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

James Le
Jul 3, 2018 · 15 min read

In part 1, I introduced the field of Natural Language Processing (NLP) and the deep learning movement that's powered it. I also walked you through 3 critical concepts in NLP: **text embeddings** (vector representations of strings), **machine translation** (using neural networks to translate languages), and **dialogue & conversation** (tech that can hold conversations with humans in real time). In part 2, I'll cover 4 other important NLP techniques that you should pay attention to in order to keep up with the fast growing pace of this research field.

Technique 4: Sentiment Analysis

Human communication isn't just words and their explicit meanings. Instead, it's nuanced and complex. You can tell based on the way a friend asks you a question whether they're bored, angry, or curious. You can tell based on word choice and punctuation whether a customer is getting furious, even in a completely text-based conversation.

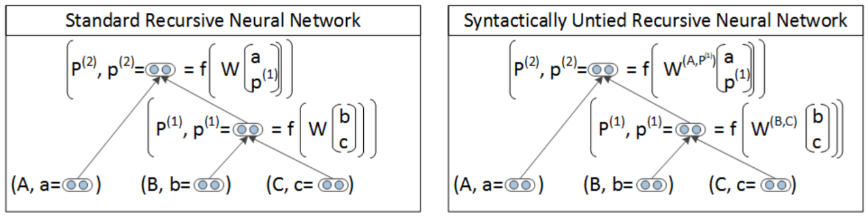
You can read an Amazon review for a product and understand whether the reviewer liked or disliked it even if they never directly said so. For computers to truly understand the way humans communicate every day, they need to understand more than the objective definitions of words; they need to understand our sentiments, what we really mean. Sentiment analysis is this process of interpreting the meaning of larger text units (entities, descriptive terms, facts, arguments, stories) by the semantic composition of smaller elements.

The traditional approach to sentiment analysis is to treat a sentence as a bag-of-words and to consult a curated list of "positive" and "negative" words to determine the sentiment of that particular sentence. This would require hand-designed features to capture the sentiment, which is extremely time-consuming and unscalable.

The modern deep learning approach for sentiment analysis can be used for morphology, syntax, and logical semantics, of which the most effective one is Recursive Neural Networks. As the name implies, the main assumption for Recursive Neural Net development is such that recursion is a natural way for describing language. Recursion is useful in disambiguation, helpful for some tasks to refer to specific phrases, and works extremely well for tasks that use a grammatical tree structure.

Recursive Neural Networks are perfect for settings that have a nested hierarchy and an intrinsic recursive structure. If we think about a sentence, doesn't this have such a structure? Take the sentence "A big crowd violently attacks the unarmed police." First, we break apart the sentence into its respective Noun Phrase and Verb Phrase—"A big crowd" and "violently attacks the unarmed police." But there's a noun phrase within that verb phrase, right? "violently attacks" and "unarmed police." Seems pretty recursive to me.

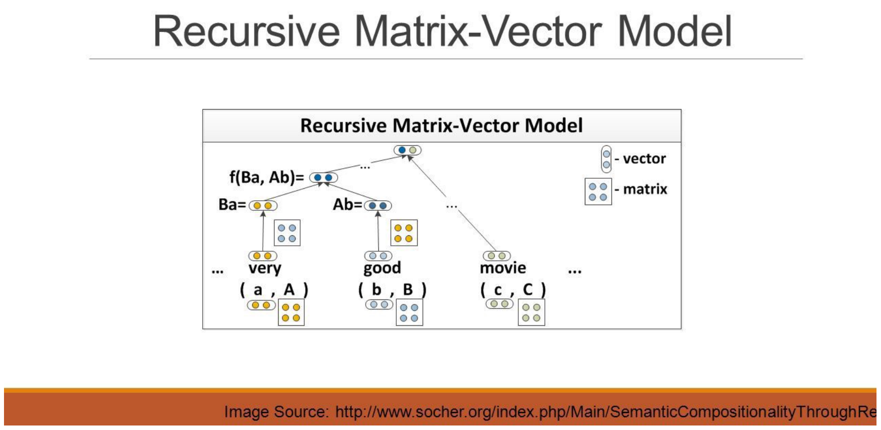
The syntactic rules of language are highly recursive. So we take advantage of that recursive structure with a model that respects it! Another added benefit of modeling sentences with RNN's is that we can now input sentences of arbitrary length, which was a huge head scratcher for using Neural Nets in NLP, with very clever tricks to make the sentence's input vector to be of equal size, despite the length of the sentences not being equal.


Standard Recursive Neural Network
$\left(P^{(2)}, p^{(2)}=\right) = f\left(W\begin{bmatrix}a\\p^{(1)}\end{bmatrix}\right)$
$\left(P^{(1)}, p^{(1)}=\right) = f\left(W\begin{bmatrix}b\\c\end{bmatrix}\right)$
(A, a=)
(B, b=)
(C, c=)
Syntactically Untied Recursive Neural Network
$\left(P^{(2)}, p^{(2)}=\right) = f\left(W^{(A,P^{1})}\begin{bmatrix}a\\p^{(1)}\end{bmatrix}\right)$
$\left(P^{(1)}, p^{(1)}=\right) = f\left(W^{(B,C)}\begin{bmatrix}b\\c\end{bmatrix}\right)$
(A, a=)
(B, b=)
(C, c=)

The Standard RNN is the most basic version of a Recursive Neural Network. It has a max-margin structure prediction architecture that can successfully recover such structure both in complex scene images as well as sentences. It's used to provide a competitive syntactic parser for natural language sentences from the Penn Treebank. For your reference, the Penn Treebank is the 1st large-scale treebank dataset composed of 2,499 stories from a three year Wall Street

Journal (WSJ) collection of 98,732 stories for syntactic annotation. Additionally, it outperforms alternative approaches for semantic scene segmentation, annotation, and classification.

However, the standard RNN captures neither the full syntactic nor semantic richness of linguistic phrases. The Syntactically Untied RNN, otherwise known as Compositional Vector Grammar (CVG), is a major upgrade that addresses this issue. It uses a syntactically untied recursive neural network that learns syntactic-semantic and compositional vector representations. The model is fast to train and implemented as efficiently as the standard RNN. It learns a soft notion of head words and improves performance on the types of ambiguities that require semantic information.

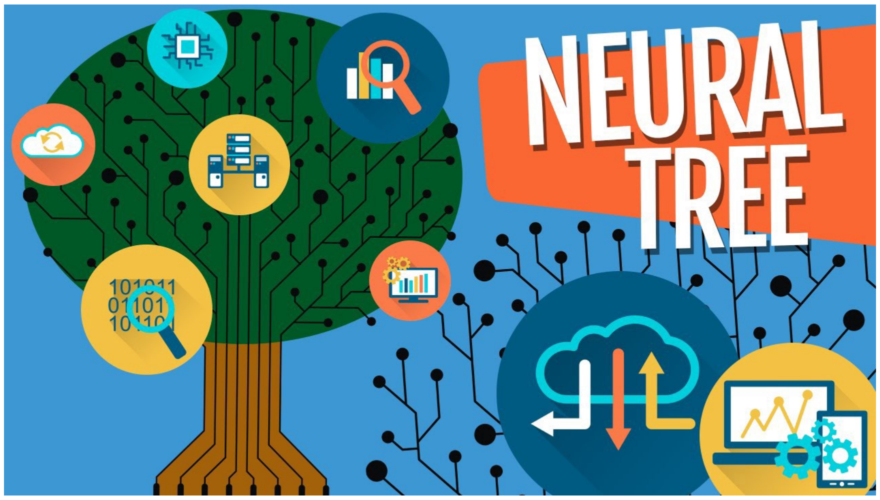

Recursive Matrix-Vector Model
Recursive Matrix-Vector Model
f(Ba, Ab)=
Ba=
Ab=
- vector
- matrix
...
very
good
movie
(a , A)
(b , B)
(c , C)
Image Source: http://www.socher.org/index.php/Main/SemanticCompositionalityThroughRe

Another evolution is the Matrix-Vector RNN, which is capable of capturing the compositional meaning of even much longer phrases. The model assigns a vector and a matrix to every node in a parse tree: the vector captures the inherent meaning of the constituent, while the matrix captures how it changes the meaning of neighboring words or phrases. This matrix-vector RNN can learn the meaning of operators in propositional logic and natural language.

As a result, the model obtains state of the art performance on three different experiments:

- Predicting fine-grained sentiment distributions of adverb-adjective pairs.
- Classifying sentiment labels of movie reviews.

- Classifying semantic relationships such as cause-effect or topic-message between nouns using the syntactic path between them.

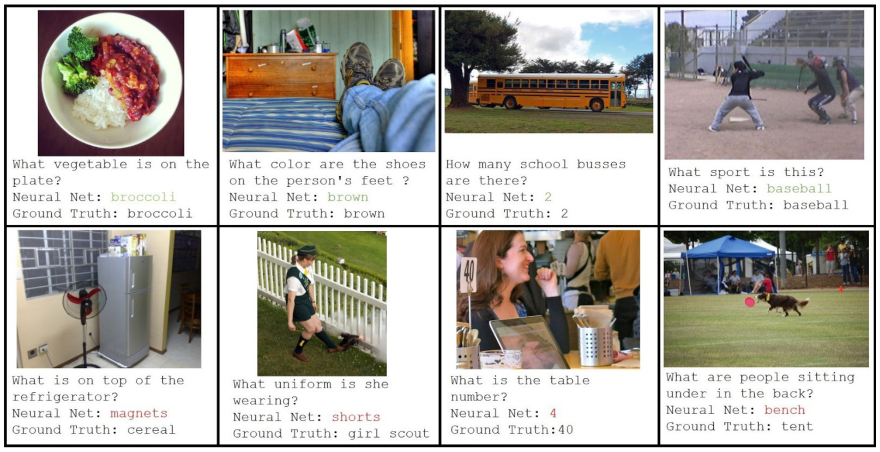


The most powerful RNN model for sentiment analysis developed thus far is Recursive Neural Tensor Network, which has a tree structure with a neural net at each node. This model can be used for boundary segmentation to determine which word groups are positive and which are negative. The same applies to sentences as a whole. When trained on the Sentiment Treebank, this model outperformed all previous methods on several metrics by more than 5%. Currently, it's the only model that can accurately capture the effects of negation and its scope at various tree levels for both positive and negative phrases.

Technique 5: Question Answering

The idea of a **Question Answering (QA) system** is to extract information, directly from documents, conversations, online searches, and elsewhere, that will meet a user's information needs. Rather than make the user read through an entire document, a QA system prefers to give a short and concise answer. Nowadays, a QA system can combine very easily with other NLP systems like chatbots, and some QA systems even go beyond the search of text documents and can extract information from a collection of pictures.

In fact, most of the NLP problems can be considered as a question answering problem. The paradigm is simple: we issue a query, and the machine provides a response. By reading through a document, or a set of instructions, an intelligent system should be able to answer a wide variety of questions. So naturally, we'd like to design a model that can be used for general QA.



A powerful deep learning architecture, known as dynamic memory network(DMN), has been developed and optimized specifically for QA problems. Given a training set of input sequences (knowledge) and questions, it can form episodic memories, and use them to generate relevant answers. The architecture has the following components:

- The **Semantic Memory Module** (analogous to a knowledge base) consists of pre-trained GloVe vectors that are used to create sequences of word embeddings from input sentences. These vectors will act as inputs to the model.

- The **Input Module** processes the input vectors associated with a question into a set of vectors termed *facts*. This module is implemented using a Gated Recurrent Unit. The GRU enables the network to learn if the sentence currently under consideration is relevant or has nothing to do with the answer.

- The **Question Module** processes the question word by word, and outputs a vector using the same GRU as the input module, and the same weights. Both facts and questions are encoded as embeddings.

- The **Episodic Memory Module** receives the fact and question vectors extracted from the input and encoded as embeddings. This uses a process inspired by the brain's hippocampus, which can retrieve temporal states that are triggered by some response, like sights or sounds.

- Finally, the **Answer Module** generates an appropriate response. By the final pass, the episodic memory should contain all the information required to answer the question. This module uses another GRU, trained with the cross-entropy error classification of the correct sequence, which can then be converted back to natural language.

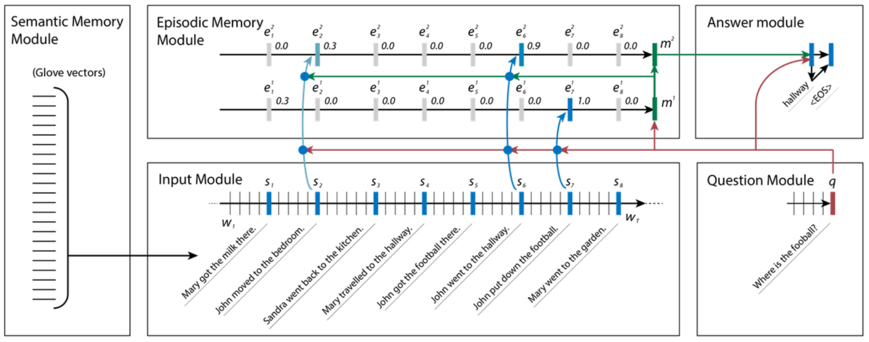


Illustration of DMN performing transitive inference.

DMN not only did extremely well for QA tasks, but also outperformed other architectures for sentiment analysis and part-of-speech tagging. Since its inception, there have been major improvements to Dynamic Memory Networks to further improve their accuracy on question answering tasks, including:

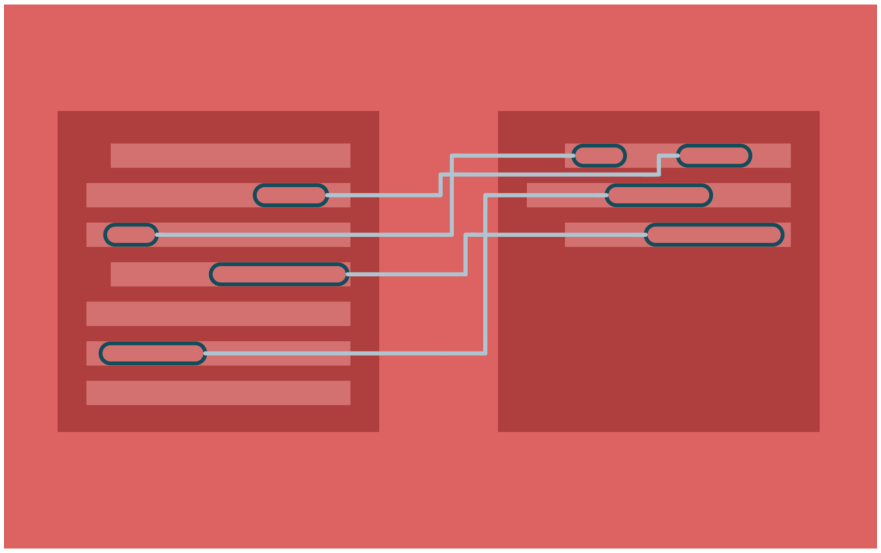
- Dynamic Memory Networks for Visual and Textual Question Answering is basically DMN being applied to images. Its memory and input modules are upgraded in order to be able to answer visual questions. This model improves the state of the art on many benchmark Visual Question Answering datasets without supporting fact supervision.

- Dynamic Coattention Networks for Question Answering addresses the problem of recovering from local maxima corresponding to incorrect answers. It first fuses co-dependent representations of the question and the document in order to focus on relevant parts of both. Then, a dynamic pointing decoder iterates over potential answer spans. This iterative procedure enables the model to recover from initial local maxima corresponding to incorrect answers.

Technique 6: Text Summarization

It’s very difficult for human beings to manually summarize large documents of text. Text summarization is the problem in NLP of creating short, accurate, and fluent summaries for source documents. It’s become an important and timely tool for assisting and interpreting text information in today’s fast-growing information age. With push notifications and article digests gaining more and more traction, the task of generating intelligent and accurate summaries for long pieces of text has been growing every day.

Automatic summarization of text works by first calculating the word frequencies for the entire text document. Then, the 100 most common words are stored and sorted. Each sentence is then scored based on how many high frequency words it contains, with higher frequency words being worth more. Finally, the top X sentences are taken and sorted based on their position in the original text.



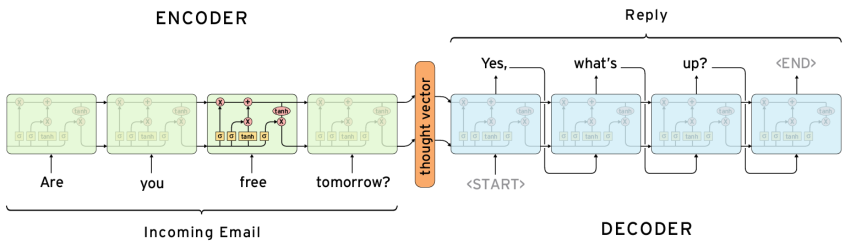
By keeping things simple and for a general purpose, the automatic text summarization algorithm is able to function in a variety of situations that other implementations might struggle with, such as documents containing foreign languages or unique word associations that aren't found in standard english language corpuses.

There are two fundamental approaches to text summarization: **extractive** and **abstractive**. The former extracts words and word phrases from the original text to create a summary. The latter learns an internal language representation to generate more human-like summaries, paraphrasing the intent of the original text.

The methods in **extractive summarization** work by selecting a subset. This is done by extracting the phrases or sentences from the actual article to form a summary. **LexRank** and **TextRank** are well known extractive summarizations. Both of them use a variation of the Google PageRank algorithm.

- LexRank is an unsupervised graph-based algorithm that uses IDF-modified Cosine as the similarity measure between two sentences. This similarity is used as weight of the graph edge between two sentences. LexRank also incorporates an intelligent post-processing step that makes sure top sentences chosen for the summary are not too similar to each other.
- TextRank is a similar algorithm to LexRank with a few enhancements, such as using lemmatization instead of stemming, incorporating Part-Of-Speech tagging and Named

Entity Resolution, extracting key phrases from the article, and extracting summary sentences based on those phrases. Along with a summary of the article, TextRank also extracts meaningful key phrases from the article.



Models for **abstractive summarization** fall under the larger umbrella of deep learning. There have been certain breakthroughs in text summarization using deep learning. Below are some of the most notable published results by some of the biggest companies in the field of NLP:

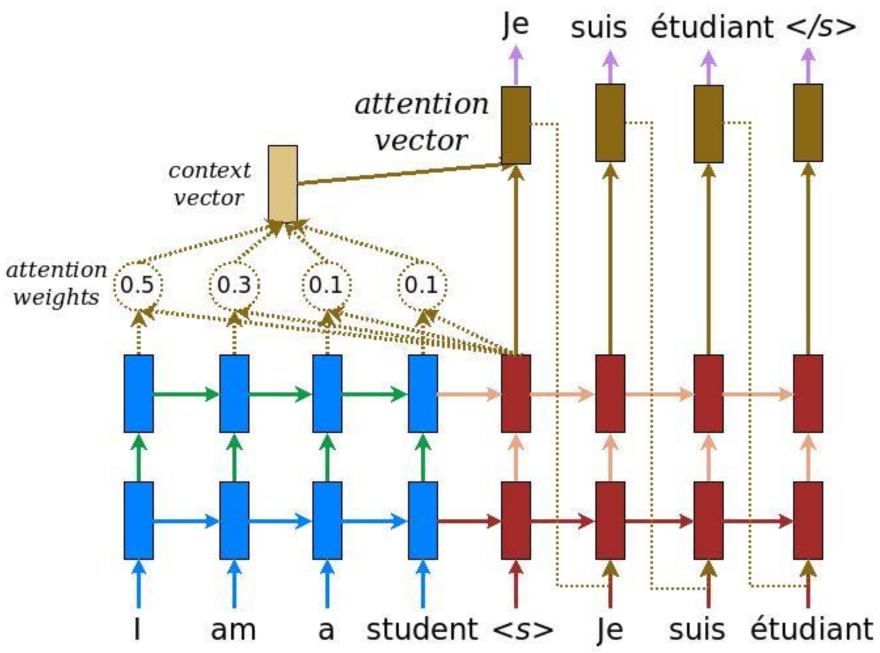
- Facebook's Neural Attention is a neural network architecture that utilizes a local attention-based model capable of generating each word of the summary conditioned on the input sentence.
- Google Brain's Sequence-to-Sequence model follows an encoder-decoder architecture. The encoder is responsible for reading the source document and encoding it to an internal representation. The decoder is a language model responsible for generating each word in the output summary using the encoded representation of the source document.
- IBM Watson uses the similar Sequence-to-Sequence model, but with attention and bidirectional recurrent neural network features.

Technique 7: Attention Mechanism

Attention Mechanisms in Neural Networks are loosely based on the visual attention mechanism found in humans. Human visual attention is well-studied and while there exist different models, all of them essentially come down to being able to focus on a certain region of an image with "high resolution" while perceiving the surrounding image in “low resolution,” and then adjusting the focal point over time.

Imagine you’re reading a whole essay: instead of going through each word or character sequentially, you subconsciously focus on a few sentences of highest information density and filter out the rest. Your attention effectively captures contextual information in a hierarchical manner, such that it’s sufficient for decision making while reducing overheads. Attention Mechanisms in Neural Networks are loosely based on the visual attention mechanism found in humans.

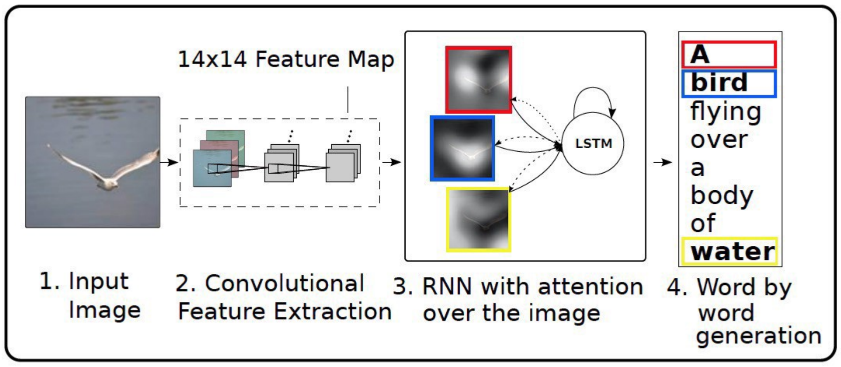
So why is this important? Models such as LSTM and GRU rely on reading a complete sentence and compressing all the information into a fixed-length vector. This requires sophisticated feature engineering based on the statistical properties of text. A sentence with hundreds of words represented by several words will surely lead to information loss, inadequate translation, etc.



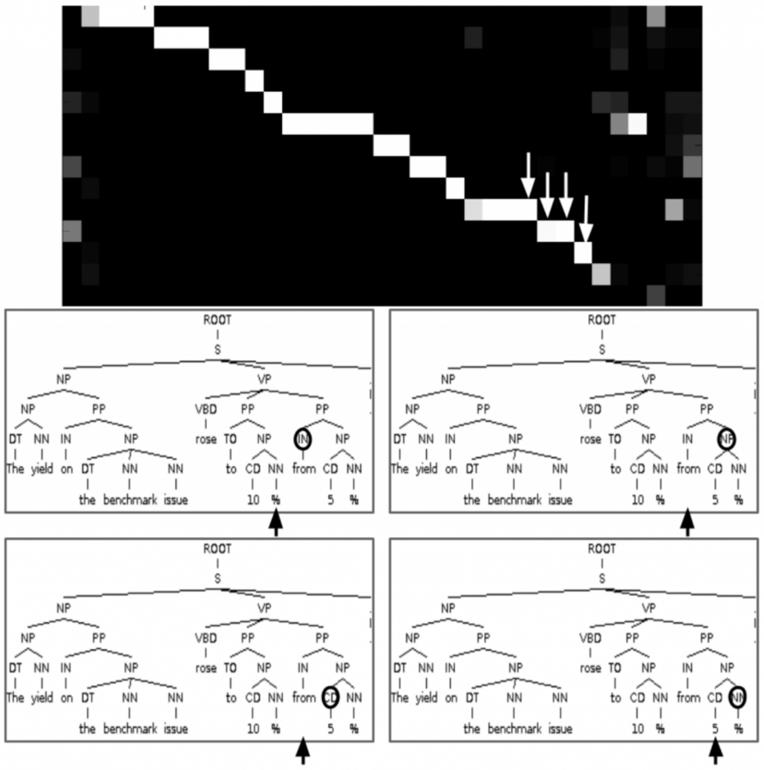
With an attention mechanism, we no longer try encode the full-surge sentence into a fixed-length vector. Rather, we allow the decoder to attend to different parts of the source sentence at each step of the output generation. We let the model learn what to attend to based on the input sentence and what it has produced so far.

According to the image above from Effective Approaches to Attention-Based Neural Machine Translation, blue represents encoder and red represents decoder, so we can see that the context vector takes all cells' outputs as input to compute the probability distribution of source language words for each single word the decoder wants to generate. By utilizing this mechanism, it's possible for the decoder to capture global information rather than solely to infer based on one hidden state.

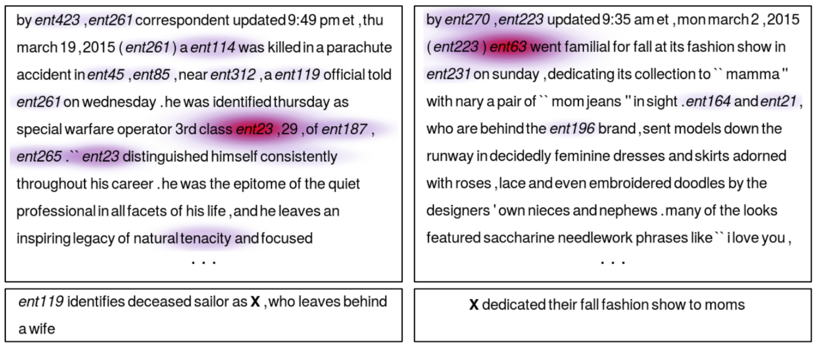
Besides Machine Translation, the attention model works on a variety of other NLP tasks. In Show, Attend and Tell: Neural Image Caption Generation with Visual Attention, the authors apply attention mechanisms to the problem of generating image descriptions. They use a Convolutional Neural Network to encode the image, and a Recurrent Neural Network with attention mechanisms to generate a description. By visualizing the attention weights, they interpret what the model is looking at while generating a word:


14x14 Feature Map
LSTM
A
bird
flying
over
a
body
of
water
1. Input Image
2. Convolutional Feature Extraction
3. RNN with attention over the image
4. Word by word generation

In Grammar as a Foreign Language, the authors use a Recurrent Neural Network with attention mechanism to generate sentence-parsed trees. The visualized attention matrix gives insight into how the network generates those trees:



In Teaching Machines to Read and Comprehend, the authors use a Recurrent Neural Network to read a text, read a question, and then produce an answer. By visualizing the attention matrix, they show where the network looks while trying to find the answer to the question:


by *ent423* ,*ent261* correspondent updated 9:49 pm et ,thu march 19 ,2015 (*ent261*) a *ent114* was killed in a parachute accident in *ent45* ,*ent85* ,near *ent312* ,a *ent119* official told *ent261* on wednesday .he was identified thursday as special warfare operator 3rd class *ent23* ,29 ,of *ent187* , *ent265* .`` *ent23* distinguished himself consistently throughout his career .he was the epitome of the quiet professional in all facets of his life ,and he leaves an inspiring legacy of natural tenacity and focused
. . .

ent119 identifies deceased sailor as **X** ,who leaves behind a wife

by *ent270* ,*ent223* updated 9:35 am et ,mon march 2 ,2015 (*ent223*) *ent63* went familial for fall at its fashion show in *ent231* on sunday ,dedicating its collection to `` mamma '' with nary a pair of `` mom jeans '' in sight .*ent164* and *ent21* , who are behind the *ent196* brand ,sent models down the runway in decidedly feminine dresses and skirts adorned with roses ,lace and even embroidered doodles by the designers ' own nieces and nephews .many of the looks featured saccharine needlework phrases like `` i love you ,
. . .

X dedicated their fall fashion show to moms

Attention does come at a cost, however. We need to calculate an attention value for each combination of input and output word. If you have a 100-word input sequence and generate a 100-word output sequence, that would be 10,000 attention values. If you do characterlevel computations and deal with sequences consisting of hundreds of tokens, the above mechanisms can become prohibitively expensive.

Natural Language Processing Obstacles

It should be noted that in each of the 7 NLP techniques I have discussed over these 2 posts, researchers have had to deal with a variety of obstacles: limits of the algorithms, scalability of the models, vague understanding of the human language. . .The good news is that the development of this field seems like a giant open-source project: researchers keep building better models to solve the existing problems and sharing their results with the community. Here are the major obstacles in NLP that have been resolved thanks to recent academic research progress:

- There is no single model architecture with consistent state-of-the-art results **across tasks**. For example, in Question Answering, we have Strongly Supervised End-to-End Memory Networks; in Sentiment Analysis, we have Tree-LSTMs; and in Sequence Tagging, we have Bidirectional LSTM-CRF. The Dynamic Memory Network I mentioned earlier in the Question Answering section somehow addressed this challenge, as it could perform well consistently across multiple domains.

- A powerful approach in machine learning is **multi-task learning,** which shares representations between related tasks in order to enable the model to generalize better on the original task. However, fully-joint multitask learning is hard, as it's usually restricted to lower layers, useful only if tasks are related (often hurts performance if tasks are not related), and has the same decoder/classifier in the proposed model. In A Joint Many-Task Model: Growing a NN for Multiple NLP Tasks, the authors pre-define a hierarchical architecture consisting of several NLP tasks as a joint model for multi-task learning. The model includes character n-grams and short-circuits as well as a state-of-the-art, purely feedforward parser, capable of performing dependency parsing, multi-sentence tasks, and joint training.

- **Zero-shot learning** is the ability to solve a task despite not having received any training examples of that task. There aren't many models capable of doing zero shot learning for NLP, as answers can only be predicted if they were seen during training and as part of the softmax function. In order to tackle this obstacle, the authors of Pointer Sentinel Mixture Models have combined a standard LSTM softmax with Pointer Networks in a mixture model. The pointer networks help with rare words and long-term dependencies, while the standard softmax can refer to words that are not in the input.

- Another challenge is the problem of **duplicate word representations**, where different encodings for the encoder and decoder in a model result in duplicate parameters / meanings. The simplest solution for this is to tie word vectors together and train single weights jointly, as demonstrated in Tying Word Vectors and Word Classifiers: A Loss Framework for Language Modeling.
- Another big obstacle is that Recurrent Neural Networks, the basic building block for any Deep NLP techniques, are quite slow compared to, say, Convolutional Neural Nets or Feedforward Neural Nets. Quasi-Recurrent Neural Networks take the best parts of RNNs and CNNs to enhance the training speed, using convolutions for parallelism across time and element-wise gated recurrence for parallelism across channels. This approach is better and faster than any other models in language modeling and sentiment analysis.
- Finally, in NLP, **architecture search**—the process of using machine learning to automate the design of artificial neural networks—is quite slow, as the traditional manual process requires a lot of expertise. What if we could use AI to find the right architecture for any problem? Neural architecture search with reinforcement learning from Google Brain is the most viable solution developed so far. The authors use a recurrent network to generate the model descriptions of neural networks and train this RNN with reinforcement learning to maximize the expected accuracy of the generated architectures on a validation set.

Conclusion

So there you go! I showed you a basic rundown of the major natural language processing techniques that can help a computer extract, analyze, and understand useful information from a single text or sequence of texts.

From machine translation that connects humans across cultures, to conversational chatbots that help with customer service; from sentiment analysis that deeply understands a human's mood, to attention mechanisms that can mimic our visual attention, the field of NLP is too expansive to cover completely, so I'd encourage you to explore it further, whether through online courses, blog tutorials, or research papers.

I'd highly recommend Stanford's CS 224 for starters, as you'll learn to implement, train, debug, visualize, and invent your own neural network models for NLP tasks. As a bonus, you can get all the lecture slides, assignment guidelines, and source code from **my GitHub repo**. I hope it'll guide you in the quest of changing how we'll communicate in the future!

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. Sign up for my newsletter to receive my latest thoughts on data science, machine learning, and artificial intelligence right at your inbox!*

Discuss the post on Hacker News.

• • •

Editor's Note: Ready to dive into some code? Check out Fritz on GitHub. You'll find open source, mobile-friendly implementations of the popular machine and deep learning models along with training scripts, project templates, and tools for building your own ML-powered iOS and Android apps.

And follow us on Twitter and LinkedIn for the all the latest content, news, and more from the mobile machine learning world.