



Article

A beginner's guide to natural language processing

Discover how natural language processing can help you to converse more naturally with computers

☆ Save 👍 Like

[Site feedback](#)

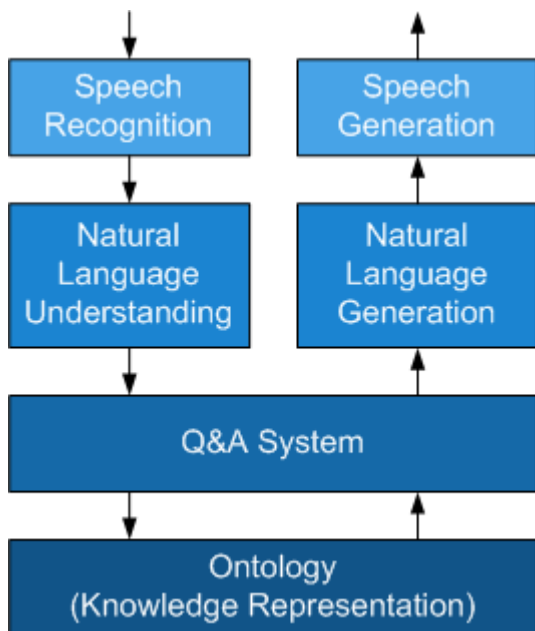
By M. Tim Jones

Published April 9, 2019

In 1954, IBM demonstrated the ability to translate Russian sentences into English using machine translation on an IBM 701 mainframe. While simple by today's standards, the demonstration identified the massive advantages of language translation. In this article, we'll examine natural language processing (NLP) and how it can help us to converse more naturally with computers.

NLP is one of the most important subfields of machine learning for a variety of reasons. Natural language is the most natural interface between a user and a machine. In the ideal case, this involves speech recognition and voice generation. Even Alan Turing recognized this in his "intelligence" article, in which he defined the "Turing test" as a way to test a machine's ability to exhibit intelligent behavior through a natural language conversation.

NLP isn't a singular entity but a spectrum of areas of research. Figure 1 illustrates a voice assistant, which is a common product of NLP today. The NLP areas of study are shown in the context of the fundamental blocks of the voice assistant application.



Beyond voice assistant technology, one of the key benefits of NLP is the massive amount of unstructured text data that exists in the world and acts as a driver for natural language processing and understanding. For a machine to process, organize, and understand this text (that was generated primarily for human consumption), we could unlock a large number of useful applications for future machine learning applications and a vast amount of knowledge that could be put to work. Wikipedia, as one example, includes a large amount of knowledge that is linked in many ways to illustrate the relationships of the topics. Wikipedia itself would be an invaluable source of unstructured data to which NLP could be applied.

Let's now explore the history and methods for NLP.

History

NLP, much like AI, has a history of ups and downs. IBM's early work in 1954 for the Georgetown demonstration emphasized the huge benefits of machine translation (translating over 60 Russian sentences into English). This early approach used six grammar rules for a dictionary of 250 words and resulted in large investments into machine translation, but rules-based approaches could not scale into production systems.

MIT's SHRDLU (named based upon frequency order of letters in English) was developed in the late 1960s in LISP and used natural language to allow a user to manipulate and query the state of a blocks world. The blocks world, a virtual world filled with different blocks, could be manipulated by a user with commands like "Pick up a big red block." Objects could be stacked and queried to understand the state of the world ("is there

anything to the right of the red pyramid?”). At the time, this demonstration was viewed as highly successful, but could not scale to more complex and ambiguous environments.

During the 1970s and early 1980s, many chatbot-style applications were developed, which could converse about restricted topics. These were precursors to what is now called conversational AI, widely and successfully used in many domains. Other applications such as Lehnert's Plot Units implemented narrative summarization. This permitted summarization of a simple story with “plot units,” such as motivation, success, mixed-blessing, and other narrative building blocks.

In the late 1980s, NLP systems research moved from rules-based approaches to statistical models. With the introduction of the Internet, NLP became even more important with the flood of textual information becoming machine accessible.

[Site feedback](#)

Early work in NLP

In the 1960s, work began on applying meaning to sequences of words. In a process called tagging, sentences could be broken down into their parts of speech to understand their relationship within the sentence. These taggers relied on human-constructed rules-based algorithms to “tag” words with their context in a sentence (for example noun, verb, adjective, etc.). But there was considerable complexity in this tagging, since in English there can be up to 150 different types of speech tags.

Using Python's Natural Language Toolkit (NLTK), you can see the product of parts-of-speech tagging. In this example, the final set of tuples represent the tokenized words along with their tags (using the UPenn tagset). This tagset consists of 36 tags, such as VBG (verb, gerund, or present participle), NN (singular noun), PRP (personal pronoun), and so on.

```
>>> quote = "Knowing yourself is the beginning of all wisdom."
>>> tokens = nltk.word_tokenize( quote )
>>> tags = nltk.pos_tag( tokens )
>>> tags
[('Knowing', 'VBG'), ('yourself', 'PRP'), ('is', 'VBZ'),
 ('the', 'DT'), ('beginning', 'NN'), ('of', 'IN'), ('all', 'DT'),
 ('wisdom', 'NN'), ('.', '.')]
>>>
```

[Show more](#)

Tagging words might not seem complicated, but since words can mean different things depending upon where they are used, the process can be complicated. Parts-of-speech tagging is used as a prerequisite to other problems and is applied to a variety of NLP tasks.

Strict rules-based approaches to tagging have given way to statistical methods where ambiguity exists. Given a body of text (or corpus), one could calculate the probabilities of a word following another word. In some cases, the probabilities are quite high, where in others they are zero. The massive graph of words and their transitions probabilities is the product of training a machine to figure out which words are more likely to follow others and can be used in a variety of ways. As an example, in a speech recognition application, this word graph could be used to identify a word that was garbled by noise (based upon the probabilities of the word sequence that preceded it). This could also be used in an auto-correct application (to recommend a word for one that was misspelled). This technique is commonly solved using a Hidden Markov Model (HMM).

[Site feedback](#)

The HMM is useful in that a human doesn't need to construct this graph; a machine can construct it from a corpus of valid text. Additionally, it can be constructed based upon tuples of words (probability that a word follows another word called a bigram) or based upon an n-gram (where $n=3$, the probability that a word follows two other words in a sequence). HMMs have been applied not just to NLP but a variety of other fields (such as protein or DNA sequencing).

The following example illustrates the construction of bigrams from a simple sentence in NLTK:

```
>>> sentence = "the man we saw saw a saw"
>>> tokens = nltk.word_tokenize( sentence )
>>> list(nltk.bigrams( tokens ))
[('the', 'man'), ('man', 'we'), ('we', 'saw'), ('saw', 'saw'),
 ('saw', 'a'), ('a', 'saw')]
>>>
```

[Show more](#) ✓

Let's explore some of the modern approaches to NLP tasks.

Modern approaches

Modern approaches to NLP primarily focus on neural network architectures. As neural network architectures rely on numerical processing, an encoding is required to process words. Two common methods are one-hot encodings and word vectors.

Word encodings

A one-hot encoding translates words into unique vectors that can then be numerically processed by a neural network. Consider the words from our last bigram example. We create our one-hot vector of the dimension of the number of words to represent and assign a single bit in that vector to represent each word. This creates a unique mapping that can be used as input to a neural network (each bit in the vector as input to a neuron) (see Figure 2). This encoding is advantageous to simply encoding the words as numbers (label encoding) because networks can more efficiently train with one-hot vectors.

[Site feedback](#)

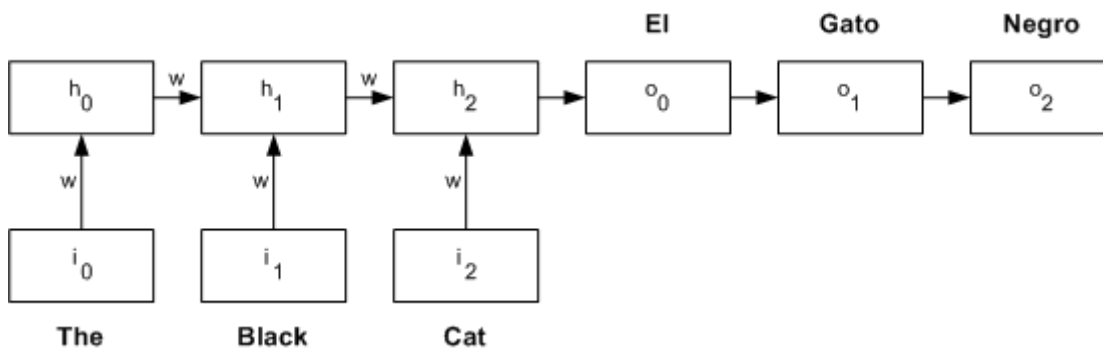
the	1	0	0	0	0
man	0	1	0	0	0
we	0	0	1	0	0
saw	0	0	0	1	0
a	0	0	0	0	1

Another encoding is word vectors, which represent words as highly dimensional vectors where units of the vector are real values. But rather than assign each unit a word (as in one-hot), each unit represents categories for the word (such as singular vs. plural or noun vs. verb) and can be 100-1,000 units wide (dimensionality). What makes this encoding interesting is that words are now numerically related, and the encoding opens up applying mathematical operations to word vectors (such as adding, subtracting, or negating).

Recurrent neural networks

Developed in the 1980s, recurrent neural networks (RNNs) have found a unique place in NLP. As the name implies, RNNs — as compared to typical feed-forward neural networks — operate in the time domain. RNNs unfold in time and operate in stages where prior outputs feed subsequent stage inputs (see Figure 3 for an unrolled network example). This type of architecture applies well to NLP since the network considers not just the words (or their encodings) but the context in which the words appear (what follows, what came before). In this contrived network example, input neurons are fed with the word

encodings, and the outputs feed forward through the network to the output nodes (with the goal of an output word encoding for language translation). In practice, each word encoding is fed one at a time and propagated through. On the next time step, the next word encoding is fed (with output occurring only after the last word is fed).


[Site feedback](#)

Traditional RNNs are trained through a variation of back-propagation called back-propagation through time (BPTT). A popular variation of RNNs is long short-term memory units (LSTMs), which have a unique architecture and the ability to forget information.

Reinforcement learning

Reinforcement learning focuses on selecting actions in an environment to maximize some cumulative reward (a reward that is not immediately understood but learned over many actions). Actions are selected based on a policy that defines whether the given action should explore new states (unknown territory where learning can take place) or old states (based on past experience).

High-level flow of reinforcement learning

In the context of NLP and machine translation, observations are sequences of words that are presented. The state represents a partial translation and the action the determination of whether a translation can be provided or if more information is needed (more observations, or words). As further observations are provided, the state may identify that sufficient information is available and a translation presented. The key to this approach is that the translation is done incrementally with reinforcement learning identifying when to commit to a translation or when to wait for more information (useful in languages where the main verb occurs at the end of the sentence).

Reinforcement learning has also been applied as the training algorithm for RNNs that implement text-based summarization.


Deep learning

Deep learning/deep neural networks have been applied successfully to a variety of problems. You'll find deep learning at the heart of Q&A systems, document summarization, image caption generation, text classification and modeling, and many others. Note that these cases represent natural language understanding and natural language generation.

Deep learning refers to neural networks with many layers (the deep part) that takes as features as input and extracts higher-level features from this data. Deep learning networks are able to learn a hierarchy of representations and different levels of abstractions of their input. Deep learning networks can use supervised learning or unsupervised learning and can be formed as hybrids of other approaches (such as incorporating a recurrent neural network with a deep learning network).

[Site feedback](#)

The most common approach to deep learning networks is the convolutional neural network (CNN), which is predominantly used in image-processing applications (such as classifying the contents of an image). Figure 5 illustrates a simple CNN for sentiment analysis. It consists of an input layer of word encodings (from the tokenized input), which then feeds the convolutional layers. The convolutional layers segment the input into many “windows” of input to produce feature maps. These feature maps are pooled with a max operation, which reduces the dimensionality of the output and provides the final representation of the input. This is fed into the final neural network that provides for the classification (such as positive, neutral, negative).

 Convolutional neural network architecture for sentiment analysis

While CNNs have proven to be efficient in image and language domains, other types of networks can also be used. The long short-term memory is a new kind of RNN. LSTM cells are more complex than typical neurons, since they include state and a number of internal gates that can be used to accept input, output data, or forget internal state information. LSTMs are commonly used in natural language applications. One of the most interesting uses of LSTMs was in concert with a CNN where the CNN provided the ability to process an image, and the LSTM was trained to generate a textual sentence of the contents of the input image.

Going further

The importance of NLP is demonstrated by the ever-growing list of applications that use it. NLP provides the most natural interface to computers and to the wealth of unstructured data available online. In 2011, IBM demonstrated Watson™, which competed with two of Jeopardy's greatest champions and defeated them using a natural language interface. Watson also used the 2011 version of Wikipedia as its knowledge

source – an important milestone on the path to language processing and understanding and an indicator of what’s to come. You can also go here to learn more about [NLP](#).

If you’re ready to start learning about and using natural language processing, see the [Get started with natural language processing](#) series.



Legend ⓘ

Categories

Artificial intelligence Deep learning Machine learning Natural language processing

Table of Contents

Resources

Related

Article

Convolutional neural networks

Site feedback

May 2, 2018



Article

A neural networks deep dive

Site feedback

July 24, 2017



Article

Recurrent neural networks deep dive

August 17, 2017



Build Smart ↓

Build Secure ↑

IBM Developer

- About
- FAQ
- Third-party notice

Follow Us

- Twitter
- LinkedIn
- Facebook

YouTube

Explore

- Newsletters
- Code patterns
- APIs
- Articles
- Tutorials
- Open source projects
- Videos
- Events

Select a language

- English
- Português
- Español

Site feedback

-
- Community
 - Career Opportunitites
 - Privacy
 - Terms of use
 - Accessibility
 - Cookie preferences