Implementation of current state-of-the-art technology for obtaining a working Dialogue Agent

Daniel Oklesinski, Wojciech Jaworski
oklesinskidaniel@gmail.com, wjaworski@mimuw.edu.pl

Institute of Informatics, Warsaw University
Banacha 2, 02-097 Warsaw, Poland
Selidor, sp. j.
Łomiańska 20b, 01-685 Warsaw, Poland

Abstract

In this paper, we describe the solution used to create a bot that is able to book services within existing booking system Reservis by conversing with a user in the Polish language. A task-oriented dialogue system that is a type of an expert system has been created. In addition to fulfilling basic functionalities, two other goals were established: to make human-computer interaction as natural as possible and not to limit user's choice by making inexplicit assumptions (and therefore showing only part of available results). In comparison to most commercial conversational agents, the described dialogue manager is strictly goal-oriented and is designed to perform a complex task through maintaining a multi-turn conversation.

1. Introduction

While first chatbot ELIZA has been created more than 60 years ago (Weizenbaum, 1966), they have been used mainly for entertainment and research purposes. It was not until the recent development of the second branch of conversational agents, task-oriented dialogue managers, that dialogue systems have become ubiquitous (eg. Amazon Alexa, Microsoft's Cortana, Google Assistant, Facebook's M and Apple's Siri). Most of them are personal assistants. They are able to handle a variety of tasks (eg. texting, finding the nearest restaurant, checking the weather), but are generally unable to process many turns of dialogue - in 2014, Siri could process only one turn, in 2017 the number increased slightly (Jurafsky and Martin, 2019).

Multi-turns conversational agents that allow performing a complex task have started appearing only recently. Areas they are developed in include e-commerce (Asadi and Hemadi, 2018), movie booking (Li et al., 2016) and restaurant reservation (Bordes and Weston, 2016).

The main challenge was to create a solution that ensures that many types of situations would be understood and supported in the desired way. Because of that, an expert system based on a complex and exhaustive state space has been chosen. It allows for creating very specific responses based on current knowledge about the state of the conversation. By using nearly infinite state space, the system has mixed initiative. A client can follow the system's prompts or steer the conversation in another direction.

2. Background

Reservis is a universal booking system for businesses. It allows any business based on client service (eg. beauty parlours, clinics, car repair shops) to manage its own bookings in a complex web app. It is highly adaptable and configurable and allows business managers to control many aspects of the process of booking. For example, for every service, a list of data needed to register a visit can be defined (eg. telephone number or zip code). Customers can be blacklisted if the number of appointments at which they fail to appear exceeds the value specified by the manager.

The role of the dialogue manager is to enable potential customers to make a reservation in any business subscribed to Reservis. Therefore, the primary requirement is to ensure that an application will be able to handle a variety of categories and any service configuration settings. The second requirement is to maintain focus on a fluid human-computer interaction (HCI). The desired user's impression of the system should be that of a human as opposed to a variation of a form. The third requirement is to not only focus on fulfilling users' needs but also on maximising their satisfaction (by not restricting their choices). For example, if there are 100 dates that satisfy the user, rather than choosing a random one, the system should offer the user the possibility to narrow down their preferences on their own. The last requirement was that all interactions must be limited to (or be easily replaceable by) text messages as a speech recognition module may be added to the project in the future and dialogue must have a form of a natural conversation. Currently, the manager handles only the scenario of making a reservation. Cancelling, altering or asking about already made reservation is planned to be done. The manager informs user that they are talking with a bot.

3. System overview

A state of conversation is defined by its current knowledge, which consists mainly of pieces of information from the client's utterances and history of the dialogue manager's actions. Every turn, the client's utterance is parsed, disambiguated and merged into already acquired knowledge.

Raw query text made by a client is fed to the Natural Language Understanding module which returns structural representation of text content in JSON format. No context is given to the module and the assumption is that the model returns every possible correct interpretation of a given utterance. For example, the phrase `at 10` can mean both `at 10 a.m.` and `at 10 p.m.`

After disambiguating JSON data and updating inner knowledge, the system acquires data concerning availability of terms from Reservis database and chooses a reply according to the current state and the set of rules.

4. NLU subsystem

The natural language understanding unit for dialogue agent is implemented using Categorial Syntactic-Semantic Parser *ENIAM* (Jaworski and Kozakoszczak, 2016). Main features that distinguish it from typical NLU is the fact that parser performs deep syntactic and semantic analysis and an implementation of a fuzzy pipeline.

A fuzzy pipeline is an approach where the system doesn't disambiguate the output of every step immediately but makes a compact representation of the ambiguous outcome and passes it to the next stage. Structure sharing assures that the growth of the representation is polynomial with respect to the length of the sentence, although the number of interpretations growths exponentially. Partial disambiguation is performed by means of applying syntactic and semantic constraints, as well as applying domain knowledge to data at various stages of the processing. The fuzzy pipeline approach significantly increases the accuracy of the parser by means of eliminating the possibility of dropping the correct interpretation in the course of disambiguation.

Chciałbym	*umówić*	*córkę*
I-would-like	to-make-an-appointment	for-a-daughter

do	*fryzjera.*
to	a-hairdresser.

'I would like to make an appointment for a daughter to a hairdresser.'

Figure 1: Exemplary sentence

During the first stages of analysis, the text is represented as a graph whose edges correspond to tokens. For each interpretation of a token sequence, another alternative edge is being added. In this setting we perform tokenisation, lemmatization, morphosyntactic tagging, word sense annotation, abbreviation extension, MWE and symbolic expression recognition, subcategorization, semantic valence annotation. For example, the sentence of Fig. 1 will be represented at this stage as on Fig. 2.

nodes	orth	lemma	interp	category
1–2	chciał	chcieć	praet:sg:m1.m2.m3	Attitude
2–3	by	by	qub	Qub
3–4	m	być	aglt:sg:pri:imperf:nwok	Aglt
4–5	umówić	umówić	inf:perf	Action
5–6	córkę	córka	subst:sg:acc:f	Person
6–7	do	do	prep:gen	Prep
7–8	fryzjera	fryzjer	subst:sg:acc.gen:m1	Profession

Figure 2: Token graph

We use several lexical resources such as Grammatical Dictionary of Polish (SGJP) (Zygmunt et al., 2017), TERYT (the database of Polish toponyms) (TER, 2018), lists of Polish first names and last names. Lemmatization is combined with word sense annotation: if there is at least one known lemma among deduced lemmata, the unknown lemmata are being discarded. If none of the lemmata is known, all of them are returned. Instead of recognizing named entities we add concepts like *FirstName*, *LastName* or *StreetName* to ontology and treat them in the same way as common words. We do not apply ML-based methods for word sense annotation since even if the system could correctly guess the sense of the word it still would be unable to use it. For example, in order to use the name of the street one needs to know where it is located.

We created a syntactic and semantic valence dictionary dedicated to the reservation domain (Fig. 3). In comparison to Walenty (Przepiórkowski et al., 2014), the general-purpose valence dictionary, our lexicon has much fewer entries, but describes them in a more precise way. The slots in the syntactic schemata in Walenty correspond to traditional argument positions (such as subject or object). Our dictionary covers all dependents for described lexemes. The other difference is that Walenty describes selectional restrictions in terms synsets taken from Słowosieć (Maziarz et al., 2016) and our lexicon uses domain ontology.

```
lemma=chcieć,pos2=verb: Attitude:
  subj,controller{pro+np(str)}:Agnt[Person] *
  sit{infp}:Thme[Action,Service];
lemma=umówić|umawiać|zapisać,pos2=verb: Action:
  subj{null+np(str)}:Agnt[Person] *
  {refl+np(str)}:Ptnt[Person,Animal] *
  {null+prepnp(do,gen)+prepnp(z,inst)}:Doer[Person] *
  {null+prepnp(na,acc)+prepnp(do,gen)}:Thme[Service] *
  {null+xp+np(nom)+np(gen)}:Tim[Time];
```

Figure 3: Simplified entries in valence dictionary.

4.1. Syntactic Analysis

The next stage of the processing pipeline is the construction of the syntactic tree (dependency tree, see Fig. 4) done using Type Logical Categorial Grammar (TLCG) (Morrill, 2010). The choice of TLCG instead of the standard CCG (Bos et al., 2004) is motivated by the wider choice of connectives that express important information, like feature vectors, ambiguity or polymorphism.

The full lexicalization of categorial grammars facilitates integration with the resources: morphological and valence dictionaries. The categorial lexicon is generated dynamically for each query from the graph obtained as a result of previous processing stages.

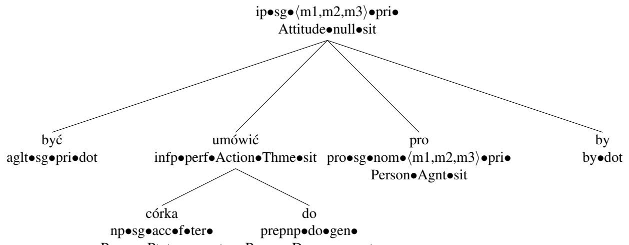

Figure 4: The syntactic tree for the exemplary sentence.

To parse a sentence is to give a proof in a non-commutative intuitionistic linear logic, therefore fragments of the proof system can be implemented to obtain quick parsers with guaranteed correctness. The parser is based on the CKY algorithm and is a direct implementation of a fragment of the proof system for linear logic. The parser generates results in a form of packed forest, a representation is polynomial with respect to the length of the query.

The grammar that includes selectional restrictions based on domain knowledge deals fairly well with ambiguity. However, there is one kind of ambiguity that cannot be resolved that way: the ambiguity of coordination arguments scope. In order to resolve this, we introduced a heuristics that prefers interpretation where coordination conjunct nodes deeper in the syntactic tree. For example `Monday or Wednesday afternoon` will be bracketed as follows: `[Monday or Wednesday] afternoon`.

4.2. Semantics

Semantic representation is created in two stages. First semantic graphs are built upon dependency trees (Fig. 5).

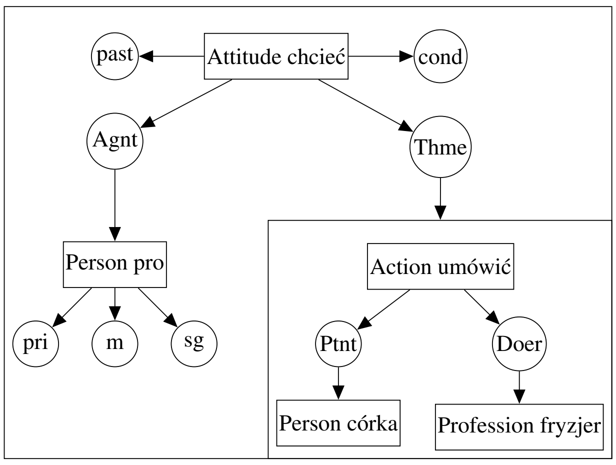

Figure 5: Semantic graph

This stage is done as direct translation without the use of additional resources. The lexemes together with their ontological categories became nodes, while semantic roles (relations between provided by valence dictionary) became labels of edges. Some nodes of dependency tree such as non-semantic prepositions are dropped.

```
[Attitude *1] ==> JObject {client_declaration: ?1}: 2;
Attitude *1 {past,cond,...} ==> ?1: 2;
Attitude *2 {Thme: [Action *1]} ==>
    JArray „and” [?1,?2]: 2;
Attitude *1 {Agnt: Person „pro” {m,pri,sg},...} ==>
    JArray „and” [?1, JObject {client_data:
        JObject {gender: JString „m”}}]: 2;
Attitude *1 {} ==>
    JObject {action: JObject {attitude: JString ?1}}: 2;
Action *2 {Ptnt: *1,...} ==>
    JArray „and” [JObject {patient: ?1},?2]: 2;
Action *2 {Doer: *1,...} ==>
    JArray „and” [JObject {doer: ?1},?2]: 2;
Action *1 {} ==>
    JObject {action: JObject {name: JString ?1}}: 2;
```

Figure 6: Inference rules

As the second stage of semantic processing we apply inference rules to the graph (Fig. 6). These rules matches with fragments of graphs and replaces them with other fragments (rule productions). Rules serves for several purposes:

- they implements synonymy relation (between words and MWE);
- they unify representation of numerals and month names written using digits with those written using words;
- finally they translate the graph into a JSON tree.

After the translation, semantic representation is treated as an algebraic expression. A canonical form of representation is produced using associative, commutative and distributive properties. This process results in further ambiguity reduction.

Finally, semantic representation is exported to JSON (Fig. 7).

```
{"text": "Chciałbym umówić córkę do fryzjera.",
 "client_declaration": {
   "action": {
     "attitude": "chcieć",
     "name": "umówić"},
   "client_data": {
     "gender": "m"},
   "doer": {
     "profession": "fryzjer"},
   "patient": {
     "person": "córka"}}}
```

Figure 7: Semantic representation in JSON

5. Dialogue Manager module

5.1. Communication with NLU

JSON data extracted from client's utternace by NLU module returns is formatted according to predefined information schema.

The core of schema consists of 9 attributes that can be recognised by NLU module. They are:

- `action`: the intent of a client in a current utterance,
- `service`: what kind of service should be performed,
- `time`: when the service should be performed,
- `location`: where the service should be performed,
- `organisation`: in what firm the service should be performed,
- `doer`: who should perform the service,
- `patient`: on whom the service should be performed,
- `price`: what price should the service have,
- `rating`: what rating should the service have.

Every attribute is additionally parametrized (eg. `location` can include amongst other `town`, `quarter`, `street`, `street_name` parameters). The schema also allows modifiers of these core attributes (such as `before` or `aprox`) and operations on them (such as conjunction or alternative).

5.2. Disambiguation

Disambiguation consists of deciding which interpretation of an utterance is correct. The system is able to handle this problem by using one of three available solutions.

1. **Using the context of the previous input.** System responses include meta-information about what kind of semantic category can be expected in response. If any NLU interpretation matches to the expected category, the system selects this interpretation as a correct one. For example, a client's utterance `10` can be interpreted amongst others as an hour, a day of a month, a price or rating value. However, if it occurred after the system's question about an hour, the hour interpretation is chosen as a correct one.

2. **Using the probability of options.** Any interpretation that is impossible in the current context (eg. interpreting `10` as a year) can be ruled out.

3. **Asking the client to clarify.** If the aforementioned methods fail, the system just asks the client to clarify what they meant.

Combining these three methods allows to make safe assumptions and resort to client's clarification only when absolutely necessary.

```
{"text": "o 6:32",
 "client_declaration": {
   "time": {
     "at": {
       "hour": {"with": [18, 6]},
       "minute": 32}}}}.
```

Figure 8: Ambiguity representation in JSON

5.3. Merging

Merging new piece of information with the old ones is a challenging task to formalise. Humans perform this part by intuitive guesses. For example, if at the beginning the client defined their time preference as `Wednesday, after 6 p.m.`, but later added `Thursday`, there are 4 possible options:

- `Wednesday, after 6 p.m.; Thursday, after 6 p.m.`,
- `Wednesday, after 6 p.m.; Thursday`,
- `Thursday, after 6 p.m.`,
- `Thursday`.

For now, a simple heuristic is used. Depending on whether the system in the current state wants to expand or limit user's preferences both attributes are summed or crossed. If they are to be crossed and the intersection is empty, then the older attribute is crossed out and only the newer one holds.

Although the space of responses from the NLU module is finite, the size of it is so big, that effectively the number of possible states is infinite. With such a design, nearly every possible state of conversation can be encoded in it. With the assumption of perfect NLU module, the only limitation is information schema. If some information from the client cannot be put in the schema, it cannot be merged into the dialogue state.

5.4. Expert system

The system, after analyzing current dialogue knowledge and data from Reservis database, decides what action to take. Then, a reply is generated (for every action, there is a predefined set of answers parametrized by user's gender and a configuration setting of a firm). Replies can be either open questions, specifying questions (accompanied by a finite set of suggested responses) or acknowledgements. The system infers both client's and business' demands and constraints from possessed data.

One of the main factors in decision making is the current aim of conversation (eg. to determine what service the client seeks, to present available times or to confirm the whole process). However, the system does not hold an initiative the whole time. The client can in any moment change a topic, revert the previous statement or retract to the previous point in the conversation. Thus, the dialogue, like in natural conversation, has mixed initiative.

The system seeks not to make decisions for the client. If necessary, it enquires for additional data, so it can present the best results. It also takes into account which medium of communication is used, as different media require different dialogue flow.

The multitude of known data allows to handle accurately any corner case and generate very specific and tailored to circumstances responses. For example, if the client defined their service and time preferences, the system may:

- present available times (if there are only a few results),
- ask for another parameter/preferable hour/preferable day of the week (if there are too many fitting results and they will be substantially reduced by this question),
- inform that the value of the time parameter does not intersect with the period in which booking is allowed (is too early/too late),
- inform that the value of the time parameter is invalid (eg. 30th February).

6. Other polish dialogue managers

According to digital transformation report (K2, 2018), most conversational agents available in the Polish language are only slightly more advanced versions of forms.

6.1. Goals of chatbots

The goal of nearly all available chatbots is to provide information to users. They are three main subsets of these bots.

Answering questions. These bots are created to answer simple (but often asked) questions from users. For example, messenger bot of WOŚP (polish charity event) is able to give information on how to become a volunteer or what is the current status of donations (WOS, 2018). Although very simple, it was able to independently conduct 64% of conversations during the event in 2018.

Newsletters. Using these bots, a user can subscribe to a newsletter and receive it via bot communicator. For example, you can ask the bot of Wirtualna Polska (one of the biggest Polish media company) to send you daily press briefing (WP, 2018).

Redirecting clients. This set of bots recognizes a need of client and redirects them to either website, another information system or a human being. In the Multikino (Polish cinema chain) bot, you can choose a film title, cinema and screening hour and then you are redirected to a seating plan on Multikino's website (Mul, 2018).

It seems that Reservis bot will be one of the first bots on Polish market that can perform substantial actions such as the full process of booking.

6.2. Dialogue flow

Most available chatbots do not aim to imitate natural conversation. They maintain the conversation by suggesting responses (used by 94,(4)% chatbots from the report), implementing NLU (66,(6)%) and machine learning (50%)

Quick replies. Most bots are integrated with Messenger on Facebook and make heavy use of its quick replies feature (set of suggested replies is presented to the client). It gives initiative completely to the bot. Although it does not necessarily lower the quality of user experience, it does substantially limit the possibilities of such bots.

NLU modules. To balance the limitations of quick replies, many bots also implement simple NLU modules. However, these modules are able to extract only basic pieces of information.

7. Conclusion

While there is still a lot of space for progress and development, already in its present form, the presented bot stands out in many ways in comparison to products available on the Polish market.

Although it does not use machine learning, by having a complex state space and being an elaborate expert system, it is able to handle very specifically many various points in a conversation. Having a mixed initiative, the flow of conversation is more natural than in most existing solutions. While extending bot does require work, it is easy to add precisely new features.

We plan to evaluate the system by producing a large collection of semirandomly generated conversations beetween the bot and an user (based on possessed corpora) and grading each conversation (in terms of achieving a communication success). Furthemore, testers will be used to evaluate a demo version of the system.

Acknowledgments

This work was co-financed by the European Regional Development Fund under the Regional Operational Programme for Mazowieckie Voivodeship 2014-2020 project RPMA.01.02.00-14-5681/16-00.

8. References

2018. Multikino S.A. chatbot. `http://m.me/MultikinoPolska`.

2018. Raport. Polskie chatboty 2018. `http://eteryt.stat.gov.pl`.

2018. Rejestr TERYT. `http://eteryt.stat.gov.pl`.

2018. Wielka Orkiestra Świątecznej Pomocy chatbot. `http://m.me/WOSP`.

2018. Wirtualna Polska chatbot. `http://m.me/WirtualnaPolska`.

Asadi, Amir Reza and Reza Hemadi, 2018. Design and implementation of a chatbot for e-commerce.

Bordes, Antoine and Jason Weston, 2016. Learning end-to-end goal-oriented dialog. *CoRR*, abs/1605.07683.

Bos, Johan, Stephen Clark, Mark Steedman, James R Curran, and Julia Hockenmaier, 2004. Wide-coverage semantic representations from a ccg parser. In *Proceedings of the 20th international conference on Computational Linguistics*. Association for Computational Linguistics.

Jaworski, Wojciech and Jakub Kozakoszczak, 2016. Eniam: Categorial syntactic-semantic parser for polish. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*.

Jurafsky, Dan and James H. Martin, 2019. Speech and language processing (3rd ed. draft). `https://web.stanford.edu/~jurafsky/slp3/`.

Li, Xiujun, Zachary C. Lipton, Bhuwan Dhingra, Lihong Li, Jianfeng Gao, and Yun-Nung Chen, 2016. A user simulator for task-completion dialogues. *CoRR*, abs/1612.05688.

Maziarz, Marek, Maciej Piasecki, Ewa Rudnicka, Stan Szpakowicz, and Paweł Kędzia, 2016. plwordnet 3.0–a comprehensive lexical-semantic resource. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*.

Morrill, Glyn, 2010. *Categorial grammar: Logical syntax, semantics, and processing*. Oxford University Press.

Przepiórkowski, Adam, Elżbieta Hajnicz, Agnieszka Patejuk, and Marcin Woliński, 2014. Extended phraseological information in a valence dictionary for NLP applications. In *Proceedings of the Workshop on Lexical and Grammatical Resources for Language Processing (LG-LP 2014)*. Dublin, Ireland: Association for Computational Linguistics and Dublin City University.

Weizenbaum, J., 1966. Eliza – a computer program for the study of natural language communication between man and machine.

Zygmunt, Saloni, Woliński Marcin, Wołosz Robert, Gruszczyński Włodzimierz, and Skowrońska Danuta, 2017. Grammatical Dictionary of Polish. `http://sgjp.pl`.