

Project Notes:

Project Title: Using Machine Learning and a new Shorthand for Faster Transcription

Name: Ronit Avadhuta

Note Well: There are NO SHORT-cuts to reading journal articles and taking notes from them. Comprehension is paramount. You will most likely need to read it several times so set aside enough time in your schedule.

Contents:

Knowledge Gaps:	2
Literature Search Parameters:	3
Article #1 Notes: New Translation Software	4
Article #2 Notes: Method for computer-assisted translation	6
Article #3 Notes: Phonetic-based text input method	9
Article #4 Notes: Statistical review of Online/Offline Gregg Shorthand Recognition using CANN and BP - A Comparative analysis	11
Article #5 Notes: Segmentation and recognition of phonetic features in handwritten Pitman shorthand	13
Article #6 Notes: Automatic recognition and transcription of Pitman's handwritten shorthand—An approach to shortforms	15
Article #7 Notes: Ontology-based parser for natural language processing	18
Article #8 Notes: English-Arabic Handwritten Character Recognition using Convolutional Neural Networks	21
Article #9 Notes: How File Compression Works	23
Article #10 Notes: Different Languages, Similar Encoding Efficiency	25
Article #11 Notes: Fast Compression Algorithm for UNICODE	29
Article #12 Notes: Natural Language Processing: An introduction	31
Article #13 Notes: Object recognition in images using convolutional neural network	35
Article #14 Notes: Calamari - A High-Performance Tensorflow-based Deep Learning Package for Optical Character Recognition	38
Article #15 Notes: Fooling OCR Systems with Adversarial Text Images	41

Article #16 Notes: Deep Learning Approach in Gregg Shorthand Word to English-Word Conversion	44
Article #17 Notes: A new image classification method using CNN transfer learning and web data augmentation	48
Article #18 Notes: Data Augmentation for Recognition of Handwritten Words and Lines Using a CNN-LSTM Network	51
Article #19 Notes: Dropout: a simple way to prevent neural networks from overfitting	55
Article #20 Notes: Combining Convolutional Neural Network With Recursive Neural Network for Blood Cell Image Classification	58

Knowledge Gaps:

This list provides a brief overview of the major knowledge gaps for this project, how they were resolved and where to find the information.

Knowledge Gap	Resolved By	Information is located	Date resolved
Research ML Softwares that could prove to be useful for my project	Creating a plan for the next two months and all the material I will need for studying	<ul style="list-style-type: none"> - MIT OpenCourseware for Linear Algebra - 3Blue1Brown for Essence of Calculus - Probability by MIT on EdX - 3 Playlists for <ul style="list-style-type: none"> - Math - Tensorflow - Deep Learning - Unacademy course ML 	9/30
Research how to download and connect the software myself	Finding some tutorials online (on youtube for example) of importing Tensorflow and its respective libraries	https://www.youtube.com/watch?v=QJQTIp5McV8&feature=emb_logo&ab_channel=CodingTheSmartWay.com	9/27
Lexers and Parsers	<p>Meeting with Mrs. Taricco</p> <p>Watching YouTube tutorials</p> <p>Reading articles about the application and purpose about mainly lexers</p>	https://www.youtube.com/watch?v=rLVX_TkGWIM&ab_channel=TobyHo	10/14

Literature Search Parameters:

These searches were performed between (Start Date of reading) and XX/XX/2019.

List of keywords and databases used during this project.

Database/search engine	Keywords	Summary of search
Science Mag	Handwritten Recognition AI Translation Information Density Efficiency + Language Natural Language Processing	Google's new translation software is powered by brainlike artificial intelligence Automatic recognition and transcription of Pitman's handwritten shorthand—An approach to shortforms (I had to consult multiple sources for this article across databases)
Google Scholar	Patents Unicode Input Natural Language Processing Shorthand	English-Arabic Handwritten Character Recognition using Convolutional Neural Networks Method for computer-assisted translation Phonetic-based text input method
Google	Character recognition Machine Learning Gregg Pitman Shorthand	Statistical review of Online/Offline Gregg Shorthand Recognition using CANN and BP - A Comparative analysis Segmentation and recognition of phonetic features in handwritten Pitman shorthand

Article #1 Notes: New Translation Software

Article notes should be on separate sheets

Source Title	<p>Google's new translation software is powered by brainlike artificial intelligence</p> <p>https://patents.google.com/patent/US8200475B2/en.</p>
Source citation (APA Format)	<p>MatacicSep. 27, C., 2016, & Pm, 2:45. (2016, September 27). <i>Google's new translation software is powered by brainlike artificial intelligence</i>. Science AAAS.</p> <p>https://www.sciencemag.org/news/2016/09/google-s-new-translation-software-powered-brainlike-artificial-intelligence</p>
Original URL	https://www.sciencemag.org/news/2016/09/google-s-new-translation-software-powered-brainlike-artificial-intelligence
Source type	Digital News
Keywords	Technology, AI, Translation
Summary of key points (include methodology)	<p>A team from Google Mountain View, California led by Quoc Le used a new deep learning algorithm to reduce errors by finding the nuances in languages not explicitly stated. By using these sensitive systems, they were able to find the small, illusive details of language. This article is also a summary on how translation is being made easier through a process called vector processing. Vectors are these relations between words that cannot be stated objectively. By asserting the surmounting task of locating and dealing with each minor linguistic anomaly, data scientists can make difficult tasks like translation easier.</p>
Research Question/Problem/Need	How can digital translators be made more accurate?
Important Figures	<p>It was 58% more accurate at translating English into Chinese, and 87% more accurate at translating English into Spanish</p> <p>Vectors: 2.5 billion sentence pairs for English and French; 500 million for English and Chinese</p> <p>reduces translation errors by up to 87%. "This ... demonstrates</p>

	<p>like never before the power of neural machine translation,” says Yoshua Bengio</p> <p>The new method, reported today on the preprint server arXiv, uses a total of 16 processors to first transform words into a value known as a vector</p> <p>A lot of the inspiration for this paper came from [the fields of] speech and computer vision,” says Thang Luong, a graduate at Stanford University in Palo Alto, California</p>
Notes	<p>There is an ever-growing shift away from the rough idea and towards one-to-one translation. Part of this movement includes the new deep learning algorithm developed by Quoc Le and his team at Google. This part of deep learning is referred to as neural machine translation. This is a series of several layers of processors working together in tandem. The fact that translating language requires deeper cognitive abilities is a debated topic. This was proven due to the old mindset of splicing the sentence which simply did not work sometimes. By using vectors to connect related words and ideas however, they were able to consistently remove the majority of translation errors. This hints at further growth in this field in the nearby future.</p>
Cited references to follow up on	<p>https://arxiv.org/pdf/1609.08144v1.pdf</p> <p>https://www.sciencemag.org/news/2016/01/huge-leap-forward-computer-mimics-human-brain-beats-professional-game-go</p>
Follow up Questions	<p>How can we track vectors?</p> <p>What are some semantic ideas and details that were unable to be picked up with the DL system but not my foresight?</p> <p>Can this be combined with sentence splicing in any way?</p>

Article #2 Notes: Method for computer-assisted translation

Article notes should be on separate sheets

Source Title	Method for computer-assisted translation
Source citation (APA Format) (Mercier, 2001)	Mercier, P. (2001). <i>US20030105621A1—Method for computer-assisted translation—Google Patents.</i> https://patents.google.com/patent/US20030105621A1/en
Original URL	https://patents.google.com/patent/US20030105621A1/en
Source type	Patent
Keywords	Translation, Remote translation, Multiple Input, Translation sequence proposals
Summary of key points (include methodology)	The patent protects a method of integrating several computers and translator servers through a communications server. The benefits of this specific system are that it allows for the target and source language sequence to be backed up and utilized for future use by the database. The system also gives simultaneous access to other users and other more remote translation servers that not only enhances efficiency but also is quite secure. The system tries to be as conservative as possible by trying to compare the source of any translation request with past queries and comparing similarity percentages.
Research Question/Problem/Need	Can we quickly and effectively translate using a variety of programs and computers should be easily accessible to each and every computer with this system?

Important Figures

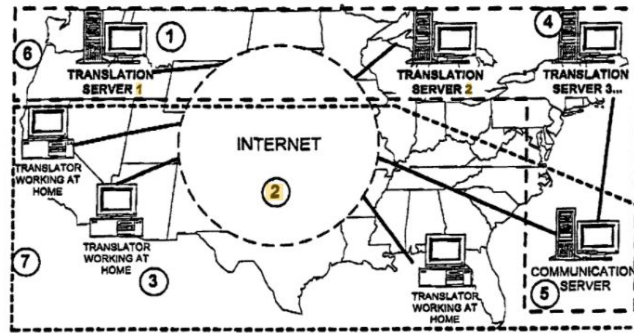


FIG. 3

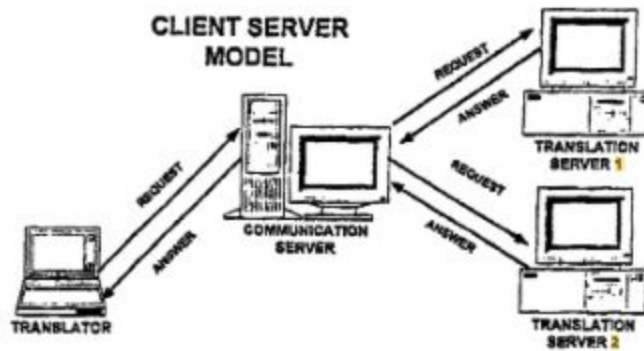


FIG. 8

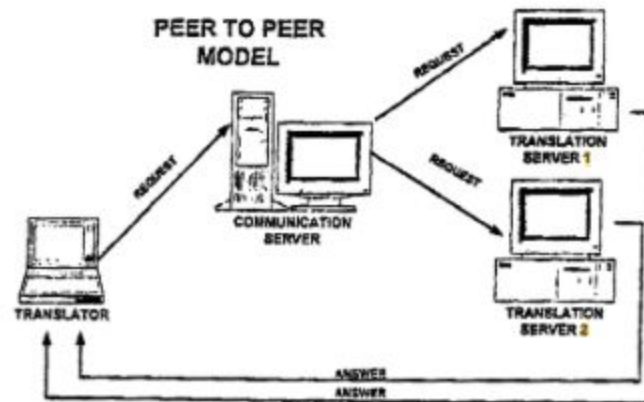


FIG. 9

Notes

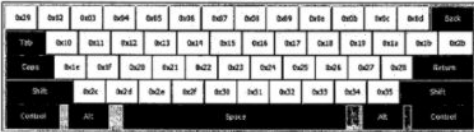
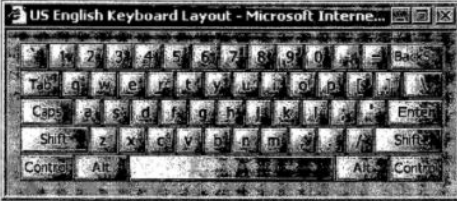

The following patent is a cross-verification/reinforcement language translating system that works by delegating a variety of tasks in a

	<p>specific manner across many computers and servers.</p> <p>Connects with two other computers that mainly help with storage and also previous quote and syntax recalling.</p>
Cited references to follow up on	<p>https://patents.google.com/patent/US4706212A/en</p> <p>https://patents.google.com/patent/US5568383A/en</p> <p>https://patents.google.com/patent/US5848386A/en</p>
Follow up Questions	<p>Why did they choose to take the percentage in common over other methods?</p> <p>How often does this process take?</p> <p>How many servers are usually working together at a given time?</p>

Article #3 Notes: Phonetic-based text input method

Article notes should be on separate sheets

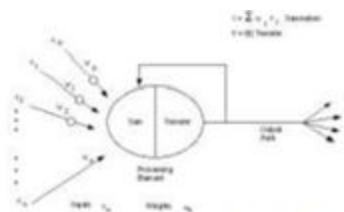
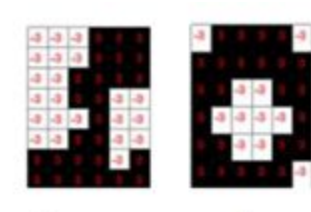
Source Title	Phonetic-based text input method
Source citation (APA Format) (Kotipalli, 2012)	Kotipalli, Krishna V. Phonetic-based text input method. United States US8200475B2, filed February 13, 2004, and issued June 12, 2012. https://patents.google.com/patent/US8200475B2/en .
Original URL	https://patents.google.com/patent/US8200475B2/en
Source type	Patent
Keywords	Character encoding, phonetics, text input, keyboard layout, phonetic key
Summary of key points (include methodology)	In order for one to accurately map the sounds in a certain language so that they can be expressed in other scripts and languages, one must use this phonetic based string layout. The system works by first taking a word, character, or phrase from a target language and giving it a specific IPA designation along with certain unicode IDs. From here, one can find the corresponding layout in the target language. The patent also protects the layout of the keyboard used for this system.
Research Question/Problem/ Need	Can we develop a way of inputting a universal phonetic for sounds in one language to be accurately represented in another?
Important Figures	<pre> graph LR A[PHONETIC INPUT STRING] -- IN --> B[PHONETIC MAPPING ENGINE] B -- OUT --> C[UNICODE OUTPUT] D[PHONETIC MAPPING SCHEMES] --- E[~] E --- B </pre>

	<p style="text-align: center;">FIG. 3</p>  <p style="text-align: center;"><i>Scan codes for U.S. Keyboard hardware (Fig. 3a)</i></p>  <p style="text-align: center;"><i>US English Keyboard Layout (Fig. 3b)</i></p>  <p style="text-align: center;"><i>Hindi Traditional Keyboard Layout (Fig. 3c)</i></p>
<p>Notes</p>	<p>Source Language >> Unicode >> Phonetic String >> Analysis >> Unicode >> Target Language</p> <p>The phonetic mapping engine has a fixed scheme that it uses to concisely and reliably produce results.</p> <p>This is intended as a reusable program on many different platforms including stationary and mobile for many different environments. It is also applicable to word processing programs.</p> <p>The system also has a series of three large processors that first work together to double check the work that works with different information for example Computer 110 works primarily on the backend.</p>
<p>Cited references to follow up on</p>	<p>https://patents.google.com/patent/US4731735A/en https://patents.google.com/patent/US5047932A/en https://patents.google.com/patent/US5136504A/en https://patents.google.com/patent/US5243519A/en</p>
<p>Follow up Questions</p>	<p>What steps did they take to streamline the process? At what points did the program think it was most crucial to double check its work? How could they take frequency into account when assigning IDs in their own system before Unicode?</p>

Article #4 Notes: Statistical review of Online/Offline Gregg Shorthand

Recognition using CANN and BP - A Comparative analysis

Article notes should be on separate sheets

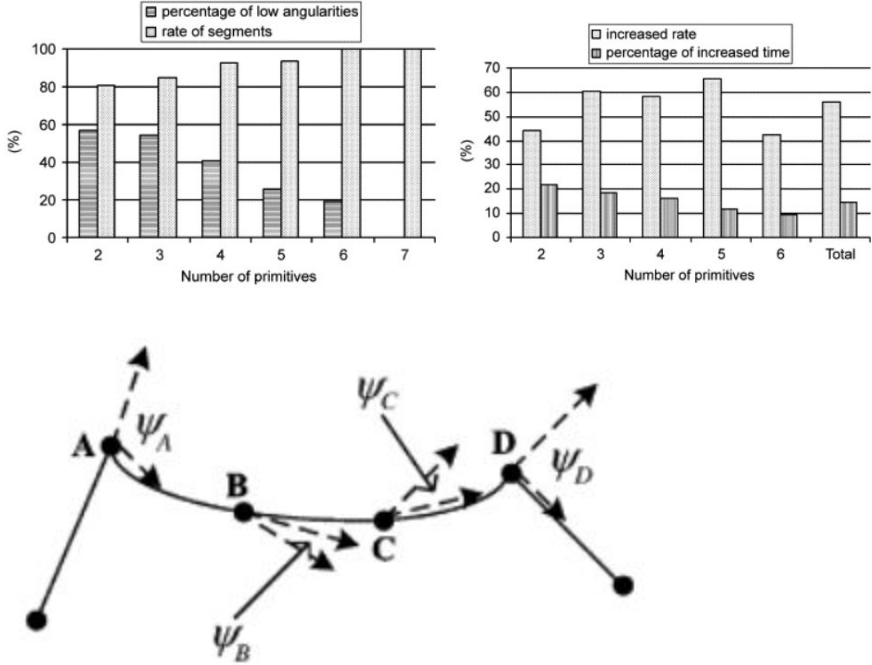
Source Title	Statistical review of Online/Offline Gregg Shorthand Recognition using CANN and BP - A Comparative analysis
Source citation (APA Format)	R. Rajasekaran, Dr. K. R. (2014). Statistical review of Online/Offline Gregg Shorthand Recognition using CANN and BP - A Comparative analysis. <i>IJAIST</i> , 24(24), 13.
Original URL	https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.672.2819&rep=rep1&type=pdf
Source type	Journal Article
Keywords	Handwriting recognition, DIU, machine learning, Gregg Shorthand
Summary of key points (include methodology)	<p>The engineers behind this project decided to use two artificial networks both offline and online and compare them in a variety of elements such as computing time, computing power needed, and accuracy. While the online and offline processes differed, they both followed this general outline shown in Figure 1A.</p> <p>A weighted matrix system categorized every pixel in a 32 x 32 area into either black or white. 32 x 32 was chosen for its reliability yet is small enough to efficiently work with. A model for this matrix would have to be prepped but after its filtering, softening, sharpening, or embossing is done it would look like this (see Figure 7A)</p>
Research Question/Problem/Need	There is a need to be able to quickly and accurately translate from handwritten Gregg Shorthand to digital text with minimal human oversight.
Important Figures	 <p>Figure 1 A Basic Artificial Neuron.</p>  <p>Figure 7 a. Weight Matrix W for Character 'a' b. Weight Matrix W for Character 'aback'</p>

Notes	<p>The p-value of 0.002 in the Lilliefors's test for Normality rejects the null hypothesis for the effectiveness of the online and offline processes.</p> <p>The p-value of 0.711 in the Jarque-Bera test which demonstrates neither online nor offline yields an advantage.</p>
Cited references to follow up on	<p>http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.259.4151&rep=rep1&type=pdf</p>
Follow up Questions	<ul style="list-style-type: none">- Can this process be generalized to other shorthands?- What characteristics of a shorthand make it easy for the computer to discern?

Article #5 Notes: Segmentation and recognition of phonetic features in handwritten Pitman shorthand

Article notes should be on separate sheets

Source Title	Segmentation and recognition of phonetic features in handwritten Pitman shorthand
Source citation (APA Format)	Ma, Y., Leedham, G., Higgins, C., & Htwe, S. M. (2008). Segmentation and recognition of phonetic features in handwritten Pitman shorthand. <i>Pattern Recognition</i> , 41(4), 1280–1294. https://doi.org/10.1016/j.patcog.2007.10.014
Original URL	https://www.sciencedirect.com/science/article/pii/S0031320307004426
Source type	Journal Article
Keywords	Pitman shorthand, Vocalized outline, Shortform, Segmentation, Classification
Summary of key points (include methodology)	The article started by addressing the fact that in this day and age shorthands are important and relevant due to their usage in digital data entry displays. Pitman's wpm rate makes it the ideal tool to use for the job. First, the article works on analyzing the patterns within the strokes of Pitman Shorthand. It marks the change of direction and also the order to check how many times the direction of writing changes. From there, the samples must be segmented so that the flow of different characters combined with each other can be realized and to prep for the Box Model. This Box Model is a helpful tool for AI to break up sensing objects or features in visual data. From here the computer represents the strokes as a series of polar functions and applies the summative vectors and weighted matrices approach standard in Machine Learning
Research Question/Problem/Need	How do certain features of Pitman shorthand enable it to be understood more or faster to write?

Important Figures	
Notes	<p>Experiments on a set of 1127 consonant outlines, 2039 vowels and diphthongs and 841 shortforms have shown that the approaches achieved 75.33%, 96.86% and 91.86% correct recognition accuracy.</p> <p>An experiment on 461 outlines each containing one smooth junction shows that, with the new proposed rule, recognition rate was improved by 55.88% (from 37.53% to 93.41%) at the cost of 14.42% increase in writing time.</p>
Cited references to follow up on	<p>Article Google Scholar Google Scholar</p>
Follow up Questions	<p>How strongly are the intermediate angles related to total sharp movements?</p> <p>How experienced were the professionals?</p> <p>Why was the process of seeing that a sample was legible selected?</p>

Article #6 Notes: Automatic recognition and transcription of Pitman's handwritten shorthand—An approach to shortforms

Article notes should be on separate sheets

Source Title	Automatic recognition and transcription of Pitman's handwritten shorthand—An approach to shortforms
Source citation (APA Format)	<p><i>Automatic recognition and transcription of Pitman's handwritten shorthand—An approach to shortforms—ScienceDirect.</i></p> <p>(n.d.). Retrieved October 15, 2020, from https://www.sciencedirect.com/science/article/abs/pii/0031320387900082</p>
Original URL	https://www.sciencedirect.com/science/article/pii/0031320387900082
Source type	Journal Article
Keywords	Handwriting, Character recognition, Template match, Dynamic programming, Pitman's shorthand
Summary of key points (include methodology)	There are a series of unofficial yet extremely helpful and efficient changes to Pitman shorthand including new abbreviations and the changes of certain vowel clusters. To track these changes and to measure whether or not these changes are significant, they trained an algorithm to read the shorthand and also evaluate its efficiency to write and legibility.
Research Question/Problem/Need	How can we identify Pitman shorthand by analyzing how it is digitally represented pixelated?

Important Figures

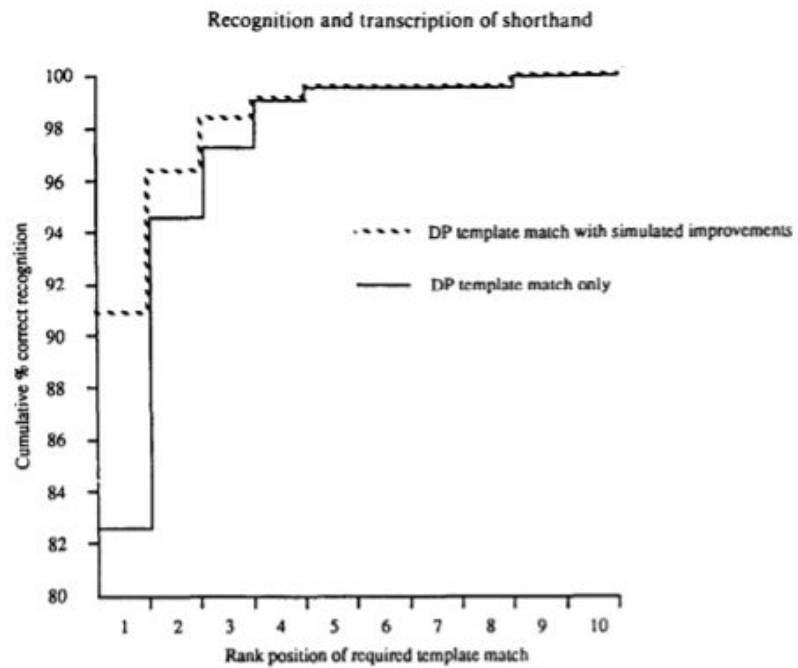
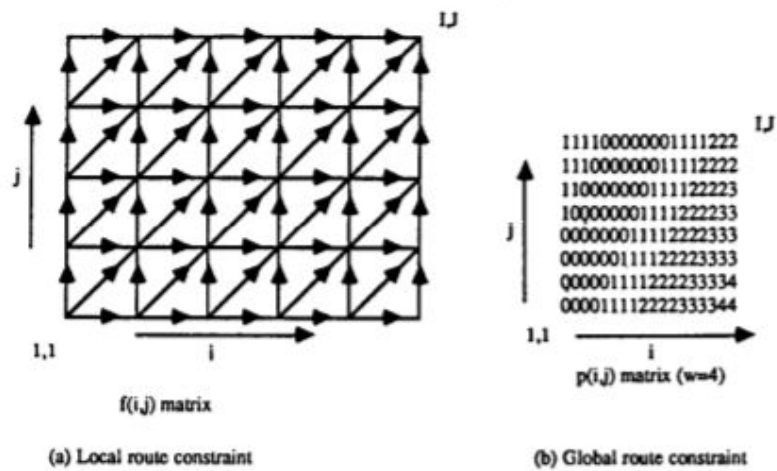


Fig. 6. Cumulative recognition performance for template match.

Notes

They took a similar approach to many matrix based vector analysis procedures as other Handwritten Shorthand Recognition systems. However, the caveat is that in this system it uses a graded scale to gauge it more in detail than just zeros and ones. However, this will lead to dealing with heuristics and thus the workload is heavier on the Machine Learning side.

Cited references to follow up on	<p>Segmentation and recognition of phonetic features in handwritten Pitman shorthand</p> <p>Segmentation and recognition of handwritten pitman shorthand outlines using an interactive heuristic search</p> <p>Evaluation of dynamic programming algorithms for the recognition of shortforms in Pitman's shorthand</p>
Follow up Questions	<p>Why didn't this approach use Polar coordinates and instead focused more on the "jaggedness" of the strokes?</p> <p>Is there a reason these particular amendments to the shorthand were selected for this project?</p>

Article #7 Notes: Ontology-based parser for natural language processing

Article notes should be on separate sheets

Source Title	Ontology-based parser for natural language processing
Source citation (APA Format)	<i>US7027974B1—Ontology-based parser for natural language processing—Google Patents.</i> (n.d.). Retrieved October 15, 2020, from https://patents.google.com/patent/US7027974B1/en
Original URL	https://patents.google.com/patent/US7027974B1/en
Source type	Patent
Keywords	Lexers, Parsers, Natural Language Processing, predicate-argument
Summary of key points (include methodology)	The natural language processing system aims to loom for specific keywords or types of keywords that may give hints to the literary subject and predicates of sentences. By identifying these attributes, the program can turn these into a format that is more easily readable by search engine and more robotic operators. Ultimately a lexer and parser will need to be used to deal with this new fundamental type of data.
Research Question/Problem/Need	Search engines, word processors, and other technological tools sometimes have a difficult time understanding human language and thus could stand to benefit from having an algorithm streamline a user's search phrases.

Important Figures

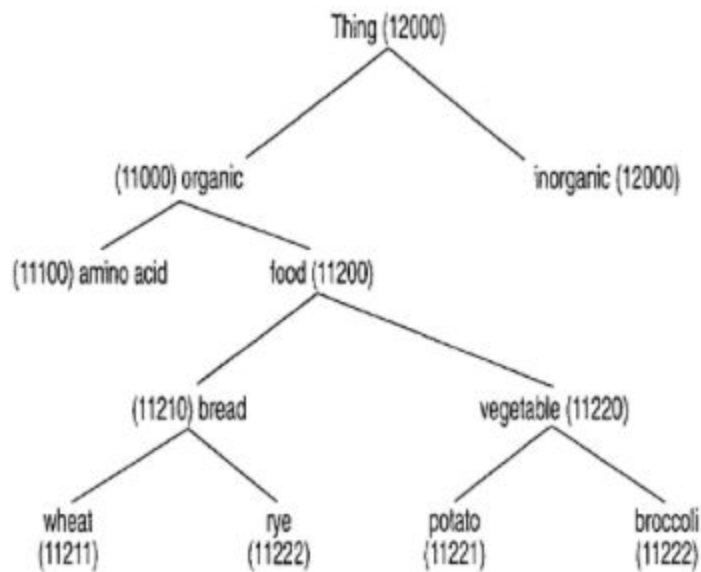
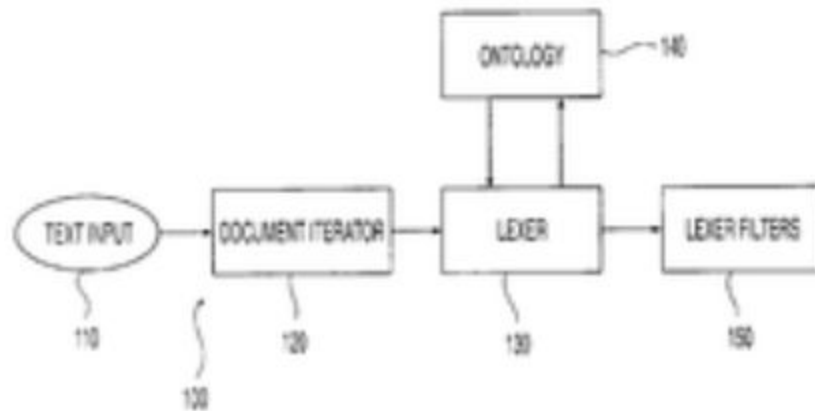


Fig. 7

Notes

The system uses a tagging system in conjunction with a database to sort the words into different classes in a way. By using this method, it starts to create percentages of how likely certain outcomes are to happen. This same sort of tagging technique can perhaps be applied in my project.

Cited references to follow up on

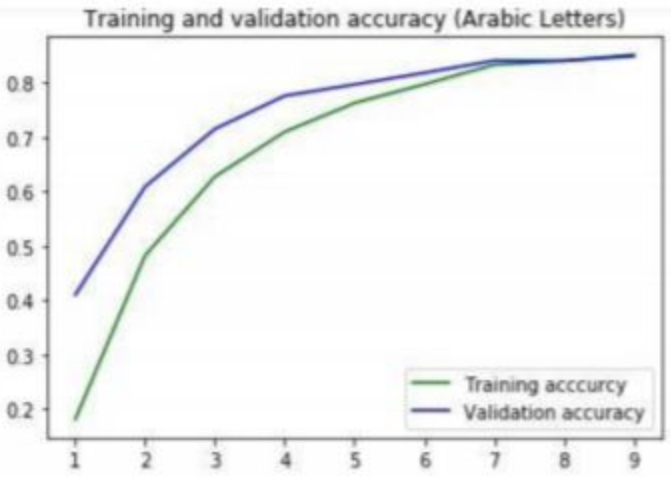
<https://patents.google.com/patent/US4270182A/en>
<https://patents.google.com/patent/US4864502A/en>

	https://patents.google.com/patent/US4887212A/en
Follow up Questions	<p>How can this process be further customized for specific search engines?</p> <p>Can this system be directly implemented into search engines?</p>

Article #8 Notes: English-Arabic Handwritten Character Recognition using Convolutional Neural Networks

Article notes should be on separate sheets

Source Title	English-Arabic Handwritten Character Recognition using Convolutional Neural Networks
Source citation (APA Format)	Mohamed, A., & Rohm-Ensing, E. (n.d.). <i>English-Arabic Handwritten Character Recognition using Convolutional Neural Networks</i> . 7.
Original URL	http://jmgphd.com/wp-content/uploads/2019/06/handwritten_ocr_report.pdf
Source type	Journal Article
Keywords	HCR (Handwritten Character Recognition), CNN (Convolutional Neural Networks), and image vectors
Summary of key points (include methodology)	The HCR System aimed to create an algorithm that could accurately read the English characters and numerals that the researchers derived from the digital font data set. They did this by using a multiple layer approach with image vectors.
Research Question/Problem/Need	This is a need for an algorithm to parse through written english arabic (as to not be overfitted to a particular language) and determine its meaning.

<p>Important Figures</p>	 <p>Figure 6: Chart of accuracy increase over epochs</p> <table border="1" data-bbox="592 231 1258 714"> <thead> <tr> <th>Epoch</th> <th>Training accuracy</th> <th>Validation accuracy</th> </tr> </thead> <tbody> <tr><td>1</td><td>0.18</td><td>0.42</td></tr> <tr><td>2</td><td>0.50</td><td>0.62</td></tr> <tr><td>3</td><td>0.63</td><td>0.72</td></tr> <tr><td>4</td><td>0.72</td><td>0.78</td></tr> <tr><td>5</td><td>0.77</td><td>0.80</td></tr> <tr><td>6</td><td>0.80</td><td>0.82</td></tr> <tr><td>7</td><td>0.83</td><td>0.84</td></tr> <tr><td>8</td><td>0.84</td><td>0.85</td></tr> <tr><td>9</td><td>0.85</td><td>0.85</td></tr> </tbody> </table>	Epoch	Training accuracy	Validation accuracy	1	0.18	0.42	2	0.50	0.62	3	0.63	0.72	4	0.72	0.78	5	0.77	0.80	6	0.80	0.82	7	0.83	0.84	8	0.84	0.85	9	0.85	0.85
Epoch	Training accuracy	Validation accuracy																													
1	0.18	0.42																													
2	0.50	0.62																													
3	0.63	0.72																													
4	0.72	0.78																													
5	0.77	0.80																													
6	0.80	0.82																													
7	0.83	0.84																													
8	0.84	0.85																													
9	0.85	0.85																													
<p>Notes</p>	<p>For my project, this can serve as a very important tool for general language recognition because it is generalized to many different types of writing. For example even with my shorthand I developed, there are aspects of english and shorthand that are both used and thus a more generalized solution may help.</p>																														
<p>Cited references to follow up on</p>	<p>N. Das, A. F. Mollah, S. Saha, and S. S. Haque. Handwritten arabic numeral recognition using a multi layer perceptron. arXiv preprint arXiv:1003.1891, 2010.</p> <p>W. Rawat and Z. Wang. Deep convolutional neural networks for image classification: A comprehensive review. Neural computation, 29(9):2352–2449, 2017.</p>																														
<p>Follow up Questions</p>	<p>How does this account for human error?</p> <p>In what ways can this same system be repurposed for other languages in general?</p> <p>There are definitely databases on kaggle for different handwriting samples, so why didn't the researchers consider using those for training?</p>																														

Article #9 Notes: How File Compression Works

Article notes should be on separate sheets

Source Title	How File Compression Works
Source citation (APA Format)	<i>How File Compression Works</i> <i>HowStuffWorks</i> . (n.d.). Retrieved October 15, 2020, from https://computer.howstuffworks.com/file-compression.htm
Original URL	https://computer.howstuffworks.com/file-compression1.htm
Source type	News Article
Keywords	File Compression, Zip Files, LZ Compression
Summary of key points (include methodology)	This article focuses on how applications like WinZip or Stuffit work by targeting the repetitive nature of language. This article then went through how such a program would cut down the size of a famous Kennedy quote: <i>"Ask not what your country can do for you -- ask what you can do for your country."</i> First, it tackles this problem by assigning each distinct word a certain numerical value. Then after the reader grasps this basic concept, it moves onto simply identifying any string of characters (including spaces) in order to maximize efficiency. Finally, the article ends with mentioning how color can be accommodated to match other already established pixels without compromising resolution.
Research Question/Problem/Need	How to save file space with a Zip File? How do they work?
Important Figures	<p>The quote has 17 words, made up of 61 letters, 16 spaces, one dash and one period. If each letter, space or punctuation mark takes up one unit of memory, we get a total file size of 79 units. To get the file size down, we need to look for redundancies.</p> <p>The sentence now takes up 18 units of memory, and our dictionary takes up 41 units. So we've compressed the total file</p>

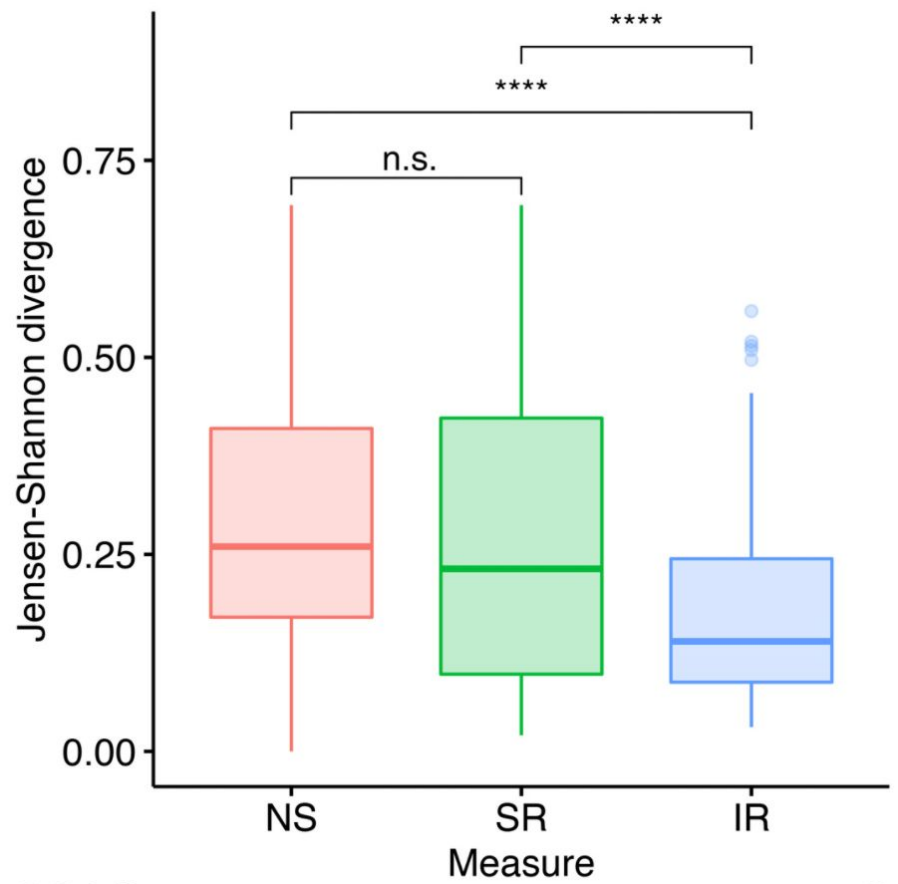
	size from 79 units to 59 units!
Notes	For my project, I plan to use this same sort of idea once the actual bulk of translating my text file is done and dealt with. While I do not think the last part is applicable, I do however definitely feel that the rest of it can definitely be applied. I am still considering whether or not I have to make a separate dictionary for each translation or should I just make one big one. I will also have to test whether or not it even matters enough in the end thanks to this new LZ adaptive dictionary-based algorithm.
Cited references to follow up on	How does a zip drive store so much more data than a floppy drive? Data-Compression.com
Follow up Questions	Is there a way we can take advantage of the repetitiveness of certain specific individuals? How commonplace is LZ Compression?

Article #10 Notes: Different Languages, Similar Encoding Efficiency

Article notes should be on separate sheets

Source Title	Different Languages, Similar Encoding Efficiency
Source citation (APA Format)	<i>Different languages, similar encoding efficiency: Comparable information rates across the human communicative niche</i> <i>Science Advances</i> . (n.d.). Retrieved October 15, 2020, from https://advances.sciencemag.org/content/5/9/eaaw2594
Original URL	https://advances.sciencemag.org/content/5/9/eaaw2594
Source type	Encoding, Efficiency, Information Density
Keywords	Languages, Information Density, Linguistic speed
Summary of key points (include methodology)	This article focuses on drawing conclusions between 17 languages across 9 language families to demonstrate the relationships between speech, information density, number of syllables or etc. The project had seemingly taken into consideration several variables that could have contributed to different rates in information and speech such as sex and language family. After a number of mathematical tests and syntagmatic analysis, Figure 1 showed that languages varied greatly in speech while they were remarkably similar in Information rate. Information rate was defined in the experiment as the product of the speech rate and information density. Figure 2 showed speech versus information on a 2 dimensional plane with vertical lines. Figure 3 showed that the box and whisker plots of all linguistic aspects and showed that information rate has smaller interquartile ranges.
Research Question/Problem/Need	Despite the differences in writing and speaking speed of languages, do all languages communicate information at roughly the same rate?

Important Figures



<p>Notes</p>	<p>For my project, it is useful to note that Vietnamese did particularly well in this study and thus I will have to consider the possibility of adding the equivalent of a tone in written language. I can also gather from this that although languages in general have a consistent information rate, they have different densities. This gives hope that the information density can be further pushed for this project. I hope to take advantage of this for my project.</p>
<p>Cited references to follow up on</p>	<p>S. C. Levinson, Turn-taking in human communication – Origins and implications for language processing. <i>Trends Cogn. Sci.</i> 20, 6–14 (2016).CrossRefPubMedGoogle Scholar</p> <p>M. Dingemanse, S. G. Roberts, J. Baranova, J. Blythe, P. Drew, S. Floyd, R. S. Gisladdottir, K. H. Kendrick, S. C. Levinson, E. Manrique, G. Rossi, N. J. Enfield, Universal principles in the repair of communication problems. <i>PLOS ONE</i> 10, e0136100 (2015).CrossRefPubMedGoogle Scholar</p>
<p>Follow up Questions</p>	<p>Are there more factors to consider in this experiment including</p>

vocabulary size of the language?

How are different phonemes and especially linguistic stress taken into account?

Article #11 Notes: Fast Compression Algorithm for UNICODE

Article notes should be on separate sheets

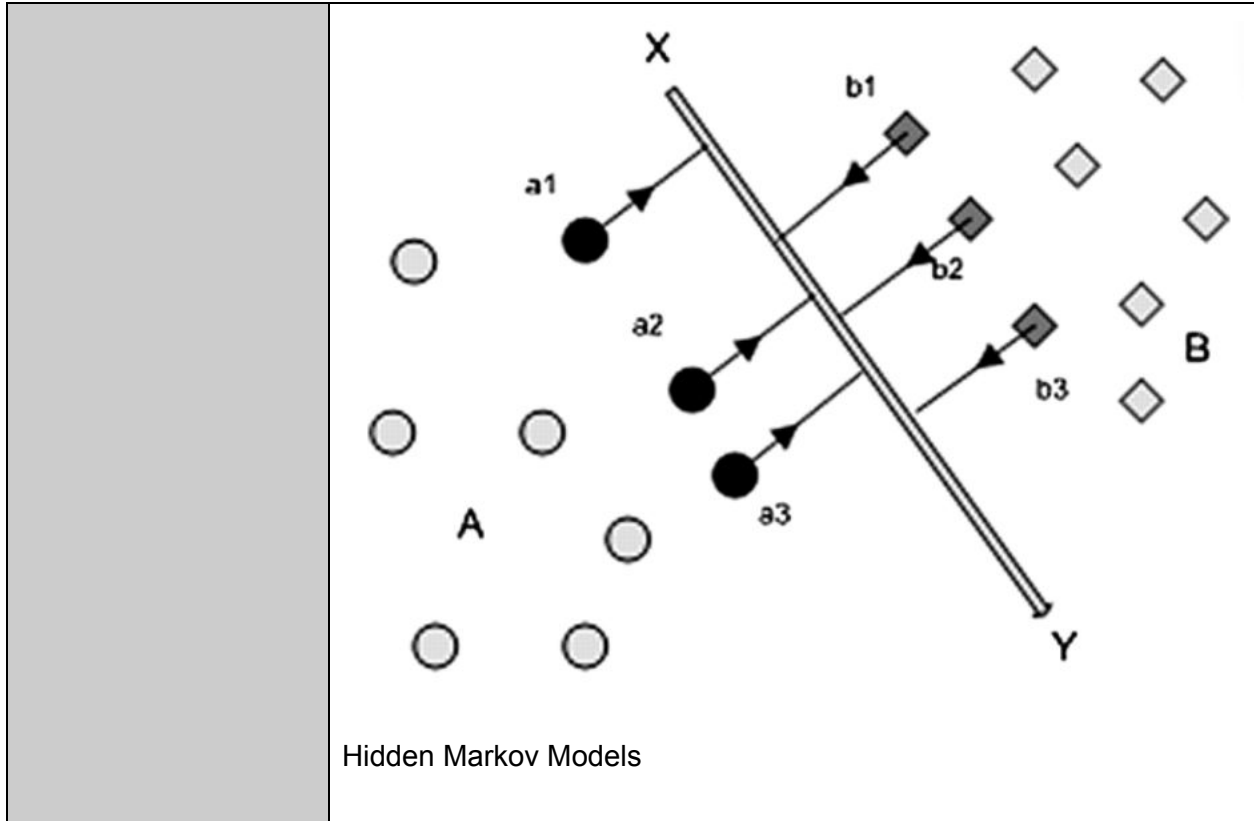
Source Title	Fast Compression Algorithm for UNICODE
Source citation (APA Format)	<i>Fast Compression Algorithm for UNICODE Text.</i> (n.d.). Retrieved October 15, 2020, from http://unicode.org/notes/tn31/
Original URL	http://unicode.org/notes/tn31/
Source type	Technical Note
Keywords	Unicode, text compression,
Summary of key points (include methodology)	This article is a more in depth look at the compression process within Unicode itself. This is similar to the first article I read on zip files but this uses slightly different methods. The first method it touches upon is Lempel-Ziv compression that takes previous segments, which may be beneficial depending on how repetitive the sample is. Then they also use a hash table which is a method almost identical to zip files. However it also goes into detail about the storage space of the characters. Compression and decompression is nothing but a matter of assigning values to different segments of the text. There are also some same algorithms and times (in microseconds) for how long this compression and etc should take.
Research Question/Problem/Need	How does Unicode use text compression to make its encoding more efficient?

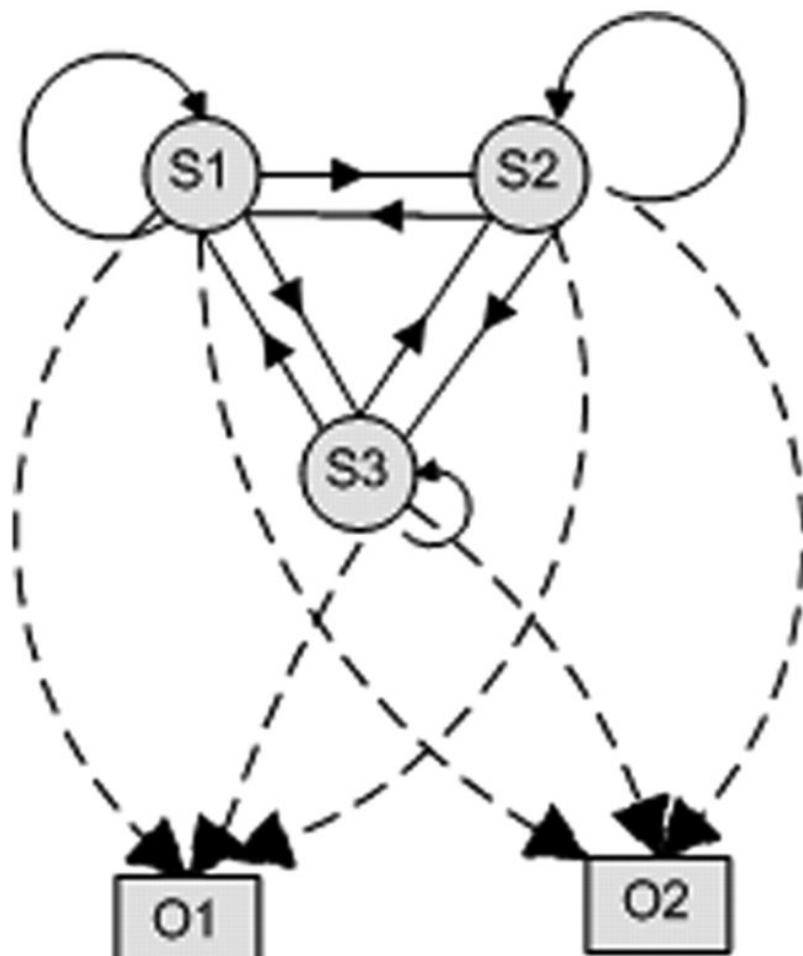
Important Figures	<table border="1"> <thead> <tr> <th></th> <th colspan="2">Unicode compression</th> <th colspan="2">zlib, level 1</th> <th colspan="2">zlib, level 9</th> </tr> <tr> <th></th> <th>compressed size [B]</th> <th>time [μs]</th> <th>compressed size [B]</th> <th>time [μs]</th> <th>compressed size [B]</th> <th>time [μs]</th> </tr> </thead> <tbody> <tr> <td>English (1014 B)</td> <td>560</td> <td>0.18</td> <td>405</td> <td>2.8</td> <td>377</td> <td>3.2</td> </tr> <tr> <td>Russian (982 B)</td> <td>618</td> <td>0.19</td> <td>464</td> <td>2.9</td> <td>443</td> <td>3.3</td> </tr> <tr> <td>Chinese (1018 B)</td> <td>841</td> <td>0.23</td> <td>726</td> <td>3.8</td> <td>719</td> <td>3.8</td> </tr> </tbody> </table>							Unicode compression		zlib, level 1		zlib, level 9			compressed size [B]	time [μs]	compressed size [B]	time [μs]	compressed size [B]	time [μs]	English (1014 B)	560	0.18	405	2.8	377	3.2	Russian (982 B)	618	0.19	464	2.9	443	3.3	Chinese (1018 B)	841	0.23	726	3.8	719	3.8
		Unicode compression		zlib, level 1		zlib, level 9																																			
		compressed size [B]	time [μs]	compressed size [B]	time [μs]	compressed size [B]	time [μs]																																		
	English (1014 B)	560	0.18	405	2.8	377	3.2																																		
Russian (982 B)	618	0.19	464	2.9	443	3.3																																			
Chinese (1018 B)	841	0.23	726	3.8	719	3.8																																			
<p><i>Fig. 2: We get to the position of the previous occurrence by hashing 'k' and 'e' and looking up the pointer in the dictionary.</i></p>																																									
Notes	<p>For the purpose of my project, I will probably end up using hash tables and maybe Lempel-Ziv compression. It will also be helpful to measure the sizes and times given in the table to the times of my own. Hopefully after putting it through my DSK, I can simply use the idea of the code in my program. Unfortunately the sample code is in C and is optimized for understanding instead of efficiency.</p>																																								
Cited references to follow up on	<p>"What is Unicode" pages from www.unicode.org</p>																																								
Follow up Questions	<p>Can this system be built upon with other compression methods?</p> <p>Is there a way to make the character encoding quicker for more frequent characters?</p>																																								

Article #12 Notes: Natural Language Processing: An introduction

Article notes should be on separate sheets

Source Title	Natural Language Processing: An introduction
Source citation (APA Format)	<i>Natural language processing: An introduction</i> <i>Journal of the American Medical Informatics Association</i> Oxford Academic. (n.d.). Retrieved October 15, 2020, from https://academic.oup.com/jamia/article/18/5/544/829676
Original URL	https://academic.oup.com/jamia/article/18/5/544/829676
Source type	Journal Article
Keywords	Natural language processing, Introduction, clinical NLP, knowledge bases, machine learning, predictive modeling, statistical learning, privacy technology
Summary of key points (include methodology)	This article details the beginnings of NLP and how it essentially started from looking at heuristics and complex lexers and parsers to its new analytical form. Then it lists a few different types of NLP designs that can be generally applied to any NLP problem. The article then wraps up by telling the read about IBM Watson and where NLP is heading.
Research Question/Problem/Need	What are the different ways to tackle NLP using different techniques?
Important Figures	Support Vector Machines





Notes

My algorithm, or at least the referencing in my system, will probably have more backwards propagation and thus will represent the second diagram more. The first diagram deals more with the sorting between two main data types, which while can be applied to the model I propose to build, it is better to get a system that will work more like a function performed on a word.

Cited references to follow up on

Manning C Raghavan P Schuetze H . *Introduction to Information Retrieval*. Cambridge, UK: Cambridge University Press, 2008.

Hutchins W . *The First Public Demonstration of Machine Translation: the Georgetown-IBM System, 7th January 1954*. 2005. <http://www.hutchinsweb.me.uk/GU-IBM-2005.pdf> (accessed 4 Jun 2011).

Chomsky N . Three models for the description of language. *IRE Trans Inf Theory* 1956;2:113–24.

Follow up Questions

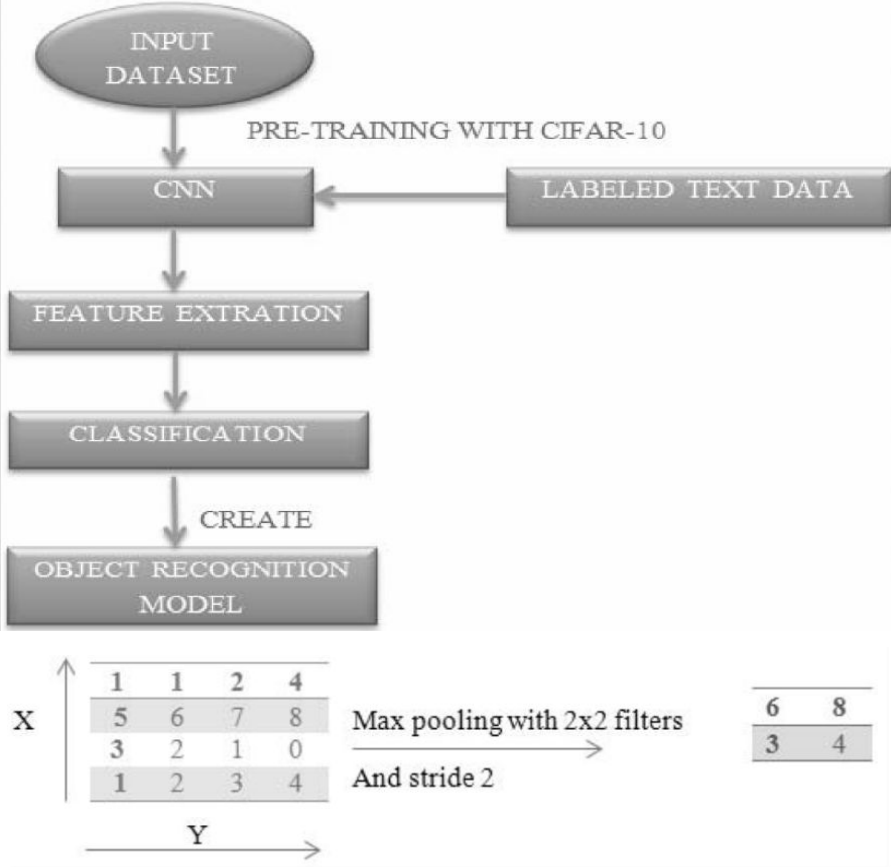
In Figure 4, does the quality of the tasks improve with more intermediate checking, or does it put too much strain on the computer?

Would a Markov model or N-grams model be more suitable for my project?

Article #13 Notes: Object recognition in images using convolutional neural network

Article notes should be on separate sheets

Source Title	Object recognition in images using convolutional neural network
Source citation (APA Format)	Sudarshan, D. (n.d.). <i>Object recognition in images using convolutional neural network—IEEE Conference Publication</i> . Retrieved December 13, 2020, from https://ieeexplore.ieee.org/document/8398912
Original URL	https://ieeexplore.ieee.org/document/8398912
Source type	Journal Article
Keywords	<ul style="list-style-type: none"> • Image recognition • Object recognition • Conferences • Control systems • DVD • Convolutional neural networks
Summary of key points (include methodology)	Object detection from repository of images is challenging task in the area of computer vision and image processing in this work we present object classification and detection using cifar-10 data set with intended classification and detection of airplane images. So we used convolutional neural network on keras with tensorflow support the experimental results shows the time required to train, test and create the model in limited computing system. We train the system with 60,000 images with 25 epochs each epoch is taking 722to760 seconds in training step on tensorflow cpu system. At the end of 25 epochs the training accuracy is 96 percentage and the system can recognition input images based on train model and the output is respective label of images.
Research Question/Problem/	To what degree of accuracy can convolutional neural networks recognize objects in images from the cifar-10 data set using a Keras

<p>Need</p>	<p>library?</p>																				
<p>Important Figures</p>	 <p>The flowchart illustrates the process of creating an object recognition model. It starts with an 'INPUT DATASET' which undergoes 'PRE-TRAINING WITH CIFAR-10' to produce a 'CNN'. This CNN is then trained with 'Labeled Text Data'. The process continues through 'FEATURE EXTRATION', 'CLASSIFICATION', and finally 'CREATE' to produce an 'OBJECT RECOGNITION MODEL'. Below the flowchart, a diagram shows a 4x4 input matrix X with values: <table border="1" style="display: inline-table; margin-right: 20px;"> <tr><td>1</td><td>1</td><td>2</td><td>4</td></tr> <tr><td>5</td><td>6</td><td>7</td><td>8</td></tr> <tr><td>3</td><td>2</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>2</td><td>3</td><td>4</td></tr> </table> This matrix is processed with 'Max pooling with 2x2 filters' and 'And stride 2' to produce a 2x2 output matrix Y with values: <table border="1" style="display: inline-table;"> <tr><td>6</td><td>8</td></tr> <tr><td>3</td><td>4</td></tr> </table> </p> <p style="text-align: center;">Fig1-2x2 max pooling</p>	1	1	2	4	5	6	7	8	3	2	1	0	1	2	3	4	6	8	3	4
1	1	2	4																		
5	6	7	8																		
3	2	1	0																		
1	2	3	4																		
6	8																				
3	4																				
<p>Notes</p>	<p>They used a dataset of about a couple thousand images to train their model using multiple layers of the Keras Library</p> <p>They used an existing dataset but a lot of their results were impacted by how well they resized and changed the resolution of the images and etc</p>																				
<p>Cited references to follow up on</p>	<p>Image Processing and Location based Image Querier(LBIQ) https://ieeexplore.ieee.org/document/9214155</p>																				
<p>Follow up Questions</p>	<p>What are the pros and cons of Max pooling?</p>																				

	How big of a role does Image Preprocessing make here?
--	---

Article #14 Notes: Calamari - A High-Performance Tensorflow-based Deep Learning Package for Optical Character Recognition

Article notes should be on separate sheets

Source Title	Calamari - A High-Performance Tensorflow-based Deep Learning Package for Optical Character Recognition
Source citation (APA Format)	<p>Wick, C., Raul, C., & Puppe, F. (n.d.). [1807.02004]</p> <p><i>Calamari—A High-Performance Tensorflow-based Deep Learning Package for Optical Character Recognition.</i></p> <p>Retrieved December 13, 2020, from https://arxiv.org/abs/1807.02004</p>
Original URL	https://arxiv.org/abs/1807.02004
Source type	Journal Article
Keywords	Tensorflow, Deep Learning, OCR
Summary of key points (include methodology)	<p>Optical Character Recognition (OCR) on contemporary and historical data is still in the focus of many researchers. Especially historical prints require book specific trained OCR models to achieve applicable results (Springmann and Lüdeling, 2016, Reul et al., 2017a). To reduce the human effort for manually annotating ground truth (GT) various techniques such as voting and pretraining have shown to be very efficient (Reul et al., 2018a, Reul et al., 2018b). Calamari is a new open source OCR line recognition software that both uses state-of-the art Deep Neural Networks (DNNs) implemented in Tensorflow and giving native support for techniques such as pretraining and voting. The customizable network architectures constructed of Convolutional Neural Networks (CNNS) and Long-ShortTerm-Memory (LSTM) layers are trained by the so-called Connectionist Temporal Classification (CTC) algorithm of Graves et al. (2006). Optional usage of a GPU drastically reduces the computation times for both training and prediction. We use two different datasets to compare the performance of Calamari to</p>


	OCRopy, OCRopus3, and Tesseract 4.																								
Research Question/Problem/Need	How effective is Calamari, a new open source OCR line recognition software that uses state-of-the art Deep Neural Networks (DNNs) implemented in Tensorflow and gives native support for techniques such as pretraining and voting, do for accuracy when applied to read characters from historical documents?																								
Important Figures	$\text{An examp} \left\{ \begin{array}{c c c} \mathbf{1} & \mathbf{0.8\%} & \mathbf{I} & \mathbf{0.2\%} & \mathbf{L} & \mathbf{0.0\%} \\ 1 & 0.4\% & \mathbf{I} & \mathbf{0.5\%} & \mathbf{L} & \mathbf{0.1\%} \\ 1 & 0.2\% & \mathbf{I} & \mathbf{0.3\%} & \mathbf{L} & \mathbf{0.2\%} \end{array} \right\} e$ <p>Figure 1: An example for the confidence voting algorithm. Each row shows a part of the output of three different voters. When choosing the most frequent top result of each voter (bold) an "I" would be predicted. However, when adding the confidences of each voter, the letter "I" is predicted.</p> <p>Table 4: Average time for training or prediction of a single line of the UW3 dataset. Note that the times measured for OCRopy and Tesseract 4 are on the CPU while Calamari and OCRopy3 run on the GPU. The prediction of OCRopy and Tesseract 4 is evaluated using a single process, using multiple multithreading highly reduces their computation time. The last row was published by Breuel (2017).</p> <table border="1"> <thead> <tr> <th>Model</th> <th>Software</th> <th>Training</th> <th>Prediction</th> </tr> </thead> <tbody> <tr> <td>C, Mp(2x2), C, Mp(2x2), LSTM(200)</td> <td>Calamari</td> <td>8 ms</td> <td>3 ms</td> </tr> <tr> <td>LSTM(200)</td> <td>OCRopy</td> <td>850 ms</td> <td>330 ms</td> </tr> <tr> <td>C, Mp(2x2), C, Mp(2x2), LSTM(200)</td> <td>Tesseract 4</td> <td>1200 ms</td> <td>550 ms</td> </tr> <tr> <td>C, Mp(2x2), C, Mp(2x2), LSTM(200)</td> <td>OCRopy3</td> <td>10 ms</td> <td>7 ms</td> </tr> <tr> <td>C, Mp(1x2), C, Mp(1x2), LSTM(100)</td> <td>OCRopy3</td> <td>-</td> <td>10 ms</td> </tr> </tbody> </table>	Model	Software	Training	Prediction	C, Mp(2x2), C, Mp(2x2), LSTM(200)	Calamari	8 ms	3 ms	LSTM(200)	OCRopy	850 ms	330 ms	C, Mp(2x2), C, Mp(2x2), LSTM(200)	Tesseract 4	1200 ms	550 ms	C, Mp(2x2), C, Mp(2x2), LSTM(200)	OCRopy3	10 ms	7 ms	C, Mp(1x2), C, Mp(1x2), LSTM(100)	OCRopy3	-	10 ms
Model	Software	Training	Prediction																						
C, Mp(2x2), C, Mp(2x2), LSTM(200)	Calamari	8 ms	3 ms																						
LSTM(200)	OCRopy	850 ms	330 ms																						
C, Mp(2x2), C, Mp(2x2), LSTM(200)	Tesseract 4	1200 ms	550 ms																						
C, Mp(2x2), C, Mp(2x2), LSTM(200)	OCRopy3	10 ms	7 ms																						
C, Mp(1x2), C, Mp(1x2), LSTM(100)	OCRopy3	-	10 ms																						
Notes	<p>Calamari reaches a Character Error Rate (CER) of 0.11% on the UW3 dataset written in modern English and 0.18% on the DTA19 dataset written in German Fraktur, which considerably outperforms the results of the existing softwares.</p> <p>The default network consists of two pairs of convolution and pooling layers with a ReLU-Activation function, a following bidirectional LSTM layer, and an output layer which predicts probabilities for the alphabet. Both convolution layers have a kernel size of 3×3 with zero padding of one pixel. The first layer has 64 filters, the second layer 128 filters. The pooling layers implement MaxPooling with a kernel size and stride of 2×2. Each LSTM layer (forwards and backwards) has 200 hidden states that are concatenated to serve as input for the final output layer. During training we apply dropout (Srivastava et al., 2014) with a rate of 0.5 to the concatenated LSTM output to prevent overfitting. The loss is computed by the CTC-Algorithm given the output layer's predictions and the GT label</p>																								

	sequence.
Cited references to follow up on	<p>BREUEL, T. M. (2008) The OCRopus open source OCR system. In: Document Recognition and Retrieval XV. International Society for Optics and Photonics, Vol. 6815, p. 68150F. BREUEL, T. M. (2017) High performance text recognition using a hybrid convolutionallstm implementation. In: Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on. IEEE, Vol. 1, pp. 11-16. BREUEL, T. M., et al. (2013) High-performance OCR for printed English and Fraktur using LSTM networks. In: Document Analysis and Recognition (ICDAR), 2013 12th International Conference on. IEEE, pp. 683-687.</p>
Follow up Questions	<p>Why did they have such a different way of presenting their confusion matrix?</p> <p>Their confidence voting seemed to work well but how was it taken into consideration with the confusion matrix?</p>

Article #15 Notes: Fooling OCR Systems with Adversarial Text Images

Article notes should be on separate sheets

Source Title	Fooling OCR Systems with Adversarial Text Images
Source citation (APA Format)	Song, C., & Shmatikov, V. (n.d.). [1802.05385] <i>Fooling OCR Systems with Adversarial Text Images</i> . Retrieved December 13, 2020, from https://arxiv.org/abs/1802.05385
Original URL	https://arxiv.org/abs/1802.05385
Source type	Journal Article
Keywords	OCR, adversarial text images, semantic filtering
Summary of key points (include methodology)	We demonstrate that state-of-the-art optical character recognition (OCR) based on deep learning is vulnerable to adversarial images. Minor modifications to images of printed text, which do not change the meaning of the text to a human reader, cause the OCR system to "recognize" a different text where certain words chosen by the adversary are replaced by their semantic opposites. This completely changes the meaning of the output produced by the OCR system and by the NLP applications that use OCR for preprocessing their inputs.
Research Question/Problem/Need	To what extent can OCR be fooled into lowering accuracies and how can we learn from these weaknesses?

Important Figures	<table border="1"> <thead> <tr> <th>Font</th> <th>Clean acc</th> <th>Target acc</th> <th>Rejected</th> <th>Avg L_2</th> </tr> </thead> <tbody> <tr><td>Arial</td><td>100.00%</td><td>94.17%</td><td>0.00%</td><td>3.10</td></tr> <tr><td>Arial B</td><td>100.00%</td><td>96.67%</td><td>0.00%</td><td>3.27</td></tr> <tr><td>Arial BI</td><td>100.00%</td><td>95.00%</td><td>0.00%</td><td>3.14</td></tr> <tr><td>Arial I</td><td>99.17%</td><td>94.17%</td><td>0.83%</td><td>2.90</td></tr> <tr><td>Courier</td><td>99.17%</td><td>79.17%</td><td>0.00%</td><td>2.73</td></tr> <tr><td>Courier B</td><td>100.00%</td><td>96.67%</td><td>0.00%</td><td>3.36</td></tr> <tr><td>Courier BI</td><td>100.00%</td><td>93.33%</td><td>0.00%</td><td>3.23</td></tr> <tr><td>Courier I</td><td>99.17%</td><td>93.33%</td><td>0.83%</td><td>2.78</td></tr> <tr><td>Georgia</td><td>100.00%</td><td>91.67%</td><td>0.83%</td><td>2.94</td></tr> <tr><td>Georgia B</td><td>100.00%</td><td>94.17%</td><td>0.83%</td><td>3.18</td></tr> <tr><td>Georgia BI</td><td>100.00%</td><td>92.50%</td><td>0.83%</td><td>3.03</td></tr> <tr><td>Georgia I</td><td>100.00%</td><td>95.00%</td><td>0.00%</td><td>2.99</td></tr> <tr><td>Times NR</td><td>100.00%</td><td>88.33%</td><td>0.00%</td><td>2.90</td></tr> <tr><td>Times NR B</td><td>100.00%</td><td>91.67%</td><td>0.00%</td><td>3.04</td></tr> <tr><td>Times NR BI</td><td>98.33%</td><td>96.67%</td><td>0.00%</td><td>2.81</td></tr> <tr><td>Times NR I</td><td>96.67%</td><td>90.00%</td><td>0.00%</td><td>2.75</td></tr> </tbody> </table>					Font	Clean acc	Target acc	Rejected	Avg L_2	Arial	100.00%	94.17%	0.00%	3.10	Arial B	100.00%	96.67%	0.00%	3.27	Arial BI	100.00%	95.00%	0.00%	3.14	Arial I	99.17%	94.17%	0.83%	2.90	Courier	99.17%	79.17%	0.00%	2.73	Courier B	100.00%	96.67%	0.00%	3.36	Courier BI	100.00%	93.33%	0.00%	3.23	Courier I	99.17%	93.33%	0.83%	2.78	Georgia	100.00%	91.67%	0.83%	2.94	Georgia B	100.00%	94.17%	0.83%	3.18	Georgia BI	100.00%	92.50%	0.83%	3.03	Georgia I	100.00%	95.00%	0.00%	2.99	Times NR	100.00%	88.33%	0.00%	2.90	Times NR B	100.00%	91.67%	0.00%	3.04	Times NR BI	98.33%	96.67%	0.00%	2.81	Times NR I	96.67%	90.00%	0.00%	2.75
Font	Clean acc	Target acc	Rejected	Avg L_2																																																																																						
Arial	100.00%	94.17%	0.00%	3.10																																																																																						
Arial B	100.00%	96.67%	0.00%	3.27																																																																																						
Arial BI	100.00%	95.00%	0.00%	3.14																																																																																						
Arial I	99.17%	94.17%	0.83%	2.90																																																																																						
Courier	99.17%	79.17%	0.00%	2.73																																																																																						
Courier B	100.00%	96.67%	0.00%	3.36																																																																																						
Courier BI	100.00%	93.33%	0.00%	3.23																																																																																						
Courier I	99.17%	93.33%	0.83%	2.78																																																																																						
Georgia	100.00%	91.67%	0.83%	2.94																																																																																						
Georgia B	100.00%	94.17%	0.83%	3.18																																																																																						
Georgia BI	100.00%	92.50%	0.83%	3.03																																																																																						
Georgia I	100.00%	95.00%	0.00%	2.99																																																																																						
Times NR	100.00%	88.33%	0.00%	2.90																																																																																						
Times NR B	100.00%	91.67%	0.00%	3.04																																																																																						
Times NR BI	98.33%	96.67%	0.00%	2.81																																																																																						
Times NR I	96.67%	90.00%	0.00%	2.75																																																																																						
Notes	 <p>OCR systems are often used as just one component in a bigger pipeline, which passes their output to applications operating on the natural-language text (e.g., document categorization or summarization). These pipelines are a perfect target for the adversarial-image attacks because the output of OCR is not intended to be read or checked by a human. Therefore, the adversary does not need to worry about the syntactic or semantic correctness of the OCR output as long as this output has the desired effect on the NLP application that operates on it.</p>																																																																																									
Cited references to follow up on	<p>[1] ABADI, M., BARHAM, P., CHEN, J., CHEN, Z., DAVIS, A., DEAN, J., DEVIN, M., GHEMAWAT, S., IRVING, G., ISARD, M., ET AL. TensorFlow: A system for large-scale machine learning. In OSDI (2016).</p> <p>[2] Abbyy automatic document classification. https://www.abbyy.com/en-eu/ocr-sdk/key-features/classification, 2016.</p>																																																																																									
Follow up Questions	How can this be applied to fit texts from modern times but of subpar																																																																																									

	<p>quality?</p> <p>Can image preprocessing change this?</p>
--	---

Article #16 Notes: Deep Learning Approach in Gregg Shorthand Word to English-Word Conversion

Article notes should be on separate sheets

Source Title	Deep Learning Approach in Gregg Shorthand Word to English-Word Conversion
Source citation (APA Format)	<p>Padilla, D., & Vitug, N. (n.d.). <i>Deep Learning Approach in Gregg Shorthand Word to English-Word Conversion—IEEE Conference Publication</i>. Retrieved December 13, 2020, from</p> <p>https://ieeexplore.ieee.org/abstract/document/9177452?casa_token=b7gXAFfpJvwAAAAA:W0wx7xd-H9fJxOqKMsW-CkzFn89081m-ZXRqSJqlAWzfZiBM8oFhLsCdbT2ukpwGFZdl8ao</p>
Original URL	https://ieeexplore.ieee.org/abstract/document/9177452?casa_token=b7gXAFfpJvwAAAAA:W0wx7xd-H9fJxOqKMsW-CkzFn89081m-ZXRqSJqlAWzfZiBM8oFhLsCdbT2ukpwGFZdl8ao
Source type	Journal Article
Keywords	TensorFlow; CNN; Inceptionv3; Gregg Shorthand
Summary of key points (include methodology)	<p>Shorthand or Stenography has been used in a variety of fields of practice, particularly by court stenographers. To record every detail of the hearing, a stenographer must write fast and accurate. In the Philippines, the stenographers still used the conventional way of writing shorthand, which is by hand. Transcribing shorthand writing is time-consuming and sometimes confusing because of a lot of characters or words to be transcribed. Another problem is that only a stenographer can understand and translate shorthand writing. What if there is no stenographer available to decipher a document? A deep learning approach was used to implement and developed an automated Gregg shorthand word to English-word conversion. The</p>

	<p>Convolutional Neural Network (CNN) model used was the Inception-v3 in TensorFlow platform, an open-source algorithm used for object classification.</p>
<p>Research Question/Problem/ Need</p>	<p>How well can the Convolutional Neural Network (CNN) model used with the Inception-v3 in TensorFlow platform translate written Shorthand into English?</p>
<p>Important Figures</p>	<pre> graph TD Start([Start]) --> A[Identify the Statement of the Problem] A --> B[Purpose of the Study] B --> C[Review of Related Literature] C --> D[Set General and Specific Objectives] D --> E[Formulate Hypothesis and Frameworks to define the outline of the research] E --> F[Integrate Hardware and Software Development] F --> G[Use Inception-v3 Model for Training] G --> H[Prepare for the Testing and compare the result] H --> I{valid result?} I -- NO --> J[Modify training parameters] J --> G I -- YES --> K[Compute for the word accuracy] K --> L([end]) </pre> <p>Figure 1. Methodology Framework</p>

	Gregg Shorthand Image (Legal Terms)	Actual Translation (English Text)	Predicted Output (English Text)	Gregg Shorthand Equivalent
		Complaint	Cause	
		Convict	Conveyance	
		Finding	Findings	
		File	Felony	
Notes	The training datasets consist of 135 Legal Terminologies with 120 images per word with a total of 16,200 datasets. The trained model achieved a validation accuracy of 91%. For testing, 10 trials per legal terminology were executed with a total of 1,350 handwritten Gregg Shorthand words tested. The system correctly translated a total of 739 words resulting in 54.74% accuracy.			
Cited references to follow up on	<p>[1] M. Yang, G. Leedham, C. Higgins, and S. Htwe, "An On-line Recognition System for Handwritten Pitman Shorthand," TENCON 2005 - 2005 IEEE Region 10 Conference, 2005.</p> <p>[2] "Efficiency and Common Problems in Writing Stenography of the Bachelor of South Philippine Adventist College", " South Philippine Adventist College. [Online]. Available: http://www.spaconline.org/home/2016/09/19/efficiency-andcommon-problems-in-writing-stenography-of-the-bachelor-of-science-in-office-administration-students-of-south-philippineadventist-college/. [Accessed: 09-Sep-2019]</p>			
Follow up Questions	How does overfitting to a specific field and or region limit this project?			

	<p>How can it be compared to simpler CNN that do not have Deep Learning?</p>
--	--

Article #17 Notes: A new image classification method using CNN transfer learning and web data augmentation

Article notes should be on separate sheets

Source Title	A new image classification method using CNN transfer learning and web data augmentation
Source citation (APA Format)	Han, G., Liu, Q., & Fan, W. (n.d.). <i>A new image classification method using CNN transfer learning and web data augmentation</i> — <i>ScienceDirect</i> . Retrieved December 14, 2020, from https://www.sciencedirect.com/science/article/abs/pii/S0957417417307844?casa_token=z97ISwPPuZQAAAAA:LEsbigcJI5OM_4KgV3tnQfMyTD15PhK9ul4-oSZfNvJmkBvSN7cR_MoE9wFSuFQAS2ivezI
Original URL	https://www.sciencedirect.com/science/article/pii/S0957417417307844?casa_token=z97ISwPPuZQAAAAA:LEsbigcJI5OM_4KgV3tnQfMyTD15PhK9ul4-oSZfNvJmkBvSN7cR_MoE9wFSuFQAS2ivezI
Source type	Journal Article
Keywords	Feature transferring Data augmentation Convolutional neural network Feature representation Parameter fine-tuning Bayesian optimization
Summary of key points (include methodology)	Since Convolutional Neural Network (CNN) won the image classification competition 202 (ILSVRC12), a lot of attention has been paid to deep layer CNN study. The success of CNN is attributed to its superior multi-scale high-level image representations as opposed to hand-engineering low-level features. However, estimating millions of parameters of a deep CNN requires a large

	<p>number of annotated samples, which currently prevents many superior deep CNNs (such as AlexNet, VGG, ResNet) being applied to problems with limited training data. To address this problem, a novel two-phase method combining CNN transfer learning and web data augmentation is proposed. With our method, the useful feature presentation of pre-trained network can be efficiently transferred to target task, and the original dataset can be augmented with the most valuable Internet images for classification. Our method not only greatly reduces the requirement of a large training data, but also effectively expand the training dataset. Both of method features contribute to the considerable over-fitting reduction of deep CNNs on small dataset. In addition, we successfully apply Bayesian optimization to solve the tuff problem, hyper-parameter tuning, in network fine-tuning. Our solution is applied to six public small datasets. Extensive experiments show that, comparing to traditional methods, our solution can assist the popular deep CNNs to achieve better performance. Particularly, ResNet can outperform all the state-of-the-art models on six small datasets. The experiment results prove that the proposed solution will be the great tool for dealing with practice problems which are related to use deep CNNs on small dataset.</p>
<p>Research Question/Problem/ Need</p>	<p>Estimating millions of parameters of a deep CNN requires a large number of annotated samples, which currently prevents many superior deep CNNs (such as AlexNet, VGG, ResNet) being applied to problems with limited training data</p>
<p>Important Figures</p>	<p>The figure consists of six bar charts, one for each dataset: Dogs, Flowers 102, Caltech 101, Event8, 15 Scene, and 67 Indoor Scene. Each chart plots accuracy (%) on the y-axis (0 to 100) against three models: AlexNet, VGG-16, and ResNet-152 on the x-axis. For each model, five bars represent different training methods: 'scratch' (blue), 'fine-tune1' (orange), 'fine-tune2' (grey), 'fine-tune3' (yellow), and 'best' (purple). In all cases, the 'best' performance is achieved through fine-tuning, with ResNet-152 consistently showing the highest accuracy across all datasets and methods.</p>
<p>Notes</p>	<ul style="list-style-type: none"> - We do image classification on training data limited dataset with deep learning.

	<ul style="list-style-type: none"> - Transfer learning is employed to overcome the serious over-fitting. - Web data augmentation is developed to improve the classification performance. - Bayesian optimization is employed to facilitate the hyper-parameter search.
Cited references to follow up on	<p>Ando and Zhang, 2005 R.K. Ando, T. Zhang A framework for learning predictive structures from multiple tasks and unlabeled data Journal of Machine Learning Research, 6 (November) (2005), pp. 1817-1853</p> <p>Y.L. Boureau, J. Ponce, Y. LeCun A theoretical analysis of feature pooling in visual recognition Proceedings of the 27th international conference on machine learning (ICML-10) (2010), pp. 111-118</p>
Follow up Questions	<p>How can the dataset be customized with adapted Bayesian techniques?</p> <p>What causes the differences between the fine-tune groups?</p>

Article #18 Notes: Data Augmentation for Recognition of Handwritten Words and Lines Using a CNN-LSTM Network

Article notes should be on separate sheets

Source Title	Data Augmentation for Recognition of Handwritten Words and Lines Using a CNN-LSTM Network
Source citation (APA Format)	<p>Wigington, C., & Stewart, S. (n.d.). <i>Data Augmentation for Recognition of Handwritten Words and Lines Using a CNN-LSTM Network—IEEE Conference Publication</i>.</p> <p>Retrieved December 14, 2020, from</p> <p>https://ieeexplore.ieee.org/abstract/document/8270041?casa_token=v9uOpRqk3-gAAAAA: hv9tt4iFRGKInfLnpYI-0hFJ7kkTBvEV8qUjDN8ynzyc0kis05a5GAE7F__plhOBeBbcPC8</p>
Original URL	https://ieeexplore.ieee.org/abstract/document/8270041?casa_token=v9uOpRqk3-gAAAAA: hv9tt4iFRGKInfLnpYI-0hFJ7kkTBvEV8qUjDN8ynzyc0kis05a5GAE7F__plhOBeBbcPC8
Source type	Journal Article
Keywords	<ul style="list-style-type: none"> • Data Augmentation , • Handwriting Recognition , • Deep Learning , • Elastic Distortion , • CNN , • LSTM
Summary of key points (include methodology)	We introduce two data augmentation and normalization techniques, which, used with a CNN-LSTM, significantly reduce Word Error Rate (WER) and Character Error Rate (CER) beyond best-reported results on handwriting recognition tasks. (1) We apply a novel profile

normalization technique to both word and line images. (2) We augment existing text images using random perturbations on a regular grid. We apply our normalization and augmentation to both training and test images. Our approach achieves low WER and CER over hundreds of authors, multiple languages and a variety of collections written centuries apart. Image augmentation in this manner achieves state-of-the-art recognition accuracy on several popular handwritten word benchmarks.

Research Question/Problem/Need
 How can Data Augmentation, when applied to using a CNN to recognize Gregg shorthand, significantly improve performance?

Important Figures

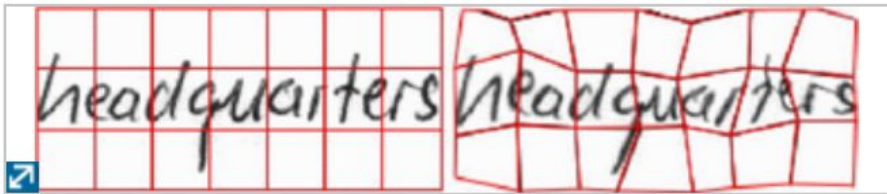










Fig. 3.
 Word image with uniform grid superimposed. 2nd image (right) with distorted grid and image distorted accordingly.

Single Example	Five Overlaid Examples
Original	
	
Shear/Rotation ($\pm 5^\circ$)	
	
Simard et al.[19] ($\sigma = 8, \alpha = 64$)	
	
Ours	
	

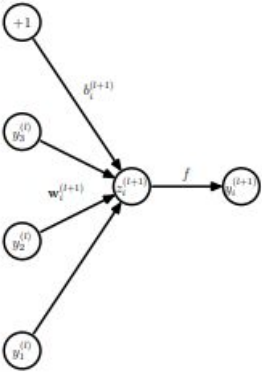
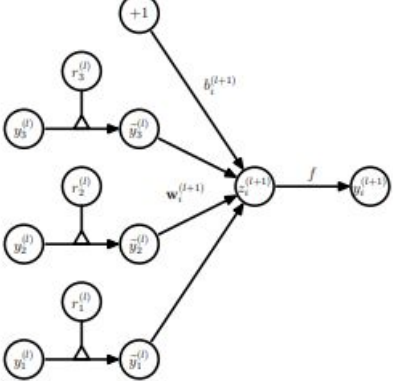
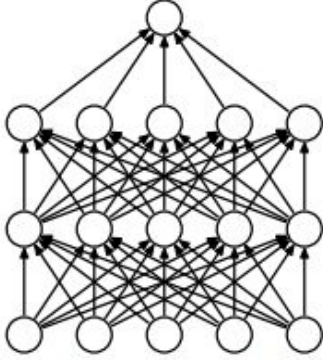
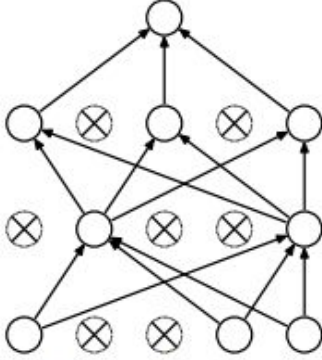
Notes
 Study mostly focused on English and German

	<p>Study was very extensive on image preprocessing</p> <p>The data augmentation tactics utilized a wide repertoire including everything from rotation to “scrunchiness”</p>
Cited references to follow up on	<p>A. Poznanski and L. Wolf, "Cnn-n-gram for handwriting word recognition", <i>Proc. CVPR 2016</i>.</p> <p>P. Krishnan, K. Dutta and C. V. Jawahar, "Deep feature embedding for accurate recognition and retrieval of handwritten text", <i>The 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)</i>, 2016.</p>
Follow up Questions	<p>Is it time efficient to use this type of image preprocessing?</p> <p>Can deep learning look beyond data augmentation?</p>

Article #19 Notes: Dropout: a simple way to prevent neural networks from overfitting

Article notes should be on separate sheets

Source Title	Dropout: a simple way to prevent neural networks from overfitting
Source citation (APA Format)	<p>Srivasta, N., Hinton, G., & Krizhevsky, A. (n.d.). <i>Dropout: A simple way to prevent neural networks from overfitting: The Journal of Machine Learning Research: Vol 15, No 1.</i></p> <p>Retrieved December 14, 2020, from https://dl.acm.org/doi/abs/10.5555/2627435.2670313</p>
Original URL	https://dl.acm.org/doi/abs/10.5555/2627435.2670313
Source type	Journal Article
Keywords	Computing Methodologies, Machine Learning, Neural Networks, Dropout
Summary of key points (include methodology)	<p>Deep neural nets with a large number of parameters are very powerful machine learning systems. However, overfitting is a serious problem in such networks. Large networks are also slow to use, making it difficult to deal with overfitting by combining the predictions of many different large neural nets at test time. Dropout is a technique for addressing this problem. The key idea is to randomly drop units (along with their connections) from the neural network during training. This prevents units from co-adapting too much. During training, dropout samples from an exponential number of different "thinned" networks. At test time, it is easy to approximate the effect of averaging the predictions of all these thinned networks by simply using a single unthinned network that has smaller weights. This significantly reduces overfitting and gives major improvements over other regularization methods. We show that dropout improves the performance of neural networks on supervised learning tasks in vision, speech recognition, document classification and computational biology, obtaining state-of-the-art results on many benchmark data sets.</p>

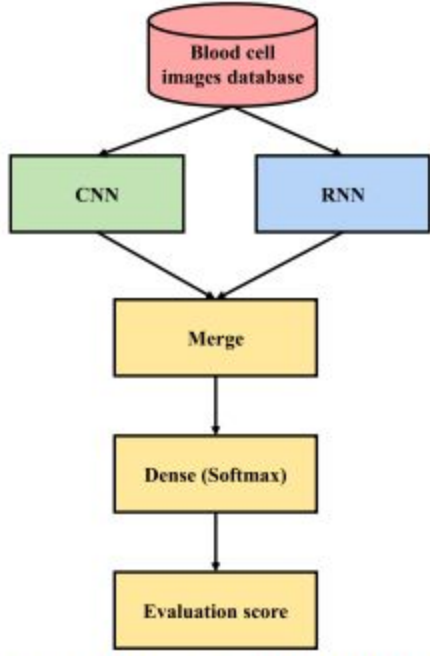
<p>Research Question/Problem/Need</p>	<p>How effective is Random Dropout at developing a ML Model?</p>
<p>Important Figures</p>	<div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;">  <p>(a) Standard network</p> </div> <div style="text-align: center;">  <p>(b) Dropout network</p> </div> </div> <p style="text-align: center;">Figure 3: Comparison of the basic operations of a standard and dropout network.</p> <div style="display: flex; justify-content: space-around; align-items: center; margin-top: 20px;"> <div style="text-align: center;">  <p>(a) Standard Neural Net</p> </div> <div style="text-align: center;">  <p>(b) After applying dropout.</p> </div> </div>
<p>Notes</p>	<p>Dropout neural networks can be trained using stochastic gradient descent in a manner similar to standard neural nets. The only difference is that for each training case in a mini-batch, we sample a thinned network by dropping out units. Forward and backpropagation for that training case are done only on this thinned network. The gradients for each parameter are averaged over the training cases in each mini-batch. Any training case which does not use a parameter contributes a gradient of zero for that parameter. Many methods have been used to improve stochastic gradient descent such as momentum, annealed learning rates and L2 weight decay. Those</p>

	were found to be useful for dropout neural networks as well.
Cited references to follow up on	<p>M. Chen, Z. Xu, K. Weinberger, and F. Sha. Marginalized denoising autoencoders for domain adaptation. In <i>Proceedings of the 29th International Conference on Machine Learning</i>, pages 767-774. ACM, 2012.</p> <p>G. E. Dahl, M. Ranzato, A. Mohamed, and G. E. Hinton. Phone recognition with the mean-covariance restricted Boltzmann machine. In <i>Advances in Neural Information Processing Systems 23</i>, pages 469-477, 2010.</p>
Follow up Questions	How can we facilitate this process to target specific areas of the model?

Article #20 Notes: Combining Convolutional Neural Network With Recursive Neural Network for Blood Cell Image Classification

Article notes should be on separate sheets

Source Title	Combining Convolutional Neural Network With Recursive Neural Network for Blood Cell Image Classification
Source citation (APA Format)	<p>Liang, G., & Hong, H. (n.d.). <i>Combining Convolutional Neural Network With Recursive Neural Network for Blood Cell Image Classification—IEEE Journals & Magazine</i>.</p> <p>Retrieved December 13, 2020, from https://ieeexplore.ieee.org/abstract/document/8402091</p>
Original URL	https://ieeexplore.ieee.org/abstract/document/8402091
Source type	Journal Article
Keywords	Artificial intelligence, convolutional neural network, recurrent neural network, transfer learning.
Summary of key points (include methodology)	<p>The diagnosis of blood-related diseases involves the identification and characterization of a patient's blood sample. As such, automated methods for detecting and classifying the types of blood cells have important medical applications in this field. Although deep convolutional neural network (CNN) and the traditional machine learning methods have shown good results in the classification of blood cell images, they are unable to fully exploit the long-term dependence relationship between certain key features of images and image labels. To resolve this problem, we have introduced the recurrent neural networks (RNNs). Specifically, we combined the CNN and RNN in order to propose the CNN-RNN framework that can deepen the understanding of image content and learn the structured features of images and to begin end-to-end training of big data in medical image analysis. In particular, we apply the transfer learning method to transfer the weight parameters that were pre-trained on the ImageNet dataset to the CNN section and adopted a custom loss</p>

	<p>function to allow our network to train and converge faster and with more accurate weight parameters. Experimental results show that compared with the other CNN models such as ResNet and Inception V3, our proposed network model is more accurate and efficient in classifying blood cell images.</p>
Research Question/Problem/ Need	<p>How can we automate the process of identifying red-blood cells with AI?</p>
Important Figures	 <pre>graph TD; A[Blood cell images database] --> B[CNN]; A --> C[RNN]; B --> D[Merge]; C --> D; D --> E[Dense (Softmax)]; E --> F[Evaluation score];</pre> <p>FIGURE 1. Overview of the proposed method using CNN-RNN framework and transfer learning for classifying blood cell images.</p>

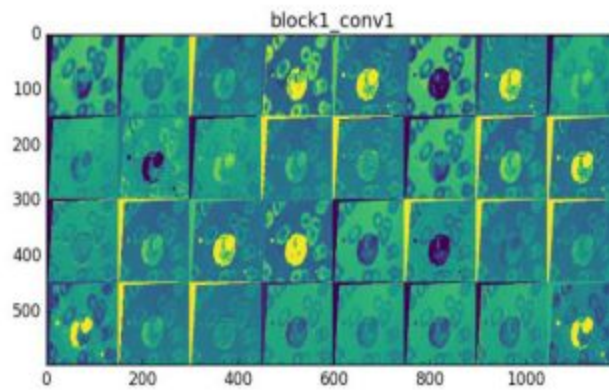


FIGURE 7. Visualization of the result of the activation (feature maps) of in the first convolutional layer of the Xception-LSTM fine-tuned on the dataset (block1_conv1 is the first convolutional layer of our model).

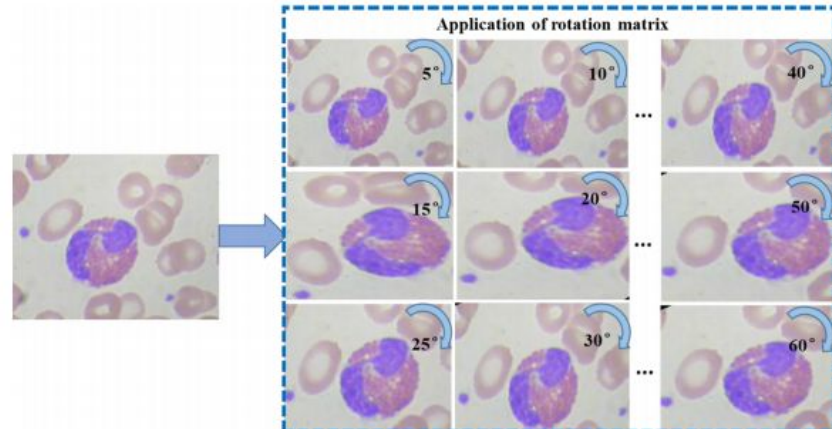


FIGURE 4. Image patches are generated from the blood cell image by application of rotation matrix so that the acquired images can be cropped to further augment the dataset.

Notes

They had a rotation matrix and focused heavily on one specific type of image preprocessing (or 2)

They had a standard confusion matrix

This was a color model and not in grayscale like many projects.

Cited references to follow up on

[1] N. Sinha and A. G. Ramakrishnan, "Automation of differential blood count," in Proc. Conf. Convergent Technol. Asia–Pacific Region (TENCON), Bengaluru, India, vol. 2, 2003, pp. 547–551.
 [2] P. Yampri, C. Pintavirooj, S. Daochai, and S. Teartulakarn, "White blood cell classification based on the combination of eigen cell and parametric feature detection," in Proc. 1st IEEE Conf.

	Ind. Electron. Appl., Singapore, May 2006, pp. 1–4.
Follow up Questions	How would its effectiveness be limited without color? Would its effectiveness be improved?