Ghi chú của một coder

Vũ Anh

Tháng 01 năm 2018

Mục lục

Phần I

Khoa học dữ liệu

Chương 1

Học sâu

24/11/2017 - Từ hôm nay, mỗi ngày sẽ ghi chú một phần (rất rất nhỏ) về Deep Learning

22/11/2017 - Phải nói quyển này hơi nặng so với mình. Nhưng thôi cứ cố gắng vậy.

Tài liệu cũ: http://magizbox.com/training/deep_learning/site/
Deep Learning is a new area of Machine Learning research, which has been introduced with the objective of moving Machine Learning closer to one of its original goals: Artificial Intelligence.

1.1 Deep Feedforward Networks

deep feedforward networks

Deep feedforward networks, (hay còn gọi là feedforward neural networks) hoặc multilayer perceptrons (MLPs) là mô hình deep learning cơ bản nhất. Mục tiêu của một feedforward network là xấp xỉ một hàm $f*$. Ví dụ, với bài toán phân loại, $y = f*(x)$ chuyển đầu vào x thành một nhãn y. Một feedforward network định nghĩa một ánh xạ $y = f(x, \theta)$ và học giá trị của θ để xấp xỉ hàm $f*$ tốt nhất.

Mô hình được gọi là feedforward vì giá trị của x đang lan truyền qua mạng đến output, mà không có chiều ngược lại. Các mô hình neural có sự lan truyền ngược lại được gọi là recurrent neural network.

Feedforward là nền tảng cho rất nhiều mô hình mạng neural hiện đại. Ví dụ, các ứng dụng trong việc nhận diện ảnh thường áp dụng mô hình convolutional, là một dạng đặc biệt của feedforward network. Feedforward network cũng là nền tảng cho mạng recurrent network, có nhiều ứng trọng xử lý ngôn ngữ.

Feedforward neural networks được gọi là network vì nó thường được biểu diễn bởi một đồ thị gồm nhiều tầng, định nghĩa các hàm được kết hợp với nhau như thế nào. Ví dụ, chúng ta có 3 hàm f(1)(x), f(2)(x), f(3)(x) được đặt với nhau thành một chuỗi, hình thành hàm f(x) = f(3)(f(2)(f(1)(x))). Cấu trúc chuỗi này là dạng phổ biến nhất trong mạng neural. Trong trường hợp này f(1)(x) được gọi là layer 1, f(2)(x) được gọi là layer 2, và cứ tiếp tục như vậy. Độ dài của chuỗi thể hiện độ sâu của mô hình. Thuật ngữ "deep learning" xuất phát từ điều này. Layer cuối cũng được goi là output layer. Trong mạng neural, mục tiêu là học hàm f*(x). Ứng với mỗi giá trị đầu vào x, sẽ có tương ứng một giá trị y. Mục tiêu của mạng là tìm các tham số để y xấp xỉ với f*(x). Nhưng việc

học của hàm không hoàn toàn phụ thuộc vào x, y, mà do learning algorithm quyết định. Vì dữ liệu huấn luyện không chỉ rõ giá trị ở những layer ở giữa, nên chúng được gọi là các hidden layers Cuối cùng, mạng neural được gọi là neural vì nó lấy cảm hứng từ ngành khoa học thần kinh. Mỗi hidden layer trong mạng thường là các giá trị vector. Mỗi phần tử trong vector sẽ quyết định việc hành xử thế nào với các input đầu vào và đưa ra output.
Ta có thể coi mỗi layer chứa rất nhiều units hoạt động song song, đóng vai trò như một hàm ánh xạ vector sang giá trị số. Mặc dù các kiến trúc của mạng neural lấy rất nhiều ý tưởng từ ngành khoa học thần kinh, tuy nhiên, mục tiêu của mạng neural không phải để mô phỏng bộ não. Một cách nhìn tốt hơn là xem mạng neural như một máy học, được thiết kế với các điểm nhấn từ những gi chúng ta biết về não người. Một cách để hiểu mạng neural là bắt đầu với các mô hình linear, và xem xét các hạn chế của các mô hình này. Linear models, như logistic regression hay linear regression, hấp dẫn ở chỗ chúng có thể fit rất hữu quả và đang tin cậy, với các close form và tối ưu hóa lồi. Một hàn chế hiển nhiên của linear là các hàm linear, nên model không thể hiểu tương tác giữa các input. Để mở rộng lienar model để biểu diễn hàm nonlinear của x, chúng ta có thể áp dụng linear model không phải cho x mà cho (x), trong đó là một nonlinear transformer. Có thể xem như một tập các feature mô tả x, hoặc một cách để biểu diễn x. Vấn đề là chọn mapping . Có 3 cách thông dụng, chọn một generic như RBF, thiết kế bằng tay, hoặc học .
Nội dung chương này:

- 1. Ví dụ cơ bản củ feedforward network
- 2. Design decisions cho việc thiết kế mạng feedforward network
- 3. Choose activation functions trong hidden layer
- 4. Thiết kế mạng neural, bao nhiêu layers, các layer kết nối với nhau như thế nào, có bao nhiêu unit trong một layer
  - chosing the optimizer
  - cost function
  - form of the output units
- 5. Thuật toán Back-propagation
- 6. Góc nhìn lịch sử

Tham khảo Chapter 6, Deep Learning Book Neural Network Zoo http://www.asimovinstitute.org/neural-network-zoo/
word2vec Example Step 1: Download word2vec example from github
dir
02/06/2017 11:45 DIR . 02/06/2017 11:45 DIR .. 02/06/2017 10:12 9,144 word2vec$_b asic.py Step 2 : Run word2vec$_b asic example$
$activate tensorflow -- gpu$ python word2vec$_b asic.py$
Found and verified text8.zip Data size 17005207 Most common words (+UNK) [['UNK', 418391], ('the', 1061396), ('of', 593677), ('and', 416629), ('one', 411764)] Sample data [5241, 3082, 12, 6, 195, 2, 3136, 46, 59, 156] ['anarchism', 'originated', 'as', 'a', 'term', 'of', 'abuse', 'first', 'used', 'against'] 3082 originated -> 5241 anarchism 3082 originated -> 12 as 12 as -> 6 a 12 as -> 3082 originated

6 a -> 195 term 6 a -> 12 as 195 term -> 2 of 195 term -> 6 a Initialized Average loss at step 0 : 288.173675537 Nearest to its: nasl, tinkering, derivational, yachts, emigrated, fatalism, kingston, kochi, Nearest to into: streetcars, neglecting, deutschlands, lecture, realignment, bligh, donau, medalists, Nearest to state: canterbury, exceptions, disaffection, crete, westernmost, earthly, organize, richland,

1.2 Tài liệu Deep Learning

Lang thang thế nào lại thấy trang này My Reading List for Deep Learning! của một anh ở Microsoft. Trong đó, (đương nhiên) có Deep Learning của thánh Yoshua Bengio, có một vụ hay nữa là bài review "Deep Learning" của mấy thánh Yann Lecun, Yoshua Bengio, Geoffrey Hinton trên tạp chí Nature. Ngoài ra còn có nhiều tài liệu hữu ích khác.

1.3 Các layer trong deep learning

1.3.1 Sparse Layers

nn.Embedding (hướng dẫn)
grep code: Shawn1993/cnn-text-classification-pytorch
Đóng vai trò như một lookup table, map một word với dense vector tương ứng

1.3.2 Convolution Layers

nn.Conv1d, nn.Conv2d, nn.Conv3d)
grep code: Shawn1993/cnn-text-classification-pytorch, galsang/CNN-sentence-classification-pytorch
Các tham số trong Convolution Layer
* *kernel_size* (hay là filter size)
Đối với NLP, $kernel\_size$ thường bằng $region\_size * word\_dim$ (đối với conv1d) hay $(region\_size, word\_dim)$ đối với conv2d
<small>Quá trình tạo feature map đối với region size bằng 2</small> ![](https://media.giphy.com/media/l2Q
* ‘in$_c$hannels‘,‘out$_c$hannels‘($lslng‘featuremaps‘$)
Kênh (channels) là các cách nhìn (view) khác nhau đối với dữ liệu. Ví dụ, trong ảnh thường có 3 kênh RGB (red, green, blue), có thể áp dụng convolution giữa các kênh. Với văn bản cũng có thể có các kênh khác nhau, như khi có các kênh sử dụng các word embedding khác nhau (word2vec, GloVe), hoặc cùng một câu nhưng biểu diễn ở các ngôn ngữ khác nhau.
* ‘stride‘
Định nghĩa bước nhảy của filter.

Hình minh họa sự khác biệt giữa các feature map đối với stride=1 và stride=2. Feature map đối với stride = 1 có kích thước là 5, feature map đối với stride = 3 có kích thước là 3. Stride càng lớn thì kích thước của feature map càng nhỏ.
Trong bài báo của Kim 2014, ‘stride = 1‘ đối với ‘nn.conv2d‘ và ‘stride = word$_d$im‘Øivi‘$nn.conv1d$‘
Toàn bộ tham số của mạng CNN trong bài báo Kim 2014,

| Description | Values | |———————|———————| | input word vectors | Google word2vec | | filter region size | (3, 4, 5) | | feature maps | 100 | | activation function | ReLU | | pooling | 1-max pooling | | dropout rate | 0.5 | | *latexlamp*; $s =$ 22 norm constraint | 3 |
Đọc thêm:
* [Lecture 13: Convolutional Neural Networks (for NLP). CS224n-2017](http://web.stanford.edu/class/cs224n 2017-lecture13-CNNs.pdf) * [DeepNLP-models-Pytorch - 8. Convolutional Neural Networks](https://nbviewer.jupyter.org/github/DSKSD/DeepNLP-models-Pytorch/blob/master/notebooks/08.CNN-for-Text-Classification.ipynb) * [A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. Zhang 2015](https://arxiv.org/pdf/1510.03820.pdf)

1.4 Recurrent Neural Networks

1.4.1 What are RNNs?

The idea behind RNNs is to make use of sequential information. In a traditional neural network we assume that all inputs (and outputs) are independent of each other. But for many tasks that's a very bad idea. If you want to predict the next word in a sentence you better know which words came before it. RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being depended on the previous computations. Another way to think about RNNs is that they have a "memory" which captures information about what has been calculated so far. In theory RNNs can make use of information in arbitrarily long sequences, but in practice they are limited to looking back only a few steps (more on this later). Here is what a typical RNN looks like:
A recurrent neural network and the unfolding in time of the computation involved in its forward computation
A recurrent neural network and the unfolding in time of the computation involved in its forward computation. Source: Nature The above diagram shows a RNN being unrolled (or unfolded) into a full network. By unrolling we simply mean that we write out the network for the complete sequence. For example, if the sequence we care about is a sentence of 5 words, the network would be unrolled into a 5-layer neural network, one layer for each word. The formulas that govern the computation happening in a RNN are as follows:
xtxt is the input at time step tt. For example, x1x1 could be a one-hot vector corresponding to the second word of a sentence. stst is the hidden state at time step tt. It's the "memory" of the network. stst is calculated based on the previous hidden state and the input at the current step: st=f(Uxt+Wst1)st=f(Uxt+Wst1). The function ff usually is a nonlinearity such as tanh or ReLU. s1s1, which is required to calculate the first hidden state, is typically initialized to all zeroes. otot is the output at step tt. For example, if we wanted to predict the next word in a sentence it would be a vector of probabilities across our vocabulary. ot=softmax(Vst)ot=softmax(Vst). There are a few things to note here:
You can think of the hidden state stst as the memory of the network. stst captures information about what happened in all the previous time steps. The

output at step otot is calculated solely based on the memory at time tt. As briefly mentioned above, it's a bit more complicated in practice because stst typically can't capture information from too many time steps ago. Unlike a traditional deep neural network, which uses different parameters at each layer, a RNN shares the same parameters (UU, VV, WW above) across all steps. This reflects the fact that we are performing the same task at each step, just with different inputs. This greatly reduces the total number of parameters we need to learn. The above diagram has outputs at each time step, but depending on the task this may not be necessary. For example, when predicting the sentiment of a sentence we may only care about the final output, not the sentiment after each word. Similarly, we may not need inputs at each time step. The main feature of an RNN is its hidden state, which captures some information about a sequence.

1.4.2 What can RNNs do?

RNNs have shown great success in many NLP tasks. At this point I should mention that the most commonly used type of RNNs are LSTMs, which are much better at capturing long-term dependencies than vanilla RNNs are. But don't worry, LSTMs are essentially the same thing as the RNN we will develop in this tutorial, they just have a different way of computing the hidden state. We'll cover LSTMs in more detail in a later post. Here are some example applications of RNNs in NLP (by non means an exhaustive list).
Language Modeling and Generating Text Given a sequence of words we want to predict the probability of each word given the previous words. Language Models allow us to measure how likely a sentence is, which is an important input for Machine Translation (since high-probability sentences are typically correct). A side-effect of being able to predict the next word is that we get a generative model, which allows us to generate new text by sampling from the output probabilities. And depending on what our training data is we can generate all kinds of stuff. In Language Modeling our input is typically a sequence of words (encoded as one-hot vectors for example), and our output is the sequence of predicted words. When training the network we set ot=xt+1ot=xt+1 since we want the output at step tt to be the actual next word.
Research papers about Language Modeling and Generating Text:
Recurrent neural network based language model Extensions of Recurrent neural network based language model Generating Text with Recurrent Neural Networks
Machine Translation Machine Translation is similar to language modeling in that our input is a sequence of words in our source language (e.g. German). We want to output a sequence of words in our target language (e.g. English). A key difference is that our output only starts after we have seen the complete input, because the first word of our translated sentences may require information captured from the complete input sequence.
RNN for Machine Translation
RNN for Machine Translation. Image Source: http://cs224d.stanford.edu/lectures/CS224d-Lecture8.pdf
Research papers about Machine Translation:
A Recursive Recurrent Neural Network for Statistical Machine Translation Sequence to Sequence Learning with Neural Networks Joint Language and Translation Modeling with Recurrent Neural Networks Speech Recognition Given an

input sequence of acoustic signals from a sound wave, we can predict a sequence of phonetic segments together with their probabilities.
Research papers about Speech Recognition:
Towards End-to-End Speech Recognition with Recurrent Neural Networks Generating Image Descriptions Together with convolutional Neural Networks, RNNs have been used as part of a model to generate descriptions for unlabeled images. It's quite amazing how well this seems to work. The combined model even aligns the generated words with features found in the images.
Deep Visual-Semantic Alignments for Generating Image Descriptions. Source: http://cs.stanford.edu/people/karpathy/deepimagesent/

1.4.3 Training RNNs

Training a RNN is similar to training a traditional Neural Network. We also use the backpropagation algorithm, but with a little twist. Because the parameters are shared by all time steps in the network, the gradient at each output depends not only on the calculations of the current time step, but also the previous time steps. For example, in order to calculate the gradient at t=4t=4 we would need to backpropagate 3 steps and sum up the gradients. This is called Backpropagation Through Time (BPTT). If this doesn't make a whole lot of sense yet, don't worry, we'll have a whole post on the gory details. For now, just be aware of the fact that vanilla RNNs trained with BPTT have difficulties learning long-term dependencies (e.g. dependencies between steps that are far apart) due to what is called the vanishing/exploding gradient problem. There exists some machinery to deal with these problems, and certain types of RNNs (like LSTMs) were specifically designed to get around them.
RNN Extensions Over the years researchers have developed more sophisticated types of RNNs to deal with some of the shortcomings of the vanilla RNN model. We will cover them in more detail in a later post, but I want this section to serve as a brief overview so that you are familiar with the taxonomy of models.
Bidirectional RNNs are based on the idea that the output at time t may not only depend on the previous elements in the sequence, but also future elements. For example, to predict a missing word in a sequence you want to look at both the left and the right context. Bidirectional RNNs are quite simple. They are just two RNNs stacked on top of each other. The output is then computed based on the hidden state of both RNNs.
Deep (Bidirectional) RNNs are similar to Bidirectional RNNs, only that we now have multiple layers per time step. In practice this gives us a higher learning capacity (but we also need a lot of training data).
Deep Bidirectional RNN LSTM networks are quite popular these days and we briefly talked about them above. LSTMs don't have a fundamentally different architecture from RNNs, but they use a different function to compute the hidden state. The memory in LSTMs are called cells and you can think of them as black boxes that take as input the previous state ht1ht1 and current input xtxt. Internally these cells decide what to keep in (and what to erase from) memory. They then combine the previous state, the current memory, and the input. It turns out that these types of units are very efficient at capturing long-term dependencies. LSTMs can be quite confusing in the beginning but if you're interested in learning more this post has an excellent explanation.

Conclusion So far so good. I hope you've gotten a basic understanding of what RNNs are and what they can do. In the next post we'll implement a first version of our language model RNN using Python and Theano. Please leave questions in the comments!
[1] : [*UnderstandingConvolutionalNeuralNetworksforNLP*](*http* : *//www.wildml.com/*2015/11/*understa*
convolutional−neural−networks−for−nlp)[2] : [*http* : *//pytorch.org/docs/master/nn.html*](*http* :
//pytorch.org/docs/master/nn.html)

Chương 2

Xử lý ngôn ngữ tự nhiên

Bản lưu cũ http://magizbox.com/training/natural_language_processing/site/
Natural language processing (NLP) is a field of computer science, artificial intelligence, and computational linguistics concerned with the interactions between computers and human (natural) languages.

2.1 Introduction to Natural Language Processing

Natural language processing (NLP) is a field of computer science, artificial intelligence, and computational linguistics concerned with the interactions between computers and human (natural) languages.
NLP is related to the area of human–computer interaction. Many challenges in NLP involve: natural language understanding, enabling computers to derive meaning from human or natural language input; and others involve natural language generation.
The input and output of an NLP system can be either speech or written text.
Components of NLP
There are two components of NLP as given
Natural Language Understanding (NLU): this task mapping the given input in natural language into useful representations and analyzing different aspects of the language. Natural Language Generation (NLG): In the process of producing meaningful phrases and sentences in the form of natural language form some internal representation. It involves text planning retrieve the relevant content from knowledge base, sentence planning choose required words, forming meaningful phrases, setting tone of the sentence, text realization map sentence plan into sentence structure. Difficulties
Natural Language has an extremely rich form and structure. It is very ambiguous. There can be different levels of ambiguity
Lexical ambiguity: it is at very primitive level such as word-level. For example, treating the word “board” as noun or verb? Syntax level ambiguity: A sentence be parsed in different ways. For example, “He lifted the beetle with the red cap?” - did he use cap to lift the beetle or he lifted a beetle that had red cap? Referential ambiguity: referring to something using pronouns. For example, Rima went to

Gauri. She said "I am tired". - Exactly who is tired? One input can mean different meanings. Many inputs can mean the same thing.

2.2 Natural Language Processing Tasks

The analysis of natural language is broken into various board levels such as phonological, morphological, syntactic, semantic, pragmatic and discourse analysis.
Phonological Analysis Phonology is analysis of spoken language. Therefore, it deals with speech recognition and generation. The core task of speech recognition and generation system is to take an acoustic waveform as input and produce as output, a string of words. The phonology is a part of natural language analysis, which deals with it. The area of computational linguistics that deals with speech analysis is computational phonology
Example: Hans Rosling's shortest TED talk
Original Sound
0:00 / 0:52
Text X means unknown but the world is pretty known it's seven billion people have seven stones. One billion can save money to fly abroad on holiday every year. One billion can save money to keep a car or buy a car. And then three billion they save money to pay the by be a bicycle or perhaps a two-wheeler. And two billion they are busy saving money to buy shoes. In the future they will get rich and these people we move over here, these people will move over here, we will have two billion more in the world like this and the question is whether the rich people over there are prepared to be integrated in the world with 10 bilions people. Auto generated sound
0:00 / 0:36
Morphological Analysis
It is the most elementary phase of NLP. It deals with the word formation. In this phase, individual words are analyzed according to their components called "morphemes". In addition, non-word taken such as punctuation, etc. are separated from words. Morpheme is basic grammatical building block that makes words.
The study of word structure is refereed to as morphology. In natural language processing, it is done in morphological analysis. The task of breaking a word into its morphemes is called morphological parsing. A morpheme is defined as minimal meaningful unit in a language, which cannot be further broken into smaller units.
Example: word fox consists a single morpheme, as it cannot be further resolved into smaller units. Whereas word cats consists two morphemes, the morpheme "cat" and morpheme "s" indicating plurality.
Here we defined the term meaningful. Though cat can be broken in "c" and "at", but these do not relate with word "cat" in any sense. Thus word "cat" will be dealt with as minimum meaningful unit.
Morphemes are traditionally divided into two types
(i) "free morphemes", that are able to act as words in isolation (e.g., "thing", "permanent", "local") (ii) "bound morphemes", that can operate only as part of other words (e.g., "is" 'ing' etc) The morpheme, which forms the center part of the world, is also called "stem". In English, a word can be made up of one or more

morphemes, e.g., word - thing -> stem "think" word - localize -> stem "local", suffix "ize" word - denationalize -> prefix "de", stem "nation", suffix "al", "ize" The computational tool to perform morphological parsing is finite state transducer. A transducer performs it by mapping between the two sets of symbols, and a finite state transducer does it with finite automaton. A transducer normally consists of four parts: recognizer, generator, translator, and relator. The output of the transducer becomes a set of morphemes.
Lexical Analysis In this phase of natural language analysis, validity of words according to lexicon is checked. Lexicon stands for dictionary. It is a collection of all possible valid words of language along with their meaning.
In NLP, the first stage of processing input text is to scan each word in sentence and compute (or look-up) all the relevant linguistic information about that word. The lexicon provides the necessary rules and data for carrying out the first stage analysis.
The details of words, like their type (noun, verb and adverb, and other details of nouns and verb, etc.) are checked.
Lexical analysis is dividing the whole chunk of text into paragraphs, sentences, and words.
Syntactic Analysis Syntax refers to the study of formal relationships between words of sentences. In this phase the validity of a sentence according to grammar rules is checked. To perform the syntactic analysis, the knowledge of grammar and parsing is required. Grammar is formal specification of rules allowable in the language, and parsing is a method of analyzing a sentence to determine its structure according to grammar. The most common grammar used for syntactic analysis for natural languages are context free grammar (CFG) also called phase structure grammar and definite clause grammar. These grammars are described in detail in a separate actions.
Syntactic analysis is done using parsing. Two basic parsing techniques are: top-down parsing and bottom-up parsing.
Semantic Analysis In linguistics, semantic analysis is the process of relating syntactic structures, from the levels of phrases, clauses, sentences and paragraphs to the level of the writing as a whole, to their language-independent meanings. It also involves removing features specific to particular linguistic and cultural contexts, to the extent that such a project is possible.
The elements of idiom and figurative speech, being cultural, are often also converted into relatively invariant meanings in semantic analysis. Semantics, although related to pragmatics, is distinct in that the former deals with word or sentence choice in any given context, while pragmatics considers the unique or particular meaning derived from context or tone. To reiterate in different terms, semantics is about universally coded meaning, and pragmatics the meaning encoded in words that is then interpreted by an audience
Discourse Analysis The meaning of any sentence depends upon the meaning of the sentence just before it. In addition, it also brings about the meaning of immediately succeeding sentence.
Topics of discourse analysis include:
The various levels or dimensions of discourse, such as sounds, gestures, syntrax, the lexicon, style, rhetoric, meanings, speech acts, moves, strategies, turns, and other aspects of interaction Genres of discourse (various types of discourse in politics, the media, education, science, business, etc.) The relations between text (discourse) and context The relations between discourse and power The relations

between discourse and interaction The relations between discourse and cognition and memory Pragmatic Analysis During this, what was said is re-interpreted on what it actually meant. It involves deriving those aspects of language which require real world knowledge.
Sentiment Analysis
MetaMind, @RichardSocher
Named Entity Recognition KDD 2015 Tutorial: Automatic Entity Recognition and Typing from Massive Text Corpora - A Phrase and Network Mining Approach
Relationship Extraction AlchemyAPI

2.3 Natural Language Processing Applications

Information Retrieval (IR) Information retrieval (IR) is the activity of obtaining information resources relevant to an information need from a collection of information resources. Searches can be based on metadata or on full-text (or other content-based) indexing.
Information Extraction (IE) Information extraction (IE) is the task of automatically extracting structured information from unstructured and/or semi-structured machine-readable documents. In most of the cases this activity concerns processing human language texts by means of natural language processing (NLP).
Machine Translation Machine translation, sometimes referred to by the abbreviation MT (not to be confused with computer-aided translation, machine-aided human translation (MAHT) or interactive translation) is a sub-field of computational linguistics that investigates the use of software to translate text or speech from one language to another.
Question Answering (QA) Question answering (QA) is a computer science discipline within the fields of information retrieval and natural language processing (NLP), which is concerned with building systems that automatically answer questions posed by humans in a natural language.

2.4 Spelling Correction

For instance, we may wish to retrieve documents containing the term carrot when the user types the query carot. Google reports (http://www.google.com/jobs/britney.html) that the following are all treated as misspellings of the query britney spears: britian spears, britney's spears, brandy spears and prittany spears
We look at two steps to solving this problem: the first based on edit distance and the second based on k-gram overlap. Before getting into the algorithmic details of these methods, we first review how search engines provide spell-correction as part of a user experience.
Implementing spelling correction There are two basic principles underlying most spelling correction algorithms.
Of various alternative correct spellings for a mis-spelled query, choose the nearest one. This demands that we have a notion of nearness or proximity between a pair of queries. When two correctly spelled queries are tied (or nearly tied), select the one that is more common. For instance, grunt and grant both seem

equally plausible as corrections for grnt. Then, the algorithm should choose the more common of grunt and grant as the correction. The simplest notion of more common is to consider the number of occurrences of the term in the collection; thus if grunt occurs more often than grant, it would be the chosen correction. A different notion of more common is employed in many search engines, especially on the web. The idea is to use the correction that is most common among queries typed in by other users. The idea here is that if grunt is typed as a query more often than grant, then it is more likely that the user who typed grnt intended to type the query grunt. Corpus Birkbeck spelling error corpus
References How to Write a Spelling Corrector. Peter Norvig. 2007 Statistical Natural Language Processing in Python. Peter Norvig. 2007 Spelling correction. Introduction to Information Retrieval. 2008

2.5 Word Vectors

Discrete Representation Use a taxonomy like WordNet that has hypernyms (is-a) relationships
from nltk.corpus import wordnet as wn panda = wn.synset("panda.n.01") hyper = lambda s: s.hypernyms() list(panda.closure(hyper)) [Synset(‘procyonid.n.01’), Synset(‘carnivore.n.01’), Synset(‘placental.n.01’), Synset(‘mammal.n.01’), Synset(‘vertebrate.n.01’), Synset(‘chordate.n.01’), Synset(‘animal.n.01’), Synset(‘organism.n.01’), Synset(‘living$_thing.n.01'), Synset(‘w$
Great as resource but missing nuances, e.g. synonyms: adept, expert, good, practiced, proficient, skillful? Missing new words (impossible to keep up to date): wicked, badass, nifty, crack, ace, wizard, genius, ninjia Subjective Requires human labor to create and adapt Hard to compute accurate word similarity Word2Vec Word2vec is a group of related models that are used to produce word embeddings. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. Word2vec takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located in close proximity to one another in the space.
Word2vec was created by a team of researchers led by Tomas Mikolov at Google. The algorithm has been subsequently analysed and explained by other researchers. Embedding vectors created using the Word2vec algorithm have many advantages compared to earlier algorithms like Latent Semantic Analysis.
Main Idea of Word2Vec
Instead of capturing cooccurrence counts directly,, Predict surrounding words of every word Both are quite similar, see “Glove: Global Vectors for Word Representation” by Pennington et at. (2014) and Levy and Goldberg (2014)... more later. Faster and can easily incorporate a new sentence/document or add a word to the vocabulary. Detail of Word2Vec
Predict surrounding words in a window of length m of every word. Objective function: Maximize the log probability of any context word given the current cenetr word: J()=1Tt=1Tmjm,j0log p(wt+j|wt) J()=1Tt=1Tmjm,j0log p(wt+j|wt) where represents all variables we optimize
Predict surrounding words in a window of length m of every word For p(wt+j|wt)p(wt+j|wt)
the simplest first formulation is p(o|c)=exp(uTovc)Ww=1exp(uTwvc) p(o|c)=exp(uoTvc)w=1Wexp(uwTvc)

where o is the outside (or output) word id, c is the center word id, u and v are "center" and "outside" vectors of o and c
Every word has two vectors! This is essentially "dynamic" logistic regression Linear Relationships in word2vec
These representations are very good at encoding dimensions of similarity!
Analogies testing dimensions of similarity can be solved quite well just by doing vector subtraction in the embedding space Syntactically
xapplexapplesxcarxcarsxfamilyxfamiliesxapplexapplesxcarxcarsxfamilyxfamilies
Similarly for verb and adjective morphological forms Semantically (Semeval 2012 task 2)
xshirtxclothingxchairxfurniturexshirtxclothingxchairxfurniture xkingxmanxqueenx-womanxkingxmanxqueenxwoman GloVe Project
Highlights Training Model Overview GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.
Pre-trained Model fastText
Pre-trained word vectors for 294 languages, trained on Wikipedia using fastText. These vectors in dimension 300 were obtained using the skip-gram model described in Bojanowski et al. (2016) with default parameters.
glove
Pre-trained word vectors. This data is made available under the Public Domain Dedication and License v1.0 whose full text can be found at: http://www.opendatacommons.org/licenses/pddl
Language: English
Wikipedia 2014 + Gigaword 5 (6B tokens, 400K vocab, uncased, 50d, 100d, 200d, 300d vectors, 822 MB download): glove.6B.zip Common Crawl (42B tokens, 1.9M vocab, uncased, 300d vectors, 1.75 GB download): glove.42B.300d.zip Common Crawl (840B tokens, 2.2M vocab, cased, 300d vectors, 2.03 GB download): glove.840B.300d.zip Twitter (2B tweets, 27B tokens, 1.2M vocab, uncased, 25d, 50d, 100d, 200d vectors, 1.42 GB download): glove.twitter.27B.zip
word2vec-GoogleNews-vectors
Language: English
Pre-trained Google News corpus (3 billion running words) word vector model (3 million 300-dimension English word vectors).
Word Analogies Test for linear relationships, examined by Mikolov et at. (2014)
Suggested Readings Simple Word Vector representations: word2vec, GloVe. cs224d.stanford.edu. Last Accessed: 2017-02-01. FastText and Gensim word embeddings. rare-technologies.com. Last Accessed: 2016-08-31. Distributed Representations of Words and Phrases and their Compositionality. papers.nips.cc. Last Accessed: 2013-12-05. Efficient Estimation of Word Representations in Vector Space. arxiv.org. Last Accessed: 2013-01-16

2.6 Conditional Random Fields in Name Entity Recognition

In this tutorial, I will write about how to using CRF++ to train your data for name entity recognition task.

Environment:
Ubuntu 14.04 Install CRF++ Download CRF++-0.58.tar.gz
Extact CRF++-0.58.tar.gz file
Navigate to the location of extracted folder through
Install CRF++ from source
./configure make sudo make install ldconfig Congratulations! CRF++ is install
$\text{crf}_{l}earnTrainingCRFTotrainaCRFusingCRF++, youneed2things$:
A template file: where you define features to be considered for training A training data file: where you have data in CoNLL format $\text{crf}_{l}earn{-}ttemplate_{f}iletrain_{d}ata_{f}ilemodel$
$\text{crf}_{l}earn - ttemplatetrain.txtmodelAbinaryofmodelisproduce.$
To test this model, on a testing data
$\text{crf}_{t}est - mmodeltestfile > output.txt$
$\text{crf}_{t}est - mmodeltest.txt > output.txtReferencesConditionalRandomFields$:
$InstallingCRF{+}{+}onUbuntuConditionalRandomFieldsTrainingandTestingusingCRF{+}{+}$

2.7 Entity Linking

In natural language processing, entity linking, named entity linking (NEL), named entity disambiguation (NED), named entity recognition and disambiguation (NERD) or named entity normalization (NEN) is the task of determining the identity of entities mentioned in text. More precise, it is the task of linking entity mentions to entries in a knowledge base (e.g., DBpedia, Wikipedia)
Entity linking requires a knowledge base containing the entities to which entity mentions can be linked. A popular choice for entity linking on open domain text are knowledge-bases based on Wikipedia, in which each page is regarded as a named entity. NED using Wikipedia entities has been also called wikification (see Wikify! an early entity linking system]). A knowledge base may also be induced automatically from training text or manually built.
NED is different from named entity recognition (NER) in that NER identifies the occurrence or mention of a named entity in text but it does not identify which specific entity it is
Examples Example 1:
For example, given the sentence “Paris is the capital of France”, the idea is to determine that “Paris” refers to the city of Paris and not to Paris Hilton or any other entity that could be referred as “Paris”.
Example 2:
Give the sentence “In Second Debate, Donald Trump and HIllary Clinton Spar in Bitter, Personal Terms”, the idea is to determine that “Donald Trump” refer to an American politician, and “Hillary Clinton” refer to 67th United States Secretary of State from 2009 to 2013.
Architecture
Mention detection: Identification of text snippets that can potentially be linked to entities Candidate selection: Generating a set of candidate entities for each mention Disambiguation: Selecting a single entity (or none) for each mention, based on the context Mention detection
Goal: Detect all “linkable” phrases
Challenges:

Recall oriented: Do not miss any entity that should be link Find entity name variants (e.g. “jlo” is name variant of [Jennifer Lopez]) Filter out inappropriate ones (e.g. “new york” matches >2k different entities) COMMON APPROACH Build a dictionary of entity surface forms entities with all names variants Check all document n-grams against the dictionary the value of n is set typically between 6 and 8 Filter out undesired entities Can be done here or later in the pipeline Examples
Candidate Selection
Goal: Narrow down the space of disambiguation possibilities
Balances between precision and recall (effectiveness vs. efficiency)
Often approached as ranking problem: keeping only candidates above a score/rank threshold for downstream processing.
COMMONNESS Perform the ranking of candidate entities based on their overall popularity, i.e., “most command sense”
Examples
Commonness can be pre-computed and stored in the entity surface form dictionary. Follows a power law with a long tail of extremely unlikely senses; entities at the tail end of distribution can be safely discarded (e.g., 0.001 is sensible threshold)
Disambiguation
Baseline approach: most common sense
Consider additional types of evidence: prior importance of entities and mentions, contextual similarity between the text surrounding the mention and the candidate entity, coherence among all entity linking decisions in the document.
Combine these signals: using supervised learning or graph-based approaches
Optionally perform pruning: reject low confidence or semantically meaning less annotations.
References “Entity Linking”. wikipedia “Entity Linking”. Krisztian Balog, University of Stavanger, 10th Russian Summer School in Information Retrieval. 2016 “An End-to-End Entity Linking Approach for Tweets”. Ikuya Yamada, Hideaki Takeda, Yoshiyasu Takefuji. 2015

2.8 Chatbot

28/01/2018 Hôm nay thấy khoá này hay quá Build Your Own Chatbot Cognitive Class CB0103EN

24/01/2018 Hôm qua vừa nhận kèo cà phê với CEO của Rabiloo. Thấy thú vị quá. Hôm nay hỏi anh Vũ qua qua về chatbot. Hehe

2.8.1 3 loại chatbot

Bài này là dịch từ bài của một bác ở IBM. Nóng hổi vừa thổi vừa dịch.

Chatbot hỗ trợ - Support Chatbots

Những còn này có xu hướng **nắm rõ về một lĩnh vực**, giống như các kiến thức của một công ty.
Có lẽ Rabiloo, otonhanh, và rất nhiều bot trên facebook thuộc loại này.

Chatbot chức năng - Skill chatbot

Các chatbot chức năng thường là loại **một câu lệnh**, và không cần phải quá chú ý về ngữ cảnh.
Ví dụ, nó có thể thực hiện các câu "Bật đèn lên"
Mấy con bot này có thể ở các nhà thông minh hay các bot trong công nghiệp.

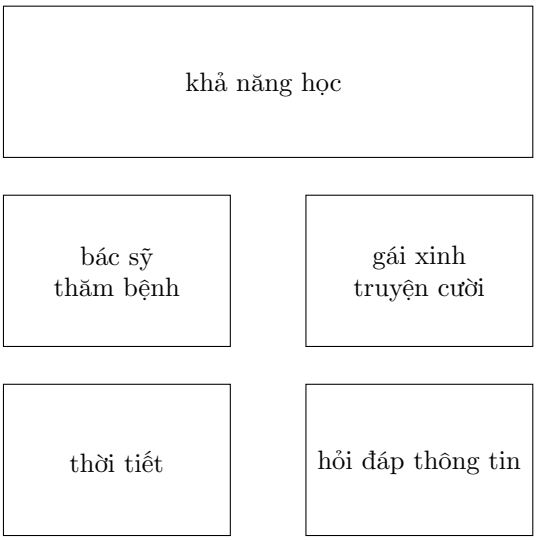
Trợ lý ảo

Các trợ lý ảo có thể kết hợp của hai loại trên. Hoạt động tốt nào biết một chút kiến thức của mỗi lĩnh vực.
Một ví dụ là bạn Siri của Apple.
Kết
Không cần biết loại chatbot bạn muốn xây dựng là gì, điều quan trọng là hãy **đưa cho nó một cuộc sống, một tính cách riêng**, **khiến nó trở nên hữu ích**, và **dễ dàng sử dụng**.
Mọi người sử dụng chatbot vì họ muốn có một cách giao tiếp tự nhiên hơn so với những cách trước đó. Có thể đó là công việc đơn giản như bật một chiếc đèn, hay đó là công việc phức tạp như cho vay thế chấp. Mỗi một công việc có những đặc tính cụ thể, chắc chắn chú bot của bạn tỏa sáng với thiết kế của nó. **Các phương pháp là không thể đếm xuể**.

2.8.2 Các câu hỏi mà chatbot cần chuẩn bị

Tham khảo bài viết này 36 Questions to Ask Your Chatbot

2.8.3 Ý tưởng chatbot



Mình muốn chatbot giao tiếp bằng ngôn ngữ tự nhiên chứ không theo kiểu mệnh lệnh. *cười -> ra chuyện cười.* Thế thì không khác gì search engine.
Mà sao không có quan bot nào khai thác tâm trạng, thông tin của người dùng. Để tìm ra nhu cầu của họ nhỉ?
Vài chức năng hữu ích

- Hỏi đáp thông tin cá nhân của bot. Nhiều bot fail bởi trò đơn giản này.
- Yêu đương an ủi tán tỉnh
- Dự báo thời tiết. Lấy thông tin địa điểm người dùng. Bình luận về thời tiết hiện tại
- Tìm ảnh gái xinh, đọc truyện cười
- Khả năng học

Chán chẳng cần làm

- Tra mã số karaoke

2.8.4 Một số chatbot

Chatbot tiếng anh

1. Mitsuku (web) (2000-). Chiến thắng Loebner Prize vào năm 2016-2017
2. Rose (web). Chiến thắng Loebner Prize vào năm 2014-2015
3. Cleverbot (1997)
4. ELIZA (1964-1966) tại MIT bởi Joseph Weizenbaum
5. poncho (messenger). Con này chỉ chuyên về thời tiết
6. Melody (Baidu). Chuyên tư vấn về bác sỹ
7. Do Not Pay (2017). Chatbot về tư vấn luật
8. meditatebot (messenger) - Hướng dẫn thiền
9. bots duolingo (ios) - Bot hướng dẫn học ngoại ngữ

Nền tảng tiếng Anh

- api.ai
- DialogFlow

Cuộc thi tiếng Anh
Loebner Prize. Được tổ chức lần đầu vào năm 1990. Mục tiêu là kiểm tra xem chương trình có vượt qua được Turning Test hay không.
Chatbot tiếng việt

1. troly.bedieu (messenger) (2016) Có mấy chức năng. Tán ngẫu. Tìm ảnh gái xinh. Ổn phết
2. Bob (skype) Cũng khá vui đấy
3. Người Bạn Tốt Miki (2016). Xịt rồi
4. hana.ai. Nghe nói có vẻ khủng

5. Simsimi. Thấy bảo "con" chatbot này bậy lắm. Không biết có thật không?
6. Sumichat Hỗ trợ cả tiếng Việt đây
7. VisualFriend (2007) bởi HungCode. Nghe nói là thần thánh lắm nhưng chưa kiểm chứng.

Nền tảng tiếng việt

1. hekate.ai (2017). 66 triệu doanh nghiệp. 1.2 tỷ người. 68 tỷ message mỗi ngày
2. harafunnel.com

Cuộc thi trên nền tảng tiếng Việt
fpt.ai: bot of the year
Đề bài: Xây dựng Chatbot để nâng cao trải nghiệm của người dùng cá nhân và doanh nghiệp
Thời gian: 22/11/2017 – 30/07/2018
Đối tượng: Cá nhân và doanh nghiệp quan tâm đến lĩnh vực Chatbot trên khắp Việt Nam.

Tài liệu tham khảo

Chỉ mục

Ghi chú