INTERNSHIP REPORT

RAJENDRA SINGH

COMPUTER SCIENCE AND ENGINEERING

IIT PALAKKAD

111601017

PREFACE

The present report is the outcome of the Internship Program at Researshala. The objective of the internship program was to familiarize the student with the implementation of the knowledge she/he earned on the campus. The practical knowledge is far different from the bookish knowledge that a student achieves in an institution.

The major problem that I faced during my internship was that there were not sufficient published documents available on the web as most of project topic was under research and quite new.

ACKNOWLEDGEMENT

I would like to thank all the people who helped me during that internship, especially my project leaders Harshit Aneja(CTO) and Shubham Jain(CEO) for the time they has devoted supervising and guiding me in my project. I would like to thank all the lab members for contributing to a very positive spirit of friendship and noncompetition.I would like to thank Varun, another intern who helped me in web scarping. I would like to thanks Researchshala for giving me the opportunity to work on research topics.

I would like to thank the developers of Tensorflow, Keras and DyNet for developing such powerful tools.

ABOUT RESEARCH LAB

Researchshala (Alazia Labs Private Limited) is a research lab that helps business schools in USA and Europe in their research projects. Their customers are from top B-schools such as Chicago Booth, London Business School, UC Berkeley to name a few.

Research requires various technical work which deviates researchers from their core research so, they work with professors/researchers as their RA/research team. Professors are using this service to analyse data, build machine learning models or simple R/stata programming. They have pre-built research specific internal tools which help them finish their task quickly.

CEO: Shubham Jain CTO: Harshit Aneja

ABSTRACT

During my intern period I learned about Nature Language Processing. Starting with web and pdf scraping data from site and cv. I used both ruled based and machine learning technique for data cleaning. My most of the work was on unstructured unlabeled data, on which I run the various nlp techniques to extract the required information. Among these technique Topic modeling was used for getting major topics out of text, Named entity Recognition for extracting various entity like person, location, dates etc., Part of speech tagging for finding the noun, verb etc. in text, sentiment analysis for measuring the level of positiveness or negativeness of text, dependency parsing for getting the relation between the various words in sentence (hence helping understanding the text). Then I worked with feed forward, Bidirectional, and Tree Recurrent neural network (RNN) architecture. Using these RNN along with convolutional neural network (CNN) and transfer learning I create model for image captioning and machine translation. Then I worked on GANs (Generative adversarial networks) for generating the image form sentence describing the image. Then I worked on dynamic computational graphs on the framework like Dynet, Dragnn, Tensorflow fold etc. for semantic analysis of text.

I also worked with convolutional neural network (CNN) and centrality graph for classifying images based on creativity.

Additionally, I also get opportunity to work on IOT (Internet of things) project where I worked with GSM module and arm processor for pollution level control of gases.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

- NLP - Natural language processing
- NLG - Natural language generation
- NLU - Natural language understanding
- NER - Named-entity recognition
- CNN - Convolutional neural network
- RNN - Recurrent neural network

- NLTK - Natural Language Toolkit
- CNTK - Cognitive Toolkit
- ORG - Organization
- LOC - Location
- ENT - Entity
- Q&A - Question and Answering
- GAN - Generative adversarial networks
- CRF - Conditional random field
- HMM - Hidden Markov model
- IOT - Internet of things
- GSM - Global System for Mobile
- TRBU - Transition Based Recurrent Unit
- DRAGNN - Dynamic Recurrent Acyclic Graphical Neural Networks
- LDA - Latent Dirichlet allocation
- LSTM - Long short-term memory
- GRU - Gated Recurrent Unit
- CBOW - Continuous bag-of-word

1) Web and pdf scraping:

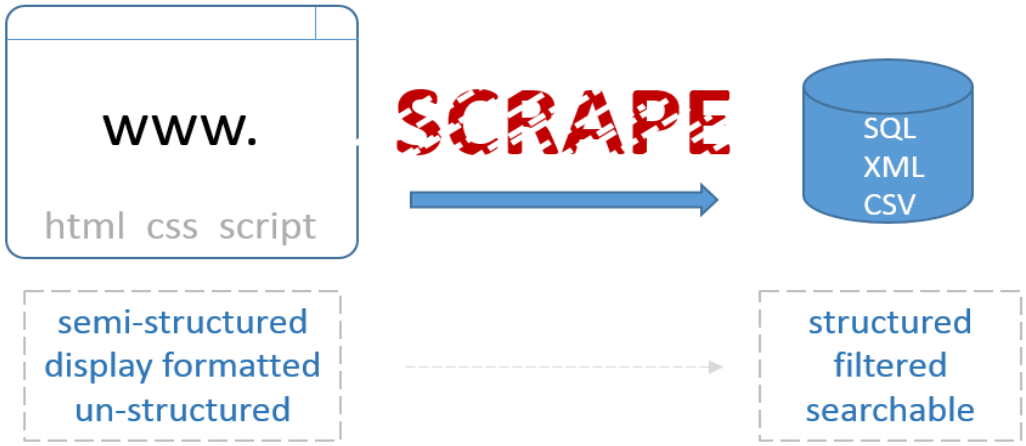


Figure 1

The need and importance of extracting data from the web is becoming increasingly loud and clear. Every few weeks, I find myself in a situation where I need lots and lots of data from the web.

What is Web Scraping?

Web scraping is a computer software technique of extracting information from websites. This technique mostly focuses on the transformation of unstructured data (HTML format) on the web into structured data (database or spreadsheet). You can perform web scrapping in various ways, including use of Google Docs to almost every programming language. I used Python because of its ease and rich eocsystem. It has a library known as 'BeautifulSoup' which assists this task.

I used pdfminer.six for pdf scarping.

Figure 2

2) Data Cleaning:

Many datasets have missing, malformed, or erroneous data. It's often unavoidable– anything from incomplete reporting to technical glitches can cause "dirty" data.

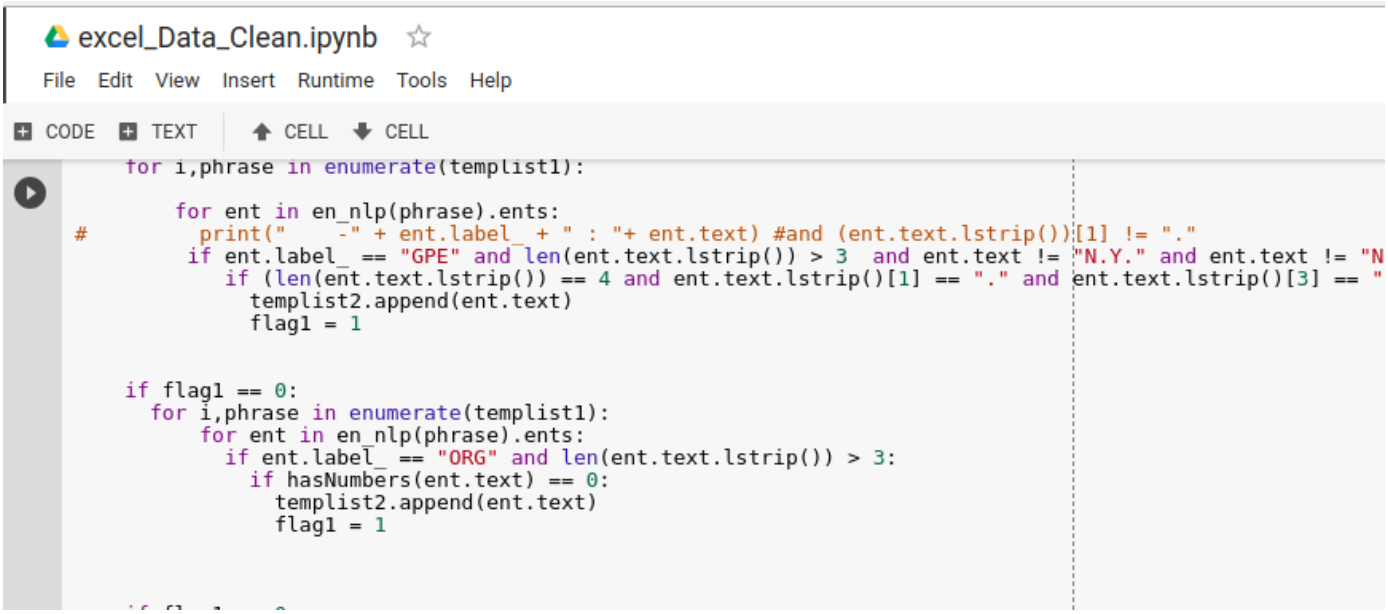


Figure 3: Screenshot of excel data cleaning project

Data cleaning method used:

- Deal with missing data
- Add default values
- Remove incomplete rows
- Deal with error-prone columns
- Normalize data types
- Change casing
- Rename columns

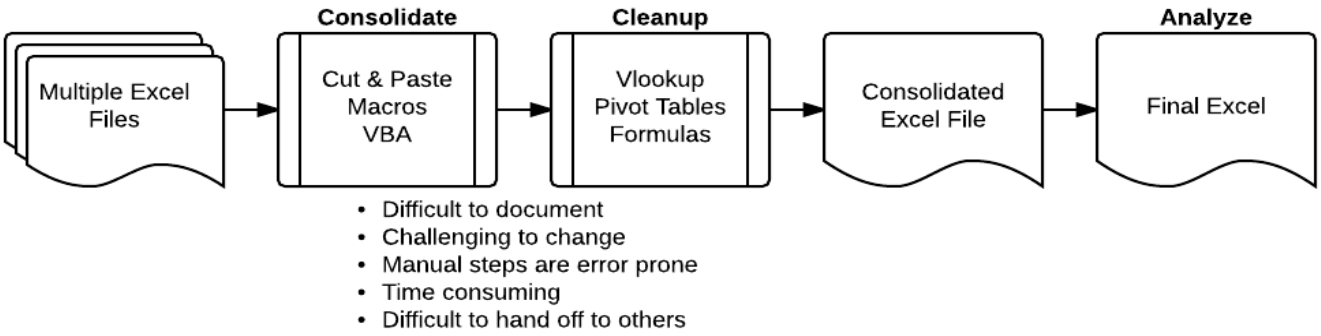


Figure 4

Cleaning a set of OCR'd free form text files and converting it to a format that makes it easy for a developer to process it using code can be difficult. Data cleaning is an ad-hoc process requiring use of every data processing, and possibly coding, tool you have in your tool belt:

Data cleaning tool used:

- Regular Expressions
- String replacement
- XML DOM traversal
- Standard string split and replacement operations
- Filter values on created lists from source data.
- Use of trained models

3) Topic modeling:

Topic Modeling is a technique to extract the hidden topics from large volumes of text. Latent Dirichlet Allocation (LDA) is a popular algorithm for topic modeling with excellent implementations in the Python's Gensim package. The challenge, however, is how to extract good quality of topics that are clear, segregated and meaningful. This depends heavily on the quality of text preprocessing and the strategy of finding the optimal number of topics. One of the primary applications of nlp is to automatically extract what topics people are discussing from large volumes of text. Some examples of large text could be feeds from social media, customer reviews of hotels, movies, etc, user feedbacks, news stories, e-mails of customer complaints etc.

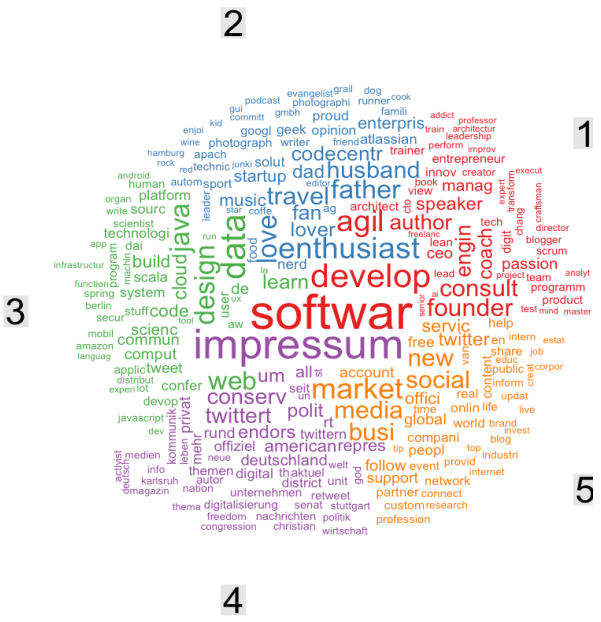


Figure 5

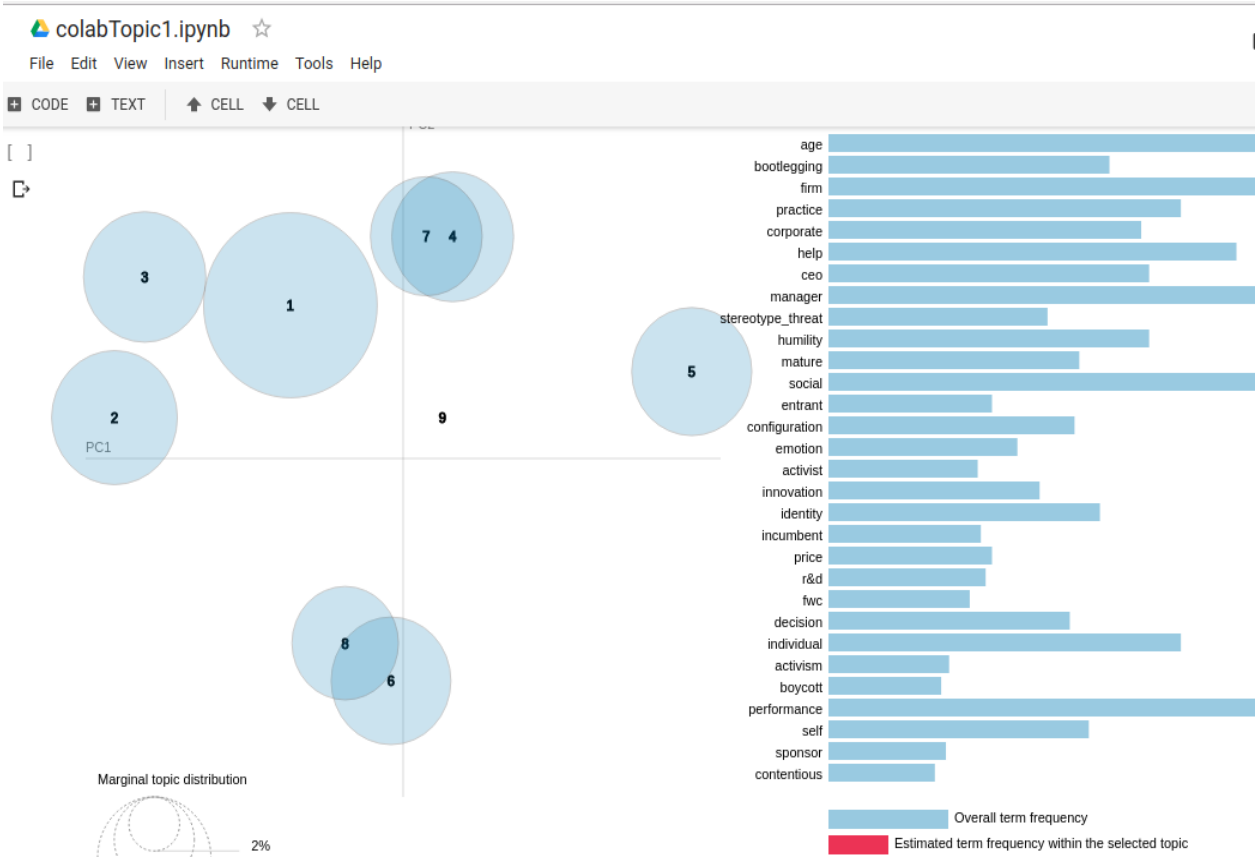


Figure 6: Screenshot of topic model of news articles

Knowing what people are talking about and understanding their problems and opinions is highly valuable to businesses, administrators, political campaigns. And it's really hard to manually read through such large volumes and compile the topics.

Thus is required an automated algorithm that can read through the text documents and automatically output the topics discussed.

LDA's approach to topic modeling is it considers each document as a collection of topics in a certain proportion. And each topic as a collection of keywords, again, in a certain proportion.

Once you provide the algorithm with the number of topics, all it does it to rearrange the topics distribution within the documents and keywords distribution within the topics to obtain a good composition of topic-keywords distribution.

A topic is nothing but a collection of dominant keywords that are typical representatives. Just by looking at the keywords, you can identify what the topic is all about.

Steps of topic modeling:

> Tokenize words and Clean-up text
> Creating Bigram and Trigram Models
> Remove Stopwords, Make Bigrams and Lemmatize
> Create the Dictionary and Corpus needed for Topic Modeling
> Building the Topic Model
> View the topics in LDA model
> Visualize the topics-keywords
> How to find the optimal number of topics for LDA?
> Finding the dominant topic in each sentence
> Find the most representative document for each topic
> Topic distribution across documents

We built a basic topic model using Gensim's LDA and visualize the topics using pyLDAvis. You saw how to find the optimal number of topics using coherence scores and how you can come to a logical understanding of how to choose the optimal model.
Finally we saw how to aggregate and present the results to generate insights that may be in a more actionable.

4) Named entity Recognition (NER):

Introduction

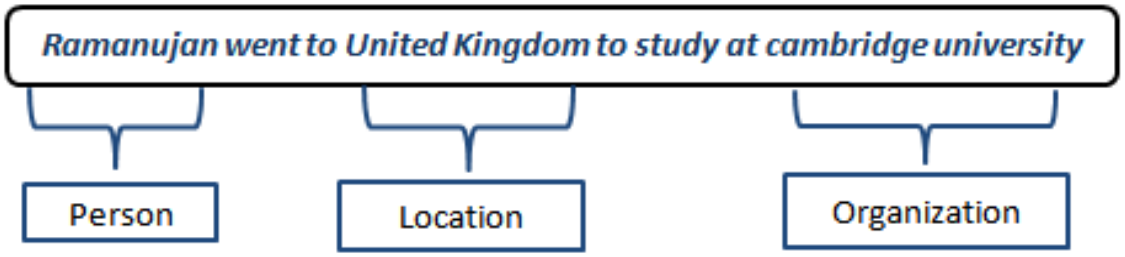


Figure 7

Named Entity Recognition is one of the very useful information extraction technique to identify and classify named entities in text. These entities are pre-defined categories such a person's names, organizations, locations, time representations, financial elements, etc.

Apart from these generic entities, there could be other specific terms that could be defined given a particular problem. These terms represent elements which have a unique context compared to the rest of the text. For example, it could be anything like operating systems, programming languages, football league team names etc. The machine learning models could be trained to categorize such custom entities which are usually denoted by proper names and therefore are mostly noun phrases in text documents.

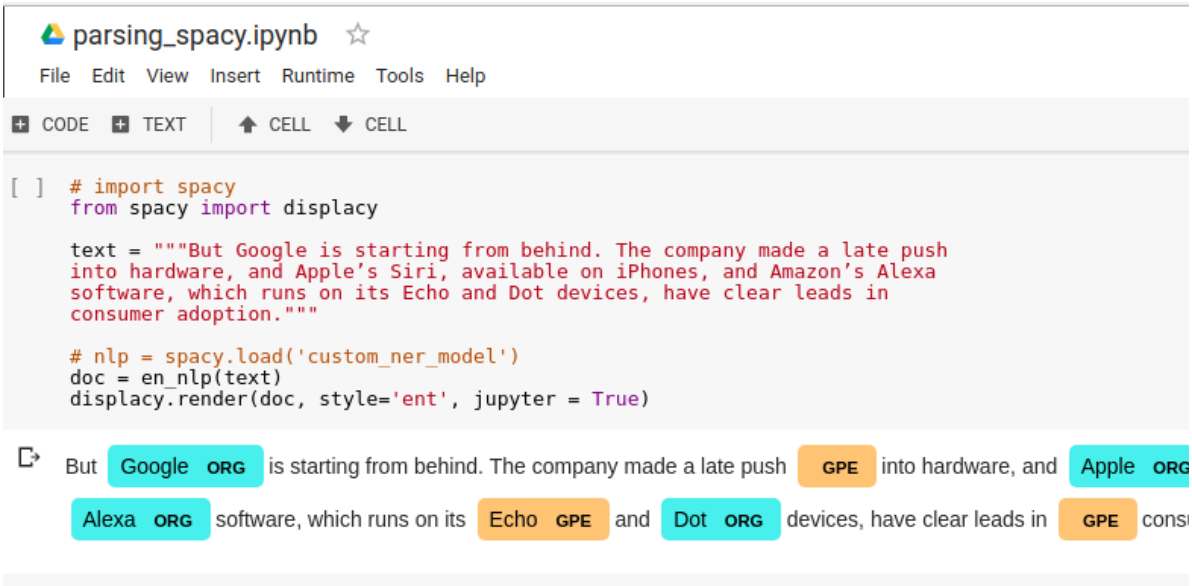


Figure 8: Screenshot of NER Tags

NER Categories

Broadly NER has three top-level categorizations - entity names, temporal expressions, and number expressions:

Entity Names represent the identity of an element, for example name of a person, title, organization, any living or nonliving thing etc.

A temporal expression is some sequence of words with time related elements for example calendar dates, times of day, durations etc.

A numerical expression is a mathematical sentence involving only numbers and or operation symbols. It could depict, financial numbers, tangible entities, mathematical expressions etc.

Named entities are often not simply singular words, but are chunks of text, e.g. "University of British Columbia" or "Bank of America". Therefore, some chunking and parsing prediction model is required to predict whether a group of tokens belong in the same entity.

Approaches to NER

To implement NER chunker to tag specific elements in the text, there are two different approaches. The classical approach is knowledge/rule based and the other way to solve the problem is by using supervised machine learning. However sometimes, the combination of both gives better results than the individual ones.

Rule based approach

A rule based NER system uses predefined language dependent rules based on linguistics which helps in the identification of named entities in a document. Rule based systems perform well but are limited to a particular language and are not flexible to changes. Most of these entities are either proper nouns or proper nouns in coalition with numbers.

Machine Learning approach

Rule based NER can be sometimes very complex and less accurate, in such cases machine learning approach is helpful. In this approach, by using supervised learning on labelled data, machines can predict custom entities on a given text.

Sent: ['Linux', 'is', 'the', 'best', 'OS']

Labels: ['OS','IR','IR','IR','IR']

Sent: ['Ubuntu', 'is', 'my', 'favorite', 'OS']

Labels: ['OS','IR','IR','IR','IR']

Conditional Random Fields (CRF)

To take advantage of the surrounding context of the tokens labelled in a sequence, a commonly used method is conditional random field (CRF). It is a type of probabilistic undirected graphical model that can be used to model sequential data. CRF calculates the conditional probability of values on classified output nodes given values assigned to the classified input nodes.

Conditionally trained CRFs can easily include large number of non independent features for example POS tags, lower/title/uppercase flags etc. but the power of model increases by adding features that are concurrent or sequential in nature.

Training CRF Model

To train a CRF Model, we need to extract features from all the sentences in our stub corpus. The features are the token sequence of the words.

NER is extensively used in question and answer systems, document clustering, textual entailment, and text analytics applications. An understanding of the named entities involved in a document provides much richer analytical frameworks and cross-referencing.

For example, news and publishing houses generate large amounts of online content on a daily basis and managing them correctly is challenging. To get maximum out of each article, entity extraction can be used to automatically scan entire articles and reveal which are the major people, organizations, and places are being talked about. Knowing the relevant tags for each article would also help in automatically categorizing and discovering the articles.

NER also plays an important role in automation of customer support. Automatically tagged locations and product names can help smoothly route customer queries to right location and people in a company with multiple branches and many employees.

5) Part of speech (POS) tagging:

POS tagging is the process of assigning a word to its grammatical category, in order to understand its role within the sentence. Traditional parts of speech are nouns, verbs, adverbs, conjunctions, etc. Part-of-speech taggers typically take a sequence of words (i.e. a sentence) as input, and provide a list of tuples as output, where each word is associated with the related tag.

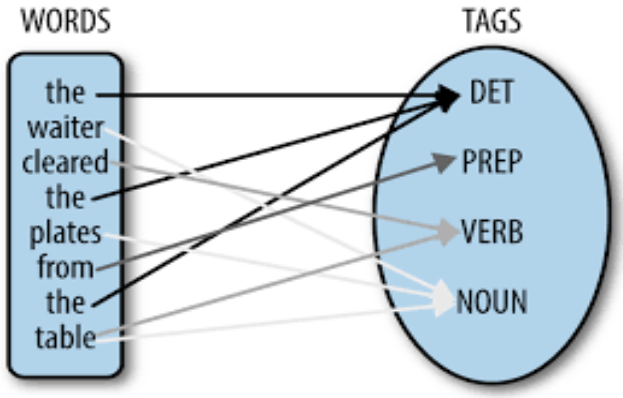


Figure 9

Part-of-speech tagging is what provides the contextual information that a lemmatiser needs to choose the appropriate lemma.

POS tag list with example:

- CC coordinating conjunction
- CD cardinal digit
- DT determiner
- EX existential there (like: "there is" ... think of it like "there exists")
- FW foreign word
- IN preposition/subordinating conjunction
- JJ adjective 'big'
- JJR adjective, comparative 'bigger'
- JJS adjective, superlative 'biggest'
- LS list marker 1)
- MD modal could, will
- NN noun, singular 'desk'
- NNS noun plural 'desks'
- NNP proper noun, singular 'Harrison'
- NNPS proper noun, plural 'Americans'
- PDT predeterminer 'all the kids'

- POS possessive ending parent's
- PRP personal pronoun I, he, she
- PRP$ possessive pronoun my, his, hers
- RB adverb very, silently,
- RBR adverb, comparative better
- RBS adverb, superlative best
- RP particle give up
- TO to go 'to' the store.
- UH interjection errrrrrrrm
- VB verb, base form take
- VBD verb, past tense took
- VBG verb, gerund/present participle taking
- VBN verb, past participle taken
- VBP verb, sing. present, non-3d take
- VBZ verb, 3rd person sing. present takes
- WDT wh-determiner which
- WP wh-pronoun who, what
- WP$ possessive wh-pronoun whose
- WRB wh-abverb where, when

6) Sentiment Analysis:

In Natural Language Processing there is a concept known as Sentiment Analysis.

Given a movie review or a tweet, it can be *automatically* classified in categories.
These categories can be user defined (positive, negative) or whichever classes you want.

Text Pre-processing
All tweets are processed to remove unnecessary things like links, non-English words, stopwords, punctuation’s, etc.

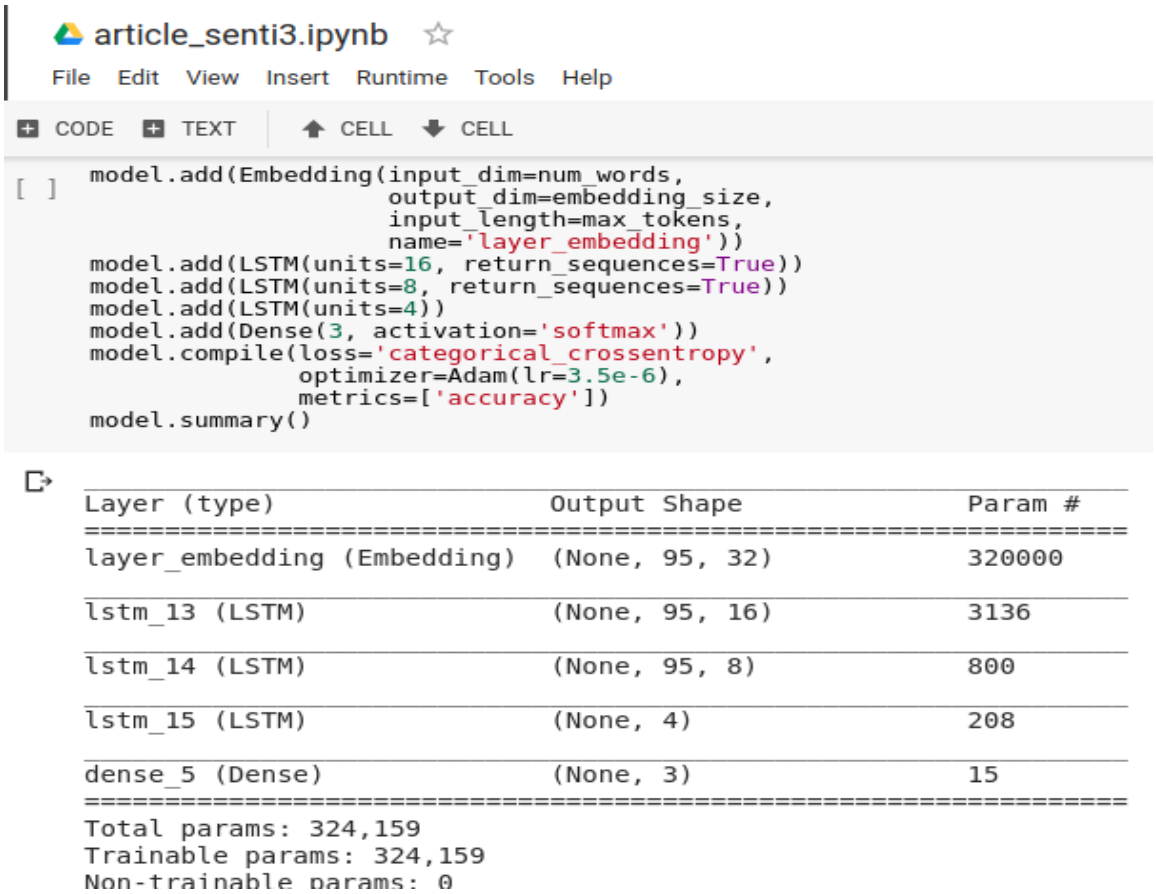


Figure 10: Screenshot of lstm model bulid for sentiment analysis training

Sentiment Analysis for Brand Monitoring

One of the most well documented uses of Sentiment Analysis is to get a full 360 view of how your brand, product, or company is viewed by your customers and stakeholders. Widely available media, like product reviews and social, can reveal key insights about what your business is doing right or wrong. Companies can also use sentiment analysis to measure the impact of a new product, ad campaign, or consumer's response to recent company news on social media.

Sentiment Analysis for Customer Service

Customer service agents often use sentiment analysis to automatically sort incoming user email into "urgent" or "not urgent" buckets based on the sentiment of the email, proactively identifying frustrated users. The agent then directs their time toward resolving the users with the most urgent needs first. As customer service becomes more and more automated through Machine Learning, understanding the sentiment of a given case becomes increasingly important.

Sentiment Analysis for Market Research and Analysis

Sentiment analysis is used in business intelligence to understand the subjective reasons why consumers are or are not responding to something (e.x. why are consumers buying a product? What do they think of the user experience? Did customer service support meet their expectations?). Sentiment analysis can also be used in the areas of political science, sociology, and psychology to analyze trends, ideological bias, opinions, gauge reactions, etc.

In addition to the definition problem, there are multiple layers of meaning in any human generated sentence. People express opinions in complex ways; rhetorical devices like sarcasm, irony, and implied meaning can mislead sentiment analysis. The only way to really understand these devices are through context: knowing how a paragraph is started can strongly impact the sentiment of later internal sentences.

Sentiment Analysis, also known as opinion mining, is a powerful tool you can use to build smarter products. It gives you a general idea about the positive, neutral, and negative sentiment of texts. Social media monitoring apps and companies all rely on sentiment analysis and machine learning to assist them in gaining insights about mentions, brands, and products.

7) Parsing

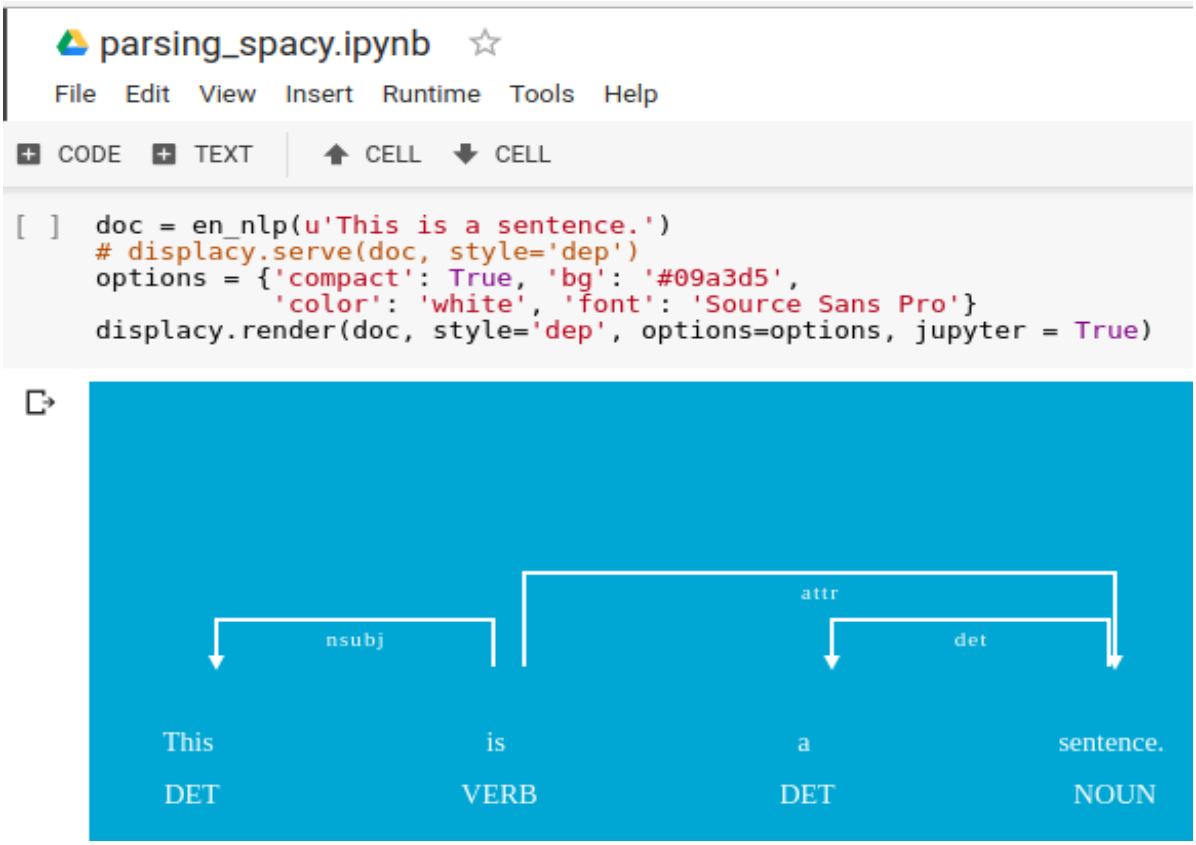


Figure 11. Screenshot of parse tree for sentence

WHY PARSING?

Parsing natural language gives structure to a sentence, which in turn allows to access its meaning.

Parsing applications:

- Data mining: extracting information (and storing in database)
- Question answering and research engine
- Automatic translation
- Grammar checkers in text editors

Parsing refers to the mental operations that establish relationships between words in sentences. Human communication requires parsing sentences to infer meanings. Thus, an artificially intelligent system with natural language processing ability is ultimately expected to analyze strings of words into a parse tree showing syntactic and/or semantic relation among the words. Thus, it is worthy to review the human parsing ability to enable an AGI system to mimic it.

Semantic Parsing in NLP

Semantic parsing can be defined as representing the meaning of a sentence in some formal knowledge representation language (or logical form) that supports automated inference. A semantic parser initially needs a formal language.

To achieve such a mapping, one initially needs formal representations and categorial tags for each word. Furthermore, the cost of mapping increases in case of ambiguities. When a full mapping of a text is achieved, it is possible to use it for various purposes such as information extraction and question answering. For example, questions can be represented in this form as well (5). In this case, a system should find the best x satisfying the logical representation.

Besides choosing the knowledge representation formalism, a semantic parser needs to deal with two other issues: parsing method choice and learning to parse.

Traditionally, semantic parsing can be divided into two categories: logical and statistical parsing. Logical parsing includes using hand-written rules, tables and dictionaries to parse texts while statistical ones try to learn parsing through machine learning algorithms. Both sides have its own advantages and disadvantages. Currently, it seems that logical and statistical techniques tend to merge especially to cope with ambiguities.

The traditional linguistic notion of grammatical relation provides the basis for the Grammatical relation binary relations that comprise these dependency structures. The arguments to these Head relations consist of a head and a dependent. The head word of a constituent was the central organizing word of a larger constituent (e.g, the primary noun in a noun phrase, or verb in a verb phrase). The remaining words in the constituent are either direct, or indirect, dependents of their head. In dependency-based approaches, the head-dependent relationship is made explicit by directly linking heads to the words that are immediately dependent on them, bypassing the need for constituent structures.

A **parsing tree** is an ordered, rooted tree that represents the syntactic structure of a string according to some context-free grammar. The term *parse tree* itself is used primarily in computational linguistics; in theoretical syntax, the term *syntax tree* is more common.

8) Various Recurrent neural network (RNN) Architecture:

8.1) Simple RNN

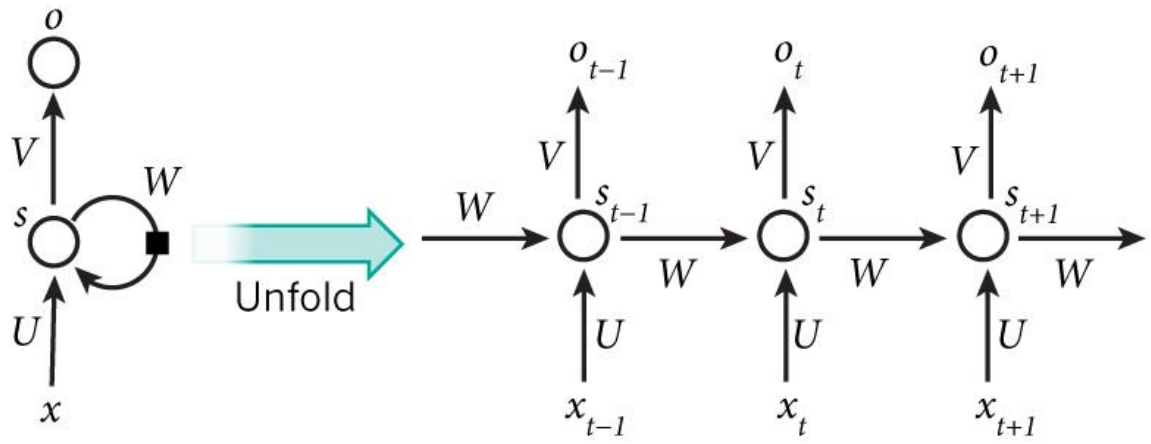


Figure 12

A **recurrent neural network** (**RNN**) is a class of artificial neural network where connections between nodes form a directed graph along a sequence. This allows it to exhibit dynamic temporal behavior for a time sequence. Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition.

A recurrent neural network (RNN) is a special type of neural network in which the computation may be done in several time steps, with loops that carry a hidden state from one time step to the next. This enables the network to process variable-length sequences, such as for instance reading a sentence by reading one word at each time step. RNNs can also be used to generate variablelength outputs, for instance it can generate a sentence word by word. Simple RNNs The simplest existing RNN model is a mode where a hidden state vector is propagated between time steps, and that hidden state is calculated by a standard neural network layer whose input vector is the concatenation of the previous state vector and the input vector at that time step: ht = (Wi xt + Wr ht¡1 + bh) ot = (Wo ht + bo) f(x1; :::; xT) = (o1; :::; oT) The parameters of such a model are the matrices Wi ; Wr and Wo, the bias vectors bh and bo, as well as the initial state h0.

Backpropagation Through Time

As long as the loss function is dened in terms of f(x) and y, the gradient of the loss with respect to the parameters can be calculated analytically. This involves calculating a gradient for the

parameters at each time step, and is usually referred to as backpropagation through time [25]. In this process, each time step produces a gradient with respect to all the parameters, and the nal gradient is the sum of all these partial gradients. The analytic calculation of these gradients can be done automatically with tools such as Theano, therefore we do not need to enter in the details of such complex derivations. Vanishing Gradient Problem An important problem with recurrent neural networks that has been extensively studied [5][19] is the

vanishing gradient problem.

It arises from the diculty of backpropagation through time to do correct credit assignment over long time ranges, as the gradient decays exponentially with time and becomes mixed with gradients from other sources. It is one of the main obstacles to the eective training of RNNs with backpropagation through time, and a very active area of research.

Long Short-Term Memory

Long short-term memory RNNs [15] are a special type of recurrent neural nets that partially solve the vanishing gradient problem. They are composed of more complex equations than standard RNNs, as they implement several gating mechanisms, allowing the RNN to ignore some inputs or to keep some state values constant at each time step. An LSTM RNN is described by the following equations:

8.2) Bidirectional RNN

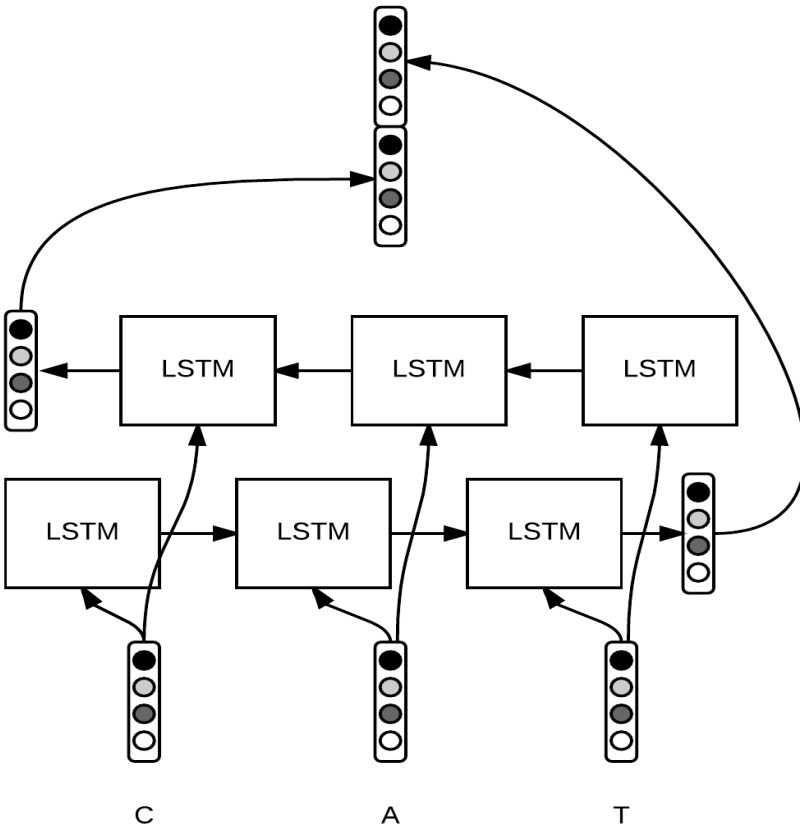


Figure 13

An important problem with the RNN approach is that the encoder RNN forgets the words it reads along the way, and the state C contains a lot of information about the last few words and very little information about the first words of the source sentence. To alleviate this problem, a bidirectional encoder can be used. This simply consists of using two RNNs to read the sentence, one that reads it normally and the other that reads it in the reverse order. The state C that is passed to the decoder is the concatenation of the last states of the two RNNs, therefore containing information about the beginning as well as the end of the sentence.

9) Transfer learning

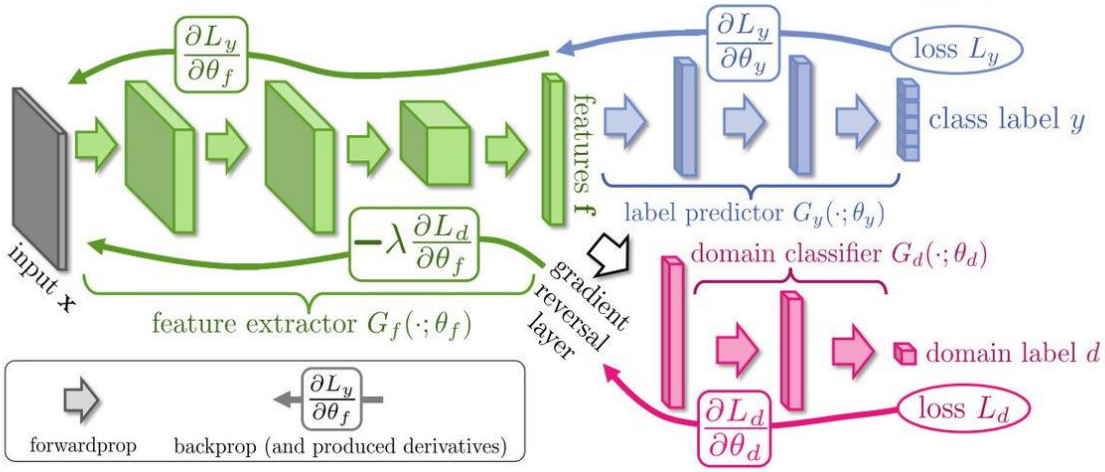


Figure 14

In practice, very few people train an entire Convolutional Network from scratch (with random initialization), because it is relatively rare to have a dataset of sufficient size. Instead, it is common to pretrain a ConvNet on a very large dataset (e.g. ImageNet, which contains 1.2 million images with 1000 categories), and then use the ConvNet either as an initialization or a fixed feature extractor for the task of interest. The three major Transfer Learning scenarios look as follows:

9.1) ConvNet as fixed feature extractor

Take a ConvNet pretrained on ImageNet, remove the last fully-connected layer (this layer's outputs are the 1000 class scores for a different task like ImageNet), then treat the rest of the ConvNet as a fixed feature extractor for the new dataset. In an AlexNet, this would compute a 4096-D vector for every image that contains the activations of the hidden layer immediately before the classifier. We call these features **CNN codes**. It is important for performance that these codes are ReLUd (i.e. thresholded at zero) if they were also thresholded during the training of the ConvNet on ImageNet (as is usually the case). Once you extract the 4096-D codes for all images, train a linear classifier (e.g. Linear SVM or Softmax classifier) for the new dataset.

9.2) **Pretrained models**.

Since modern ConvNets take 2-3 weeks to train across multiple GPUs on ImageNet, it is common to see people release their final ConvNet checkpoints for the benefit of others who can use the networks for fine-tuning. For example, the Caffe library has a Model Zoo where people share their network weights.

9.3) **Fine-tuning the ConvNet**.

The second strategy is to not only replace and retrain the classifier on top of the ConvNet on the new dataset, but to also fine-tune the weights of the pretrained network by continuing the backpropagation. It is possible to fine-tune all the layers of the ConvNet, or it's possible to keep some of the earlier layers fixed (due to overfitting concerns) and only fine-tune some higher-level portion of the network. This is motivated by the observation that the earlier features of a ConvNet contain more generic features (e.g. edge detectors or color blob detectors) that should be useful to many tasks, but later layers of the ConvNet becomes progressively more specific to the details of the classes contained in the original dataset. In case of ImageNet for example, which contains many dog breeds, a significant portion of the representational power of the ConvNet may be devoted to features that are specific to differentiating between dog breeds.

10) Generative adversarial networks (GANs)

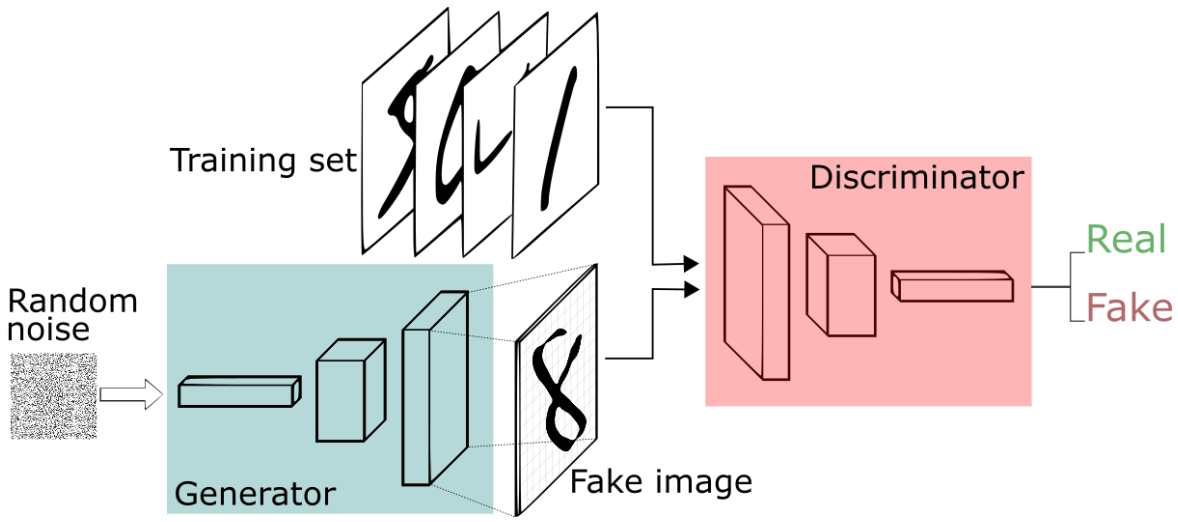


Figure 15

GANs are a class of artificial intelligence algorithms used in unsupervised machine learning, implemented by a system of two neural networks contesting with each other in a zero-sum game framework. They were introduced by Ian Goodfellow *et al.* in 2014. This technique can

generate photographs that look at least superficially authentic to human observers, having many realistic characteristics

Here are the steps a GAN takes:

- The generator takes in random numbers and returns an image.

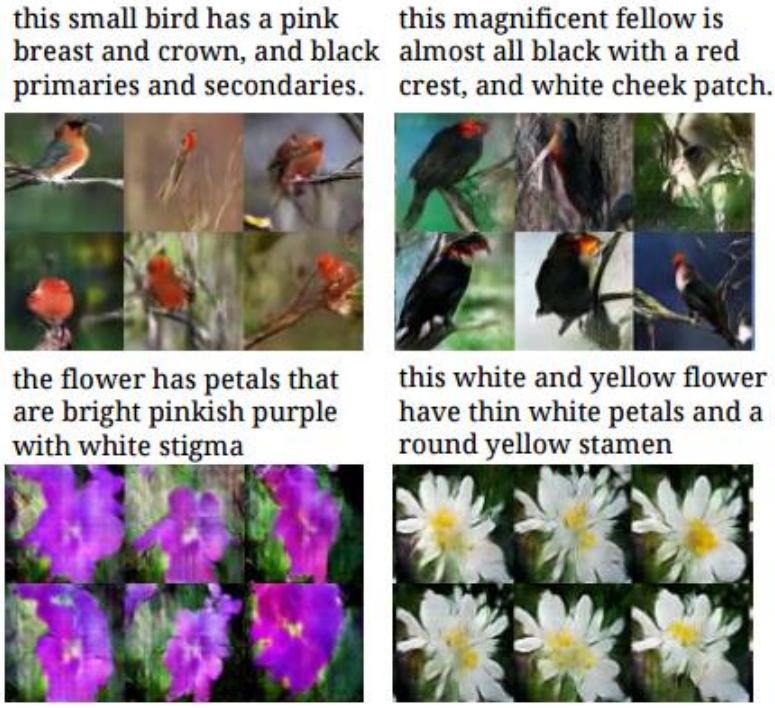


Figure 16

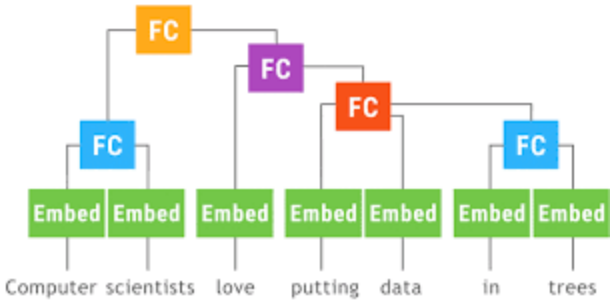
- This generated image is fed into the discriminator alongside a stream of images taken from the actual dataset.
- The discriminator takes in both real and fake images and returns probabilities, a number between 0 and 1, with 1 representing a prediction of authenticity and 0 representing fake.

So you have a double feedback loop:

- The discriminator is in a feedback loop with the ground truth of the images, which we know.
- The generator is in a feedback loop with the discriminator.

11) Dynamic computational graphs:

Figure 17



Neural networks that compute over graph structures are a natural fit for problems in a variety of domains, including natural language (parse trees) and cheminformatics (molecular graphs). However, since the computation graph has a different shape and size for every input, such networks do not directly support batched training or inference. They are also difficult to implement in popular deep learning libraries, which are based on static data-flow graphs. We introduce a technique called dynamic batching, which not only batches together operations between different input graphs of dissimilar shape, but also between different nodes within a single input graph. The technique allows us to create static graphs, using popular libraries, that emulate dynamic computation graphs of arbitrary shape and size. We further present a high-level library1 of compositional blocks that simplifies the creation of dynamic graph models. Using the library, we demonstrate concise and batch-wise parallel implementations for a variety of models from the literature.

DYNAMIC BATCHING

In deep learning libraries like TensorFlow, computations are manually batched. The computation is expressed as a static graph of mathematical operations, such as $y = \sigma(x \cdot w + c)$, which are polymorphic in batch size; an input x of dimensions (b, n) will yield an output of dimensions (b, m), where b is the batch size. With DCGs, the graph of operations is not static, but is assumed to be different for every input, so multiple inputs no longer naturally batch together in the same way. The dynamic batching algorithm overcomes this difficulty. Given a set of computation graphs as input, each of which has a different size and topology, it will rewrite the graphs by batching together all instances of the same operation that occur at the same depth in the graph. The rewriting process inserts additional concat and gather operations to move data between the batched operations; the indices to gather encode the topology of the original input graphs.

12) GSM Module and Arm Processor

GSM is a mobile communication modem; it is stands for global system for mobile communication (GSM). The idea of GSM was developed at Bell Laboratories in 1970. It is widely used mobile communication system in the world. GSM is an open and digital cellular technology used for transmitting mobile voice and data services operates at the 850MHz, 900MHz, 1800MHz and 1900MHz frequency bands.

Figure 18

GSM system was developed as a digital system using time division multiple access (TDMA) technique for communication purpose. A GSM digitizes and reduces the data, then sends it down through a channel with two different streams of client data, each in its own particular time slot. The digital system has an ability to carry 64 kbps to 120 Mbps of data rates.

Features of GSM Module:

- Improved spectrum efficiency
- International roaming
- Compatibility with integrated services digital network (ISDN)
- Support for new services.
- SIM phonebook management
- Fixed dialing number (FDN)
- Real time clock with alarm management
- High-quality speech
- Uses encryption to make phone calls more secure
- Short message service (SMS)

ARM Processor :

An ARM processor is one of a family of CPUs based on the RISC (reduced instruction set computer) architecture developed by Advanced RISC Machines (ARM).

Figure 19

ARM makes 32-bit and 64-bit RISC multi-core processors. RISC processors are designed to perform a smaller number of types of computer instructions so that they can operate at a higher speed, performing more millions of instructions per second (MIPS). By stripping out unneeded instructions and optimizing pathways, RISC processors provide outstanding performance at a fraction of the power demand of CISC (complex instruction set computing) devices.

An ARM processor is one of a family of CPUs based on the RISC (reduced instruction set computer) architecture developed by Advanced RISC Machines (ARM).

ARM makes 32-bit and 64-bit RISC multi-core processors. RISC processors are designed to perform a smaller number of types of computer instructions so that they can operate at a higher speed, performing more millions of instructions per second (MIPS). By stripping out unneeded instructions and optimizing pathways, RISC processors provide outstanding performance at a fraction of the power demand of CISC (complex instruction set computing) devices.

ARM processors are extensively used in consumer electronic devices such as smartphones, tablets, multimedia players and other mobile devices, such as wearables. Because of their reduced instruction set, they require fewer transistors, which enables a smaller die size for the integrated circuitry (IC). The ARM processor's smaller size, reduced complexity and lower power consumption makes them suitable for increasingly miniaturized devices.

ARM processor features include:

- Load/store architecture.
- An orthogonal instruction set.
- Mostly single-cycle execution.
- Enhanced power-saving design.
- 64 and 32-bit execution states for scalable high performance.
- Hardware virtualization support.

13) FUTURE SCOPE OF NLP

- NLP Attempt to make AI more human like, which is a hard task now. (For example, making a virtual assistant more conversational)
- Proliferate existing AI technologies. (For example, extending automatic image captioning to healthcare and other applications to get a better understanding of images)
- Healthcare

  Information discovery and retrieval

  Diagnostic assistance

  Virtual healthcare assistant

  Image classification and report generation

- Personal Virtual Assistance
  most current virtual AI assistants (such as Siri, Alexa, Echo, etc.) understand and respond to vocal commands in a sequence. That is, they execute one command at a time.

However, to take on more complex tasks, they have to be able to converse, much like a human.

- Automotive
  The automotive industry, particularly cars, is a “preeminent AI platform.” Automotive is a fast-moving consumer tech area, and car OEMs are evolving fast. Cars will be increasingly used as autonomous robots whose transportational capabilities could be augmented with other onboard computational capabilities and sensors.

- Customer Service
  In the customer service field, advanced NLP technologies could be used to analyze voice calls and emails in terms of customer happiness quotient, prevalent problem topics, sentiment analysis, etc.

- Smart search: Users will be able to search via voice commands rather than typing or using keywords. NLP systems will increasingly use image and object classification methods to help users search using images. For example, a user will be able to take a picture of a vehicle they like and use it to identify its make and model so they can buy similar vehicles online.

14) CONCLUSION

During this internship I had the opportunity to learn and work on a great variety of domains, which has been very enriching. However, I wasn't able to bring to a conclusion some projects I worked on. In this section I will quickly summarize what has been done, and what remains to be done for all these projects. Learning About Deep Learning and NLP, by being in the lab, in contact with many other students and working on various projects, I have gained a good overview of the domain that is deep learning and NLP. I acquired a good understanding of the basic principles of neural networks and optimization techniques, and have been able to study in details many aspects such as recurrent neural nets, optimization algorithms, regularization methods, practical GPU programming, etc. I have seen from afar a variety of applications of NLP and am confident that I have acquired the knowledge and skill necessary to apply deep learning and NLP to many problems.

15) CERTIFICATE

researchshala

(Alazia Labs Private Limited)
CIN : U72900DL2018PTC335580

July 10, 2018

TO WHOM IT MAY CONCERN

This is to certify that Rajendra Singh (Student ID - 111601017) s/o Mr. Bhagwat Singh, 2nd year student of IIT Palakkad, has completed his internship at Researchshala(Alazia Labs Pvt Ltd) starting 10th May 2018 to 10th July 2018 at our Chandigarh office. He worked on Data Science and Machine Learning Projects.

Harshit Aneja
Co-Founder & CTO

Registered Office - 77C, WP Block, Pitampura, Delhi-110034
Email ID - contact@researchshala.com