Natural Language Processing

(Computational Linguistics)

Sorcha Gilroy

February 23, 2018

University of Edinburgh

Where to find stuff

Papers: mostly in conferences e.g. ACL, NAACL, EACL, EMNLP, COLING.

Blogs: Vered Schwartz, Hal Daume III, Chris Olah, Andrej Karpathy.

Book: Speech and Language Processing, Jurafsky and Martin (pdf online)

Overview

Inputs

Applications

Language Modelling with RNNs

Difficulties

Inputs

One-hot vectors

Used all across NLP. Take a vocabulary

$$V = [\text{the}, \text{cat}, \text{sat}, \text{on}, \text{mat}]$$

the cat sat on the mat $= [1, 0, 0, 0, 0]$, $[0, 1, 0, 0, 0]$, $[0, 0, 1, 0, 0]$, $[0, 0, 0, 1, 0]$, $[1, 0, 0, 0, 0]$, $[0, 0, 0, 0, 1]$

Word Embeddings

Problem with one-hot: cat and dog encoded completely independently

Word embeddings map words to vectors that reflect the contexts (surrounding words) they appear in.

Distributional hypothesis: “a word is characterized by the company it keeps" - John Firth

Word Embeddings

"The cat sat on the mat"

Map each word to a 1-hot vector: $[1, 0, 0, 0, 0], [0, 1, 0, 0, 0], \ldots$.

Train on contexts:
(input,output) = (word$_i$, [word$_{i-1}$, word$_{i+1}$]) or
(input,output) = ([word$_{i-1}$, word$_{i+1}$], word$_i$)

Want to learn a matrix W such that $\sigma(W \cdot v)$ predicts the output from the input.

Word Embeddings

Common examples, e.g. word2vec, glove.

Train for *semantic similarity*. End up with king - man = queen - woman.

Can also get embeddings for tasks.

Have a neural network for a task, input sentence as sequence of 1-hot vectors, first layer of network = embedding.

Packages

NLTK (python): huge amount of functions/documentation.

Stanford CoreNLP (python): more production/industry level than NLTK.

Tensorflow/Pytorch (python): focused on neural networks, useful tutorials.

Applications

Named Entity Recognition

Input = “John Smith of Apple went to Sydney”

Output = PER PER _ ORG _ _ LOC

Uses: classifying content, news article recommendations etc.

Sentiment Analysis

(1) "I love ice cream" = positive
(2) "I hate ice cream" = negative
(3) "I don't hate ice cream" = neutral

Mostly not great, will probably classify (3) as negative.

Lots of packages to do this - just keep an eye out for common mistakes.

Uses: lots of use on twitter - companies see how they're viewed.

Summarisation

Extractive: highlights specific words in a text.

Classify each word as being in or out of the the summary.

Abstractive: creates new sentences, possibly with words not in the original.

Read in the whole document. Generate a new shorter document (hard).

Uses: generating short summaries, generating headlines.

Machine Translation

Lots of different methods - phrase based machine translation vs neural machine translation.

Neural models encode entire source sentence as one vector - then predict an output sentence in the target language. Called an encoder-decoder.

Takes lots of examples of pairs of sentences that are translations of each other (e.g. European/Canadian Parliament proceedings).

Language Modelling with RNNs

Language Modelling

Language model = probability distribution over sequences of words.

Want "I eat my dinner" to have high probability and "he eat my dinner" to have low probability.

Lots of different ways of doing this. Traditionally, count contexts of words over a document/set of documents (corpus). Recently - recurrent neural networks (RNNs)

Language Modelling with RNNs

Basic RNN. Input is sequence of 1-hot vectors $x_1, \ldots, x_n$. Hidden state $h_1, \ldots, h_n$. Output $y_1, \ldots, y_n$.

$$h_t = \sigma(W \cdot h_{t-1} + U \cdot x_t)$$

$$y_t = \text{softmax}(E \cdot h_t)$$

U, W, E are matrices, σ is the activation function e.g. ReLU, sigmoid. Softmax turns y into a distribution.

Language Modelling with RNNs

For each i, y_i is a vector over the vocab predicting the next word.

For vocab of size 4, if $y_i = [0.2, 0.1, 0.1, 0.6]$ then the model says that word 4 is most likely to come next.

Trained by comparing predicted next word with actual word at each time step.

Language Modelling with RNNs

Can use this to score a sequence of words. Given a sentence "I like pie" change to "BOS I like pie EOS".

$$V = [\text{i}, \text{like}, \text{pie}, \text{BOS}, \text{EOS}]$$

$$\text{score}(\text{I like pie}) = y_1[0] \cdot y_2[1] \cdot y_3[2] \cdot y_4[4]$$

Language Modelling with RNNs

That model took words as input other options:

characters

tri-grams

morphological components (e.g. cats $\rightarrow$ cat s)

...

Difficulties

Types of Data

Most NLP models are trained on formal writing - e.g. Wall Street Journal, European Parliament etc.

Doesn't work so well at new domains.

If you need to do NLP on informal writing, look for something that works on social media- lots of stuff for twitter.

Rare and Unknown Words

What happens when we see a word that we didn't encounter in our training data?

Most people reserve an unknown word token in the vocabulary - often written UNK.

People also often map all words below certain frequency rank to UNK in training.

Can also use stemming/lemmatisation.

Stemming and Lemmatisation

Word = flies

Stem = fli/flie

Lemma = fly

NLTK packages to do both.

Miscellaneous

Things that generate sentences: e.g. translation, abstractive summarisation- really hard to evaluate. Uses BLEU which measures word overlap - can't handle synonyms.

Languages: almost everything geared towards English. Usually means the same models work badly when trained on other languages. E.g. morphologically rich languages.

Takeaways

Check NLTK etc to see if it does what you want.

Think about the type of data you have - formal/informal, english/other language.

Do some qualitative analysis of output - language is messy, see where it completely fails.