

City University of New York
School of Professional Studies

**Generating domain-specific language using a long short-term memory recurrent neural
network to infer relative linguistic complexity between astrophysics subdisciplines**

Daina Bouquin

DATA 698

Professor Robin Lee

December 11, 2017

Source Code and Data Access

All code, scripts, and datasets described herein are available via GitHub (Bouquin, 2017a).

Introduction

In 1987, a very simple definition of an "artificial neural network" (ANN) was presented by Maureen Caudill, an expert on artificial intelligence. Caudill wrote that an artificial neural network is "...a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs" (Caudill, 1987). Artificial neural networks are therefore able to progressively improve their performance through exposure to examples and can thus “learn” to do tasks that traditional rule-based algorithms are challenged by. For example, an ANN can “learn” to identify faces in images by being shown examples of images that have been tagged as containing faces. Similarly, ANNs can be trained to emulate a specific type of text when exposed to examples of that text. With this description in mind, what I describe herein is a small study on the application of a particular type of recurrent ANN, long short-term memory (LSTM), to generate domain-specific text in various subdisciplines of astronomy and astrophysics: black holes research, research in astrobiology, and exoplanet science.

I obtained examples of article titles published in the aforementioned subdomains and used those titles to train an LSTM ANN with the intention of generating titles that closely resemble the text that I used to train the model. Subsequent to training the ANN and sampling text from the model, I applied methods of assessment including term frequency-inverse

document frequency (tf-idf) and cosine similarity, along with a vector space model to assess the degree to which the text generated by the ANN accurately reflects the training text. I also used the proportion of non-ASCII characters generated by the LSTM models as an indicator of the model's ability to successfully replicate the domain-specific titles.

In conducting this study I was able to use the efficacy with which the LSTM ANN was able to emulate astronomy-related text as an proxy for linguistic complexity. If the LSTM ANN was not able to emulate the text well using the metrics I defined above, I have inferred that that text may have a higher degree of linguistic complexity than the text that the model was more effective at emulating. This analysis then stands to help foster conversations within the library science community about the challenges non-scientists face in engaging with academic publications; by examining relative linguistic complexity between subdisciplines of astronomy and astrophysics we may be able to infer the degree of difficulty that different scientific texts present to non-scientists and scientists from fields outside of astronomy. Moreover, if we are able to determine differences in linguistic complexity between similar domains, we would be able to help librarians better target their efforts to address scientific literacy in those fields.

In addition, the domains within astronomy and astrophysics that I choose to compare to one another using the above methods represent fields with differing longevity, which allows me to make inferences as to how the age of the field impacts the LSTM model's ability to emulate the text. This approach may then help us to better quantify how established a scientific subdiscipline is in relation to other fields of research.

Data Source

The SAO/NASA Astrophysics Data System (ADS Search, n.d.), is an online database of over eight million astronomy and physics papers from both peer reviewed and non-peer reviewed sources. The ADS is a highly used resource within the astronomy and astrophysics communities and has many levels of indexing. The ADS API makes it possible to query this valuable resource to better understand authorship and publishing behavior in these fields among many other applications. The ADS allows for full-text searching, and in the near future, will be fully incorporating Unified Astronomy Thesaurus (n.d.) keywords into their indexing schema. The ADS is managed by the Smithsonian Astrophysical Observatory at the Harvard–Smithsonian Center for Astrophysics with funding from NASA.

Background

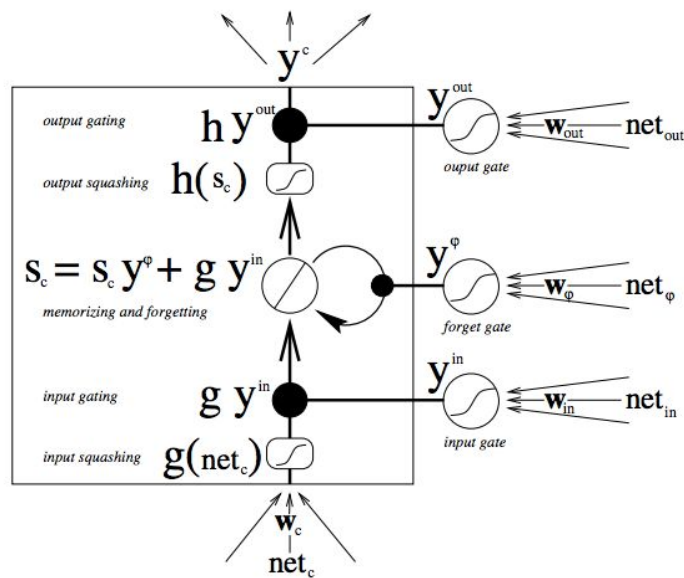
Artificial neural networks can be understood as being organized into "layers" with layers being made up of a number of interconnected "nodes" which each contain an "activation function". Data are passed into the "input layer" of the system, which communicates to "hidden layers" where the actual processing is done via a system of weighted "connections". Subsequently, the hidden layers connect with an "output layer" where the answer is output (Burger, n.d.). Since the creation of ANNs, these layered connections of nodes have been implemented in many different ways. Each implementation has been designed to meet different needs and experiments. Perhaps the most recognizable application of ANNs in popular culture is Google's "Deep Dream", a computer vision program that uses a convolutional neural network to find and enhance patterns in images via algorithmic pareidolia (Szegedy, et al., 2014).

In text-based applications of ANNs though, recurrent neural networks have become favored over convolutional systems. For instance, Twitter bots like "DeepDrumpf" (DeepDrumpf 2016, n.d.) have been used to demonstrate the powerful effectiveness of recurrent ANNs by emulating the linguistic style and syntax of individual people. Similarly, people have used recurrent ANNs to generate text based on the plays of Shakespeare (Angus, 2016), the essays of Paul Graham (Ramamoorthy, n.d.) and even to generate baby names (Kantrowitz, n.d.). Andrej Karpathy, Tesla Motors' leader on artificial intelligence research, has stated that some of the reasoning behind the "unreasonable effectiveness of recurrent neural networks" is that with convolutional systems, "their API is too constrained" in that they accept only a fixed-size vector as input and subsequently produce a fixed-size vector as output using a fixed amount of computational steps. Unlike these convolutional ANNs, recurrent ANNs allow us to operate over sequences of vectors: sequences in the input, output, or even both (Karpathy, n.d.).

To further define and justify the approach to be used for the herein described experiment, it is necessary to define the specific recurrent ANN I will be employing: long short-term memory (LSTM). LSTM ANNs are "capable of learning long-term dependencies" ('Long short-term memory', 2017) and do not start from scratch whenever they are presented with new data. LSTMs were introduced by Hochreiter and Schmidhuber (1997) and were refined iteratively following their work. The LSTM approach to recurrent ANNs is ideal for the type of text-based analysis I have defined because by using LSTM you can train the system on a large corpus of text and it will "learn" to generate text like it one character at a time. This, along with LSTM's relative insensitivity to "gap length" gives an advantage to LSTM over alternative recurrent

neural networks (Olah, 2015). Note, I have not found any evidence so far showing this approach being applied to text from astrophysics articles or using language from such a scientific-jargon heavy field.

The below LSTM diagram by DL4J's Skymind (2017) is helpful in defining this approach more granularly:



The writer of the DL4J guide explains the diagram thusly:

"Starting from the bottom, the triple arrows show where information flows into the cell at multiple points. That combination of present input and past cell state is fed... to each of its three gates, which will decide how the input will be handled. The black dots are the gates themselves, which determine respectively whether to let new input in, erase the present cell state, and/or let that state impact the network's output at the present time step. s_c is the current state of the memory cell, and g_y^{in} is the current input to it...

The cell can forget its state, or not; be written to, or not; and be read from, or not, at each time step, and those flows are represented here." (Skymind, 2017)

In order to better understand the motivations behind employing the LSTM in this context, it is necessary to also define “scientific literacy” and to touch on why it is important to further explore ways to support it among non-scientists. The United States National Center for Education Statistics defines scientific literacy as, “the knowledge and understanding of scientific concepts and processes required for personal decision making, participation in civic and cultural affairs, and economic productivity" (Council, 1999). Without fostering a scientifically literate populous, people who are not actively engaged in the scientific research enterprise are not capable of contributing to the sciences or fully enjoying the benefits that science brings to society. Libraries have taken up the mission of fostering literacy of all kinds as a core part of their work, and finding better ways to identify challenges to scientific literacy would help libraries expend their limited resources and advocate for their communities more effectively.

Test Cases

In order to test out the capability of a LSTM recurrent ANN to accurately emulate astronomy-related text, I began by subsetting increasingly large sets of article titles from the metadata available through the Astrophysics Data System. I used the titles to train the herein described ANN and to generate test titles. I did this for three different subdisciplines within the field (below) and gathered baseline metrics for each training dataset. I was then able to compare differences between "real" training data and "fake" data generated from the LSTM models. The metrics I gathered are as follows:

- Total number of unique words in various sizes of training sets within a domain
- Average proportion of non-ASCII characters in various sizes of training sets within a domain

Non-ASCII characters are characters that do not fall within the American Standard Code for Information Interchange (ASCII) character encoding scheme.

Data Sets

I chose to extract from the SAO/NASA Astrophysics Data System metadata from three different domains within astronomy and astrophysics. The datasets I built to train the LSTM artificial neural network are available for download on GitHub (Bouquin, 2017c) and are made up of increasingly large lists of titles from three specific domains. The lists range in length from the most recently published 1,000 titles of papers written in their prospective fields, up to the most recent 35,000 papers indexed by ADS. It is important to note here though that due to limitations in processing capability I was only able to perform baseline analyses for training datasets up to 20,000 titles. The three domains within astronomy and astrophysics that I chose to analyse were:

- Black Holes - earliest publication available in ADS is 1799 (Laplace, 1799)
- Astrobiology - earliest publication available in ADS is 1870 (Monro, 1870)
- Exoplanets - earliest publication available in ADS is 1943 (van den Bos, 1943)

These domains were chosen in order to enable the comparison of increasingly recent subdisciplines within astronomy and astrophysics and to ensure that there is substantial data to pull from ADS for building the LSTM model training datasets.

Hypotheses

My initial hypothesis was that some fields within astronomy and astrophysics would be more accurately generated by the LSTM ANN than others. I suspected that accuracy would be related to the diversity of unique words used within the training corpus of text representing each subdomain. This is to say that a subdomain with a larger variety of unique words (combinations of letters) would be more challenging to for the LSTM ANN to generate than a training corpus with fewer unique words. I similarly hypothesized that more data would be needed to achieve a higher degree of similarity between generated text and training text and that this would correspond to unique word count as well.

In addition, I hypothesized that the ability to generate similar titles to those used to train the models would be different between fields that were more emergent compared with those that are more established. For instance, I hypothesized that the LSTM ANN would be more effective in generating titles based on the "black hole" data than on the "exoplanet" data as the exoplanet field is much less established and therefore likely less regular from a linguistics perspective than text discussing black holes.

Torch for Data Analysis

For my analyses, I decided to use Torch (n.d.), a scientific computing framework that makes available a robust library of Python- and Lua-based machine learning tools. The Torch

implementation of LSTM recurrent neural network models is straightforward and works to ensure that programmers, "have maximum flexibility and speed" in building scientific algorithms (Torch, n.d.). The biggest deciding factor for me in selecting Torch over comparable LSTM implementation tools available through platforms like TensorFlow was the comprehensive guides available for those of us just getting started in creating neural networks. The other primary advantage of using Torch is that it does not require you to learn an entirely new set of tools and data structures; I was able to use Python, a language I'm familiar with, to generate the files needed to build my models. The primary downside to selecting Torch was that I am personally not able to capitalize as fully on the tool as I would be able to if I had access to a powerful GPU.

Installing Torch

Working with Torch meant working with a system that has multiple dependencies including Python and Lua, and if you have access to hardware like an NVidia graphics card, CUDA and OpenCL for GPU acceleration are also helpful. Standard instructions for installing Torch on various operating systems are available here (Johnson, 2017b). I did not have access to GPU resources locally and followed Torch's standard OSX installation instructions (Thompson, 2016). Unfortunately, this means that the amount of time that it took to train my LSTM models was extensive and served as a limitation to this study. With more time and resources it would be possible to expand this analysis into the cloud to speed the computation along, however, it was still possible to run the herein described tests without cloud computing.

TF-IDF and Cosine Similarity


Two of the measures I decided to employ to begin comparing LSTM-generated (“fake”) titles to the real titles used to train the models were tf-idf and cosine similarity. Both of these have become popular and impactful in the field of digital libraries. Tf-idf (2017) is a numerical statistic that is intended to reflect how important a word is to a document in a corpus of text.

$$\text{tfidf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D)$$

I used tf-idf to assign value to words within titles and then map those values into a high dimensional vector space so that the titles could be compared to each other using cosine similarity (detailed below). In the subsequent code examples I show how I can compare a titles generated by my LSTM network to the corpus of real titles on which the model was trained. I did this with for all training data sizes of the three subdisciplines; the code used to do this is available here (Bouquin, 2017b).

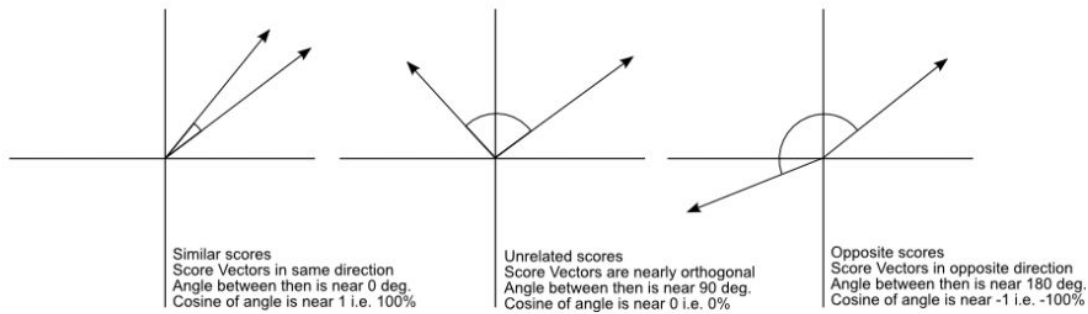
Cosine Similarity Explained

Cosine similarity (2017) is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them and is commonly applied in settings where tf-idf vectors are being compared. Cosine similarity can be calculated thusly:

$$\text{sim}(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|}$$
A diagram showing two blue vectors, A and B, originating from a common point. Vector A points towards the upper right, and vector B points towards the lower right. The angle between them is labeled with the Greek letter theta (θ).

As illustrated below, the similarity between two vectors (strings of characters) increases as the angle between the two vectors decreases. When the angle is 90 degrees, the two vectors

are unrelated.



Analysis of Non-ASCII Characters

I also assessed the quantity of non-ASCII characters in the LSTM-generated titles compared to the quantity of non-ASCII characters in the training data as another indicator of how well the model is doing at emulating the training text.

Data Acquisition, Processing, and Analysis Workflow

The code cells and comments in the following sections are used to demonstrate the creation of the initial datasets for training the LSTM models and include all necessary preprocessing steps. They also show how baseline metrics about the training sets can be calculated. The creation of file formats needed for use with Torch are also shown as script examples, and all scripts used to process data, train models, and sample from them are available on GitHub (Bouquin, 2017a). To reproduce the torch analysis below here without needing to install Torch or create a separate virtual environment, it is possible to make use of Torch's various docker images (Baldi, 2017). The final code cells are used to show the training of the model and subsequent sampling of data resulting from the training. I then demonstrated how cosine similarity can be calculated between a the titles generated with the LSTM model and the training corpus itself.

Pulling Data from the Astrophysics Data System API

Example for 1,000 "Black Hole" titles:

```
# modules for ADS API and data transformations
import os
import ads as ads
import pandas as pd
import numpy as np

# configure API
os.environ["ADS_DEV_KEY"] = "kNUoTurJ5TXV9hsw9KQN1k8wH4U0D7Oy0CJoOvyw"
ads.config.token = 'ADS_DEV_KEY'

papers1 = list(ads.SearchQuery(q= "black hole", fl=['bibcode', 'title', 'abstract'], sort='pubdate', max_pages=20 ))

# find titles
t = []
for i in papers1:
    title1 = i.title
    t.append(title1)
title = t

# create an initial df (only 1 column) and clean it up
df = pd.DataFrame({'Title' : title
})
df['Title'] = df['Title'].str.get(0)

# write to .txt
df.to_csv("blackhole_1000.txt", sep=' ', header=None, index=None, encoding='utf-8')

# get average number of characters in a title for use in torch
avg_char_len = sum(df['Title'].str.len())/1000

print "Average Character Length per Training Title:", avg_char_len

Average Character Length per Training Title: 85
```

The above calculation was used to specify the length of the title that Torch will generate.

In this case I needed to have Torch generate "fake" titles that are 85 characters long.

Calculating Average Proportions of Non-ASCII Characters in Training Data

```
import re
import pandas as pd
import csv

# Remove unnecessary white space and read data
lines1K = [line.rstrip('\n') for line in open('blackhole_1000.txt')]

# Find titles with non-ascii characters
nonascii_lines1K = []
for line in lines1K:
    a = re.sub('[\x00-\x7f]', '', line)
    nonascii_lines1K.append(a)

# Drop empty elements in the list (titles with no ascii characters)
nonascii_notempty1K = filter(None, nonascii_lines1K)
count_nonascii_notempty1K = len(nonascii_notempty1K)

# Calculate average number of non-ascii characters in a title when the title contains non-ascii characters
actual_nonascii_1K = sum([len(i) for i in nonascii_notempty1K])/count_nonascii_notempty1K

print "Average count of non-ASCII characters per training title with non-ASCIIIs:", actual_nonascii_1K
```

Average count of non-ASCII characters per training title with non-ASCIIIs: 3

To summarize, the non-ASCII analysis shows that when titles in this training set contain characters that are not ASCII-encoded, there are on average three non-ASCII characters in the title.

Calculate Total Number of Unique Words in Training Set

```
from collections import Counter
from string import punctuation

with open('blackhole_1000.txt', 'r') as myfile:
    blackhole_1000=myfile.read().replace('\n', '')

print "Number of Unique words:", len(Counter(blackhole_1000.split()))
```

Number of Unique words: 4134

Now that I had built a training dataset made up of 1,000 "black hole" titles from the ADS and calculated the average length of a title, number of unique words in the training set, and average number of non-ASCII characters in a title when non-ASCII characters are present, I began processing the training dataset for use with Torch.

Building JSON and H5 files for Torch

Torch requires an analyst to create two files in order to take advantage of Torch's library of machine learning code for LSTM ANN models: a JSON file representing the text in a more standard structure, and a hierarchical data format (2017) called an H5 file, which is designed to organize and store very large amounts of data.

To create the two necessary files for training the LSTM ANN, Torch makes available preprocessing Python scripts that can easily be run in the terminal that are available here (Johnson, 2017a). The below scripts show how to run the preprocessing scripts and generate the needed JSON and H5 files.

```
Restructuring and sampling data for fresh analysis:

## blackhole1K
# preprocess X
python scripts/preprocess.py --input_txt
data/Final_Datasets/blackholes/blackhole_1000/blackhole_1000_clean.txt --output_h5
data/Final_Datasets/blackholes/blackhole_1000/blackhole_1000_clean.h5 --output_json
data/Final_Datasets/blackholes/blackhole_1000/blackhole_1000_clean.json
```

Training the LSTM Model Using Torch

I could now take advantage of the Torch's code library to train the LSTM recurrent ANN:

```
# train X
th train.lua -input_h5 data/Final_Datasets/blackholes/blackhole_1000/blackhole_1000_clean.h5 -
input_json data/Final_Datasets/blackholes/blackhole_1000/blackhole_1000_clean.json -gpu -1
```

Creating Test Samples using the LSTM Model Generated by Torch

The model was then used to generate a sample of 10,000 new titles based on training titles. I performed these steps for each training size and each subdiscipline: Black Holes, Astrobiology, and Exoplanets. You can see 10,000 example titles generated by Torch using the 1,000 title training set of black hole titles here (Bouquin, 2017d).

```
# sample X
for ((i=1;i<=10000;i++)) ; do th sample.lua -checkpoint cv/checkpoint_10000.t7 -length 82 -
sample 1 -gpu -1 | tr -d "\r\t\n" | sed 's/./&\n/82' >>
data/Final_Datasets/blackholes/blackhole_1000/blackhole_1000_sample_n1.txt ; done
```

Cosine Similarity Example

The below process was completed for each sample of 10,000 LSTM-generated titles in each training size/domain:

Black hole titles generated after training on 20,000 real black hole titles

```
blackhole_20K = [line.rstrip('\n') for line in open('blackhole_20000_clean.txt')]
df_blackhole_20K = pd.DataFrame(blackhole_20K)
df_blackhole_20K.columns = ['Title']
```

```
test_results = []

with open("blackhole_20000_sample_n1.txt", "r") as ins:
    for line in ins:
        df_blackhole_20K_test = df_blackhole_20K
        df_blackhole_20K_test.loc[-1] = [line]
        df_blackhole_20K_test.index = df_blackhole_20K_test.index + 1
        df_blackhole_20K_test = df_blackhole_20K_test.sort_index()
        title_tuple = tuple(list(df_blackhole_20K_test['Title']))
        tfidf_matrix = tfidf_vectorizer.fit_transform(title_tuple)
        cosine_sim_array = cosine_similarity(tfidf_matrix[0:1], tfidf_matrix)
        cos_sim = np.partition(cosine_sim_array.flatten(), -2)[-2]
        angle_in_degrees = math.acos(cos_sim)
        most_similar20K = math.degrees(angle_in_degrees)
        test_results.append(most_similar20K)
```

For each subdiscipline, all results were combined for each training size and restructured to allow for plotting:

Combine and reshape all black hole results

```
blackhole_allresults = pd.concat([blackhole_1K_results, blackhole_5K_results, blackhole_10K_results, blackhole_20K_results], axis=1)
```

```
blackhole_allresults.head()
```

| | CosineSim1K | CosineSim5K | CosineSim10K | CosineSim20K |
|---|-------------|-------------|--------------|--------------|
| 0 | 83.379437 | 80.988346 | 81.341854 | 80.916998 |
| 1 | 80.112625 | 79.940088 | 79.202072 | 80.241259 |
| 2 | 83.557122 | 80.874947 | 78.306722 | 40.039122 |
| 3 | 81.901079 | 83.587092 | 82.274476 | 66.288257 |
| 4 | 90.000000 | 79.252060 | 73.501383 | 60.427735 |

```
blackhole_allresults = pd.melt(blackhole_allresults)
blackhole_allresults.columns = ['TrainingDataSize', 'CosineSim']
blackhole_allresults.head()
```

| | TrainingDataSize | CosineSim |
|---|------------------|-----------|
| 0 | CosineSim1K | 83.379437 |
| 1 | CosineSim1K | 80.112625 |
| 2 | CosineSim1K | 83.557122 |
| 3 | CosineSim1K | 81.901079 |
| 4 | CosineSim1K | 90.000000 |

```
# Find minimum
blackhole_allresults['CosineSim'].min()
```

```
30.614290662608298
```

Next, the minimum cosine similarity angle achieved across all training sizes was then calculated to represent the best result achieved for the domain. In this case the best similarity measure achieved for all black hole models was 30.6 degrees.

Non-ASCII Characters Analysis

Below I compared 10,000 LSTM-generated titles trained on 5,000 real titles to their training data to see how many non-ASCII characters on average were in a title.

Presence of non-ASCII characters in 10,000 real training titles:

```
# Find titles with non-ascii characters
nonascii_lines10K = []

for line in blackhole_10K:
    a = re.sub('[\x00-\x7f]', '', line)
    nonascii_lines10K.append(a)

# Drop empty elements in the list (titles with no ascii characters)
nonascii_notempty_10K = filter(None, nonascii_lines10K)
count_nonascii_notempty_10K = len(nonascii_notempty_10K)

# Calculate average number of non-ascii characters in a title when the title contains non-ascii characters
actual_nonascii_10K = sum([len(i) for i in nonascii_notempty_10K])/count_nonascii_notempty_10K

print "Average count of non-ASCII characters per training title with non-ASCIIIs:", actual_nonascii_10K
```

Average count of non-ASCII characters per training title with non-ASCIIIs: 3

Presence of non-ASCII characters in sampled titles trained on 10,000 real titles:

```
blackhole_10K_test = [line.rstrip('\n') for line in open('blackhole_10000_sample_nl.txt')]

# Find titles with non-ascii characters
nonascii_lines10K_test = []

for line in blackhole_10K_test:
    a = re.sub('[\x00-\x7f]', '', line)
    nonascii_lines10K_test.append(a)

# Drop empty elements in the list (titles with no ascii characters)
nonascii_notempty_10K_test = filter(None, nonascii_lines10K_test)
count_nonascii_notempty_10K_test = len(nonascii_notempty_10K_test)

# Calculate average number of non-ascii characters in a title when the title contains non-ascii characters
actual_nonascii_10K_test = sum([len(i) for i in nonascii_notempty_10K_test])/count_nonascii_notempty_10K_test

print "Average count of non-ASCII characters per training title with non-ASCIIIs:", actual_nonascii_10K_test
```

Average count of non-ASCII characters per training title with non-ASCIIIs: 6

Summary of Findings

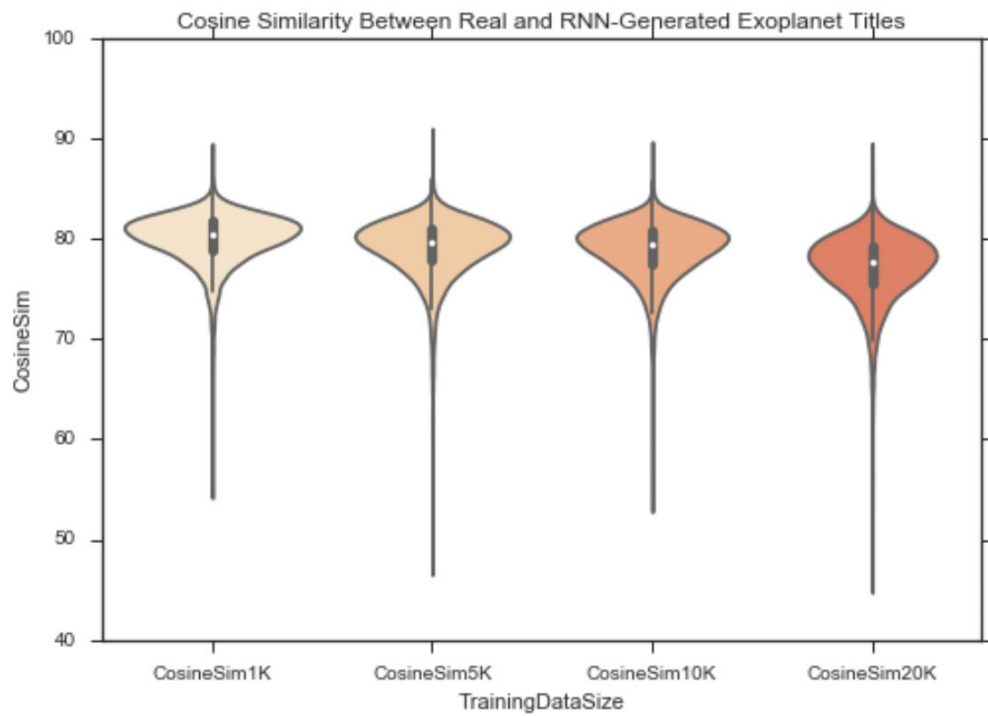
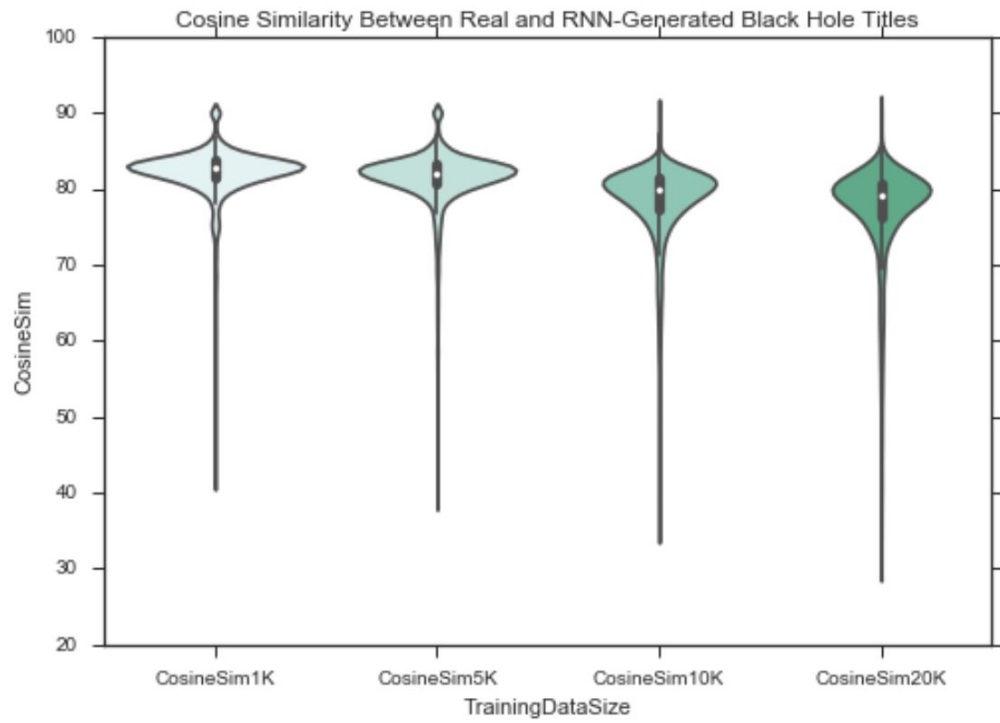
The subsequent sections outline the results of the full analysis outlined in previous sections.

I had initially hypothesized that the LSTM ANN would be more effective in generating titles based on the "black hole" data than on the "exoplanet" data as the exoplanet field is much less established and therefore likely less regular from a linguistics perspective than text

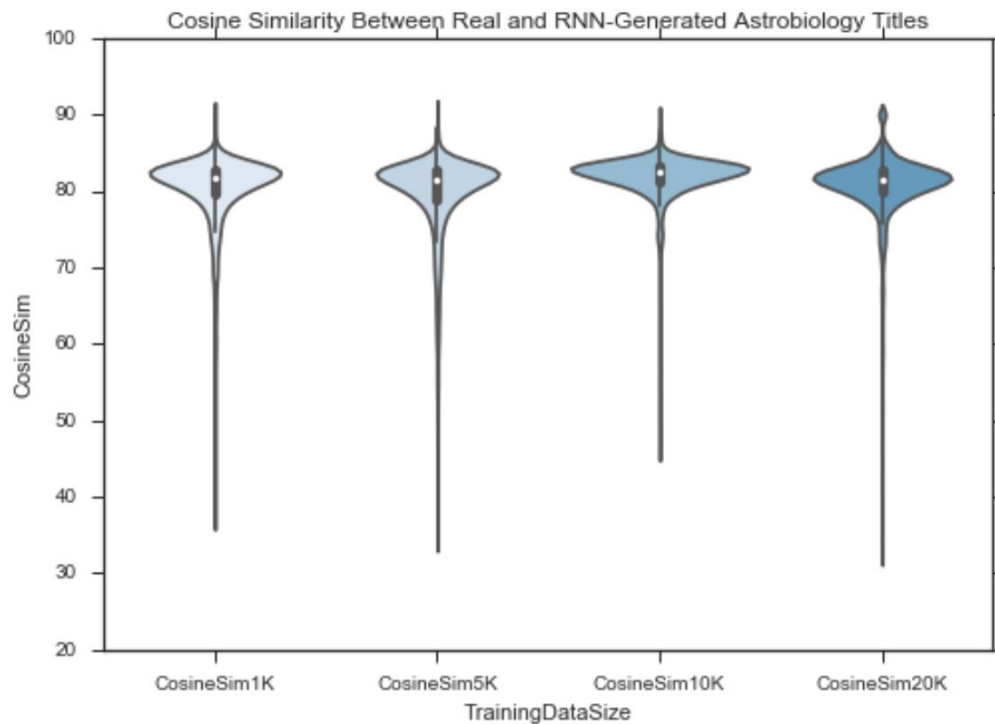
discussing black holes. This hypothesis has been shown to be true in regards to an LSTM trained real astronomy titles from the Astrophysics Data System. The real titles were used to train the neural network and each trained model output 10,000 RNN-generated titles based on the training data; training datasets consisted of corpus sizes of 1,000 to 20,000 titles. The highest degree of accuracy achieved by the LSTM RNN for each subdomain of astronomy is listed below. You can see that the most established field (black holes) has the lowest cosine similarity measure achieved across all tests, while the most emergent field (exoplanets) was not able to achieve a cosine similarity measure better than 45.81 degrees.

| Astronomy Domain | Minimum Cosine Similarity Angle |
|-------------------------|--|
| Black Holes | 30.6 |
| Astrobiology | 32.4 |
| Exoplanets | 45.81 |

Moreover, as training sizes increased, the accuracy as measured through cosine similarity improved for both the black hole and exoplanets tests (violin plots below); the improvements were subtle and this is likely due to not having a much higher capacity to process larger datasets. Running torch (the analysis module I implemented in Python) without access to a GPU and GPU acceleration through CUDA necessarily limited my ability to train models using larger datasets.



Interestingly, for the cosine similarity analysis performed on the RNN-generated titles that were trained on the most interdisciplinary field (astrobiology), there was no obvious improvement across training set sizes. There seemed to be no obvious pattern emerging from these initial astrobiology results.



I suspect that this lack of model improvement may be due to the relatively higher unique word count for astrobiology papers compared to exoplanet or black hole titles. As is shown below, astrobiology has a significantly higher unique word counts across training set sizes.

| Astronomy Domain | Training Size | Unique Word Count |
|------------------|---------------|-------------------|
| Black Hole | 1K | 4134 |
| Black Hole | 5K | 15203 |

| | | |
|--------------|-----|-------|
| Black Hole | 10K | 26095 |
| Black Hole | 20K | 45761 |
| Astrobiology | 1K | 5217 |
| Astrobiology | 5K | 20324 |
| Astrobiology | 10K | 34104 |
| Astrobiology | 20K | 56703 |
| Exoplanets | 1K | 4162 |
| Exoplanets | 5K | 14143 |
| Exoplanets | 10K | 23175 |
| Exoplanets | 20K | 38301 |

As for my non-ASCII character analysis, I compared the proportion of non-ASCII characters per title containing non-ASCII characters to that of titles in the training corpus itself with the below results:

| Astronomy Domain | Training Size | Non-ASCII Training | Non-ASCII Test |
|------------------|---------------|--------------------|----------------|
| Black Hole | 1K | 3 | 25 |
| Black Hole | 5K | 3 | 26 |
| Black Hole | 10K | 3 | 6 |

| | | | |
|--------------|-----|---|----|
| Black Hole | 20K | 3 | 6 |
| Astrobiology | 1K | 3 | 8 |
| Astrobiology | 5K | 2 | 9 |
| Astrobiology | 10K | 3 | 15 |
| Astrobiology | 20K | 3 | 15 |
| Exoplanets | 1K | 3 | 7 |
| Exoplanets | 5K | 3 | 7 |
| Exoplanets | 10K | 2 | 4 |
| Exoplanets | 20K | 2 | 4 |

Again you can see from the above results that astrobiology is the only subdomain that deviates from the trend wherein increasing the training set size makes the prevalence of non-ASCII characters more realistic. For both other disciplines increasing the size of the training dataset resulted in LSTM-generated titles that are more clearly similar to their training titles.

Conclusions

The above described analysis can be effective in examining the complexity inherent in different subfields of astronomy and astrophysics. The analysis shows that an LSTM RNN has more difficulty emulating the interdisciplinary field (Astrobiology) titles than the titles of less interdisciplinary fields (Black Holes, Exoplanets). The LSTM RNN is also better at emulating text in fields that are more well established (Black Holes) than the field that is most emergent

(Exoplanets). These results suggest that we may be able to infer the relative degree of difficulty that different scientific texts present to non-scientists so that we can better target efforts to address scientific literacy in those fields. We may also be able to better quantify how established a scientific subdiscipline is by examining its relative linguistic complexity using neural networks.

Limitations

The process that I have defined here of sampling data from the Astrophysics Data System, processing it for use in training an LSTM ANN model using Torch, generating sample titles from the trained model, and subsequently analyzing the results, can effectively be used to test the hypotheses I have laid out for this project. There are though limitations to this approach:

1. Topic modeling within the ADS is imperfect, therefore my initial search results were imperfect.
 - a. Defining what an "exoplanet paper" is was a known challenge. In the near future the ADS will implement the Unified Astronomy Thesaurus, which will allow for much more controlled searches and topic modeling.
2. I had very limited access to computational resources (GPUs) that could have improved both computational speed and the capacity to process much larger datasets for training and sampling.
3. The training dataset sizes will always be limited by the number of papers published in a given field, thus impacting the fields in which this sort of analysis could be performed.

Future Work

The next steps to be taken in furthering the work outlined here will first be to scale up the study with improved computational resources. The next challenge will then be to emulate these results in other subdisciplines of astronomy and astrophysics and to try conducting this study in another scientific discipline.

References

- ADS Search (no date). Available at: <https://ui.adsabs.harvard.edu/> (Accessed: 10 December 2017).
- Angus, R. (2016) Fakespeare: Shakespeare lives. Available at:
<https://github.com/RuthAngus/Fakespeare> (Accessed: 10 December 2017).
- Baldi, C. (2017) docker-torch-rnn: Docker images for using torch-rnn. Available at:
<https://github.com/crisbal/docker-torch-rnn> (Accessed: 10 December 2017).
- Bouquin, D. (2017a) DATA_698: Code for DATA 698/ CUNY MS in Data Analytics. Available at: https://github.com/dbouquin/DATA_698 (Accessed: 10 December 2017).
- Bouquin, D. (2017b) DATA_698: Code for DATA 698/scale_up. Available at:
https://github.com/dbouquin/DATA_698/tree/master/analysis/scale_up (Accessed: 10 December 2017).
- Bouquin, D. (2017c) DATA_698: Code for DATA 698/Training_Datasets. Available at:
https://github.com/dbouquin/DATA_698/tree/master/Training_Datasets (Accessed: 10 December 2017).
- Bouquin, D. (2017d) DATA_698: Code for DATA 698 - sample example. Available at:
https://github.com/dbouquin/DATA_698/blob/master/analysis/scale_up/blackhole_1000_sample_nl.txt (Accessed: 10 December 2017).
- Burger, J. (no date). A Basic Introduction To Neural Networks. Available at:
<http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html> (Accessed: 10 December 2017).

Caudill, M. (1987) ‘Neural Networks Primer, Part I’, *AI Expert*, 2(12), pp. 46–52. Available at:
<http://dl.acm.org/citation.cfm?id=38292.38295> (Accessed: 10 December 2017).

‘Cosine similarity’ (2017) Wikipedia. Available at:
https://en.wikipedia.org/w/index.php?title=Cosine_similarity&oldid=808635444
(Accessed: 10 December 2017).

Council, N. R. (1999) *Every Child a Scientist: Achieving Scientific Literacy for All*. doi:
10.17226/6005.

DeepDrumpf 2016 (no date). Available at: <http://www.deepdrumpf2016.com/about.html>
(Accessed: 10 December 2017).

Hierarchical Data Format (2017) Wikipedia. Available at:
https://en.wikipedia.org/w/index.php?title=Hierarchical_Data_Format&oldid=812956224
(Accessed: 10 December 2017).

Hochreiter, S. and Schmidhuber, J. (1997) ‘LONG SHORT-TERM MEMORY’, *Neural Computation*, 9(8), pp. 1735–1780. Available at:
<http://www.bioinf.jku.at/publications/older/2604.pdf>.

Johnson, J. (2017a) torch-rnn: Efficient, reusable RNNs and LSTMs for torch. Available at:
<https://github.com/jcjohnson/torch-rnn/blob/master/scripts/preprocess.py> (Accessed: 10
December 2017).

Johnson, J. (2017b) torch-rnn: Efficient, reusable RNNs and LSTMs for torch. Available at:
<https://github.com/jcjohnson/torch-rnn> (Accessed: 10 December 2017).

Kantrowitz, M. (no date). Available at:

<https://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/nlp/corpora/names/>

(Accessed: 10 December 2017).

Karpathy, A (no date). The Unreasonable Effectiveness of Recurrent Neural Networks. Available

at: <https://karpathy.github.io/2015/05/21/rnn-effectiveness/> (Accessed: 10 December

2017).

Laplace, P. S. (1799) ‘Beweis des Satzes, dass die anziehende Kraft bey einem Weltkörper so

groß seyn könne, dass das Licht davon nicht ausströmen kann’, 4(1), p. 1. Available at:

<https://ui.adsabs.harvard.edu/#abs/1799AllGE...4....1L/abstract> (Accessed: 10 December

2017).

‘Long short-term memory’ (2017) Wikipedia. Available at:

https://en.wikipedia.org/w/index.php?title=Long_short-term_memory&oldid=814397652

(Accessed: 10 December 2017).

Monro, C. J. (1870) ‘The Difficulties of Natural Selection’, *Nature*, 3(57), p. 86. doi:

10.1038/003086b0.

Olah, C. (2015). Understanding LSTM Networks -- colah’s blog. Available at:

<https://colah.github.io/posts/2015-08-Understanding-LSTMs/> (Accessed: 10 December

2017).

Ramamoorthy, S. (no date) Unfolding RNNs II. Available at:

<http://suriyadeepan.github.io/2017-02-13-unfolding-rnn-2/> (Accessed: 10 December

2017).

Skyrmind (2017). A Beginner's Guide to Recurrent Networks and LSTMs - Deeplearning4j:

Open-source, Distributed Deep Learning for the JVM. Available at:

<https://deeplearning4j.org/lstm.html#long> (Accessed: 10 December 2017).

Szegedy, C. et al. (2014) 'Going Deeper with Convolutions', arXiv:1409.4842 [cs]. Available at:

<http://arxiv.org/abs/1409.4842> (Accessed: 10 December 2017).

tf-idf (2017) Wikipedia. Available at:

<https://en.wikipedia.org/w/index.php?title=Tf%E2%80%93idf&oldid=813026023>

(Accessed: 10 December 2017).

Thompson, J. (2016) Torch-rnn: Mac Install. Available at:

<http://www.jeffreythompson.org/blog/2016/03/25/torch-rnn-mac-install/> (Accessed: 10 December 2017).

Torch | Scientific computing for LuaJIT. (no date). Available at: <http://torch.ch/> (Accessed: 10

December 2017).

Unified Astronomy Thesaurus (no date). Available at: <http://astrothesaurus.org/> (Accessed: 10

December 2017).

van den Bos, W. H. (1943) 'An Extra-Solar Planet?', Monthly Notes of the Astronomical Society

of South Africa, 2, p. 14. Available at:

<https://ui.adsabs.harvard.edu/#abs/1943MNSSA...2...14V/abstract> (Accessed: 10

December 2017).