James Le
Software Engineering || Product Management || Data Science (https://jameskle.com/)
Jun 6 · 14 min read

The 7 NLP Techniques That Will Change How You Communicate in the Future (Part I)

What is NLP?

Natural Language Processing (NLP) is a field at the intersection of computer science, artificial intelligence, and linguistics. The goal is for computers to process or "understand" natural language in order to perform tasks like Language Translation and Question Answering.

With the rise of voice interfaces and chat-bots, NLP is one of the most important technologies of the information age a crucial part of artificial intelligence. Fully understanding and representing the meaning of

language is an extremely difficult goal. Why? Because human language is quite special.

What's special about human language? A few things actually:

- Human language is a system specifically constructed to convey the speaker/writer's meaning. It's not just an environmental signal but a deliberate communication. Besides, it uses an encoding that little kids can learn quickly; it also changes.
- Human language is mostly a discrete/symbolic/categorical signaling system, presumably because of greater signaling reliability.
- The categorical symbols of a language can be encoded as a signal for communication in several ways: sound, gesture, writing, images, etc. human language is capable of being any of those.
- Human languages are ambiguous (unlike programming and other formal languages); thus there is a high level of complexity in representing, learning, and using linguistic / situational / contextual / word / visual knowledge towards the human language.

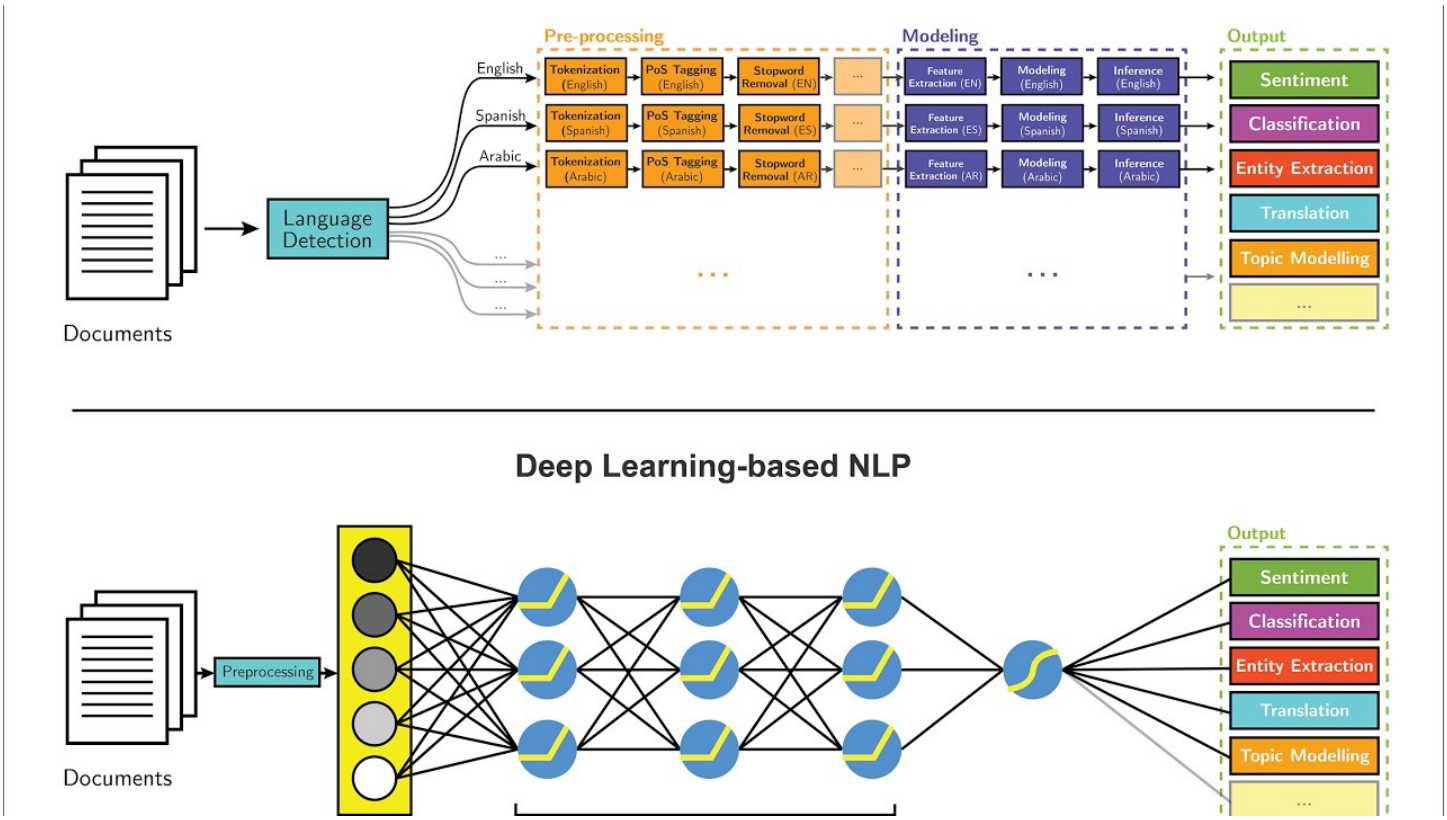
Why study NLP?

There's a fast-growing collection of useful applications derived from this field of study. They range from simple to complex. Below are a few of them:

- Spell Checking, Keyword Search, Finding Synonyms.
- Extracting information from websites such as: product price, dates, location, people, or company names.
- Classifying: reading level of school texts, positive/negative sentiment of longer documents.
- Machine Translation.
- Spoken Dialog Systems.
- Complex Question Answering.

Indeed, these applications have been used abundantly in industry: from **search** (written and spoken) to online advertisement **matching**; from automated/assisted **translation** to **sentiment analysis** for marketing or finance/trading; and from **speech recognition** to **chatbots/dialog**

agents (automating customer support, controlling devices, ordering goods).



Deep Learning

Most of these NLP technologies are powered by **Deep Learning**—a subfield of machine learning. Deep Learning only started to gain momentum again at the beginning of this decade, mainly due to these circumstances:

- Larger amounts of training data.
- Faster machines and multicore CPU/GPUs.
- New models and algorithms with advanced capabilities and improved performance: More flexible learning of intermediate representations, more effective end-to-end joint system learning, more effective learning methods for using contexts and transferring between tasks, as well as better regularization and optimization methods.

Most machine learning methods work well because of human-designed representations and input features, along with weight optimization to best make a final prediction. On the other hand, in deep learning,

representation learning attempts to automatically learn good features or representations from raw inputs. Manually designed features in machine learning are often over-specified, incomplete, and take a long time to design and validate. In contrast, deep learning's learned features are easy to adapt and fast to learn.

Deep Learning provides a very flexible, universal, and learnable framework for representing the world, for both visual and linguistic information. Initially, it resulted in breakthroughs in fields such as speech recognition and computer vision. Recently, deep learning approaches have obtained very high performance across many different NLP tasks. These models can often be trained with a single end-to-end model and do not require traditional, task-specific feature engineering.

I recently finished Stanford's comprehensive CS224n course on Natural Language Processing with Deep Learning. The course provides a thorough introduction to cutting-edge research in deep learning applied to NLP. On the model side, it covers word vector representations, window-based neural networks, recurrent neural networks, long-short-term-memory models, recursive neural networks, and convolutional neural networks, as well as some recent models involving a memory component.

On the programming side, I learned to implement, train, debug, visualize, and invent my own neural network models. In this 2-part series, I want to share the 7 major NLP techniques that I have learned as well as major deep learning models and applications using each of them.

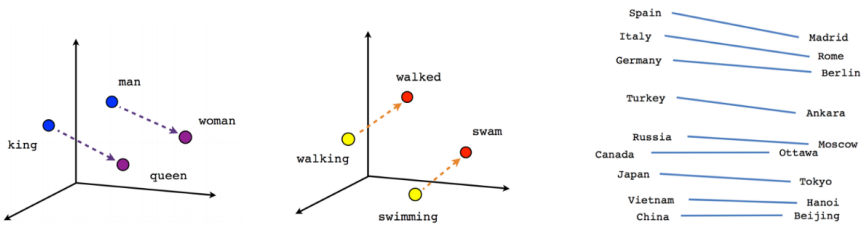
- **Quick Note:** You can access the lectures and programming assignments from CSS 224 at this GitHub Repo.

Technique 1: Text Embeddings

In traditional NLP, we regard words as discrete symbols, which can then be represented by one-hot vectors. A vector's dimension is the number of words in entire vocabulary. The problem with words as discrete symbols is that there is no natural notion of similarity for one-hot vectors. Thus, the alternative is to learn to encode similarity in the vectors themselves. The core idea is that **a word's meaning is given by the words that frequently appear close by**.

Text Embeddings are real valued vector representations of strings. We build a dense vector for each word, chosen so that it's similar to vectors

of words that appear in similar contexts. Word embeddings are considered a great starting point for most deep NLP tasks. They allow deep learning to be effective on smaller datasets, as they are often the first inputs to a deep learning architecture and the most popular way of transfer learning in NLP. The most popular names in word embeddings are **Word2vec** by Google (Mikolov) and **GloVe** by Stanford (Pennington, Socher and Manning). Let's delve deeper into these word representations:

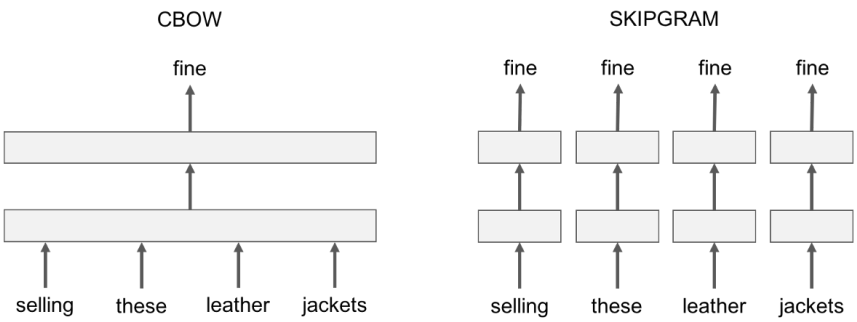


Male-Female Verb tense Country-Capital

In Word2vec, we have a large corpus of text in which every word in a fixed vocabulary is represented by a vector. We then go through each position *t* in the text, which has a center word *c* and context words *o*. Next, we use the similarity of the word vectors for *c* and *o* to calculate the probability of *o* given *c* (or vice versa). We keep adjusting the word vectors to maximize this probability.

For efficient training of Word2vec, we can eliminate meaningless (or higher frequency) words from the dataset (such as **a**, **the**, **of**, **then**…). This helps improve model accuracy and training time. Additionally, we can use negative sampling for every input by updating the weights for all the correct labels, but only on a small number of incorrect labels.

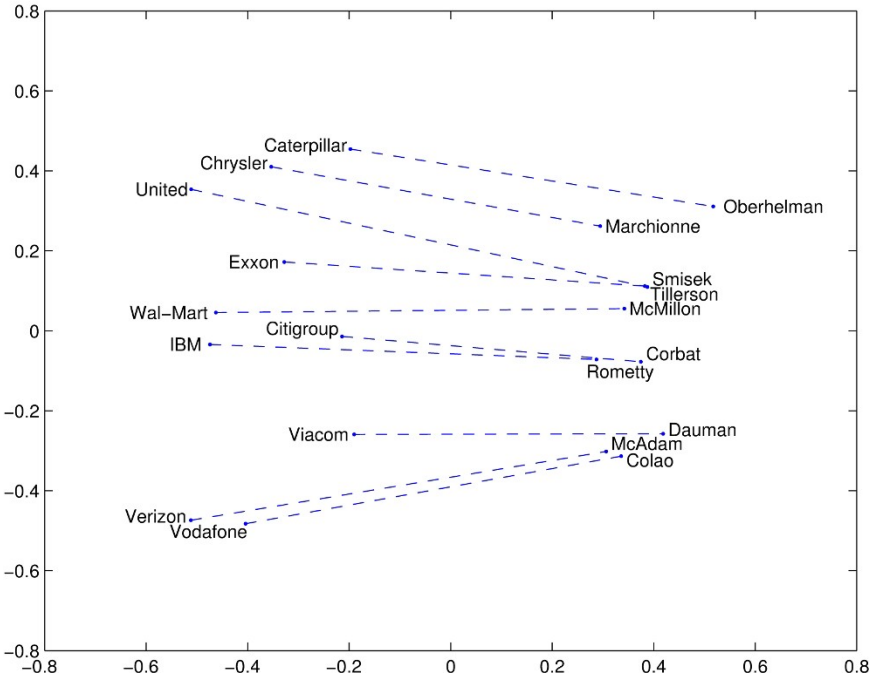
Word2vec has 2 model variants worth mentioning:



I am selling these fine leather jackets

1. **Skip-Gram:** We consider a context window containing *k* consecutive terms. Then we skip one of these words and try to learn a neural network that gets all terms except the one skipped and predicts the skipped term. Therefore, if 2 words repeatedly share similar contexts in a large corpus, the embedding vectors of those terms will have close vectors.

2. **Continuous Bag of Words:** We take lots and lots of sentences in a large corpus. Every time we see a word, we take the surrounding word. Then we input the context words to a neural network and predict the word in the center of this context. When we have thousands of such context words and the center word, we have one instance of a dataset for the neural network. We train the neural network and finally the encoded hidden layer output represents the embedding for a particular word. It so happens that when we train this over a large number of sentences, words in similar context get similar vectors.



One grievance with both Skip-Gram and CBOW is such that they are both window-based models, meaning the co-occurrence statistics of the corpus are not used efficiently, resulting in suboptimal embeddings. The GloVe model seeks to solve this problem by capturing the meaning of one word embedding with the structure of the whole observed corpus. To do so, the model One grievance with both Skip-Gram and CBOW is that they're both window-based models, meaning the co-occurrence statistics of the corpus are not used efficiently, resulting in suboptimal embeddings.

The GloVe model seeks to solve this problem by capturing the meaning of one word embedding with the structure of the whole observed corpus. To do so, the model trains on global co-occurrence counts of words and makes a sufficient use of statistics by minimizing least-squares error and, as a result, produces a word vector space with meaningful substructure. Such an outline sufficiently preserves words' similarities with vector distance.

Besides these 2 text embeddings, there are many more advanced models developed recently, including FastText, Poincare Embeddings, sense2vec, Skip-Thought, Adaptive Skip-Gram. I highly encourage you to check them out.

Technique 2: Machine Translation

Machine Translation is the classic test of language understanding. It consists of both language analysis and language generation. Big machine translation systems have huge commercial use, as global language is a $40 Billion-per-year industry. To give you some notable examples:

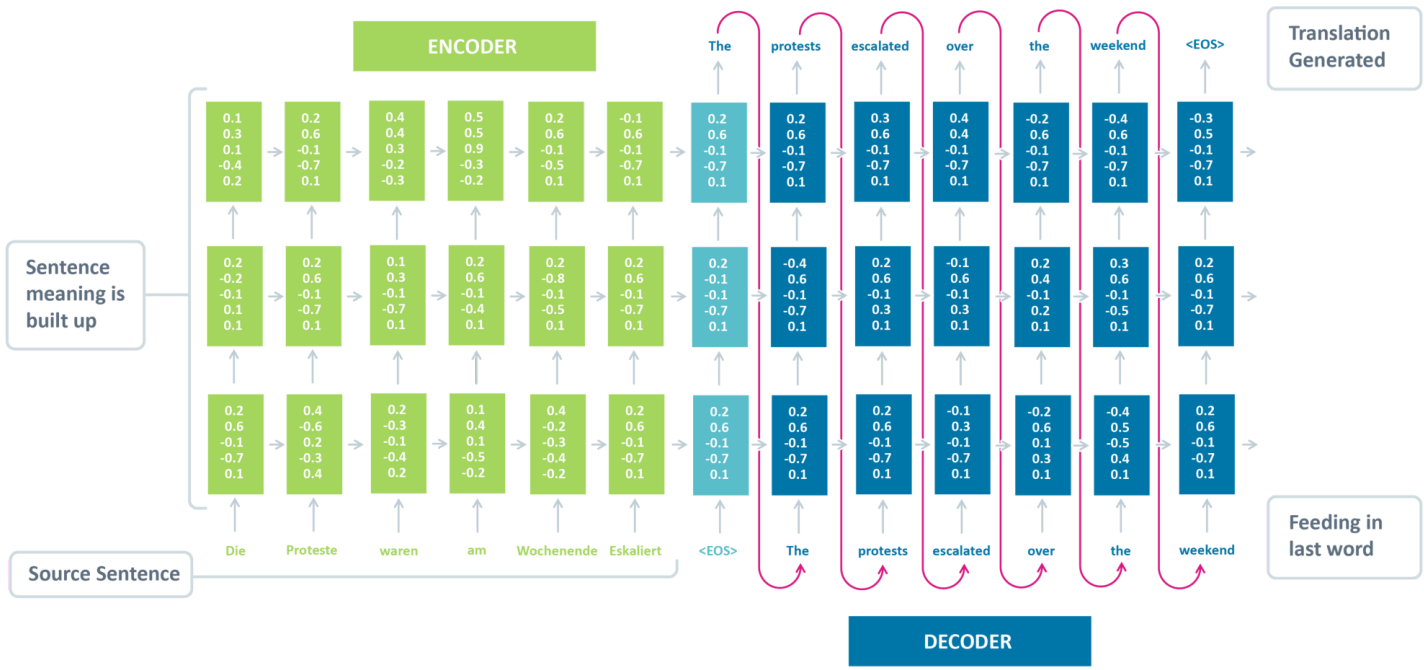
- Google Translate goes through 100 billion words per day.
- Facebook uses machine translation to translate text in posts and comments automatically, in order to break language barriers and allow people around the world to communicate with each other.
- eBay uses Machine Translation tech to enable cross-border trade and connect buyers and sellers around the world.
- Microsoft brings AI-powered translation to end users and developers on Android, iOS, and Amazon Fire, whether or not they have access to the Internet.
- Systran became the 1st software provider to launch a Neural Machine Translation engine in more than 30 languages back in 2016.

In a traditional Machine Translation system, we have to use parallel corpus—a collection of texts, each of which is translated into one or more other languages than the original. For example, given the source language f (e.g. French) and the target language e (e.g. English), we need to build multiple statistical models, including a probabilistic formulation using the Bayesian rule, a translation model p(f|e) trained on the parallel corpus, and a language model p(e) trained on the English-only corpus.

Needless to say, this approach skips hundreds of important details, requires a lot of human feature engineering, consists of many different & independent machine learning problems, and overall is a very complex system.

Neural Machine Translation is the approach of modeling this entire process via one big artificial neural network, known as a **Recurrent Neural Network**(RNN). RNN is a stateful neural network, in which it has connections between passes, connections through time. Neurons are fed information not just from the previous layer but also from themselves from the previous pass. This means that the order in which we feed the input and train the network matters: feeding it "Donald" and then "Trump" may yield different results compared to feeding it "Trump" and then "Donald".

A Recurrent Neural Network for Machine Translation



Standard Neural Machine Translation is an end-to-end neural network where the source sentence is encoded by a RNN called **encoder,** and the target words are predicted using another RNN known as **decoder.** The RNN Encoder reads a source sentence one symbol at a time, and then summarizes the entire source sentence in its last hidden state. The RNN Decoder uses back-propagation to learn this summary and returns the translated version. It's amazing that Neural Machine Translation went from a fringe research activity in 2014 to the widely adopted

leading way to do Machine Translation in 2016. So what are the big wins of using Neural Machine Translation?

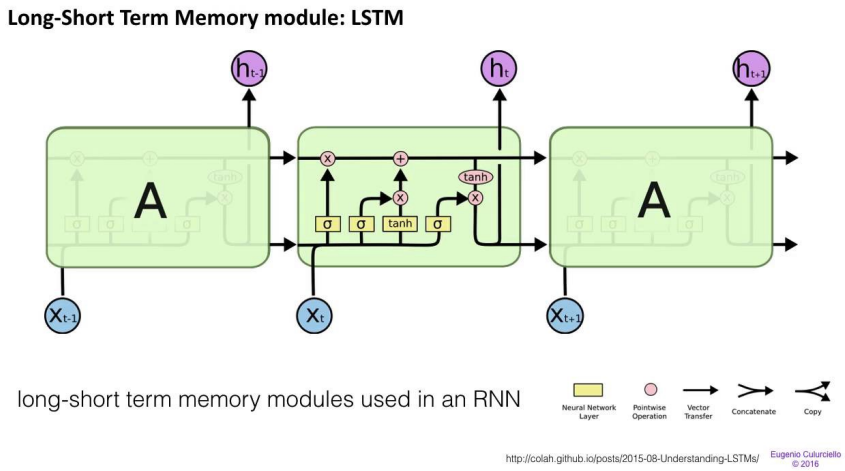
1. **End-to-end training**: All parameters in NMT are simultaneously optimized to minimize a loss function on the network's output.
2. **Distributed representations share strength**: NMT has a better exploitation of word and phrase similarities.
3. **Better exploration of context**: NMT can use a much bigger context—both source and partial target text—to translate more accurately.
4. **More fluent text generation**: Deep learning text generation is of much higher quality than the parallel corpus way.

One big problem with RNNs is the vanishing (or exploding) gradient problem where, depending on the activation functions used, information rapidly gets lost over time. Intuitively, this wouldn't be much of a problem because these are just weights and not neuron states, but the weights through time is actually where the information from the past is stored; if the weight reaches a value of 0 or 1,000,000, the previous state won't be very informative. As a consequence, RNNs will experience difficulty in memorizing previous words very far away in the sequence and are only able to make predictions based on the most recent words.

Long / short term memory (LSTM) networks try to combat the vanishing / exploding gradient problem by introducing gates and an explicitly defined memory cell. Each neuron has a memory cell and three gates: input, output and forget. The function of these gates is to safeguard the information by stopping or allowing the flow of it.

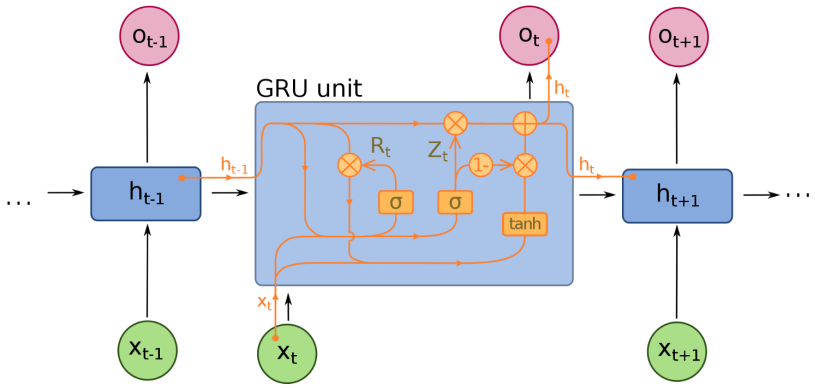
- The input gate determines how much of the information from the previous layer gets stored in the cell.
- The output layer takes the job on the other end and determines how much of the next layer gets to know about the state of this cell.
- The forget gate seems like an odd inclusion at first but sometimes it's good to forget: if it's learning a book and a new chapter begins, it may be necessary for the network to forget some characters from the previous chapter.

LSTMs have been shown to be able to learn complex sequences, such as writing like Shakespeare or composing primitive music. Note that each of these gates has a weight to a cell in the previous neuron, so they typically require more resources to run. LSTMs are currently very hip and have been used a lot in machine translation. Besides that, It is the default model for most sequence labeling tasks, which have lots and lots of data.



Gated recurrent units (GRU) are a slight variation on LSTMs and are also extensions of Neural Machine Translation. They have one less gate and are wired slightly differently: instead of an input, output, and a forget gate, they have an update gate. This update gate determines both how much information to keep from the last state and how much information to let in from the previous layer.

The reset gate functions much like the forget gate of an LSTM, but it's located slightly differently. They always send out their full state—they don't have an output gate. In most cases, they function very similarly to LSTMs, with the biggest difference being that GRUs are slightly faster and easier to run (but also slightly less expressive). In practice these tend to cancel each other out, as you need a bigger network to regain some expressiveness, which in turn cancels out the performance benefits. In some cases where the extra expressiveness is not needed, GRUs can outperform LSTMs.



Besides these 3 major architecture, there have been further improvements in neural machine translation system over the past few years. Below are the most notable developments:

- Sequence to Sequence Learning with Neural Networks proved the effectiveness of LSTM for Neural Machine Translation. It presents a general end-to-end approach to sequence learning that makes minimal assumptions on the sequence structure. The method uses a multilayered Long Short-Term Memory (LSTM) to map the input sequence to a vector of a fixed dimensionality, and then another deep LSTM to decode the target sequence from the vector.

- Neural Machine Translation by Jointly Learning to Align and Translate introduced the attention mechanism in NLP (which will be covered in the next post). Acknowledging that the use of a fixed-length vector is a bottleneck in improving the performance of NMT, the authors propose to extend this by allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word, without having to form these parts as a hard segment explicitly.

- Convolutional over Recurrent Encoder for Neural Machine Translation augments the standard RNN encoder in NMT with additional convolutional layers in order to capture wider context in the encoder output.

- Google built its own NMT system, called Google's Neural Machine Translation, which addresses many issues in accuracy and ease of deployment. The model consists of a deep LSTM network with 8 encoder and 8 decoder layers using residual connections as well as attention connections from the decoder network to the encoder.

- Instead of using Recurrent Neural Networks, Facebook AI Researchers uses convolutional neural networks for sequence to sequence learning tasks in NMT.

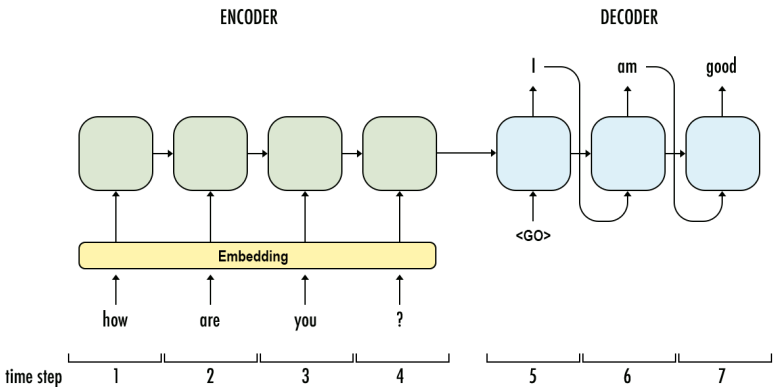
Technique 3: Dialogue and Conversations

A lot has been written about conversational AI, and a majority of it focuses on vertical chatbots, messenger platforms, business trends, and startup opportunities (think Amazon Alexa, Apple Siri, Facebook M, Google Assistant, Microsoft Cortana). AI's capability of understanding natural language is still limited. As a result, creating fully-automated, open-domain conversational assistants has remained an open challenge. Nonetheless, the work shown below serve as great starting points for people who want to seek the next breakthrough in conversation AI.

Researchers from Montreal, Georgia Tech, Microsoft and Facebook built a neural network that is capable of generating context-sensitive conversational responses. This novel response generation system can be trained end-to-end on large quantities of unstructured Twitter conversations. A Recurrent Neural Network architecture is used to address sparsity issues that arise when integrating contextual information into classic statistical models, allowing the system to take into account previous dialog utterances. The model shows consistent gains over both context-sensitive and non-context-sensitive Machine Translation and Information Retrieval baselines.

Developed in Hong Kong, Neural Responding Machine (NRM) is a neural-network-based response generator for short-text conversation. It takes the general encoder-decoder framework. First, it formalizes the generation of response as a decoding process based on the latent

representation of the input text, while both encoding and decoding are realized with Recurrent Neural Networks. The NRM is trained with a large amount of one-round conversation data collected from a microblogging service. Empirical study shows that NRM can generate grammatically correct and content-wise appropriate responses to over 75% of the input text, outperforming state-of-the-arts in the same setting.



Last but not least, Google's Neural Conversational Model is a simple approach to conversational modeling. It uses the sequence-to-sequence framework. The model converses by predicting the next sentence given the previous sentence(s) in a conversation. The strength of the model is such it can be trained end-to-end and thus requires much fewer hand-crafted rules.

The model can generate simple conversations given a large conversational training dataset. It is able to extract knowledge from both a domain specific dataset, and from a large, noisy, and general domain dataset of movie subtitles. On a domain-specific IT help-desk dataset, the model can find a solution to a technical problem via conversations. On a noisy open-domain movie transcript dataset, the model can perform simple forms of common sense reasoning.

That's the end of Part I. In the next post, I'll go over the remaining 4 Natural Language Processing techniques as well as discuss important limits and extensions in the field. Stay tuned!

. . .

If you enjoyed this piece, I'd love it if you hit the clap button 👏 so others might stumble upon it. You can find my own code on GitHub, and more of

my writing and projects at https://jameskle.com/. You can also follow me on Twitter, email me directly or find me on LinkedIn.

Discuss this post on Hacker News.

. . .

Editor's Note: Want to learn more about how machine learning can impact language? Check out these helpful Heartbeat resources:

- *Extractive Text Summarization Using Neural Networks*
- *Implementing a Natural Language Classifier in iOS with Keras + Core ML*
- *Explore this glossary of mobile machine learning tools, frameworks, and more*
- *Want to join the conversation? Join Heartbeat's new Slack community for everything and anything machine learning and mobile development*