A long short-term memory recurrent neural network for generating astrophysics-specific language and assessment using tf-idf, cosine similarity, and presence of non-ASCII characters to determine model effectiveness

Daina Bouquin

This is an expanded version of the initial proposal [1] that includes discussion of prior approaches, baseline results of my approach, and a final assessment of whether this project can effectively address the goals stated in the initial proposal.

Introduction

In 1987, a very simple definition of an "artificial neural network" (ANN) was presented by Maureen Caudill, an expert on artificial intellegence. Caudill wrote that an artifical 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" [2]. With this definition in mind, what I propose herein is a small study on the application of a particular type of recurant ANN, long short-term memory (LSTM), to generate domain-specific text in various subdisciplines of astronomy and astrophysics (research on black holes, astrobiology, and exoplanets). Subsequent to the training and sampling of astronomy and astrophysics related text, I will apply measures 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 text used to train the system. I will also use the proportion of non-ASCII characters generated by the LSTM models as an indicator of the success of the model.

In conducting this study I will be applying the aforementioned approaches to a dataset that has not yet been used to test the capacity for this type of system to "learn": metadata from the SAO/NASA Astrophysics Data System. I will also be applying an LSTM ANN to in a jargon-heavy scientific domain, which is an area of limited research. Moreover, this analysis stands to help foster conversations within the library science community about the challenges non-scientists face in improving their scientific literacy and engaging with academic publications. Specifically, the ANN-generated "fake" astronomy text could potentially be used as control text in future literacy studies and the degree of success achieved in emulating the text could serve as an indicator of the complexity inherent to this kind of discipline-specific language. Moreover, the domains within astronomy and astrophysics that I choose to compare to one another will reepresent fields with differing longevity and I will therefore be able to make inferences as to how the age of the field may impact the model's ability to emulate the text. Furthermore, sharing the ANN-generated "fake" astronomy text with the astronomy community and others will allow us to further frame conversations about "fake news" and scientific literacy.

Data Source

The SAO/NASA Astrophysics Data System (ADS) [3], 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 in 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 [4] 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.

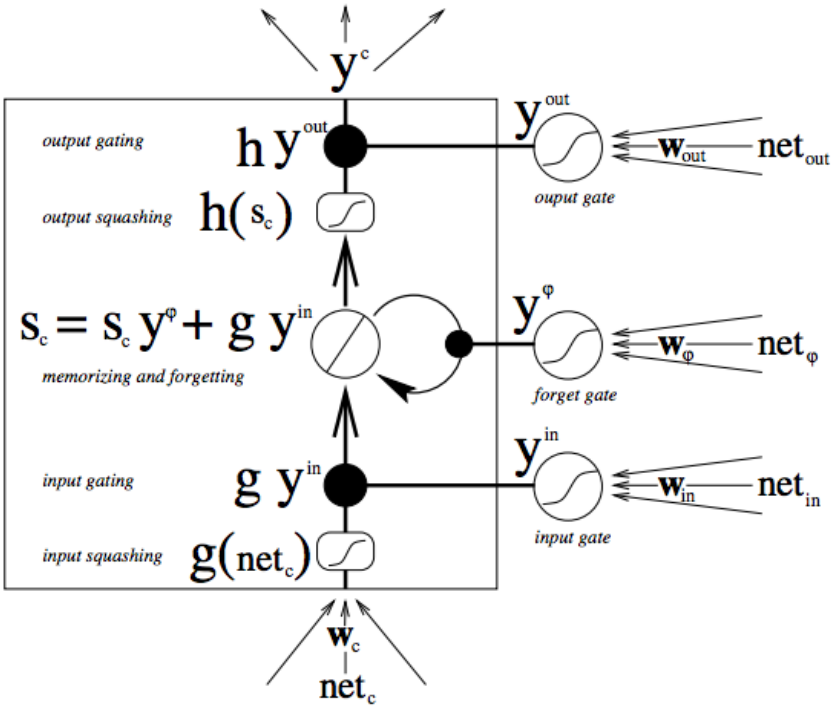
Background

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 [5]. Since the creation of ANNs, these layered connections of nodes have been implementated 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 [6].

In text-based applications of ANNs though, recurant neural networks have become favored over convolutional systems. For instance, Twitter bots like "Deep Drumpf" [7] have been used to demonstrate the powerful effectiveness of recurrant ANNs by emulating the linguistical style and syntax of individual people. Similarly, people have used recurrant ANNs to generate text based on the plays of Shakespeare [8], the essays of Paul Graham [9] and even to generate baby names [10]. Andrej Karpathy, Tesla Motors' leader on artifical intelligence reserach, has stated that some of the reasoning behind the "unreasonable effectiveness of recurrant neural networks" is that with convolutional systems, "their API is too constrained" in that they accept only a fixed-sized vector as input and subsequently produce a fixed-sized vector as output using a fixed amount of computational steps. Unlike these convolutional ANNs, recurant ANNs allow us to operate over sequences of vectors: sequences in the input, output, or even both [11].

To further define and justify the approach to be used for the herein described experiment, it is necessary to define the specific recurant ANN I will be employing: long short-term memory (LTSM). LTSM ANNs are "capable of learning long-term dependencies" [12] and do not start from scratch whenever they are presented with new data. LTSMs were introduced by Hochreiter and Schmidhuber in 1997 [13] and were refined iteratively following their work. The LTSM approach to recurant ANNs is ideal for the type of text-based analysis I will be conducting because by using LTSM 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 LTSM's relative insensitivity to "gap length" gives an advantage to LSTM over alternative recurant neural networks [14]. 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 [15] 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." [15]

Test Cases

In order to test out the capability of a LSTM recurant ANN to work effectively with astronomy-related text, I will subset increasingly large datasets from the metadata available through the Astrophysics Data System. I will use the titles to train the herein described ANN and to generate test titles. I will do this for different subdisciplines within the field. I will gather baseline metrics for each training dataset that I will be able to use to compare differences between "real" training data and "fake" data generated from the LSTM models. The metrics I will gather 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 [16]) character encoding scheme.

Data Sets

I have chosen to extract from the SAO/NASA Astrophysics Data System metadata from three different domains within astronomy and astrophysics. The datasets I have built to train the LSTM artificial neural network are available for download here [17] 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 have only yet performed baseline analyses for training datasets up to 20,000 titles and may not reach a capacity to process 35,000 titles. The three domains within astronomy and astrophysics that I chose to analyse are:

- Black Holes - earliest publication available in ADS is 1799 [18]
- Astrobiology - earliest publication available in ADS is 1870 [19]
- Exoplanets - earliest publication available in ADS is 1943 [20]

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.

Hypothesis

My hypthothesis is that some fields will be more acurately generated by the LSTM ANN than others. The relationship will be correlated with the diversity of unique words used within the training corpus of text representing the subdomain. This is to say that a subdomain with a larger variety of unique words (combinations of letters) will be more challenging to for the LSTM ANN to generate than a training corpus with fewer unique words. I hypothesize that more data will be needed to achieve a high degree of similarity between generated text and training text in some fields than others and that this will similarly correspond to unique word count. I also hypothesize that the ability to generate a similar title to those available to train the model in the original training sets will be different between fields that are more emergent compared with those that are more established. For instance, I hypothesize that the LSTM ANN will 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

Torch [21] is a scientific computing framework that makes available a robust library of Python- and Lua-based machine learning tools. The Torch implementation of LSTM recurant neural network models is straighforward and works to ensure that programmers, "have maximum flexibility and speed" in building scientific algorithms [21]. The biggest pro 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 am 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 is 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 means 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 accelleration are also helpful. Standard instructions for installing Torch on various operating systems are available here [22]. I do not have access to GPU resources locally and followed Torch's standard OSX installation instructions [23]. Unfortunately this means that the amount of time that it takes to train my LSTM models is extensive and serves 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 compuation along, however, it is still possible to run the herein described tests without cloud computing.

Data Analysis Full Workflow Demonstration

The below code cells 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 screenshots (running Torch in the notebook is impractical and will vary between local operating systems). To reproduce the torch analysis below without needing to install Torch or create a separate virtual environment, it is possible to make use of Torch's various docker images [24]. The final code cells are used to show the training of the model and subsequent sampling of data resulting from the training. I then demonstrate how cosine similarity can be calculated between a randomly sampled "fake" title generated by the LSTM model and the training corpus itself.
This demonstration is meant to establish the feasibility of the study and show that these methods can be used to explore the hypotheses outlined herein. The work of the remainer of the project will be to compare these analyses across disciplines and training dataset sizes. Visualizations will be created as part of the final draft of this project to illustrate whether or not my hypotheses can be confirmed as predicted.

Data Extraction and Preprocessing

The below scripts represent and example of how data is extracted from the NASA Astrophysics Data System API and converted into a text files for subsequent processing and analysis using Torch.

Pulling Data from the Astrophysics Data System

Example for 1,000 most recent "Black Hole" titles

In [35]:
```
# 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', 'abstra
ct'], 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 will be used to specify the length of the title that Torch will generate. In this case we will ask Torch to generate "fake" titles that are 85 characters long.

Calculating Average Proportions of Non-ASCII Characters in Training Data

In [36]:
```
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 co
ntains 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-ASCII
s:", actual_nonascii_1K
```

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

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

Calculate Total Number of Unique Words in Training Set

In [37]:
```
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
```

I have hypothesized that having more unique words in a training set will impact the ability for the model to accurately generate "fake" titles. This is because the LSTM recurrant ANN will generate the titles one character at a time, and I hypothesize that more unique words will impact the system's ability to predict upcoming characters.

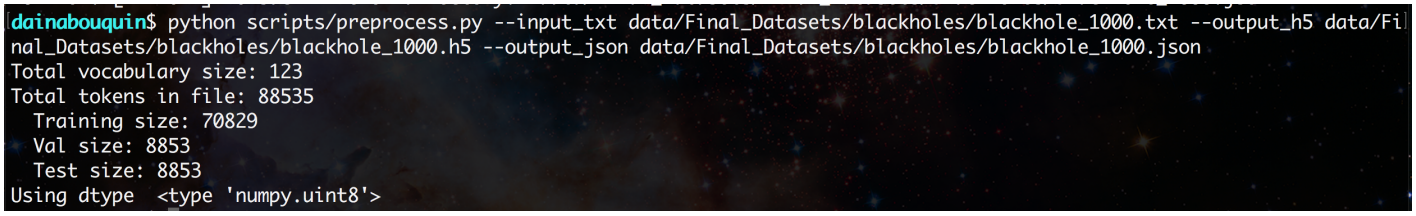
Now that we have built our training dataset made up of 1,000 "black hole" titles from the ADS and calculated the average length of a title, number of unique words in the training set, and average number of non-ASCII titles in a title when non-ASCII characters are present, we can begin processing the 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 [25] called an H5 file, which is designed to organize and store very large amounts of data.

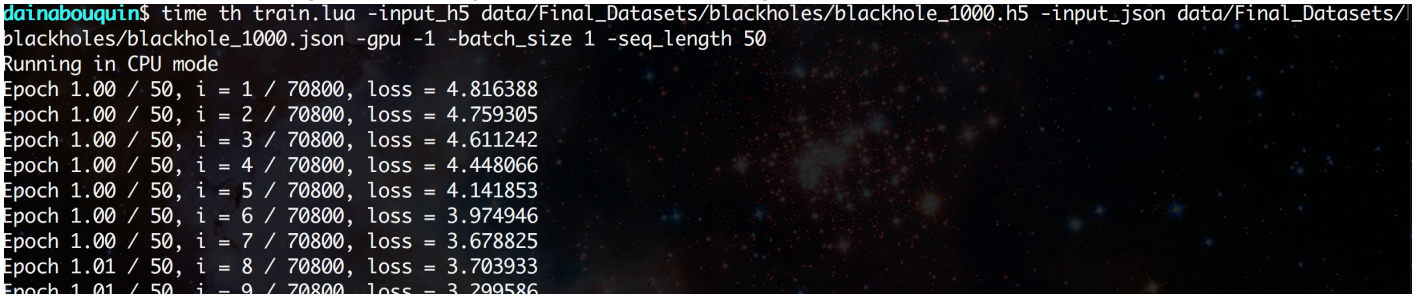
To create the two necessary files for training the LSTM ANN, Torch makes available preprocessing Python scripts can can easily be run in the terminal that are available here [26]. It is important to make sure that when running the preprocessing scripts that your file directory structure matches the designated structure (i.e. run the preprocessing code while in the torch-rnn directory: `~/torch/torch-rnn`). The below screenshot shows how I can run the preprocessing scripts and generate the needed JSON and H5 files. The output from the preprocessing script also prints metrics representing the training data.

Note, this screenshot results from applying the preprocessing scripts to the final "blackhole_1000.txt" dataset [27], rather than from the newly run code demonstration written here. The code written here pulls the most recent 1,000 papers on black holes from ADS, which will differ from my original analysis.



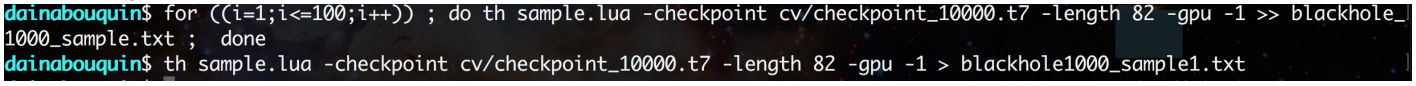
Training the LSTM Model Using Torch

We can now take advantage of the library of code available through Torch to train our LSTM recurrant ANN:



Creating a Test Sample using the LSTM Model Generated by Torch

The script run in the screenshot below does two things: it generates a single example title, and it also builds a file made up of 100 example titles generated by the model. You can see 1,000 example titles generated by Torch using the 1,000 title training set of black hole titles here [28].



Assessment

The following code cells demonstrate how cosine similarity calculated on tf-idf vectors will be used to assess the effectiveness of the LSTM model. Similarly, I will also demonstrate an assessment involving the proportion of non-ASCII characters in training and sampled datasets.

Cosine Similarity

Now that we have processed our data, trained our model, and generated sample titles, we can apply various measures to assess how the model faired in generating titles that are representative of the training titles themselves. To do this we will take advantage of tools that have become popular and impactful in the field of digital libraries. First we will need to use the measure "term frequency-inverse document frequency" (tf-idf), which is a numerical statistic that is intended to reflect how important a word is to a document in a corpus of text [29]. Tf-idf is a measure that is already employed by the ADS in topic modeling and visualizations available through the ADS "explore" features in their "Bumblebee" interface. This value will allow me to take advantage of both cosine similarity and a vector space model to determine how similar two titles actually are to each other.

Below I will show how I can compare a random title that was generated by my LSTM network to the corpus of real titles on which the model was trained. I did this with four different datasets of increasing size for illustration here [30].

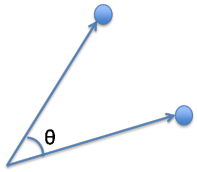
In [48]:
```
# import modules needed for tf-idf and cosine similarity calculations
import math
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

tfidf_vectorizer = TfidfVectorizer() #term frequency inverse document frequency

# import test file
# ** the first line of the test file is randomly selected title
# generated by LSTM model that was trained on 1000 black hole titles **
blackhole_test1K = [line.rstrip('\n') for line in open('blackhole_1000_test.tx
t')]
df_blackhole_test1K = pd.DataFrame(blackhole_test1K)
df_blackhole_test1K.columns = ['Title']

# convert title column to tuple
title_tuple = tuple(list(df_blackhole_test1K['Title']))

# create TFIDF matrix
tfidf_matrix = tfidf_vectorizer.fit_transform(title_tuple)

# calculate the cosine similarity of sample title compared to the whole trainin
g set
# Compare the first title (the test title) to all other titles
cosine_sim_array = cosine_similarity(tfidf_matrix[0:1], tfidf_matrix)

# Show most similar title in degrees
cos_sim = np.partition(cosine_sim_array.flatten(), -2)[-2] # second largest val
ue in array (most similar title)
angle_in_degrees = math.acos(cos_sim)
most_similar1K = math.degrees(angle_in_degrees)

# find index of most similar title
itemindex = np.where(cosine_sim_array==cos_sim)
itemindex = itemindex[1]
itemindex = itemindex[0]

# find most similar title and print it along with first title used for comparis
on
comparison_title = title_tuple[0]
most_similar_title = title_tuple[itemindex]

print "LSTM ANN Generated Title:"
print comparison_title
print "-------------------------------------------"
print "Most Similar Real Title:"
print most_similar_title
print "-------------------------------------------"
print "Cosine Similarity Angle:", most_similar1K
print "-------------------------------------------"
```

```
LSTM ANN Generated Title:
αheodiation in black hole atoveriom durings in rateriations in variability in
 rela  "Dynamic collapses of relativistic degenerate stellar cores and radiati
on pressure dominated stellar interiors"
-----------------------------------------
Most Similar Real Title:
"Stellar populations, stellar masses and the formation of galaxy bulges and di
scs at $z < 3$ in CANDELS"
-----------------------------------------
Cosine Similarity Angle: 78.4887491248
-----------------------------------------
```

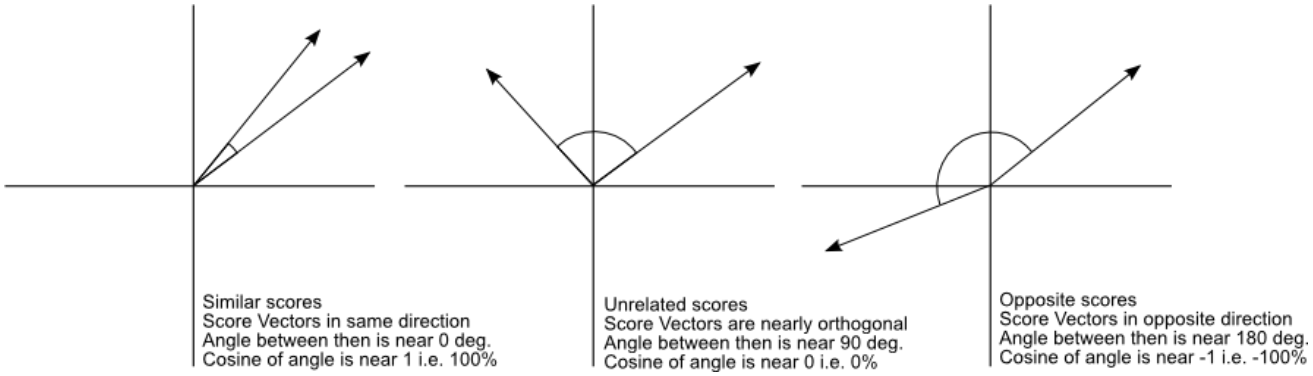
Cosine Similarity Explained

Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them [31] and is commonly applied in settings where tf-idf vectors are being compared. Cosine similarity can be calculated thusly:

$$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$



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.



The angle calculated between the test case above comparing a randomly selected title generated using the LSTM model trained on 1,000 real black hole titles to the corpus of training titles itself shows that the sampled title is measurably similar to the training data (~78.5 degrees). Although there is much room for improvement, this proof of concept will act as the foundation on which I will scale my study.

Analysis of Non-ASCII Characters

The below code analyses the results of the training and sampling of text from the LSTM ANN from the perspective of the presence or absence of non-ASCII characters. For this analysis I will make use of 100 LSTM generated titles and compare the proportion of non-ASCII characters per title containing non-ASCII characters to that of the training corpus itself.

In [50]:
```
# Remove unnecessary white space and read data
test_lines100 = [line.rstrip('\n') for line in
open('blackhole_1000_sample.txt')]

# Find titles with non-ascii characters
nonascii_test_lines100 = []
for line in test_lines100:
  a = re.sub('[\x00-\x7f]', '', line)
  nonascii_test_lines100.append(a)

# Drop empty elements in the list (titles with no ascii characters)
nonascii_notempty_test100 = filter(None, nonascii_test_lines100)
count_nonascii_notempty_test100 = len(nonascii_notempty_test100)

# Calculate average number of non-ascii characters in a title when the title co
ntains non-ascii characters
actual_nonascii_100 = sum([len(i) for i in nonascii_notempty_test100])/count_no
nascii_notempty_test100

print "Average count of non-ASCII characters per LSTM generated title with non-
ASCIIs:", actual_nonascii_100
```

```
Average count of non-ASCII characters per LSTM generated title with non-ASCII
s: 37
```

As you can see from the above analysis, when an LSTM generated title contains non-ASCII characters, it contains far more non-ASCII characters than the training titles do. This analysis was done using sample text from an LSTM model trained on 1,000 real black hole titles. To supplement this analysis I have demonstrated here [32] how the proportion of non-ASCII characters gets increasingly realistic as the training size of the data increases.

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 be used to test the hypotheses I have layed out for this project. The limitations of this approach are not inherent to the process, but instead are linked to the capabilities of the hardware I am employing to train the neural network and the bandwidth throttling of the Astrophysics Data System API, which times out at attempts to pull datasets greater than 35,000 article records. My personal computer has proved incapable of running processes to train models using datasets greater than 20,000 titles. LSTM neural networks are inherently slow to train despite their effectiveness.

Conclusions

The above described analysis can be effective in examining the complexity inherent in different subfields of astronomy and astrophysics. Once scaled to include analyses of text generated using larger training datasets from other subfields, I will be able to test my hypotheses by making analytical and visual comparisons between the fields in question. The LSTM ANN approach does work as desired, however, training time is slow and would be more practical with additional computing resources. Future analysis using cloud infrastructure may be explored as a addendum to this project.

Next Steps

The next steps for this project are to execute the above workflow at a larger scale. I will compare the above results to the results of the same analyses using training sets from three different domains within astronomy with increasingly large datasets to train my models (1K, 5K, 10K, and 20K titles). I will then be able to make generalizations about how the LSTM does in emulating text from three fields and infer ideas about how the fields differ.

References

[1] https://github.com/dbouquin/DATA_698/blob/master/DATA698_proposal.Rmd (https://github.com/dbouquin/DATA_698/blob/master/DATA698_proposal.Rmd)
[2] https://dl.acm.org/citation.cfm?id=38295 (https://dl.acm.org/citation.cfm?id=38295)
[3] https://ui.adsabs.harvard.edu/ (https://ui.adsabs.harvard.edu/)
[4] http://astrothesaurus.org/ (http://astrothesaurus.org/)
[5] http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html (http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html)
[6] https://arxiv.org/abs/1409.4842 (https://arxiv.org/abs/1409.4842)
[7] http://www.deepdrumpf2016.com/about.html (http://www.deepdrumpf2016.com/about.html)
[8] https://github.com/RuthAngus/Fakespeare (https://github.com/RuthAngus/Fakespeare)
[9] https://suriyadeepan.github.io/2017-02-13-unfolding-rnn-2/ (https://suriyadeepan.github.io/2017-02-13-unfolding-rnn-2/)
[10] https://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/nlp/corpora/names/ (https://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/nlp/corpora/names/)
[11] https://karpathy.github.io/2015/05/21/rnn-effectiveness/ (https://karpathy.github.io/2015/05/21/rnn-effectiveness/)
[12] https://en.wikipedia.org/wiki/Long_short-term_memory (https://en.wikipedia.org/wiki/Long_short-term_memory)
[13] http://www.bioinf.jku.at/publications/older/2604.pdf (http://www.bioinf.jku.at/publications/older/2604.pdf)
[14] https://colah.github.io/posts/2015-08-Understanding-LSTMs/ (https://colah.github.io/posts/2015-08-Understanding-LSTMs/)
[15] https://deeplearning4j.org/lstm.html#long (https://deeplearning4j.org/lstm.html#long)
[16] https://en.wikipedia.org/wiki/ASCII (https://en.wikipedia.org/wiki/ASCII)
[17] https://github.com/dbouquin/DATA_698/tree/master/Training_Datasets (https://github.com/dbouquin/DATA_698/tree/master/Training_Datasets)
[18] https://ui.adsabs.harvard.edu/#abs/1799AllGE...4....1L/abstract (https://ui.adsabs.harvard.edu/#abs/1799AllGE...4....1L/abstract)
[19] https://ui.adsabs.harvard.edu/#abs/1870Natur...3...86M/abstract (https://ui.adsabs.harvard.edu/#abs/1870Natur...3...86M/abstract)
[20] https://ui.adsabs.harvard.edu/#abs/1943MNSSA...2...14V/abstract (https://ui.adsabs.harvard.edu/#abs/1943MNSSA...2...14V/abstract)
[21] http://torch.ch/ (http://torch.ch/)
[22] https://github.com/jcjohnson/torch-rnn (https://github.com/jcjohnson/torch-rnn)
[23] http://www.jeffreythompson.org/blog/2016/03/25/torch-rnn-mac-install/ (http://www.jeffreythompson.org/blog/2016/03/25/torch-rnn-mac-install/)
[24] https://github.com/crisbal/docker-torch-rnn (https://github.com/crisbal/docker-torch-rnn)
[25] https://en.wikipedia.org/wiki/Hierarchical_Data_Format (https://en.wikipedia.org/wiki/Hierarchical_Data_Format)
[26] https://github.com/jcjohnson/torch-rnn/blob/master/scripts/preprocess.py (https://github.com/jcjohnson/torch-rnn/blob/master/scripts/preprocess.py)
[27] https://github.com/dbouquin/DATA_698/blob/master/Final_Datasets/blackholes/blackhole_1000/blackhole_1000.txt (https://github.com/dbouquin/DATA_698/blob/master/Final_Datasets/blackholes/blackhole_1000/blackhole_1000.txt)
[28] https://github.com/dbouquin/DATA_698/blob/master/Final_Datasets/blackholes/blackhole_1000/blackhole_1000_sample.t (https://github.com/dbouquin/DATA_698/blob/master/Final_Datasets/blackholes/blackhole_1000/blackhole_1000_sample.
[29] https://en.wikipedia.org/wiki/Tf%E2%80%93idf (https://en.wikipedia.org/wiki/Tf%E2%80%93idf)
[30] https://github.com/dbouquin/DATA_698/blob/master/analysis/cosine_sim_analysis/Cosine_similarity.ipynb (https://github.com/dbouquin/DATA_698/blob/master/analysis/cosine_sim_analysis/Cosine_similarity.ipynb)
[31] https://en.wikipedia.org/wiki/Cosine_similarity (https://en.wikipedia.org/wiki/Cosine_similarity)
[32] https://github.com/dbouquin/DATA_698/blob/master/analysis/ascii_analysis/ascii_exoplanets_vis.ipynb (https://github.com/dbouquin/DATA_698/blob/master/analysis/ascii_analysis/ascii_exoplanets_vis.ipynb)