installation time, annotation time using 4 randomly ordered dialogues) were created by the Assis team. This can generate selection biases in the observed characteristics.

An example would be a company with shared computers whose users do not need to complete the installation task.

Active Learning is an area prone to expansion. In the current methodology, we did not evaluate the "correctness" of the suggestions given by the annotating Artificial Intelligence (for example, the ratio of suggestions accepted by the operator compared to those rejected) because this evaluation is not essential for the tool's use - but it is definitely of interest for future work.

A potential application of the ASSIS tool is in annotating conversations obtained from Internet forums. This includes dialogues from different domains mined from publicly available websites on the network [34]. Training language models requires large amounts of data. Specifically, for conversational models, the demand for data is even greater.

In the process of specializing these models in particular tasks, data is expected to be annotated with respect to intentions and entities. Thus, a suitable methodology and software tool for annotation are crucial. This can be achieved effortlessly through the DAB methodology, as it is oriented towards dialogues. The implemented software tool allows annotation focused on this specific type of data.

The Assis tool can modularly implement existing features in other tools without local deployment and innovating annotating using topics and propagation. It demonstrates potential for future innovations, such as new backend modules for Active Learning. Finally, the Assis tool and its documentation are available under the open-source MIT license.

7 Conclusion

Addressing the challenge of generating well-annotated conversational datasets, especially in non-English languages like Portuguese, remains a significant research gap. In response to limitations identified in existing tools, we developed a comprehensive software tool to aid manual human annotation of written dialogues. Our solution incorporates an active learning module, refining annotation suggestions over time. The software tool was then applied in practical scenarios to annotate conversations recorded in Portuguese, contributing valuable datasets for future Natural Language Processing (NLP) applications. Quantitatively, the tool demonstrated enhanced annotation speeds, and qualitatively, user satisfaction with the provided tools was confirmed. Moving forward, our research aims to utilize these annotated datasets to train new models and refine existing language models, with the ultimate goal of applying and evaluating such models in constructing customer service chatbots.

References

[1] Hannes Max Hapke, Hobson Lane, and Cole Howard. Natural language processing in action, 2019.

[2] Marwan Omar, Soohyeon Choi, DaeHun Nyang, and David Mohaisen. Robust natural language processing: Recent advances, challenges, and future directions. *arXiv preprint arXiv:2201.00768*, 2022.

[3] Bonan Min, Hayley Ross, Elior Sulem, Amir Veyseh, Thien Nguyen, Oscar Sainz, Eneko Agirre, Illana Heinz, and Dan Roth. Recent advances in natural language processing via large pre-trained language models: A survey. *arXiv preprint arXiv:2111.01243*, 2021.

[4] Jacob Eisenstein. Natural language processing, 2018.

[5] Wen Zhang, Heng Wang, Kaijun Ren, and Junqiang Song. Chinese sentence based lexical similarity measure for artificial intelligence chatbot. In *2016 8th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, pages 1–4. IEEE, 2016.

[6] Riccardo Coppola and Luca Ardito. Quality assessment methods for textual conversational interfaces: A multivocal literature review. *Information*, 12(11):437, 2021.

[7] Maria Skeppstedt, Carita Paradis, and Andreas Kerren. Pal, a tool for pre-annotation and active learning. *Journal for Language Technology and Computational Linguistics*, 31(1):91–110, 2017.

[8] Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N Bennett, Kori Inkpen, et al. Guidelines for human-ai interaction. In *Proceedings of the 2019 chi conference on human factors in computing systems*, pages 1–13, 2019.

[9] Erinc Merdivan, Deepika Singh, Sten Hanke, Johannes Kropf, Andreas Holzinger, and Matthieu Geist. Human annotated dialogues dataset for natural conversational agents. *Applied Sciences*, 10(3):762, 2020.

[10] Alan Ritter, Colin Cherry, and Bill Dolan. Data-driven response generation in social media. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2011.

[11] Henrique Theodor Schutz Foerste, Andreis Gustavo Malta Purim, Rafael Roque Souza, and Julio Cesar Dos Reis. Assis: Online semi-automatic dialog annotation tool. In *Proceedings of the XIX Brazilian Symposium on Information Systems*, SBSI '23, page 37–44, New York, NY, USA, 2023. Association for Computing Machinery.

[12] Sophia Ananiadou and Jun'ichi Tsujii. stav: text annotation visualiser. 2012.

[13] Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. Brat: a web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107, 2012.

[14] Seid Muhie Yimam, Iryna Gurevych, Richard Eckart de Castilho, and Chris Biemann. Webanno: A flexible, web-based and visually supported system for distributed annotations. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 1–6, 2013.

[15] Thomas S Morton and Jeremy LaCivita. Wordfreak: an open tool for linguistic annotation. In *Companion Volume of the Proceedings of HLT-NAACL 2003-Demonstrations*, pages 17–18, 2003.

[16] Minh-Quoc Nghiem, Paul Baylis, and Sophia Ananiadou. Paladin: an annotation tool based on active and proactive learning. *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 238–243, 2021.

[17] Jan-Christoph Klie. Inception: Interactive machine-assisted annotation. In *DESIRES*, page 105, 2018.

[18] Hiroki Nakayama, Takahiro Kubo, Junya Kamura, Yasufumi Taniguchi, and Xu Liang. doccano: Text annotation tool for human, 2018. Software available from https://github.com/doccano/doccano.

[19] Jie Yang, Yue Zhang, Linwei Li, and Xingxuan Li. Yedda: A lightweight collaborative text span annotation tool. *arXiv preprint arXiv:1711.03759*, 2017.

[20] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[21] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*, 2015.

[22] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[23] Fábio Souza, Rodrigo Nogueira, and Roberto Lotufo. Bertimbau: pretrained bert models for brazilian portuguese. In *Brazilian Conference on Intelligent Systems*, pages 403–417. Springer, 2020.

[24] Kaleem Razzaq Malik, Muhammad Asif Habib, Shehzad Khalid, Mudassar Ahmad, Mai Alfawair, Awais Ahmad, and Gwanggil Jeon. A generic methodology for geo-related data semantic annotation. *Concurr. Comput.*, 30(15):e4495, August 2018.

[25] Lea Canales, Walter Daelemans, Ester Boldrini, and Patricio Martínez-Barco. Emolabel: Semi-automatic methodology for emotion annotation of social media text. *IEEE Transactions on Affective Computing*, 13(2):579–591, 2022.

[26] Matheus Sanches, Jader C. de Sá, Allan M. de Souza, Diego Silva, Rafael R. de Souza, Julio Reis, and Leandro Villas. Mccd: Generating human natural language conversational datasets. In *Proceedings of the 24th International Conference on Enterprise Information Systems*. SCITEPRESS - Science and Technology Publications, 2022.

[27] Pawel Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Inigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. Multiwoz - a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *arXiv preprint arXiv:1810.00278*, 2018.

[28] Fredrik Olsson. *Bootstrapping named entity annotation by means of active machine learning: a method for creating corpora*. PhD thesis, 2008.

[29] Kostiantyn Kucher, Andreas Kerren, Carita Paradis, and Magnus Sahlgren. Visual analysis of text annotations for stance classification with alva. In *EuroVis (Posters)*, pages 49–51, 2016.

[30] Xiaoxue Zang, Abhinav Rastogi, Srinivas Sunkara, Raghav Gupta, Jianguo Zhang, and Jindong Chen. Multiwoz 2.2: A dialogue dataset with additional annotation corrections and state tracking baselines. *arXiv preprint arXiv:2007.12720*, 2020.

[31] Matheus Ferraroni Sanches, Jáder M. C. de Sá, Allan Mariano de Souza, Diego A. Silva, Rafael Roque de Souza, Júlio Cesar dos Reis, and Leandro A. Villas. MCCD: generating human natural language conversational datasets. In Joaquim Filipe, Michal Smialek, Alexander Brodsky, and Slimane Hammoudi, editors, *Proceedings of the 24th International Conference on Enterprise Information Systems, ICEIS 2022, Online Streaming, April 25-27, 2022, Volume 2*, pages 247–255. SCITEPRESS, 2022.

[32] Todd Lingren, Louise Deleger, Katalin Molnar, Haijun Zhai, Jareen Meinzen-Derr, Megan Kaiser, Laura Stoutenborough, Qi Li, and Imre Solti. Evaluating the impact of pre-annotation on annotation speed and potential bias: natural language processing gold standard development for clinical named entity recognition in clinical trial announcements. *Journal of the American Medical Informatics Association*, 21(3):406–413, 2014.

[33] Daniel Albright, Arrick Lanfranchi, Anwen Fredriksen, William F Styler IV, Colin Warner, Jena D Hwang, Jinho D Choi, Dmitriy Dligach, Rodney D Nielsen, James Martin, et al. Towards comprehensive syntactic and semantic annotations of the clinical narrative. *Journal of the American Medical Informatics Association*, 20(5):922–930, 2013.

[34] Matheus Ferraroni Sanches, Jáder MC de Sá, Allan Mariano de Souza, Diego A Silva, Rafael R de Souza, Júlio Cesar dos Reis, and Leandro A Villas. Mccd: Generating human natural language conversational datasets. In *ICEIS (2)*, pages 247–255, 2022.

A User Evaluation Questionnaire

For the case studies, the form presented in Table 3 was applied to compare different tools. For user accuracy, previous experience was explained as:

- **Low proficiency**: "For example, casual use of computer tools without understanding or developing systems"
- **Knowledge and/or intermediate proficiency**: "For example, I've studied programming, but I don't consider myself to have complete knowledge in the field"
- **Advanced proficiency**: "For example, I study or work in the field of computing and find it easy to understand how tools operate"

The form is presented in its original portuguese language.

Table 3: Questionário de avaliação de ferramentas anotadoras

Pergunta	Respostas
1. Como você avalia sua experiência prévia com informática, usando ferramentas ou plataformas:*	Baixo / Médio / Avançado
2. Você tem experiência prévia com ferramentas anotadoras (seja de texto, imagem, etc...) para fins de Inteligência Artificial?	Sim / Não
3. As informações disponíveis no site satisfatórias para instalar e/ou entender o funcionamento da ferramenta	Sim / Não
4. Instalar/acessar a ferramenta é amigável e fácil e foi feito sem maiores dúvidas	Sim / Não
5. A interface da ferramenta é amigável e é fácil de entender	Sim / Não
6. Não tive problemas navegando entre os diferentes menus para visualizar os dados dos dialogos	Sim / Não
7. Criar novas entidades e/ou intenções é fácil e intuitivo.	Sim / Não
8. Essa ferramenta ajuda a reduzir o tempo de anotação de forma significativa	Sim / Não
9. Essa ferramenta ajuda a reduzir a quantidade de erros de anotação	Sim / Não
10. Se eu trabalhasse com anotação, eu consideraria essa ferramenta para meu uso diário	Sim / Não
11. Ao final da avaliação, você prefere esta ferramenta do que as outras testadas.	Sim / Não