Talking to Computers in Natural Language

Natural language understanding is as old as computing itself, but recent advances in machine learning and the rising demand of natural-language interfaces make it a promising time to once again tackle the long-standing challenge.

By Percy Liang

DOI: 10.1145/2659831

As you read this sentence, the words on the page are somehow absorbed into your brain and transformed into concepts, which then enter into a rich network of previously-acquired concepts. This process of language understanding has so far been the sole privilege of humans. But the universality of computation, as formalized by Alan Turing in the early 1930s—which states that any computation could be done on a Turing machine—offers a tantalizing possibility that a computer could understand language as well. Later, Turing went on in his seminal 1950 article, "Computing Machinery and Intelligence," to propose the now-famous Turing test—a bold and speculative method to evaluate whether a computer actually understands language (or more broadly, is "intelligent"). While this test has led to the development of amusing chatbots that attempt to fool human judges by engaging in light-hearted banter, the grand challenge of developing serious programs that can truly understand language in useful ways remains wide open. This article provides a brief glimpse into the history of language understanding systems, the challenges associated with understanding language, and how machine learning is emerging as a major character in the story.

THE EARLY YEARS, RULE-BASED SYSTEMS

The first natural language understanding systems emerged in the early 1960s in Cambridge, MA, a hotbed for artificial intelligence at the time. Daniel Bobrow built a system for his Ph.D. thesis at MIT to solve algebra word problems found in high-school algebra books, for example: "*If the number of customers Tom gets is twice the square of 20% of the number of advertisements he runs, and the number of advertisements is 45, then what is the numbers of customers Tom gets?*" [1]. Another landmark was the LUNAR system, developed by Bill Woods in the early 1970s at BBN [2]. LUNAR provided a natural-language interface into a database about moon rocks that had been brought back on the recent Apollo 11 mission. Scientists could ask LUNAR to: "list all the rocks that contain chronite and ulvospinel."

Around the same time, Terry Winograd, then a Ph.D. student at MIT, developed another system called SHRDLU [3] that lived in a toy blocks world (see Figure 1). SHRDLU could both answer questions and execute actions, for example: "Find a block that is taller than the one you are holding and put it into the box." In this case, SHRDLU would first have to understand that the blue block is the referent and then perform the action by moving the small green block out of the way and then lifting the blue block into the brown box. Interpretation and execution are performed jointly, and the key idea was that true language understanding is only sensible when it connects with the world.

For their time, these systems were significant achievements. They were able to handle fairly complex linguistic phenomena and integrate syntax, semantics, and reasoning in an end-to-end application. What is even more impressive is that these systems ran on a modicum of resources by today's stan-

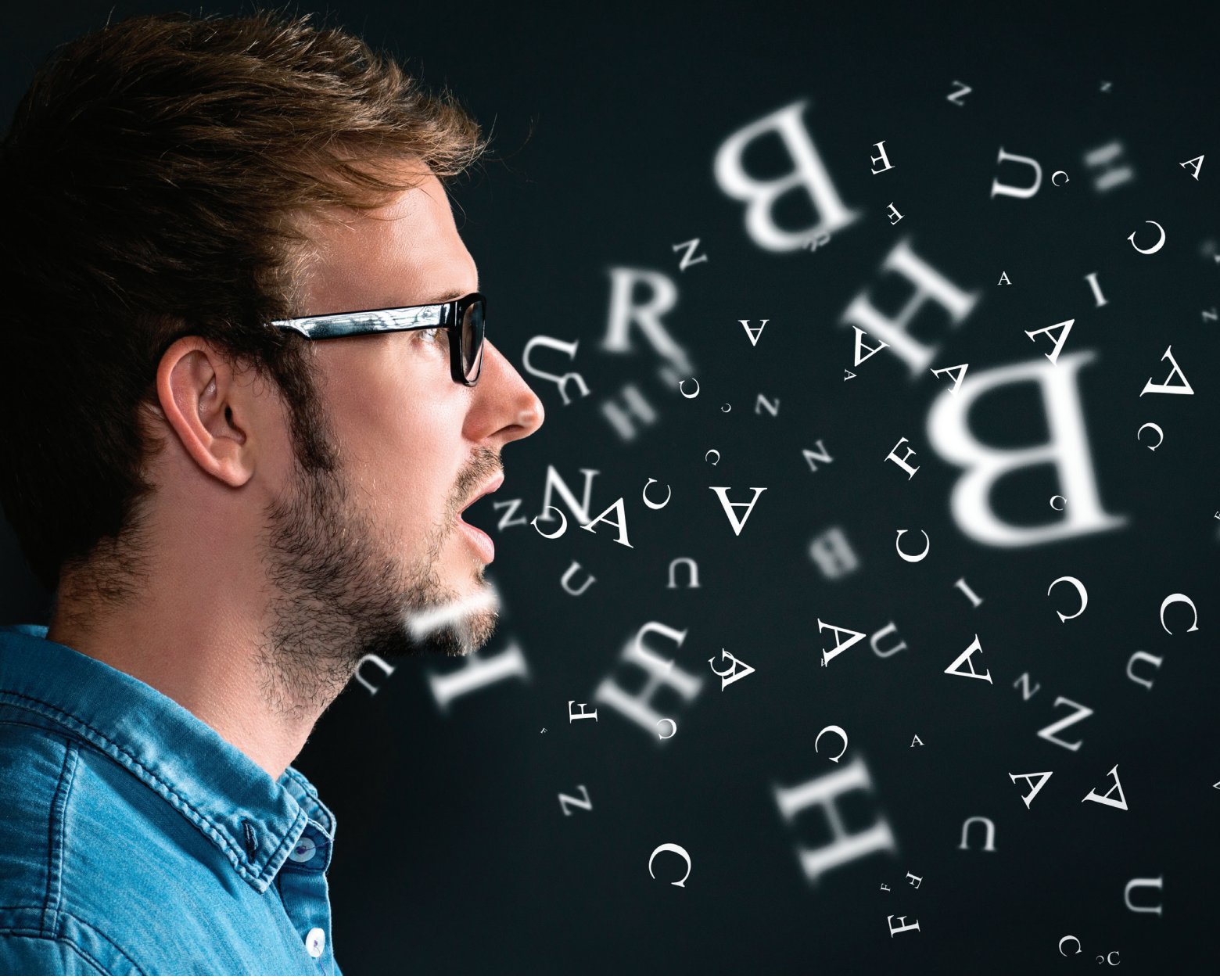


Image by Lasse Behnke

dards. For example, LUNAR was written in LISP for the DEC PDP-10, and all the code and data fit in only 80 KB of memory. In contrast, today, just starting a Python interpreter alone eats up 5 MB, which is 60 times more.

Over the following 20 years, these systems were extended, but soon it became increasingly difficult to make progress. Systems built for rocks and blocks did not automatically generalize to other domains, so adaptation was very burdensome. Also, to handle the never-ending intricacies of natural language, the complexity of these systems spiraled out of control.

READING BETWEEN THE LINES, THE CHALLENGES OF LANGUAGE

Why is it so difficult for a computer to understand natural language? To answer this, it is helpful to contrast natural languages such as English with programming languages such as Python. Python is unambiguous. The expression `lambda x : x.split(" ")[0:3]` has exactly one denotation (meaning) as dictated by the Python language specification. English, on the other hand, can be ambiguous and vague.

Consider the following pair of sentences from Yehoshua Bar-Hillel: "The pen is in the box" and "The box is in the pen." [4] In the first sentence, "pen" is most likely a writing instrument; in the second, it is most likely an enclosure for animals. Another example from Winograd: "The city councilmen refused the women a permit because they feared violence" and "The ... because they advocated revolution" [3]. Who does "they" refer to in each case? In the first sentence, "the city councilmen" is the likely interpretation, whereas in the latter, it is "the women."

In both of these examples, it is clear the words alone do not fully specify the meaning. They are only a few impressionistic brushstrokes, leaving the rest to be filled in by the reader. Humans perform this completion based on knowledge about the world that we cultivate throughout our lives. Computers lack this knowledge, and therefore these inferential leaps are currently extremely difficult. This is why systems such as SHRDLU were confined to a microcosm, where such inferences are possible, and why programming languages live solely in the computer world.

Despite differences between English and Python, there are some important similarities. The first is "compositionality"—an idea often

attributed to German logician Gottlob Frege—that the meaning of the whole is derived from the meaning of the parts. Just as a Python interpreter computes (4 – 2) + 3 by understanding numbers, operators, and a few combination rules, humans understand "red house" by understanding the constituent words. This compositionality is what allows us to communicate a dazzling array of different meanings given just a relatively small vocabulary, or as German philosopher Wilhelm von Humboldt put it, "make infinite use of finite means."

The ambiguous nature of natural language might seem like a flaw, but in fact, it is exactly this ambiguity that makes natural language so powerful. Think of language as a (cooperative) game between a speaker and a listener. Game play proceeds as follows: 1. the speaker thinks of a concept, 2. she chooses an utterance to convey that concept, and 3. the listener interprets the utterance. Both players win if the listener's interpretation matches the speaker's intention. To play this game well, the speaker should thus choose the simplest utterance that conveys her intended concept—anything the listener can infer can be omitted. For example, "cities that are in United States" can be shortened to "U.S. cities." How can a computer fill in these gaps, which depend on the breadth of human experience involving perception of the world and social interactions? Projects such as Cyc attempted to manually write down this world knowledge (e.g., every tree is a plant), but there is a lot of world out there, and it's messy too. Cyc fell prey to the same problems that plagued all purely rule-based systems.

Language is an amazing vehicle for human expression, capable of conveying everything from intense emotions to intricate scientific arguments.

THE STATISTICAL REVOLUTION

In the early 1990s, a revolution occurred. Up until then, natural language processing (NLP) research had been rule-based, where one directly writes a program to perform a task. But with the increased availability of more data and more computing power, NLP went statistical. The statistical paradigm requires a different mode of thinking: one (1) first collects examples of the desired input-output behavior of the program, (2) writes a partial program with unknown parameters, and (3) uses a machine learning algorithm to automatically tune these parameters based on the examples.

Statistical techniques have a long history. As far back as 1795, Carl Friedrich Gauss developed the least squares method for fitting a line to a set of points determined from measurement data. An early example within artificial intelligence was Arthur Samuel's 1959 checkers program that learned to play from its own moves and game outcomes. Today, machine learning plays a vital role in applications spanning spam filtering, speech recognition, advertisement placement, robotics, medical diagnosis, etc.

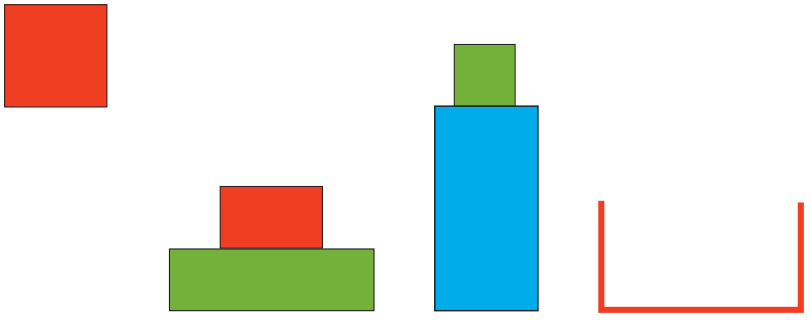
Machine learning also drives many NLP tasks: part-of-speech tagging (e.g., identifying "London" as a proper noun), named-entity recognition (e.g., identifying "London" as a location), syntactic parsing (e.g., identifying "London" as the direct object of a sentence), and machine translation (e.g., converting "London" to "*Londres*" in French). One task closely tied to language understanding is question answering (e.g., responding to "What is the largest city in England?" with "London"). A question-answering system uses the question to retrieve relevant Web pages using a search engine, extracts candidate answers from those pages, and then scores and ranks the answers.

A shining achievement of question answering was IBM's Watson [5], a computer built for the quiz show "Jeopardy!" David Ferrucci's team of 20 researchers worked tirelessly for four years, and in 2011, Watson took on former "Jeopardy!" champions Brad Rutter and Ken Jennings in a widely publicized match. Watson was victorious. IBM had pulled off a similarly impressive feat in 1997 with Deep Blue, which defeated world chess champion, Gary Kasparov. Somehow, "Jeopardy!" hit closer to home, as it dealt with language, something uniquely human, in contrast to the calculated thinking involved in playing chess. It was a soft blow, though: The stylized trivia questions on "Jeopardy!" are more about memorization and pattern-matching than reasoning; human-level natural language understanding remains a distant goal.

SYNTHESIS

We have climbed to an interesting vantage point over the last 50 years. We have seen early rule-based systems such as LUNAR and SHRDLU perform relatively deep analyses of natural language, but only in narrow domains. We have also seen the blossoming of statistical techniques in a variety of applications,

Figure 1. SHRDLU was a natural language understanding system that allowed users to interact in a blocks world environment.



including question answering, which operates more broadly and robustly by absorbing huge amounts of data but has limited reasoning capabilities. For the geography question above, a system might answer "London" not because it understands what "largest" means, but because a document contains the text: "London is the capital of England, as well as its largest city, ..."

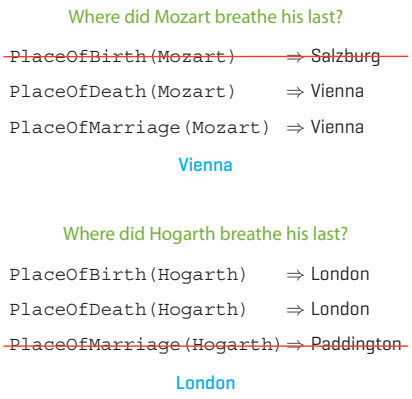
Fortunately, the rule-based and statistical views are complementary, although a large gulf has historically separated them [6]. Ideally, we would fuse the logically sophisticated representations of rule-based systems with the robustness and automation of statistical techniques. The area of (statistical) semantic parsing provides an initial step toward bridging this gulf by applying machine learning to the problem of parsing sentences into logical forms. Instead of treating "largest city in England" as merely words, semantic parsing, like rule-based methods, uses a logical form: `argmax(x, City(x) ∧ LocatedIn(x, France), Population(x))`, not unlike a Python program. Moreover, like statistical methods, the mapping from sentences to logical forms is learned from data, not manually hand-coded.

Ray Mooney [7] and Luke Zettlemoyer [8] were among the first to develop statistical semantic parsers. However, their work was limited by data, which in their case were pairs of questions annotated with their logical forms. Creating such data was burdensome, so the practical result was not much better than building a rule-based system.

In the last five years, there has been another important technical development: new machine learning techniques whose input data consists of answers to questions rather than logical forms (e.g., "London" for "What is the largest city in England?"). The key advantage is that answers, unlike logical forms, do not require specialized expertise to produce and are thus much easier to obtain. Using crowdsourcing, much larger question-answering datasets are now emerging [9].

Figure 2 illustrates a simplified version of how a system would conceivably learn from two question-answer pairs. For each question, the system generates a set of possible logical forms and executes each one to yield an answer. The logical forms that do not yield the correct answer are discarded. Based on the remaining logical forms, the system would find that "breathe his last" occurs more consistently with PlaceOfDeath than the alternatives and choose this mapping. This is the core intuition. In an actual system, thousands of question-answer pairs are used, each generating hundreds of potential logical forms that are more complex. The system also maintains probability distributions over logical forms reflecting ambiguity in language and uncertainty due to noise in the data. Semantic parsing thus draws strength from both machine learning and logic, two powerful but disparate intellectual traditions.

Figure 2. A sketch of how a system can infer the logical from question-answer pairs..



THE FUTURE OF NATURAL LANGUAGE UNDERSTANDING

On the technical front, new semantic parsing techniques are being rapidly developed. But there is another important enabling factor. In 2012, Apple launched Siri, a virtual personal assistant that could understand basic user requests. Google and Microsoft soon followed suit. Home entertainment systems and cars are now being equipped with natural-language interfaces. All these applications require precise language understanding, and this demand will undoubtedly stress current semantic parsers, causing more innovation. At the same time, these applications will also generate large amounts of data from which semantic parsers can learn, thus covering the Achilles heel of statistical methods—having adequate data.

Language is an amazing vehicle for human expression, capable of conveying everything from intense emotions to intricate scientific arguments. Fully answering Turing's original question of whether computers can match this capability remains, at least for now, more of a philosophical enterprise than an empirical one. But natural language understanding is not solely a scientifically interesting endeavor but also one loaded with practical potential. The confluence of demand, robust machine learning techniques, and expressive logical representations makes the present time a particular exciting one for natural language understanding.

References

[1] Bobrow , D. G. Natural Language Input for a Computer Problem Solving System. *AI Technical Reports (1964-2004)*. MIT. 1964. http://hdl.handle.net/1721.1/6903

[2] Woods, W.A., Kaplan, R.M., and Webber, B.N. The Lunar Sciences Natural Language Information System: Final report. Technical report. BBN Report 2378. Bolt Beranek and Newman Inc., 1972.

[3] Winograd, T. *Understanding Natural Language*. Academic Press, New York, 1972.

[4] Bar-Hillel, Y. *Language and Information: Selected Essays on Their Theory and Application*. Jerusalem, Addison-Wesley/The Jerusalem Academic Press, 1964.

[5] Ferrucci, D., Brown, E., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A.A., Lally, A., Murdock, J.W., Nyberg, E., Prager, J., Schlaefer, N., and Welty, C. Building Watson: An overview of the DeepQA project. *AI Magazine* 31, 3 (2013), 59–79.

[6] Liang, P. and Potts, C. Bringing Machine Learning and Compositional Semantics Together. *Annual Reviews of Linguistics* (to appear).

[7] Zelle, M. and Mooney, R.J. Learning to Parse Database Queries Using Inductive Logic Programming. In *Proceedings of the Thirteenth National Conference on Artificial intelligence- Vol. 2* (Portland, Aug.) AAAI Press, 1996, 1050–1055.

[8] Zettlemoyer, L.S. and Collins, M. Learning to Map Sentences to Logical Form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence* (Edinburgh, June). AUAI Press, 2005, 658–666.

[9] Berant, J., Chou, A., Frostig, R., and Liang, P. Semantic Parsing On Freebase From Question-Answer Pairs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (Seattle, Oct.). Association for Computational Linguistics Stroudsburg (ACL), 2013.

Biography

Percy Liang is an assistant professor of Computer Science at Stanford University. He received his B.S. from MIT, Ph.D. from University of California Berkeley, and was a post-doc at Google.