Economics 2355: Models of Words

Melissa Dell

Harvard University

March 2021

TOC

Language Modeling

Is a sentence probable? Language modeling aims to answer this question using the following framework:

$$P(X) = \prod_{i=1}^{I} P(x_i | x_1, \ldots, x_i - 1) \tag{1}$$

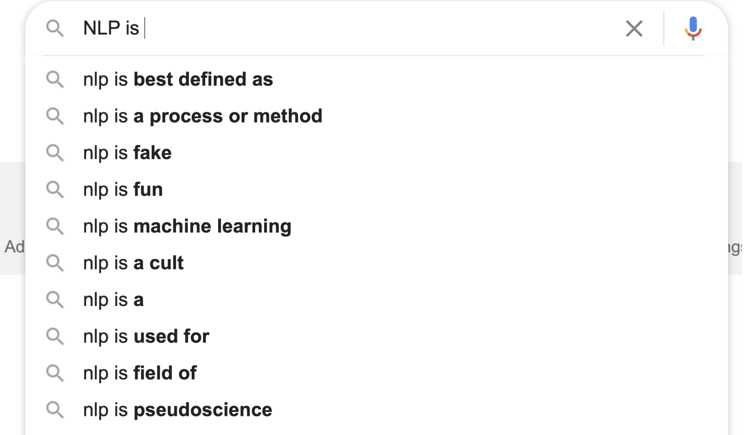
$P(x_i)$ is the probability of next word given the context

i.e. a sentence like “The girl went to school” should have high probability, whereas “School to the girl went” should have low probability

What Would We Use This For?

- Score sentences (grammar check)
- Generate sentences (auto-complete)
- Most often for us, a means to another end (the features representation from a language model is useful for another downstream task)

What Would We Use This For?



This commercial value has driven the development of mind-blowing language models, that we can use to do totally different tasks downstream (an amazing, unintended gift from Google to social science research)

Human Engineered Approaches

- Suppose we wanted to use a human engineered approach (i.e. rules) to compute the probabilities
- We'd need to encode a massive amount of information about grammar, morphology, context, etc, as well as the many exceptions

Engineering a Language Model

Jane went to the store.

store to Jane went the.

Jane went store.

} Create a grammar of the language

Jane goed to the store. } Consider morphology and exceptior

The store went to Jane. } Semantic categories, preferences

The food truck went to Jane.} And their exceptions

Carnegie Mellon CS 11-747

Need a Completely New Set of Rules if Want to Look at Similar Sentences in Japanese

ジェインは店へ行った。

は店行ったジェインは。

ジェインは店へ行た。

店はジェインへ行った。

屋台はジェインのところへ行った。

Carnegie Mellon CS 11-747

Outline

Count Based Models

- Linguists traditionally used count based approaches to model language
- The idea is to predict the next word using the frequencies of word sequences

$$P(x_i|x_{i-n}, \ldots, x_i - 1) = \frac{c(x_{i-n} \ldots x_i)}{c(x_{i-n} \ldots x_{i-1})} \tag{2}$$

i.e. count the frequency of “The teacher read” and divide it by the frequency of “The teacher”.

Evaluating this Language Model

Calculate log-likelihood of all sentences in the corpus

$$LL(corpus) = \sum_{S \in corpus} logP(S) \tag{3}$$

Where S is a sentence. Recall:

$$P(S) = \prod_{i=1}^{l} P(x_i | x_1, \ldots, x_i - 1) \tag{4}$$

Take a corpus of test data (not seen when the counts were computed on the training data).

Use the model to compute the probability of each sentence and sum up over all sentences in the corpus.

The more extensible the model, the higher the likelihood.

Can also normalize by number of words in the corpus.

N-Grams

An N-gram is a sequence of *n* consecutive words, i.e.:

- unigram: “This”, “class’, “covers”, “NLP”
- bigram: “This class’, “class covers”, “covers NLP”
- trigram: “This class covers”, “class covers NLP”
- 4-gram: “This class covers NLP”

Counts of different N-grams used to predict the next word

Problems with Count Based Models

- Treats all words as iid. Cannot account for synonyms.
- Cannot handle long-distance dependencies (i.e. for **ballet** lessons she needed to bring her new **shoes**; for **tennis** lessons she need to bring her new **ball**)
- Long-range dependencies are very common in human language

Dependencies and Sparsity

Take the example of ”This class covers *mask*”. We want to correctly predict NLP. But what if “This class covers NLP” never occurred in the training data? Even worse, what if “This class covers” never appeared (can’t divide by zero). Could handle this by a technique called “backoff” (i.e. look for “class covers”).

This sparsity problem becomes more severe as n increases. In practice, cannot have n bigger than 5.

Moreover, size of the model grows exponentially in n.

N-gram text generation is incoherent

Stanford Linguistics 284 gives the following example of text generated by a 3-gram:

> *today the price of gold per ton , while production of shoe lasts and shoe industry , the bank intervened just after it considered and rejected an imf demand to rebuild depleted european stocks , sept 30 end primary 76 cts a share .*

Need long-range dependencies to generate coherent text. But increasing *n* will significantly worsen sparsity and exponentially increase model size.

The Deep Learning Alternative: Featurized Models

- Calculate features from the context
- Learn the weights used to calculate the features
- Compute probabilities from those features

Compare to Deep Learning Generated Text

SYSTEM PROMPT (HUMAN-WRITTEN)

A train carriage containing controlled nuclear materials was stolen in Cincinnati today. Its whereabouts are unknown.

MODEL COMPLETION (MACHINE-WRITTEN, FIRST TRY)

The incident occurred on the downtown train line, which runs from Covington and Ashland stations.

In an email to Ohio news outlets, the U.S. Department of Energy said it is working with the Federal Railroad Administration to find the thief.

“The theft of this nuclear material will have significant negative consequences on public and environmental health, our workforce and the economy of our nation,” said Tom Hicks, the U.S. Energy Secretary, in a statement. “Our top priority is to secure the theft and ensure it doesn’t happen again.”

The stolen material was taken from the University of Cincinnati’s Research Triangle Park nuclear research site, according to a news release from Department officials.

The Nuclear Regulatory Commission did not immediately release any information.

According to the release, the U.S. Department of Energy’s Office of Nuclear Material Safety and Security is leading that team’s investigation.

“The safety of people, the environment and the nation’s nuclear stockpile is our highest priority,” Hicks said. “We will get to the bottom of this and make no excuses.

Open AI

The Power of Deep Learning for NLP

Differences on other downstream tasks like sentiment analysis or retrieval are also striking.

If you want to use text data in a meaningful way in your research, there's simply no excuse for not understanding and being able to apply modern deep learning methods (based on an architecture called the transformer).

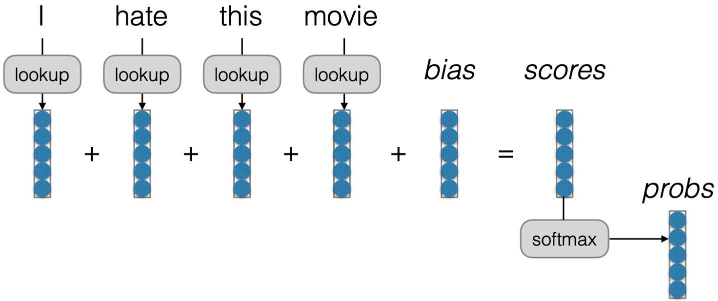
What was acceptable for a paper five years ago should not necessarily be used today.

Outline

Bag of Words

- But first, let's see a traditional approach to an important downstream task
- **Bag of Words**: take all words in a sentence and make predictions, regardless of syntax
- For example, in doing sentiment analysis, may have a score for probability each word in a sentence is associated with each sentiment to be classified. BoW adds up those probabilities.

Bag of Words



Carnegie Mellon CS 11-747

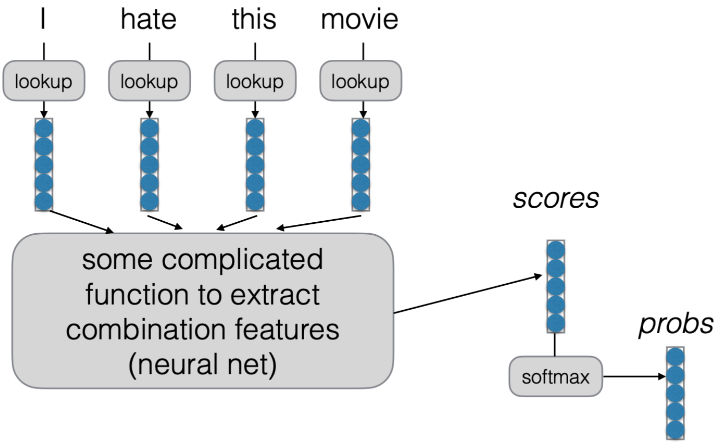
Problems

Consider the following:

- I hate this class
- I don't love this class
- I don't hate this class
- There's nothing I don't love about this class

Can't capture this with bag of words. Need combination features.

Bag of Words v. Neural Networks



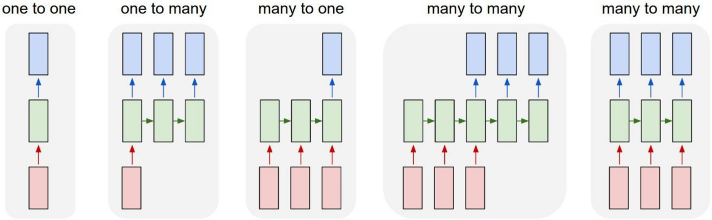
Carnegie Mellon CS 11-747

How do we approximate complicated functions?

Outline

Cases We've Covered So Far

RNN: a deep learning model that allows outputs or inputs to be variable length sequences.



Stanford CS231n Lecture Notes

RNNs

A RNN is fed an input vector at every time step and also maintains an internal state. It can modify that state as a function of what it receives at each time step.

$$h_t = f_w(h_{t-1}, x_t)$$

- h_t is the state (history)
- x_t is the input vector at each time step
- f is a recurrence function and W are its parameters - the same function is applied at every t. This is what allows us to use an RNN with **arbitrary length sequences**, regardless of sequence length, we use the same function at every step.

Produce an output y_t based on the RNN state

Plain Vanilla RNN

$$h_t = tanh(W_{hh}h_{t-1} + W_{xh}x_t) \tag{5}$$

$$y_t = W_{hy}h_t \tag{6}$$

Project, add, and squish the sum of the previous history and input vector. Then use another matrix projection on that hidden state to produce the output vector.

Outline

RNNs in Practice

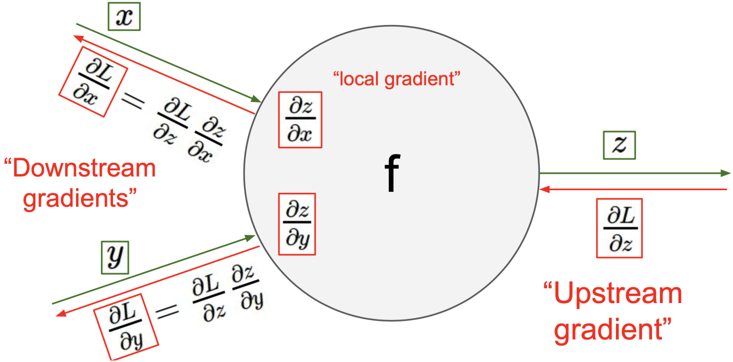
This RNN formulation is not used in practice:

$$h_t = tanh\left(W\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

To understand why, we need to return to an earlier theme of the course: how gradients flow through networks

How Gradients Flow in Networks

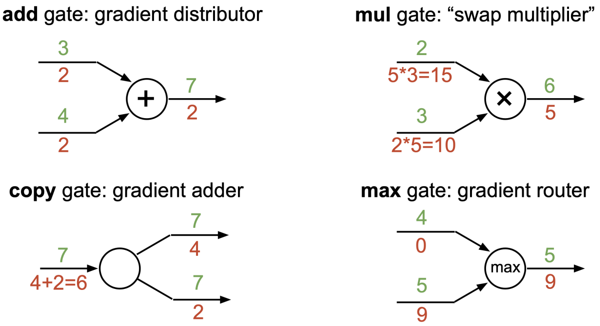
To backprop, we chain the local gradient with the upstream gradient



Stanford CS231n Lecture Notes

Computational Graphs

Green numbers denote the outputs from the forward pass, red numbers are the gradients computed from backprop.



Stanford CS231n Lecture Notes

$$f(x) = x + y \rightarrow \frac{\partial f}{\partial x} = 1, \frac{\partial f}{\partial y} = 1$$

$$f(x) = ax \rightarrow \frac{\partial f}{\partial x} = a, \text{ etc.}$$

Gradient Flow in RNN

- Gradient flow complicates learning long-run dependencies (Bengio et al., 1994)
- Recall, what allows us to model arbitrary sequence lengths is using the same weight matrix at every step
- When we backprop to the earlier layers, the chain rule will entail matrix multiplication by W over and over again, since the network structure multiples by W at every time step in the forward pass
- If the largest singular value of W is > 1 the gradient will explode, if < 1 will vanish

Challenges of Learning Long-Term Dependencies

- We can avoid exploding gradients by clipping
- Unfortunately, vanishing gradients prevent us from training the network and necessitate a different architecture

RNNs in Practice

To solve this problem, we use LSTM (Hochreiter et al., 1997):

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

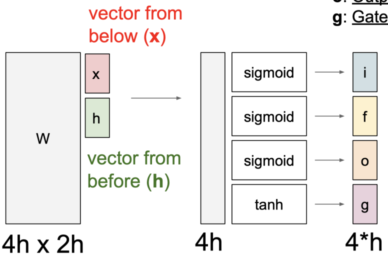
$$h_t = o \odot tanh(c_t)$$

$\odot$ is component wise multiplication for matrices

LSTMs

[Hochreiter et al., 1997]

i: Input gate, whether to write to cell
f: Forget gate, Whether to erase cell
o: Output gate, How much to reveal cell
g: Gate gate (?), How much to write to cell



$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

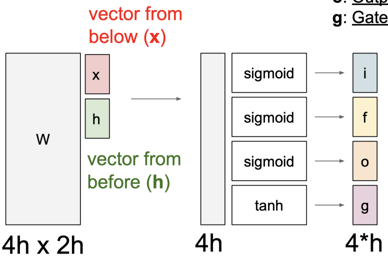
Stanford CS231n Lecture Notes

Think of these gates as being 0/1 (even though we squish them into sigmoids because we need differentiability)

LSTMs

[Hochreiter et al., 1997]

i: Input gate, whether to write to cell
f: Forget gate, Whether to erase cell
o: Output gate, How much to reveal cell
g: Gate gate (?), How much to write to cell



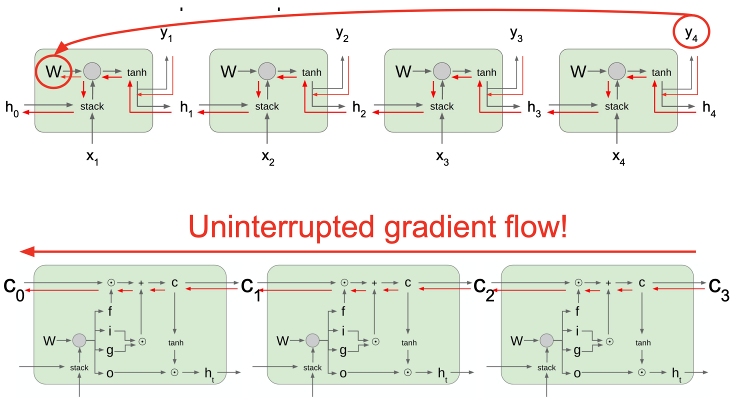
$$\boxed{\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

Update the cell state by deciding whether to remember (forget gate), and deciding whether and how much (i and g) to write to a cell. Only allow part of cell value to enter the history (which determines output), in a learnable way (o).

Uninterrupted Gradient Flow

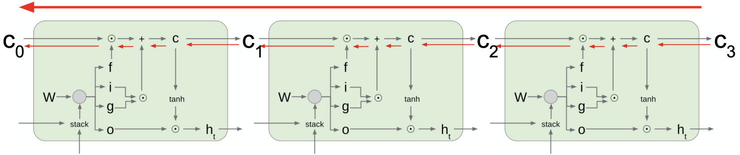


Stanford CS231n Lecture Notes

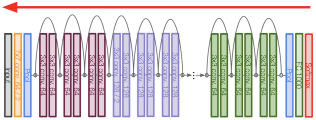
Provides a path for uninterrupted gradient flow because of the add gate (assuming you don't forget everything - avoid this by initializing the forget gate near 1).

Analogous to something we've seen before

Uninterrupted gradient flow!



Similar to ResNet!



In between:
Highway Networks

$$g = T(x, W_T)$$
$$y = g \odot H(x, W_H) + (1 - g) \odot x$$

Srivastava et al, "Highway Networks", ICML DL Workshop 2015

Stanford CS231n Lecture Notes

Skip connections in ResNet also act as add gates, allowing for unimpeded gradient flow.

Outline

Dependency Parsing

Dependency parsing maps the dependency relationships between words in a sentence.

Dependency Parsing

- This was one of the dominant topics in computational linguistics, pre-deep learning, and the first dependency parser was created by Hays in 1962
- I don't want to get into the details of how dependency parsing works, as it takes us too far afield into technical material
- However, one can imagine this being quite useful for some typical social science tasks, so I want you to understand broadly what this is and the packages you can use if you need to parse dependencies or do related tasks (part-of-speech tagging, named entity recognition)

Uses of Dependency Parsing

- It is fairly common for data that could have been organized as a database to be written as text
- Take the example of historical biographical compendia. They may be written in a fairly formulaic way: "Person A studied in Department in University in Place during Years under the tutelage of Person B." You'd like to turn this into a database, which requires extracting the named entities and understanding their dependencies (i.e. Person A studied under Person B).
- I suspect it is not uncommon for useful information to be presented in this format.

Dependency Parsing is hard

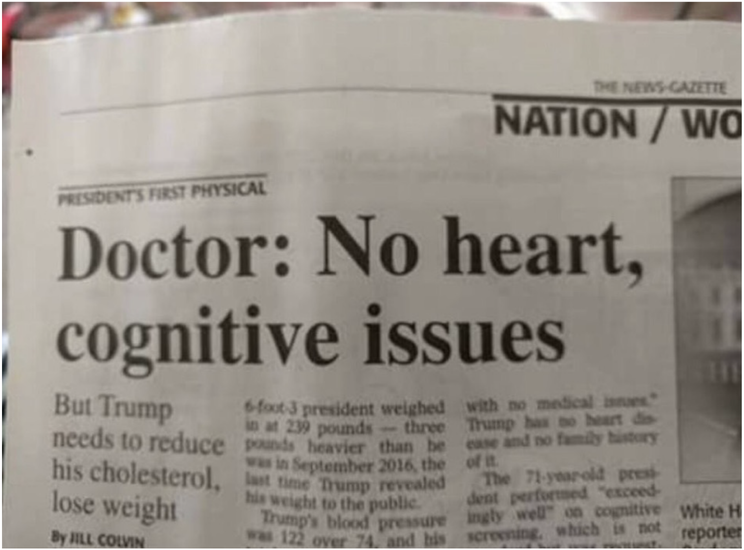
- While we don't stop to think about it, there are many, many ambiguities in human language.
- I will pull just a few examples from a lecture by a leading scholar on dependency parsing (Christopher Manning of Stanford). They are in English, but this is a much more broadly true fact across languages.

Prepositional Phrase Attachment Ambiguity

Stanford Linguistics 284

These ambiguities can multiply with the length of the sentence, with some longer sentences having 5 or more ambiguous PP in succession

Coordination Scope Ambiguity


THE NEWS-GAZETTE

NATION / WO

PRESIDENT'S FIRST PHYSICAL

Doctor: No heart, cognitive issues

But Trump needs to reduce his cholesterol, lose weight

By JILL COLVIN

6-foot-3 president weighed in at 239 pounds — three pounds heavier than he was in September 2016, the last time Trump revealed his weight to the public.

Trump's blood pressure was 122 over 74, and his

with no medical issues." Trump has no heart disease and no family history of it.

The 71-year-old president performed "exceedingly well" on cognitive screening, which is not

Stanford Linguistics 284

Verb Phrase Attachment Ambiguity

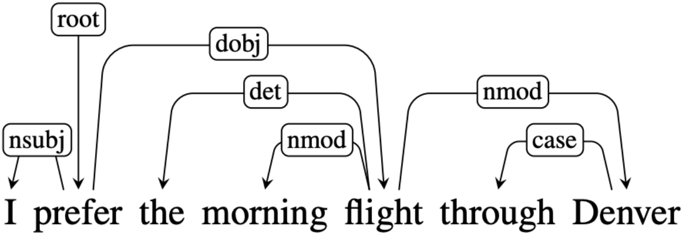

theguardian
home › world › americas asia all
Rio de Janeiro
Mutilated body washes up on Rio beach to be used for Olympics beach volleyball

Stanford Linguistics 284

Ambiguity

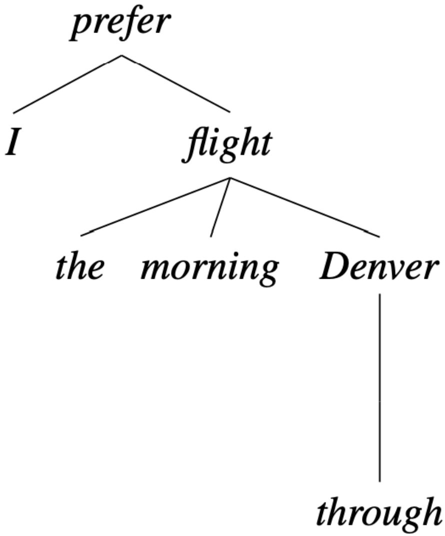
- Human languages are very different from programming languages, where such ambiguity is not allowed
- This is a feature, not a bug, as it allows human languages to be very efficient
- It does make it difficult though to teach a computer to resolve these ambiguities

How Dependencies are Denoted



Jurfasky and Martin, 2020

How Dependencies are Denoted



Jurfasky and Martin, 2020

Parts of Speech

Clausal Argument Relations	Description
NSUBJ	Nominal subject
DOBJ	Direct object
IOBJ	Indirect object
CCOMP	Clausal complement
XCOMP	Open clausal complement
Nominal Modifier Relations	**Description**
NMOD	Nominal modifier
AMOD	Adjectival modifier
NUMMOD	Numeric modifier
APPOS	Appositional modifier
DET	Determiner
CASE	Prepositions, postpositions and other case markers
Other Notable Relations	**Description**
CONJ	Conjunct
CC	Coordinating conjunction

Jurfasky and Martin, 2020

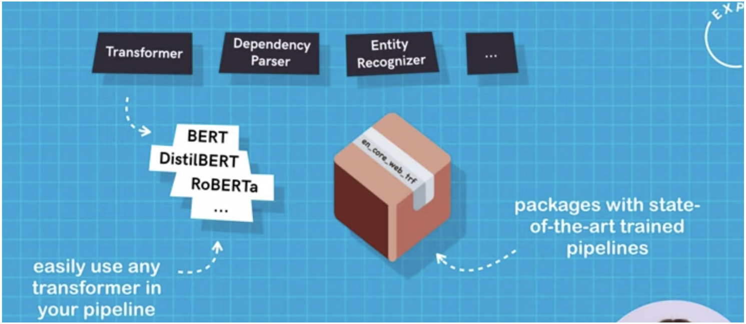
Dependency Data

- Labeled data for dependency parsing are called tree banks
- Universal Dependencies is the best known one, that has labeled data for over a hundred languages (`https://universaldependencies.org/`). It is constantly being updated
- Going back to our biography example, if you find inadequate results with pre-trained models, you might get a lot more mileage by doing some of your own labeling

spaCy

- Fortunately, there is a very well-developed, easy-to-use package for dependency parsing, called spaCy (`spacy.io`)
- You'll see an option to use a fast or accurate version. The accurate version uses a Transformer architecture to embed the input, and they say it leads on average to 30% more accurate results (a pretty big deal for our applications)

spaCy



Transformer
Dependency Parser
Entity Recognizer
...
BERT
DistilBERT
RoBERTa
...
en_core_web_trf
packages with state-of-the-art trained pipelines
easily use any transformer in your pipeline

spaCy features

- Dozens of languages
- Pre-trained pipelines for common tasks in 17 languages
- Multi-task learning with pre-trained transformers
- Components for dependency parsing as well as named entity recognition, part-of-speech tagging, entity linking, text classification, and other common tasks
- Support for custom models in PyTorch
- Visualizers for syntax and named entity recognition...

Outline

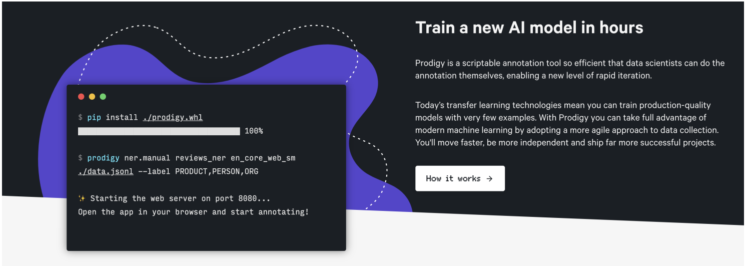
Named Entity Recognition

TYPE	DESCRIPTION
PERSON	People, including fictional.
NORP	Nationalities or religious or political groups.
FAC	Buildings, airports, highways, bridges, etc.
ORG	Companies, agencies, institutions, etc.
GPE	Countries, cities, states.
LOC	Non-GPE locations, mountain ranges, bodies of water.
PRODUCT	Objects, vehicles, foods, etc. (Not services.)
EVENT	Named hurricanes, battles, wars, sports events, etc.
WORK_OF_ART	Titles of books, songs, etc.
LAW	Named documents made into laws.
LANGUAGE	Any named language.
DATE	Absolute or relative dates or periods.
TIME	Times smaller than a day.
PERCENT	Percentage, including "%".
MONEY	Monetary values, including unit.
QUANTITY	Measurements, as of weight or distance.
ORDINAL	"first", "second", etc.
CARDINAL	Numerals that do not fall under another type.

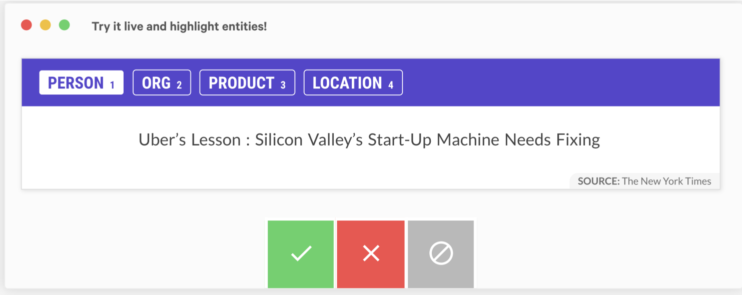
Outline

Prodigy

The makers of spaCy also have a labeling tool called Prodigy, designed to make it easy to label for NLP



Labeling for NER



Labeling for Topics