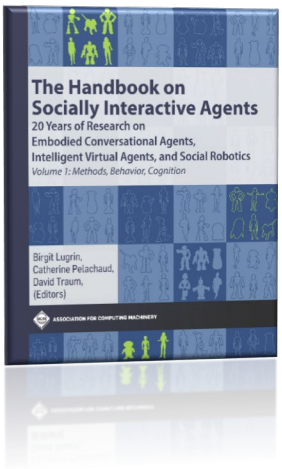
Natural Language Understanding in Socially Interactive Agents

Roberto Pieraccini



Author note:

This is a preprint. The final article is published in
"The Handbook on Socially Interactive Agents" by ACM books.

Citation information:

Pieraccini, R. (2021). Natural Language Understanding in Socially Interactive Agents. In B. Lugrin, C. Pelachaud, D. Traum (Eds.), *Handbook on Socially Interactive Agents – 20 Years of Research on Embodied Conversational Agents, Intelligent Virtual Agents, and Social Robotics*, Volume 1: Methods, Behavior, Cognition (pp. 147-172). ACM.

DOI of the final chapter: 10.1145/3477322.3477328

DOI of volume 1 of the handbook: 10.1145/3477322

Correspondence concerning this chapter should be addressed to Roberto Pieraccini, robertopieracc@gmail.com

5. Natural Language Understanding in Socially Interactive Agents

Roberto Pieraccini

5.1 Natural Language Understanding in Interactive Agents

Natural Language has fascinated and challenged research scientists and technologists from the early times of computer science. Claude Shannon's 1948 seminal paper on information theory [Shannon, C., 1948] uses language as an example of complex information and discusses its statistical properties based on the modern notion of n-grams. In 1950 Alan Turing [Turing, A., 1950] proposed what today is known as the Turing test, i.e. a criterion to determine the level of intelligence of a machine based on its capability to converse in natural language with humans. Since then there has been a proliferation of studies trying to unlock the complexity of natural language for various purposes, including machine translation, automatic summarization, questions answering, sentiment analysis, market intelligence, grammar checking, and of course Natural Language Understanding (NLU).

Besides the intellectual and scientific interest in it as a challenging AI problem, NLU occupies a prominent role in the realization of spoken language systems that belong to the category of virtual interactive agents. Figure 1 describes a classic *conversational* architecture for a virtual interactive agent. The user speaks to the system, where speaking may just be one of the modalities of interaction. Touch, gesture and images or text on a display are other typical modalities of input/output interaction that can be often combined with speech. However, in this chapter, we consider only speech input and output.

User's speech is converted by a speech recognizer (ASR) into its textual transcription, i.e. the string of words that were presumably spoken. For the sake of completeness, we need to be aware that modern speech recognizers can generate several alternative hypotheses ranked in terms of confidence, known as N-best, and even a graph of word hypotheses, generally called a *lattice*. Even though one can apply the techniques described here to alternative speech recognition transcriptions, and re-rank them based on combined speech and language scores, we will restrict our considerations here to a single, first-best string of word hypotheses.

The natural language understanding system (NLU) converts a sequence of words into some actionable symbolic representation of the utterance meaning. That representation is actionable in the sense that subsequent processes can use it to interact with the external digital world, for

instance through well defined APIs, and with the physical world through actuators (e.g. in home automation applications), or simply define and generate a response back to the user.

The function of the Dialog Manager, or DM., is deciding what the next action that agent should accomplish based on the meaning representation of the input utterance, and contextual knowledge, derived from instance from the history of the conversation, and from the knowledge of the user environment (e.g. her location, time of the day, etc.).

The generation of a response back to the user is one of the potential actions that the DM can initiate. The response can, for instance, provide a spoken answer or request more information from the user. The dialog manager would send a request to the language generation module (NLG) to generate a specific textual representation of the utterance to speak out. The speech generation, or text-to-speech (TTS) system would finally generate an utterance based on the textual representation provided by the NLG.

To summarize, NLU is thus a fundamental module of a conversational interactive agent that converts a raw textual transcription of an utterance into a symbolic representation that is used by the dialog manager to select and perform the next action. In this chapter we will provide more details on how NLU modules for interactive agents are built, and what the technologies involved, the problems, and the possible solutions are.

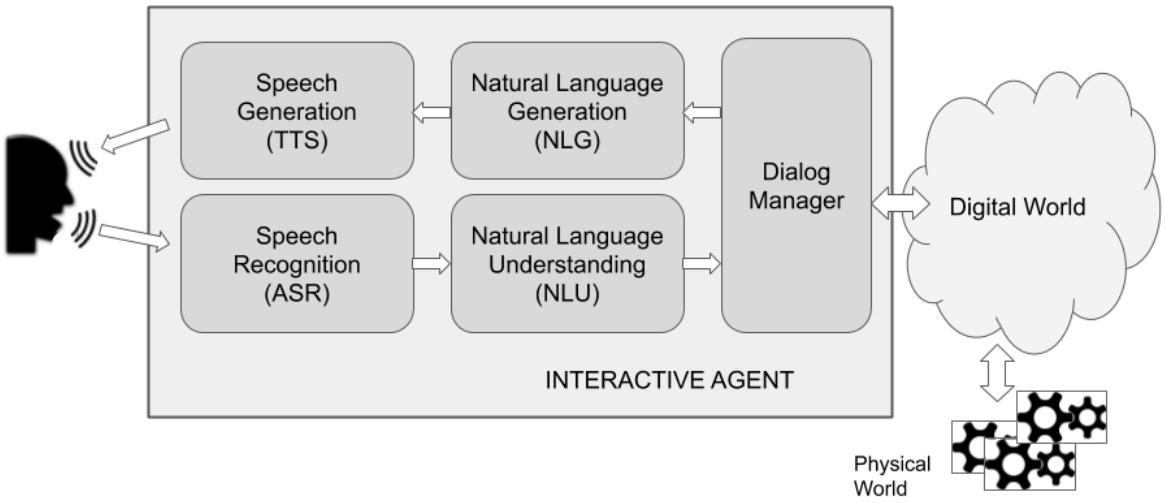


Figure 5.1: A reference architecture of a virtual interactive agent

5.2 NLU and Interactive Virtual Agent Features

A virtual agent, within the scope of this chapter, is a machine that is able to entertain a spoken conversation interaction for the fulfillment of a user goal, whether that goal be a well defined task, the answer to a question, or the satisfaction of a social function. Examples of modern commercial interactive agents include automated telephone customer assistance systems, and consumer personal virtual assistants, such as Siri, Alexa, Bixby, and the Google Assistant. While all of these agents do not have a body, there are a few examples of embodied agents. Embodied agents with a virtual body implemented as graphical avatars have been utilized to answer customer questions on websites, for instance for banking or travel. Research is providing advanced embodied agents, like Zara developed at the Hong Kong University of Science and Technology [Fung, 2016]. Finally, as far as embodied agents with a physical body, we should mention Jibo, a consumer robot that was shipped in 2017 [Jibo, WSJ]. Other examples of agents with a physical body are the robot Pepper produced by Softbank [Pepper] and FurHat [FurHat].

In general there are a couple of considerations that we should keep in mind when implementing NLU for an embodied agent, especially for a physically embodied one. First, depending on its function, a physical embodied agent may require the NLU system to be running fully embedded in the device. Imagine for instance a robot that may be in situations where the internet is not available, or where the latency requirements are very strict. Depending on the available computational capabilities of the device, the implementation of an on-device NLU may require some architectural and scope limitations with respect to a NLU that runs in the cloud. Moreover, a physically embodied agent is situated in the physical world, and thus it may be exposed to linguistic expressions that refer to the space around it. That is especially true when the device is capable, or presumed capable by the user, of visually understanding objects and gestures. The NLU system needs to be able ground spatially some referential expressions (e.g. l*ook here*, *take a picture of that object),* in the sense that they need to be resolved with the knowledge of the space around the device. Spatial grounding [Guadarrama], and grounding in general is beyond the scope of this chapter, and it is arguable whether, architecturally, it should be part of the NLU system or belong to a successive interpretation phase. Besides these specific issues, NLU for embodied and non embodied agents share similar requirements.

Table 1 shows a non-exhaustive taxonomy of request types, or features, that are common in today's personal virtual assistants where we can see a clear distinction between goal oriented and social interactions.

Goal Oriented	Home automation	Turn the lights on in the living room Start the air conditioning in the kitchen Lock the front gate
	Media Requests	Play Hello by Adele Play the Shawshank Redemption on my family room TV
	Personal Help	Set an alarm for tomorrow at 7am Remind me to take the garbage out this evening Set a timer for 10 minutes
	Personal Information	When is my next flight to London? What is my hotel in Paris next week? When is my next personal trainer appointment?
	Games	Let's play a trivia game
	Question answering	Who was the 23rd president of teh United States?
	Technical support	How do I change my email password? How do I change privacy settings?
	Third party applications	Make a hotel reservation in Geneva for this weekend Call a taxi for going to the airport Buy tickets for the Joker at the closest movie theater
Social	Fun	Tell me a joke Surprise me
	Chit-chat	How are you?

Table 5.1: A taxonomy of the type of requests fulfilled by a modern virtual assistant

A goal oriented interaction is generally characterized by a request targeted at the fulfillment of a concrete user need for information or for the execution of an action. Within that scope, different types of requests may have different natural language requirements and may pose different problems for the NLU and the successive stages of processing. In the rest of this section we will highlight some of the issues arising with the different types of features reported in Table 1.

NLU for home automation has some specific issues. The functionality of most of the devices are quite limited…(e.g. turn a device on off and change some of its characteristics). One of the issues with home automation requests is the identification of the specific device referred to by the request. Users generally give arbitrary names to the devices once they are installed on the home network. For instance one can distinguish between the “living room TV” and the “bedroom TV” or the “ceiling lights”, the “lights on the wall” or give them an arbitrary name, like “the foobar lights.” That use of arbitrary names to identify a specific device poses problems for

the speech recognizer (i.e. the names can be confused by the ASR with other similar sounding words, or they may not exist in the speech recognition vocabulary). Another problem is that users may not use the names associated with the devices in a consistent way. In any case, the NLU system needs to be able to account for arbitrary dynamic entities in the grammars and parsers that are user specific, and account for all possible variations that users may apply when they issue a command.

Media is a broad and complex area for a virtual interactive agent. That will cover the choice of music, videos and movies (on the appropriate device), podcasts, audiobook, etc. Referring to music, as an example, the user may request a specific title (e.g. “Play Hello”) , a title specifically interpreted by an artist (e.g. “Play Hello by Adele”), an album that may have the same name of a song (e.g. “Play Let it Be by the Beatles), a genre (e.g. play classical music), or a category (e.g. play piano pieces by Chopin), etc. Thus, the NLU issues related to media requests are generally related to the vast number of artists, albums, movies, and titles that need to be accounted for and the different selections, some of them ambiguous, that users may request. Ambiguity may be handled best at the level of dialog manager, so the NLU may propose a number of alternative candidates for the dialog manager to disambiguate by asking. Another serious problem with media is that generally people do not remember the exact title of songs, albums, and movies, but refer to them partially (e.g. “Play the Winding Road”) or by using a recurring phrase in lyrics. These are NLU grounding problems that need to be addressed specifically for this task.

Personal assistance, like setting timers and alarms, may be quite straightforward. However NLU and dialog complications arise with the interpretation of complex commands for changing existing objects, e.g.

> *User: Change my alarm to 7:30 am*
> *Assistant: Which one? You have 12 alarms.*
> *U: My morning alarm*
> *A: You have an alarm for 6:30 am, one for 7am, on for 8:30am one for … which one do you want to change?*

These are typical grounding problems that arise when dealing with potentially large lists of objects. Similar problems arise with personal information:

> *U: Where is my hotel?*
> *A: Which hotel?*
> *U: The one I am staying at tonight.*
> *A: Sorry, I did not understand you.*

General question answering is a special category that can rely on the power of existing search engines, like Google’s, that today can provide direct answers for most of the specific questions that users can ask on the vast number of entities on the Web. However, without relying on existing general solutions, building a general question answering system may require the

availability of a knowledge repository, such as a knowledge graph that can be either created in a manual manner, for small private domains (for instance using open source Knowledge Graph software [Data Cloud][neo4j] or derived automatically, or semi-automatically from an analysis of a collection of documents. The input query needs to be analyzed so as to determine the category of the requested answer (e.g. a who, what, where, when question) and the attributes of the requested entity (e.g. a person, an animal, a fact, etc.). Question answering is a specific area of Natural Language Processing that abstracts from the classic NLU paradigm used, for instance, as in the architecture of Figure 1.

Technical support and third party applications, including booking of events, travel reservations, order, etc., are quite more structured, and once triggered in a virtual assistant by a simple request, like “I want to make a restaurant reservation,” they fall into the category of directed dialog. The solution of directed dialog generally relies on well crafted prompts, as we will see later in this chapter. Most of these applications require some form filling logic, or FIA (form Interpretation Algorithm) [VoiceXML, 2004][Pieraccini et al., 2005][Stoyanchev et al., 2016],

5.2.1 NLU for not Goal Oriented Social Agent

The broad category of social interactions is generally not goal oriented, but targeted at providing conversational delight for the user, like jokes, fun facts, games, and conversational exchanges that can be characterized as chit-chat. One way to develop agents for social interactions consists in creating a repository of answers, generally written by creative writers. Table 2 shows examples of queries among the many that users may ask to a social interactive agent. To be interesting, a social interactive agent requires witty answers to thousands of those queries, and also it requires to maintain that content fresh, and have special seasonal answers, for instance in coincidence with public holidays, or large events, such as the Superbowl. The amount of potential queries can be so large, that creating and maintaining grammars, or training neural NLU systems can become prohibitive. Thus a solution to this could be based on typical information retrieval mechanisms. For instance one can create a repository of all potential queries, with links to the corresponding answers. When a user query comes, the retrieval algorithm can try to find the closest query in the repository and play the corresponding answer.

How are you?
When were you born?
Who built you?
Are you listening to me all the time?
Do you celebrate any holidays?
What is your favorite food?
Do you like me?

I love you!
You make me laugh!
May the force be with you

Table 2: examples of chit-chat requests to a social interactive agent.

Multi-turn social interactions may fall into the domain of social chat-bots. One of the most recent and successful examples of that is Meena, a chatbot recently developed by Google Brain [Adiwardana et al., 2020]. At a high level Meena is based on an end to end neural systems trained on a 40B word corpus of public social media conversations. An interesting characteristic of Meena is that it is optimized over a human-like metric that weights the sensibility and the specificity of each answer, and thus producing quite impressive results.

5.3 Developing NLU

The goal of an NLU module is to transform a natural language textual sentence, possibly the transcription of an utterance generated by a speech recognition system, into a non ambiguous symbolic representation of its meaning. In other words the problem of NLU is that of translating from natural language to a defined formal language, and thus it can be functionally considered analogous to that of machine translation. In fact some approaches to NLU that leveraged machine translation (MT) models have been proposed in the past, as described in [Della Pietra at al., 1997]. However, the major, and not insignificant difference between NLU and MT is that the latter acts upon defined and existing source and target natural languages (e.g. English and Italian), while the target language for NLU is not natural and needs to be created for any domain of use. The specification of the target formal language for a NLU system, in other words the meaning representation is often a non negligible part of the development of NLU.

A typical approach to the development of a meaning representation for NLU is based on the definition of an ontology of intents and their corresponding arguments.

Let's make a simple example. If we are building a system for home automation, one may consider intents that express the actions a user may want to accomplish, for instance:

```
light-control;
door-control;
TV-control;
temperature-control;
```

In order to be able to represent all the possible requests and act upon them to change the status of some home automation device, each one of these intents needs to be complemented by a number of arguments. Let's consider the following examples of user requests:

Turn the dining room TV on
Switch on the light in the living room and make it blue.

The following expressions are suitable representations of the respective meaning of the above requests. The include, at a symbolic level, all the information meaning that will be used by the subsequent stages, for instance by the dialog manager, to fulfill the user requests:

```
TV-control(target-status=on, location=dining-room);

light-control(target-status=on,
      location=living-room, target-color=blue);
```

The definition of the intents and their arguments is often called an *intent schema*. Although there have been attempts to learn ontologies automatically from large collections of text [Lourdusamy ert al., 2020], the most reliable approach for practical human-machine interaction systems still relies on crafting the intent schema based on the developer's knowledge of domain and the corresponding applications. As of today there is still not a universal standard ontology of intent schemas, so each system is generally engineered in a proprietary manner.

The design of intents and their level of granularity is somewhat arbitrary. For instance a different design criterion may have consolidated the four intents defined above into a single one, by adding an additional argument that specifies the device to be controlled, and device specific arguments. In that case the semantic representation of the above example sentences would be:

```
device-control(device=TV, target-status=on,
           location=dining-room);

device-control(device=light, target-status=on,
           location=living-room, target-color=blue);
```

The arbitrary nature of the intent schema design poses a problem for large systems, such as modern virtual assistants with an increasing number of features. Crafting and maintaining a collection of thousands of intents requires a well coordinated team effort. When new features are developed the team designated to maintain the intent ontology needs to decide whether to create new intents, or derive them by specializing existing ones. The intent specification team needs to dedicate special care to making sure there are no duplications and ambiguities.

Incidentally, It is important to consider the ambiguity that may be inherent to certain applications. For instance, consider the following query:

Switch the blue light off

The term “blue” here could refer to the color as well as the name given by the user to that particular light. So, the parsing of the query alone clouds not resolve that ambiguity unless the context is taken into consideration. For instance if there is the user actually has a device called “blue”, which happens to be a light, or if there is a light which happens to be of color blue at that particular moment. One way to resolve that is to leave the ambiguity at the NLU level and generate multiple parsing candidates. Successive stages of processing may select the correct candidate based on contextual knowledge (e.g. whether there is actually a light named “blue” or of characterized by a “blue” color)

When dealing with a large number of intents the creation of annotated corpora for testing and training can be complex. Corpus annotations are generally created via crowdsourcing. For a reasonably small number of intents, crowdsourcers can be easily trained to choose the correct intent for each utterance transcription that needs to be annotated. However, when the number of intents is large that operation is difficult and requires carefully designed annotation tools, and human annotators who become experts of the domain. Using a hierarchical approach, e.g. showing the crowdsourcers a choice of high level intent clusters, and letting them drill down to the specific ones, is a viable, but rather costly approach.

5.4 NLU and Interaction Modalities

The interaction modality of the system we are going to build has a strong influence on the type of NLU system we need to use. The rest of this section is devoted to the description of the different types of interaction modality, and the implication of that on NLU.

5.4.1 System initiated, directed prompts interaction

In this interaction modality, also denoted as *directed dialog,* the system always initiates the interaction by clearly prompting the user to elicit a specific and desired response. Prompts are designed, typically by Conversation designers or VUI (Voice User Interface) designers with the goal to constrain the user responses so as the NLU can have a higher chance to be able to understand a higher percentage of utterances [Cohen et al, 2004]. The directed dialog modality is typical of menu-based systems, and has been used quite extensively in telephone customer care services (also called IVRs, Interactive Voice Response systems). Directed prompts can be effectively used when the number of choices for the user is limited, generally between 3 and 5, the response belongs to a known closed set (e.g. city names, dates, times of day etc.), or it is a well defined and structured numeric value. (e.g. a date, credit card number, telephone number, etc.). In all these situations the transcription of an utterance performed by the ASR can be parsed by a well defined context free grammar.

Context free grammars (CFG) have been traditionally used for directed prompt interactions. In the early systems, grammars were integrated in such a way to provide a set of strong constraints for the speech recognition systems. In other words the recognizers were trying to match the input utterance to, and only to, the sentences and phrases described by the associated grammars. If the user said something different, the speech recognizer was still trying to match it to what was described by the grammar, and return the closest match, or a no-match. Modern recognizers do not need that constraint, since they are capable of recognizing speech with a high accuracy over vocabularies of millions of words. However, that constraint could be used to further reduce the error rate of the ASR by biasing the recognizer towards the expressions expected as a response to a prompt, and represented by a grammar. Commercial cloud ASR systems, like Google, provide an API to bias the recogniser towards expected responses. See for instance *SpeechContext* in [Cloud Google]. One could for instance suggest possible phrases that are likely to be spoken by the user based on the prompt. For instance if the prompt asks for a yes/no response, the recognizer could be biased towards *yes, no*, and their synonyms.

As an example, imagine we want to write a CFG grammar that would act as an NLU parser for the responses to the following prompt:

> *Do you want flight arrival, departure, or gate information?*

A grammar covering the potential responses may look as follows:

```
$menu-choice  = [$initial] [flight] (arrival | departure | gate)
[information]  [$final];
$initial = (I would like) | (I want) | (please give me);
$final = please | (thank you) | thanks;
```

The above grammar snippets follow a convention where symbols preceded by a `$` sign are non terminals (e.g. `$initial`) and square brackets represent optional elements (e.g. `[flight]`). There are many formats that have been used, and are still used to write CFG grammar. W3C, the World Wide Web Consortium, embraced several standards for interactive commercial systems, and SRGS (Speech Recognition Grammar Specification) [SRGS, 2004] is the one used almost universally in the industry. The convention used in this chapter is a simplified form inspired by ABNF, which is part of the SRGS specification.

The above grammar excerpt can parse the following sentences, and several others:

> *arrival*
> *gate information please*
> *I would like flight arrival information thank you.*

In order to be able to use a CFG grammar for NLU we need a way for it to generate symbols that represent the meaning of a sentence, for instance intents and arguments, as the result of a

successful parsing. One way to do that, that has been used in practice mostly for system-initiated directed-prompt interactions, consists of assigning external variables within the body of the rules. In the example above, only the first rule is carrying a meaning relevant for the application, and thus we can augment it by assigning a proper value to an existing external variable, let's call it `choice`, that is passed on to the dialog manager, as in the following example:

```
$menu-choice  = [$initial] [flight]
      (
            arrival {choice = arr-info;) |
            departure {choice = dep-info;} |
            gate {choice = gate-info;}
      ) [information]  [$final];
```

The CFG industry standards, such as SRGS, allow building sophisticated grammars with arbitrarily complex code (typically JavaScript) attached to the rules, and not just variable assignments. Code assigned to rules can be used by the developers, for instance, to perform string operations, calculation, or validate the values resulting from parsing an input utterance transcription. Some typical cases are aggregating natural number expressions into a number (e.g. "forty five thousand and three hundred twenty six" into a variable assignment like "amount=45,326"), normalizing date expressions, or validating the correctness of a credit card number based on the digit checksum. Of course it is important to understand that resolving these issues with code attached to the grammar terminals may not be the best architectural solution. An alternative way is to left the grammar assign intermediate values that would be then processed further at the level of the dialog manager, that may use more contextual information.

The example above highlights one of the main problems with using context free grammars to parse natural language. In fact, a slight variation from the responses prescribed by the grammar will not parse, and would produce a failure, also known as a *no-match*. For instance if the user says:

I would like to get gate information

The above rules will fail to parse and the NLU will not produce a result, even though the sentence is very close to those accounted for by the grammar..

When using CFGs, the best way to reduce the number of no-matches consists of analyzing the utterances that did not parse and making sure that they would be represented in the grammar rules. This can be typically done in an experimental phase that precedes the production of the virtual agent. Or later on, after the grammar is in production, by analyzing the logs and instituting an operational "continuous improvement cycle" [Suendermann et al, 2009].

Improving CFG grammars is a labor heavy process that, besides transcribers and annotators, requires specialists that can look at the data and modify the grammars in order to reduce the

number of no-matches. In today's world, transcribing and looking at user utterances is also a delicate process that often conflicts with user privacy.

In any case the use and maintenance of CFGs is problematic for systems with a large number of intents for the reasons discussed above. In order to reduce the amount of labor required by vanilla CFG and/or increase their generalization, one can consider robust parsing or the use of statistical or neural classifiers that will be discussed in the next sections. Deep neural networks, and in particular sequence to sequence or transformer mechanisms are one of the modern solutions. However the choice of the NLU technology depends very much on the type of interaction and the scale of the assistant, and whether a developer would prefer an immediate solution, like writing a simple grammar, or start a process of example collection and use a statistical or neural induction process. In fact, for small scale systems, CFG can still constitute an advantage since, contrary to neural methods they can be crafted in a short time without the need of data.

5.4.2 System initiated, open prompt interaction

Using CFG grammars is not practical when the response of the user is not directed towards a limited choice or a bound value (like a date or city name) by a specific prompt, but rather unconstrained, for instance as the result of generic open prompt. This was a typical situation that arose in the past in applications such as call routing [Gorin et al, 1997] or technical support [Evanini et al., , 2007]. In those situations the opening prompt of a virtual agent is quite generic, like

Please tell me the reason you are calling about.

The response to such an open prompt can exhibit a vast variety of linguistic expressions and meanings. However, if the application domain is limited, for instance technical support in a well defined domain, all the possible linguistic expressions can be clustered into a number of well defined meanings, or intents. Once the set of intents is defined, one can proceed with the collection and annotation of a corpus for the purpose of training and testing the NLU system. In some instances the corpus can be collected and annotated at the same time using a Wizard of Oz Procedure, like in [Gorin et al, 1997].

The annotated corpus can be then used to train a ML classifier to produce the right intent based on feature of the input utterances, such word ngrams. In [Suendermann et al., 2009] statistical classifiers were built not only for the initial open prompt responses, but also for direct responses that were handled, initially, with CFG grammar. As the system started to operate, data for each interaction point was collected, and sent to transcribers and annotators through an automated procedure [Suendermann et al., 2008] The automated procedure would also verify the consistency of the transcriptions and annotations, request crowdsourcers to resolve possible conflicts, create training, and test corpora for each dialog state, train the classifiers and if the accuracy resulted to be higher than those in use, push the new ones to production. All of that

was done automatically and in a mostly unsupervised manner. Eventually the overall accuracy was shown to increase with time and become superior to that obtained by using grammars only [Suendermann et al., 2009].

It is important to consider that the labor required to semantically annotate transcription data for a specific domain, does not grow linearly with the size of the corpus, since we need to annotate each unique expression only once. [Suendermann et al., 2010] shows that the automation rate obtained for an open prompt interaction (e.g. “What is the reason you are calling about”) reaches 50% (i.e. only one out of 2 transcriptions need to be annotated) after roughly 500,000 utterances, while for a specific yes/no question it is above 90% only after 1,000 utterances.

5.4.3 User Initiated interaction

What characterizes a user initiated interaction is the spontaneous nature of it. As opposed to the system initiated interaction, in a user initiated interaction the system does not prompt or guide the user on what to say. Users interact in a spontaneous manner.

NLU for spontaneous interactions requires being able to understand a large number of potential expressions. In restricted domains, this is not much dissimilar to an open prompt situation. However, for unrestricted domains, especially when the user does not know the capabilities of the system, user initiated interactions represent a substantial challenge for NLU.

This is the typical situation for personal virtual assistants like Siri, Alexa, and the Google Assistant. A virtual assistant is generally idle and the user is never prompted to start an interaction. Users start interactions at their will, typically with a wake up phrase, for instance *Hey Google*, or by touching an icon on a display (like for instance the assistant logo or the search microphone on an Android phone). Users then follow the invocation of the assistant with a spontaneous query. Often the assistant responds with an answer, or executes a command to fulfill the user’s request. However, a user query may require a followup request by the assistant through an open or directed prompt. When the system has all the necessary information, it can proceed to fulfill the user request.

When the system is in an idle state, it needs to be able to understand any possible initial user query addressed to invoke a specific feature. At the same time NLU also needs to be able to understand when a query will not produce any reasonable answer, and either return a *punt* prompt (e.g. “sorry I don’t understand your request), or as most virtual assistants do, pass the request to a search engine as a fallback, hoping it will return something useful for the user.

Historically, the first time the spoken dialog community faced the issue of recognizing and understanding spontaneous speech was during the DARPA ATIS [Price, 1990] challenge at the end of the 1980s. ATIS (Air Travel Information System) was a project where a corpus of spontaneous requests to a flight database was used for training and testing systems developed

by the participating labs. During the first years of the challenge, many of the developed NLU systems used legacy linguistic approaches based on CFGs, and in some cases also on more sophisticated grammars (e.g. context sensitive grammars). The grammars had been originally developed by linguists for parsing written text.

However, in the first experiments, feeding the traditional parsers with the transcriptions obtained by the speech recognition systems on spontaneous user utterances resulted in very low correct understanding rates. On the one hand, the errors produced by the speech recognizers dramatically impacted the accuracy of the understanding systems. On the other hand, the grammars, generally developed by hand by expert linguists to parse written text with high accuracy, could not account for phenomena occurring in spontaneous speech. Even though the grammars in use were quite sophisticated, they could not cope with disfluencies, broken sentences, repetitions, etc. Some labs started to experiment with less traditional approaches that were cognizant of the problems of spontaneous speech. One of the first non traditional approaches relied on stochastic models to represent the relationship between a meaning representation and the utterance words. These models, called Conceptual Hidden Markov Models, or CMMs [Pieraccini et al., 1991], represented sets of concepts (analogous to intents and arguments) as Hidden Markov Models with state specific statistical language models based on ngrams. The intuition here is that an utterance representing an intent is composed of a number of conceptual entities that can be mapped directly to segments of a sentence, and that can be effectively represented by statistical models.

In fact, the main cause of variability of natural language is due to the large number of equivalent expressions that carry the same meaning. However we need to consider that a significant part of that variability can be attributed to the combinatorics of different conceptual entities within a sentence that can appear in different order, and by the presence of words that do not contribute directly to the meaning, like pleasantries and fluff words. The advantage that CHMMs had over other systems developed during the ATIS challenge is that they structurally accounted for all the potential orderings of the conceptual entities while ignoring the non-meaning carrying phrases. In fact, the AT&T system based on CHMMs was the one with the highest text understanding score at the last ARPA ATIS evaluation in 1994 [Levin et al., 1995].

CHMMs are a statistically motivated instance of a broader category of parsers strategy that resulted to be best suited for spontaneous speech, known as *robust parsers* [Seneff, 1992]. The concept behind robust parsing is that a grammar does not need to account for all the words in a sentence. In a restricted domain application there are phrases that carry most of the information, while additional words can be ignored.

A robust parser can be implemented as CFGs with the addition of a wildcard or match-all operator. The following is an example of a robust parsing rule.

```
$weather-information = \w* weather \w* [$city \w*];
$city = (boston | new york | san francisco | …);
```

Where the symbol "`\w`" is a wildcard that matches any word and the symbol "`*`" is analogous to the Kleene star in regular expressions (e.g. it matches zero or more repetitions of the previous element). The following are examples of sentences accepted by this simple robust parsing grammar:

Weather
Weather in New York City please
Can you please tell me what is the weather like in Boston

However the following sentence cannot be parsed:

I am traveling tomorrow to Boston. How is the weather there?

In fact the concepts of "weather" and "city" are appearing in a different order than the one defined in the previous rules. A simple way to modify the above parsing rule in order to make it independent of the order of the concepts relies on the Kleene plus operator "`+`" that allows for one or more repetitions of the previous element. Thus:

```
$weather-information = \w* ($concept \w*)+;
$concept = (weather | $city);
$city = (boston | new york | san francisco | …);
```

This set of rules allows for any order of the weather qualifier and the city and would parse a vast array of sentences.

The one above is a simple example chosen to illustrate the power of robust parsing, but also the issues associated with it. Overgeneration is one of the main problems of robust parsing rules. While the above rules cover most of the positive examples, they also cover a vast amount of negative examples. For instance, all of the following queries are parsed by the same rules:

Weather weather Boston New York San Francisco weather weather.
I don't want to know the weather in New York
Weather I don't want to know Boston is not my city

One of the arguments in favor of using a robust parser is that those sentences are quite uncommon, and negation can be dealt with special rules and heuristics. The advantage of a robust parser is that it is easy to handcraft rules that allow for a high recall, at the expense of low precision. Robust parsing is a practical solution for small-scale systems (i.e. systems with a limited number of intents).

5.5 NLU Induction from examples

Rather than creating handcrafted grammars, a different approach consists in learning an NLU system from a number of defined examples. In particular, the use of Deep Learning and the availability of libraries like TensorFlow and PyTorch made the process of creating NLU parsers from examples quite accessible to developers that do not necessarily have a deep understanding of the intricacies of machine learning. However, we should not forget that, as we discussed earlier, the complexity of a large NLU system is not just in the training of the parser. As the number of intent grows, the work required for maintaining and creating new intents and annotating data for training the NLU can become a celar bottleneck.

5.5.1 Deep Learning for NLU

In less than a decade Deep Learning (DL) [DeepLearning] has revolutionized the whole field of AI. Speech recognition, language generation, text-to-speech synthesis, machine dialog, and natural language understanding are among the most successful applications of DL in the field of human-machine communication. DL is based on the notion of Deep Neural Networks (DNNs), i.e. neural networks with more than one hidden layer. DNNs are being used to solve a number of problems, like modeling, classification, and mapping between input and output strings of discrete elements.

More advanced types of DNNs have been recently developed to solve increasingly complex problems, and with higher and higher performance. Those include recurrent Long Short Term Memory networks, or LSTMs [LSTM], Transformers [transformers], and bi-directional transformers, like BERT [BERT].

In general, a neural NLU receives a sequence of words as input, and generates a meaning representation as output. We have to notice that because of the nature of the NLU problem, both the input sequence of words, as well as the output meaning representation do not have a fixed size. The input can have arbitrary length, i.e. an arbitrary long sequence of words, and the output can be a structure, e.g. an intent/argument set, or a serialized structure with an arbitrary number of elements. A general solution for accommodating inputs and outputs of arbitrary size is that of using a recurrent sequence to sequence DNN [seq2seq] (such as, for instance, a LSTMs). In a recurrent neural network you feed each input word to a different instance of the same network. However, for each word in the sequence, the hidden neurons of the corresponding instance of the network also receive the output values that were calculated for the instance that processed the immediately preceding word. As a result of this incremental processing the network internal values for a specific word of the input sentence are influenced by all the preceding words. The final result will thus depend on all the words in the input sentence.

The problem of representing the input words in a way to allow for generalization, for instance across synonyms and semantically similar expressions, is common to every neural architectural solution. In modern language-processing neural networks the input sequence is represented by projecting words into a multidimensional space that preserves the semantic relationships between them. In other words words that are semantically close, are also close to each other in that space. The solution to this problem is central to the notion of word embeddings [Mikolov et al., 2013].

Word embeddings are extremely popular in today's neural NLU and NLP systems in general, and can be reused across them. Multilingual embeddings [Faruqui et al, 2014][Chen et al., 2018] are an interesting and useful evolution of word embedding.

To summarize, the use of DNNs for NLU is still a topic of research and there is not a mainstream solution yet. Whether to use a sequence to sequence model or a transformer like BERT, is still an open question. Using DNNs we trade the problem of writing a grammar with that of providing labeled training data. While that may seem a quite simpler and effective approach, it is still cumbersome for many reasons. For one, getting realistic data is quite hard. Unless the system is deployed, and we have access to a large number of logs, the creation of a large corpus of data may require a collection campaign relying on recruited subjects. Thus, depending on the collection paradigm, the data may not represent the reality of the language used by real customers in real situations. Even though we had access to a large number of logs, semantic annotation is quite hard and imprecise. This is especially true if we are in presence of a large number of meanings, each of them represented by a symbolic meaning representation structure, like intent and arguments as described earlier in this chapter. Exposing a human annotator to unlabelled utterance transcriptions and asking them to select a corresponding intent/argument structure out of a catalog of thousands requires properly designed tools and processes. Moreover, human annotators need to be trained and have a certain familiarity with the ontology of intents and arguments. As discussed above, any hand-crafted ontology may include duplications, ambiguities, and inconsistencies that make the annotation task harder. The precision of the annotation done in this way may not be satisfactory, thus requiring to deploy multiple human annotators per each utterance transcription, and a mechanism to resolve the conflicts.

5.4.2 Commercial NLU tools

There are several available tools for creating NLU systems in the cloud. Major companies like Amazon, IBM, Microsoft, and Google offer those tools for experimentation or for commercial use. Google's DialogFlow is a typical example of that. DialogFlow is characterized by a dashboard that allows a developer to build intents and define entities as arguments. The interface allows developers to specify a number of sample sentences that define a certain intent. Known entities, such as numbers, dates, and geographical entities are automatically

detected in the examples and used to generalize across them. So for instance, an intent like “travel-destination” cold be created by specifying a number of example sentences, such as:

I want to go to San Francisco
My destination is Boston
Going to Paris

The entities like “San Francisco,” “Boston,” and “Paris,” used in the examples are automatically detected and internally expanded by all entities in the same category, so in the future the system can classify as “travel-destination” sentences using different entities, such as “I want to go to New York”. The system has also some inbuilt generalization across common expressions, such as “I want” and “I would like”. The interface allows entities to be associated with roles, or arguments. It also allows the developer to create their own list of entities.

5.6 The NLU Usability Paradox

The NLU Usability Paradox was initially stated by Mike Phillips [Phillips, 2006] and restated in terms of “habitability gap” by Roger Moore [Moore, 2019] and further developed by Bruce Balentine [Balentine, 2020]. Consider the qualitative chart of Figure 3, based on the original presentation by Mike Phiips.

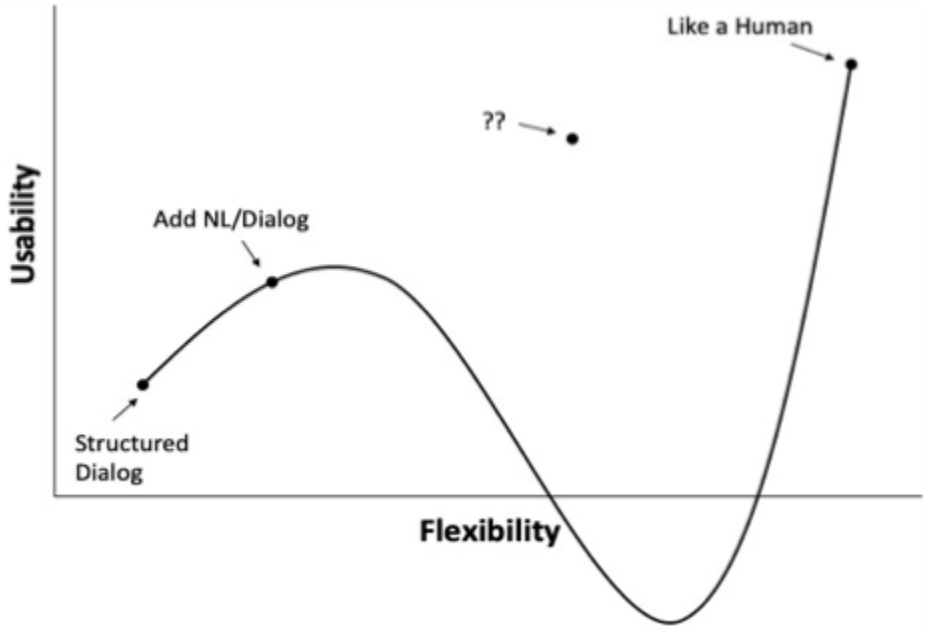


Figure 2: A qualitative description of the NLU usability paradox [Moore, 2019]

With structured dialog, where the user is specifically instructed by the system on what to say, one can reach a quite reasonable level of usability. The usability would slightly increase by

increasing the flexibility of the system to understand not only the commands suggested by the prompts, but also natural language expressions. One can safely increase the flexibility of the system, e.g. the number of available commands and the related accepted expression, up to a point, beyond which the usability will drop, without ever reaching the point marked as “??.” This is because the dialog cannot be structured with directed prompts when too many options are available, and the user is left to guess what to say. Unfortunately if the natural language coverage is limited, a high percentage of the user attempts will fail. And even though the user may learn which expressions work, by increasing the flexibility of the system, users will have a hard time remembering what works and what doesn’t. The usability of a NLU system will continue to drop, until the coverage of NLU expression reaches a point that is closer to the human capabilities. At that point whatever the user says will be understood and dealt with by the system.

Early NLU systems fell into the low usability minimum of NLU paradox curve. Systems with a relatively large number of features and low NLU coverage resulted in very extremely poor usage and the lack of feature discovery mechanisms made the systems practically unusable.

An open question for large NLU systems is that of being able to discern among expressions that the system understands and can fulfill, expressions that the system understands and cannot fulfill, and expressions that are totally not understable by the system. If we could do that, the usability of NLU systems will be much higher than what we have today.

5.7 Context

Humans carry on efficient conversations by relying on contextual information. Let’s consider the following dialog

User: Where is the Empire State building?
Assistant: In New York City
U: How tall is it?
A: 381 meters
U: Give me that in feet
A: 1,250 feet
U: When was it built?
…

This is a very efficient and natural exchange. That’s how we talk, and that’s how we would like interactive agents and assistants to talk. Each query is grounded, if possible, based on the previous conversation, and previously mentioned entities are referenced by the use of pronouns. Think how cumbersome and unnatural the same conversation would be without context references:

U: *Where is the Empire State building?*

A: The Empire State Building is In New York City
U: How tall is the Empire State Building?
A: The Empire State Building is 381 meters high
U: How much would 381 meters be in feet?
A: 381 meters correspond to 1,250 feet
U: When was the Empire State Building built?
…

Pronouns are not the only way to reference the context. In fact, sometimes the context is obvious even without any linguistic reference. For instance:

U: What is the weather in Boston?
A: 65 degrees and sunny
U: What about Chicago?
A: 58 degrees and rainy.

In order to build an efficient conversational dialog system, NLU needs to be able to account for these linguistic phenomena, generally referred to as anaphoric references, i.e. the use of pronouns, and ellipsis, i.e. the use of incomplete sentences.

There are many ways to account for anaphora and ellipsis. One method is to deal with those phenomena at the symbolic level after the NLU process has generated a meaning representation. For instance, a contextual resolution module can assume that a missing intent to be the same as the previous query and simply change the argument of the previous query for the new argument (e.g. intent= weather-information; ~~city=Boston;~~ city=Chicago). One can devise other rules that work most of the time. However a query may not reference only a previous user query but also a previous agent answer, as in:

U: Who was the 44th president of the United States
A: Barack Obama
U: How old is he?

Or, in a multimodal interface, for instance a virtual assistant with a display or on a smartphone, the user may refer to a displayed picture and ask questions or refine the search using incomplete utterances. For instance, one may ask to the Google assistant on a smartphone :

U: Show me the pictures I took in Tunisia last year
A: Here they are <showing a number of pictures>
U: Only those with camels
A: <assistant shows picture with camels taken in Tunisia last year>
U: <enlarging one of the picture> When was this picture taken
A: This picture was taken on December 20th 2019.

Even though commercial virtual assistants, like Alexa, Siri, and the Google Assistant, are able to handle context quite well, this is still an interesting topic of research where machine learning and deep learning in particular can provide increasing levels of quality.

5.8 Conclusions

We have seens how Natural Language Understanding is an essential part of any interactive agent. For an agent to be able to respond or to fulfill a user's requests, first it needs to understand the request itself. Converting an arbitrary textual representation of an utterance into an actionable representation of its meaning, for instance intents and arguments, is the goal of NLU.

In order to develop an NLU it is important to consider not just the actual parsing mechanism but the whole process, starting from the creation and management of a meaning representation, the annotation of data for testing and training, and the actual training of the NLU module. That whole process can become quite complex as the number of meanings (e.g. intents) increases, which is the case in today’s virtual interactive assistant.

We have also analyzed how the choice of the NLU system is influenced by the type of interaction. System initiated interactions characterized by well defined prompts can be handled quite well by more or less strict context free grammars (CFG), since the prompts lead the user into a restricted number of expressions. Open prompts, which need to be used in system initiated applications where a directed prompt may not work, may expect a large unbound number of expressions, even though restricted to a specific domain. In that case an ML classifier training on large corpora of annotated transcriptions needs to be used.

Finally user initiated interactions are the ones that require the most sophisticated NLU systems. That is especially true for virtual interactive assistants that span through a vast number of domains and features. Robust parsing is a simple extension of CFG grammars that can be useful in these situations. With the advent of Deep Learning, epitomized by Deep Neural Networks, the task of learning NLU from a corpus of annotated transcription has become easier and more effective.

There is a clear dichotomy between structural NLU, such as grammars and robust parsers, and induced NLU. The choice between the two depends very much on the type of system one wants to build. Structured methods allow a high precision, and a high degree of control by the developer. If the NLU does not work for a given number of expressions, accounting for them in a grammar is relatively straightforward, even though labor intensive. On the other hand ML-based NLU, and especially DNN-based systems, suffer from the problem of interpretability of the results. Besides using more data and larger networks, correcting errors is a process that requires some level of intuition, and a lot of experimentation. And there is no guarantee that the process would succeed.

Large scale internationalization is another big issue. Translating grammars is labor intensive and requires specialist grammar writers proficient in different languages. As machine translation improves one can use that, for instance, to create an initial corpus in a target language from a corpus in English. In [Sendermann et al, 2009-2] we used publicly available MT to translate an annotated corpus from English to Spanish, and thus create a set of ML-based intent classifiers in the target language. Once the Spanish system went into production, a continuous improvement loop [Suendermann et al, 2009] continued to improve the performance of the initial classifier.

As we have seen in this chapter, NLU for Interactive Virtual Agents is complex, and even though one may find some off the shelf solution for some limited domains, there is not a general solution to the NLU problem that work on any potential application

Bibliography

[Shannon, C., 1948] Claude Shannon, “A Mathematical Theory of Communication”, 1948.

[Turing, A., 1950] Alan Turing, “Computing Machinery and Intelligence”, 1950.

[Fung, 2016] Pascale Fung, Anik Dey, Farhad Bin Siddique,Ruixi Lin, Yang Yang, Wan Yan, Ricky Chan Ho Yin, “Zara The Supergirl: An Empathetic Personality Recognition System,” Proceedings of NAACL-HLT 2016 (Demonstrations), pages 87–91,
San Diego, California, June 12-17, 2016

[jibo, WSJ] This Cute Little Robot Made My Family Mad, By Joanna Stern, The Wall Street Journal, Nov. 29, 2017 1:41 pm ET.

[Pepper] https://www.softbankrobotics.com/emea/en/pepper

[FurHat] https://furhatrobotics.com/

[Guadarrama] Guadarrama, Sergio & Riano, Lorenzo & Golland, Dave & Gouhring, Daniel & Jia, Yangqing & Klein, Dan & Abbeel, Pieter & Darrell, Trevor. (2013). “Grounding spatial relations for human-robot interaction,” Proceedings of the ... IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE/RSJ International Conference on Intelligent Robots and Systems. 1640-1647.

[Data Cloud] See for instance “The Linked Open Data Cloud” (https://lod-cloud.net/) or DBPedia (https://wiki.dbpedia.org/).

[Neo4j] https://neo4j.com/

[Pieraccini et al., 2005] Pieraccini, R. Huerta, J., “Where do we go from here? Research and Commercial Spoken Dialog Systems,” Proc. of 6th SIGdial Workshop on Discourse and Dialog, Lisbon, Portugal, 2-3 September, 2005.

[VoiceXML, 2004] Voice Extensible Markup Language (VoiceXML) Version 2.0, W3C Recommendation 16 March 2004, https://www.w3.org/TR/voicexml20/

[Stoyanchev et al., 2016] Svetlana Stoyanchev, Pierre Lison, Srinivas Bangalore, “Rapid Prototyping of Form-driven Dialogue Systems Using an Open-source Framework, ” in *Proceedings of the SIGDIAL 2016 Conference,* pages 216–219, Los Angeles, USA, 13-15 September 2016.

[Adiwardana et al., 2020] Daniel Adiwardana, Minh-Thang Luong, David R. So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, Quoc V. Le, “Towards a Human-like Open-Domain Chatbot,” arXiv:2001.09977v3 [cs.CL] 27 Feb 2020.

[Lourdusamy ert al., 2020] Lourdusamy, R., Abraham, S., “A Survey on Methods of Ontology Learning from Text” in *Learning and Analytics in Intelligent Systems* book series (LAIS, volume 9), 2020.

[Della Pietra et al., 1997] Della Pietra, S., Epstein, M., Roukos, S., Ward, T., "Fertility Models for Statistical Natural Language Understanding", "35th Annual Meeting of the Association Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics", July 1997, Madrid, Spain.

[Cohen et al, 2004] Cohen, M., Giangola, J., Balogh, J., “Voice User Interface Design”, *Addison-Wesley Professional, 2004.*

[Cloud Google] https://cloud.google.com/speech-to-text/docs/context-strength.

[Gorin et al, 1997] Gorin, A., Riccardi, G., Wright, J., “How may I help you?”, Speech Communication, October 1997.

[SRGS, 2004] Speech Recognition Grammar Specification Version 1.0, W3C Recommendation, 16 March 2004, http://www.w3.org/TR/2004/REC-speech-grammar-20040316/.

[Evanini et al., , 2007] Evanini, K., Suendermann, D., Pieraccini, R., Call Classification for Automated Troubleshooting on Large Corpora, ASRU 2007, Kyoto, Japan, December 9-13, 2007.

[Suendermann et al., 2009] Suendermann, D., Evanini, K., Liscombe, J., Hunter., P, Dayanidhi, K., Pieraccini, R., From Rule-Based to Statistical Grammars: Continuous Improvement of Large-Scale Spoken Dialog Systems, Proceedings of the 2009 IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP 2009), Taipei, Taiwan, April 19-24, 2009.

[Suendermann et al., 2008] Suendermann, D., Liscombe, J., Evanini, K., Dayanidhi, K., Pieraccini, R., C5, in Proc. of 2008 IEEE Workshop on Spoken Language Technology (SLT 08), December 15-18, 2008, Goa, India.

[Suendermann et al., 2010] Suendermann, D., Liscombe, J., Pieraccini, R., How to Drink from a Fire Hose: One Person Can Annoscribe 693 Thousand Utterances in One Month, SIGDIAL 2010, The 11th Annual SIGDIAL Meeting on Discourse and Dialogue, Sep 2010, Tokyo, Japan.

[Price, 1990] Price, P., "Evaluation of Spoken Language Systems: the ATIS Domain" in *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania*, June 24-27,1990.Levin

[Pieraccini et al., 1991] Pieraccini, R., Levin, E. and Lee, C-H., Stochastic representation of conceptual structure in the ATIS task, Proc. Fourth Joint DARPA Speech and Natural Lang. Workshop, Pacific Grove, CA, Feb. 1991.

[Levin et al., 1995] Levin, E. and Pieraccini, R., Concept-based spontaneous speech understanding system, Proc. EUROSPEECH '95, Madrid, Spain, 1995.

[Seneff, 1992] Seneff, S., “Robust parsing for spoken language systems,” Proceeding of the International Conference on Acoustic, Speech, and Signal Processing, Year: 1992, Volume: 1, Pages: 189-192

[Mikolov et al., 2013] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J., “Distributed Representations of Words and Phrases and their Compositionality,” in *Advances in neural information processing systems*, 26 · October 2013.

[DeepLearning] Aaron Courville, Ian Goodfellow, and Yoshua Bengio, “Deep Learning,” MIT Press, 2015.

[LSTM] Sak, H. & Senior, Andrew & Beaufays, F., “Long short-term memory recurrent neural network architectures for large scale acoustic modeling,”. Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH 2014, 338-342.

[transformers] Polosukhin, Illia; Kaiser, Lukasz; Gomez, Aidan N.; Jones, Llion; Uszkoreit, Jakob; Parmar, Niki; Shazeer, Noam; Vaswani, Ashish (2017-06-12). "Attention Is All You Need". arXiv:1706.03762.

[BERT] Devlin, Jacob; Chang, Ming-Wei; Lee, Kenton; Toutanova, Kristina (11 October 2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," arXiv:1810.04805v2.

[seq2seq] Michel Kana, “Natural Language Understanding with Sequence to Sequence Models,” Medium. Sep 5, 2019.

[Faruqui et al, 2014] Faruqui M., Dyer, C., “Improving vector space word representations using multilingual correlation,” in *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471, 2014

[Chen et al., 2018] Chen, X., Cardie, C., “Unsupervised Multilingual Word Embeddings,” in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing,* October-November, 2018.

[Phillips, 2006]] Mike Phillips. “Applications of spoken language technology and systems”. In *IEEE/ACL Workshop on Spoken Language Technology* (SLT), Mazin Gilbert and Hermann Ney (Eds.). IEEE, Aruba,

[Moore, 2019] Moore, R. K., “A 'Canny' Approach to Spoken Language Interfaces,” 2019, https://arxiv.org/pdf/1908.08131.pdf.

[Balentine, 2020] Balentine B., Personal communication at Dagstuhl Seminar 20021 “Spoken Language Interaction with Virtual Agents and Robots (SLIVAR): Towards Effective and Ethical Interactions,” January 2020.

[Sendermann et al, 2009-2] Suendermann, D., Liscombe, J., Dayanidhi, K., Pieraccini, R., “Localization of Speech Recognition in Spoken Dialog Systems:
How Machine Translation Can Make Our Lives Easier.” Proceedings of Interspeech 2009, Brighton, UK.