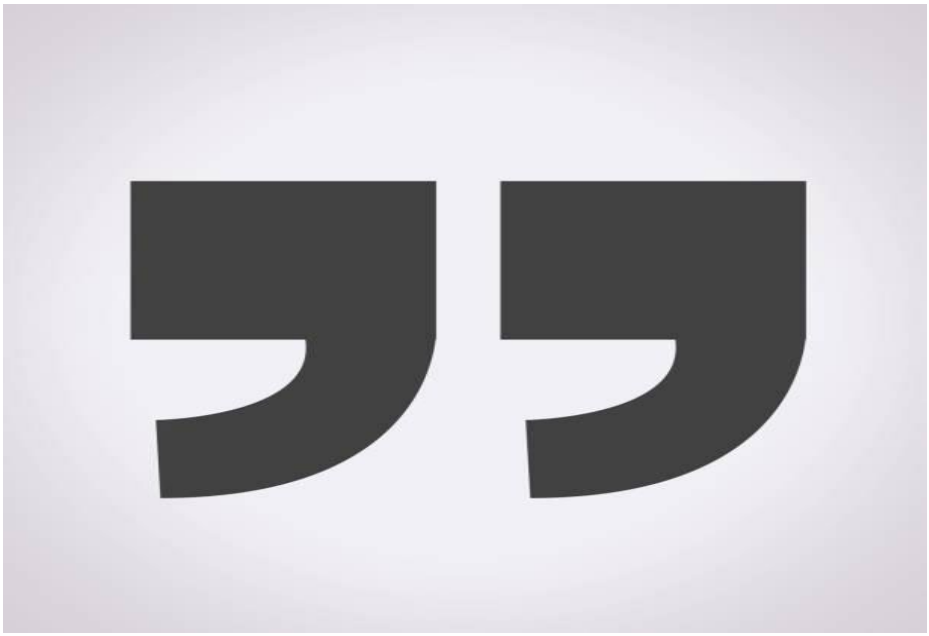
JINX



A Video Transcript Analysis Service

jiNx utilizes IBM Watson Machine Learning APIs and a GPT2 Machine Learning model to analyze social media video transcripts and predict an approximate range of views the poster can anticipate receiving based on the content of the video.

jinx

A VIDEO TRANSCRIPT ANALYSIS SERVICE

IMPLEMENTATION

jiNx 1.0 obtains the transcript of a video from a provided Youtube URL. The transcript is obtained utilizing the IBM Speech to Text service. The transcript is in turn sent to the IBM Tone Analyzer service. The Tone Analyzer service outputs ratings of the emotional tone of the transcript based on these eight emotions: Anger, Disgust, Fear, Joy, Sadness, Confident, Analytical, Tentative. The tone analysis ratings, the video URL, and a current view count are stored as a record in a PostGreSQL database through the IBM Databases as PostGreSQL service.

Once sufficient data has been collected on the PGSQL instance, the database is exported to a local CSV file. The CSV file is utilized as training data for the GPT2 Machine Learning model. Once the model has been sufficiently trained, the model is utilized to predict view count of a test video. The test video is analyzed using IBM Watson Natural Language Understanding to provide corrective insight as necessary to the end user.

Architecture

Logical view:

- IBM Watson APIs are used to parse Youtube video into text and analyze the emotional content of the text
- Thin Client displays the outcome of the analysis from IBM Watson.
- Video analysis outcomes are stored in PostgreSQL database.

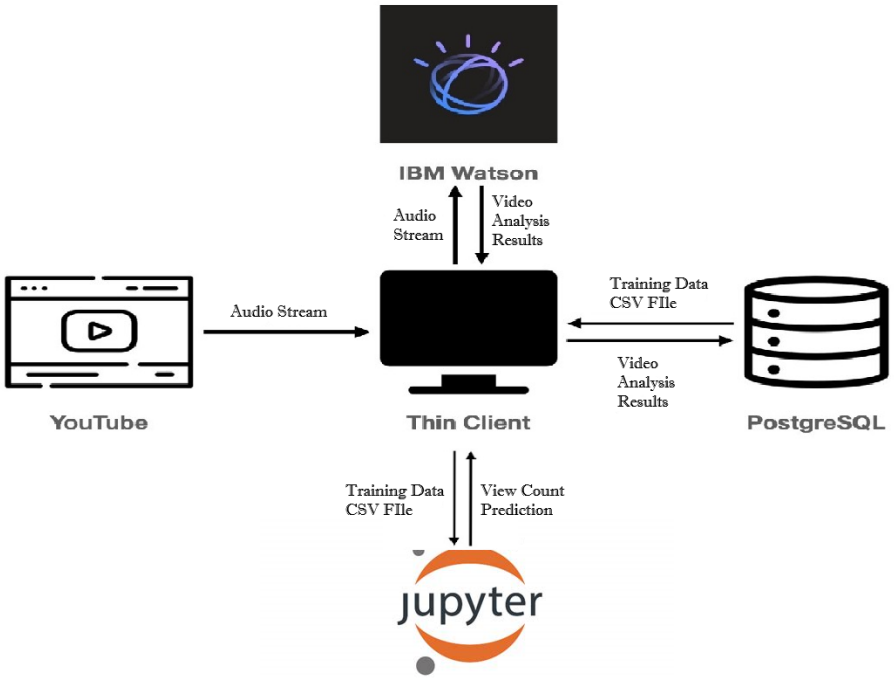
Setup Process view:

- Video analysis of training data automatically sent to PostgreSQL database
- PostgreSQL stores the analysis statistics in TranscriptStats table
- PostgreSQL database automatically exports TranscriptStats table to CSV File
- CSV File used to train GPT2 in Jupyter Notebook
- GPT2 model exported to core code

End User Process view:

- Thin Client gets transcript from YouTube.
- The Thin Client sends transcript to IBM Watson.
- Thin Client receives video analysis outcomes
- Thin Client gets views prediction
- Thin Client displays full video analysis

Physical view:


IBM Watson
Audio Stream
Video Analysis Results
Audio Stream
Training Data CSV FIle
Video Analysis Results
YouTube
Thin Client
PostgreSQL
Training Data CSV FIle
View Count Prediction
jupyter

Development view:

- Programming language: Python.
- IBM Watson Speech to Text API
- IBM Watson Tone Analysis API
- IBM Watson Natural Language Understanding
- Database: PostgreSQL.
- User interface: Flask App thin Client.
- Machine Learning Model: Jupyter Notebook with GPT2 Algorithm and Multivariate Regression

Scenarios:

- User launches the Thin Client on localhost and posts the url to a youtube video, and gets the Video Analysis Statistics displayed on the web client.
- User launches the Thin Client on localhost and posts the path to a locally stored video and get the Video Analysis Statistics displayed on the web client including a prediction for view count.

jiNx is a tongue in cheek reference to "the aforementioned", but with better utilization of inherent characteristics for the purpose of improved functionality. We take what has been said, and make it more informative by leveraging the context of the content for additional insight.

Design Process

jiNx was designed, tested and deployed using a toolkit primarily comprised of open source software. These software include Github, Travis-CI, Codecov, Google Collaboratory, IBM Cloud with the lite plan, and Jet Brains Intellij IDE with the free to students license. The benefit of utilizing this particular combination of resources meant that jiNx could be developed with zero initial financial investment. With time being the only necessary commodity to create this social media prediction platform, the expectation of the organization responsible for the development of jiNx is that the return on investment will be one hundred percent profitable regardless of the level of initial user engagement at the time of release of jiNx 1.0.

Github offers an ideal platform for open source code development via collaboration. Github provides remote storage for repositories for code base throughout the software development life cycle. With the added functionality of Continuous Integration provided by Travis-CI, each time code is committed and pushed to the remote repository, a test prototype is built. The success or failure of these builds help guide development efforts. Added insight is provided by adding the functionality of CodeCov to Github. CodeCov provides statistics on the line coverage once unit tests have been added to the code base. When developing from a Test Driven approach, CodeCov is an incredibly useful resource in making refactoring decisions. Finally, Github provides Kanban boards for projects to assist in organization and collaboration to identify, communicate, and address development activities or issues as they arise.

Paired with Github, the local development environment may be armed with an equally versatile and powerful tool in Intellij. As an integrated development environment, it has an intuitive GUI and is easily interfaced with Github through a quick authentication process that will ultimately make version control accessible through simple push button functionality. This eliminates the need to implement commands through a chosen command line terminal. IntelliJ also provides a local platform to build in-process prototypes before code is pushed to the remote in order to help guide the pace and direction of ad-hoc development efforts.

While developing the code base using local and remote version control and build environs, perhaps the most powerful in the toolkit for jiNx development is Google Collaboratory project. Google provides free of charge access to TPUs and GPUs to train, finetine, and export machine

learning algorithms. Google Jupyter Notebooks were powerful enough to train and finetune the 355 million point dataset utilized in the jiNx GPT2 implementation. In addition, the multivariate regression used to supplement the jiNx predictive analysis platform was designed, trained and exported using Jupyter.

These key development tools are elevated by the diverse framework of resources offered by IBM Cloud. Free of charge when signed up for the lite plan, IBM provides all of the necessities developers need to bring a project to fruition. For jiNx this included key Machine Learning APIs, data management, and a cloud-based container to publish a finished product. IBM Cloud offers a "Why Reinvent the Wheel" solution so that developers can focus on true innovation. While time as a commodity lends negligible financial impact on the software development process, it is, as any commodity, rather limited. IBM Cloud is essentially a time management resource that makes it possible to focus on what's most important.

Product Testing

JiNx employs a unit testing framework that is primarily utilized to ensure data integrity for the backend. Unit tests run checks to confirm that data is held securely in the PostgreSQL database. In addition these tests confirm that stored data can be retrieved successfully.

jiNx test framework is implemented automatically through Travis-CI and test metrics are analyzed using Codecov.

In addition, the build environment is rigorously tested using Travis-CI. Core requirements for jiNx are collected in a document that is utilized in build testing.

Code Review

Code review was conducted using Radon. Radon is a Python library that may be used as a command line tool to obtain static code analysis. Metrics compiled by Radon include Cyclomatic complexity, Raw, Maintainability index, and Halsted.

The raw metrics report the number of lines of code as LOC, the number of logical lines of code as LLOC, the number of source lines of code as SLOC, the number of comments, multi-line strings, and lines of whitespace.

The Maintainability Index provides a qualifier which indicates the relative ease at which the code base can be supported and changed as necessary.

An ideal ranking for Cyclomatic Complexity ranges from A, denoting a low for a simple block of code, to C, denoting moderate, for a slightly complex block of code. Modularized sections of code will be designated by type, where M refers to a method, C refers to a class, and F refers to a function.

The Halstead metric is implemented in an effort to provide a quantification of the code. In addition to a number of descriptive metrics, Halstead analysis reports the number of distinct operators as h1, the number of distinct operands as h2, the total number of operators as N1, and the total number of operands as N2.

A summarization of the Radon output for key classes implemented in jiNx are as follows.

jiNx\youtubePredictor\youtubePredictor_backend

youtubePredictor_dataManager.py

Raw:

LOC: 164

LLOC: 121

SLOC: 138

Comments: 6

Single comments: 6

Multi: 0

Blank: 20

- Comment Stats

(C % L): 4%

(C % S): 4%

(C + M % L): 4%

Maintainability Index: A, very high

Cyclomatic Complexity:

M 116:4 DataManager.import_data_from_file - B

C 23:0 DataManager - A

M 38:4 DataManager.init - A

M 100:4 DataManager.delete_record_from_database - A

M 152:4 DataManager.remove_all_data_from_database - A

C 18:0 YoutubePredictorError - A

M 75:4 DataManager.add_record_to_database - A

M 87:4 DataManager.get_record_from_database - A

M 19:4 YoutubePredictorError.__init__ - A

M 25:4 DataManager.__init__ - A

M 57:4 DataManager.reset_cursor - A

M 62:4 DataManager.get_all_records_from_database - A

M 70:4 DataManager.get_column_headers - A

Halstead:

h1: 2

h2: 3

N1: 3

N2: 5

vocabulary: 5

length: 8

calculated_length: 6.754887502163469

volume: 18.575424759098897

difficulty: 1.6666666666666667

effort: 30.95904126516483

time: 1.7199467369536017

bugs: 0.006191808253032966

jiNx\youtubePredictor\youtubePredictor_databaseBuilder\youtubePredictor_databaseBuilder_analysis.py

Raw:

LOC: 171

LLOC: 118

SLOC: 133

Comments: 9

Single comments: 9

Multi: 0

Blank: 29

- Comment Stats

(C % L): 5%

(C % S): 7%

(C + M % L): 5%

Maintainability Index: A, very high

Cyclomatic Complexity:

M 78:4 YoutubePredictorRecord.set_tones - B

M 54:4 YoutubePredictorRecord.set_tones_helper - B

C 29:0 YoutubePredictorRecord - A

C 107:0 DataBuilder - A

C 24:0 YoutubePredictorError - A

M 97:4 YoutubePredictorRecord.get_average_tone - A

M 108:4 DataBuilder.__init__ - A

M 124:4 DataBuilder.get_view_count - A

M 139:4 DataBuilder.api_manager - A

M 153:4 DataBuilder.create_csv_file - A

M 25:4 YoutubePredictorError.__init__ - A

M 30:4 YoutubePredictorRecord.__init__ - A

M 47:4 YoutubePredictorRecord.initialize - A

M 103:4 YoutubePredictorRecord.get_record - A

M 144:4 DataBuilder.ytp_record_helper - A

Halstead:

h1: 5

h2: 19

N1: 13

N2: 26

vocabulary: 24

length: 39

calculated_length: 92.32026322986493

volume: 178.81353752812512

difficulty: 3.4210526315789473

effort: 611.7305231225333

time: 33.98502906236296

bugs: 0.05960451250937504

jiNx\youtubePredictor\youtubePredictor_databaseBuilder

youtube_databaseBuilder_transcription.py

Raw:

LLOC: 22

SLOC: 22

Comments: 4

Single comments: 4

Multi: 0

Blank: 8

- Comment Stats

(C % L): 12%

(C % S): 18%

(C + M % L): 12%

Maintainability Index: A, very high

Cyclomatic Complexity:

F 12:0 get_urls - A

F 26:0 get_subtitles - A

Halstead:

h1: 1

h2: 2

N1: 1

N2: 2

vocabulary: 3

length: 3

calculated_length: 2.0

volume: 4.754887502163469

difficulty: 0.5

effort: 2.3774437510817346

time: 0.1320802083934297

bugs: 0.0015849625007211565

jiNx\youtubePredictor_frontend

youtubePredictor_flask_wrapper.py

Raw:

LOC: 56

LLOC: 25

SLOC: 35

Comments: 8

Single comments: 8

Multi: 0

Blank: 13

- Comment Stats

(C % L): 14%

(C % S): 23%

(C + M % L): 14%

Maintainability Index: A, very high

Cyclomatic Complexity:

F 24:0 submission - A

F 34:0 analysis - A

jiNx\youtubePredictor\youtubePredictor_frontend

youtubePredictor_form.py

Raw:

LOC: 16

LLOC: 7

SLOC: 7

Comments: 5

Single comments: 5

Multi: 0

Blank: 4

- Comment Stats

(C % L): 31%

(C % S): 71%

(C + M % L): 31%

Maintainability Index: A, very high

Cyclomatic Complexity:

C 13:0 VideoForm - A

jiNx\youtubePredictor\youtubePredictor_frontend\youtubePredictor_logger.py

Raw:

LOC: 41

LLOC: 20

SLOC: 24

Comments: 6

Single comments: 6

Multi: 0

Blank: 11

- Comment Stats

(C % L): 15%

(C % S): 25%

(C + M % L): 15%

Maintainability Index: A, very high

Cyclomatic Complexity:

C 20:0 YoutubePredictorLogger - A

M 21:4 YoutubePredictorLogger.__init__ - A

M 24:4 YoutubePredictorLogger.log_method_started - A

M 28:4 YoutubePredictorLogger.log_method_completed - A

M 32:4 YoutubePredictorLogger.log_info - A

M 35:4 YoutubePredictorLogger.log_warn - A

Product Release

Packaging: @TODO Full description of how jiNx is packaged and how the user runs the package

Distribution: @TODO Instructions on where jiNx can be found and how to download it

Demo: @TODO video demonstration of jiNx

Post Mortem

Why jiNx

Social media prediction is currently based on leveraging metadata. The accuracy of those predictions relies on the assumption that the content itself is an unnecessary source of influence on prediction. jiNx posits that

predicting the response to social media content on the basis of the context of the content can be at least as reliable if not more accurate. In the context of the global posture by which we socialize, we are connected most effectively, efficiently, and broadly through social media. No matter what the message, ensuring that it reaches an ever-widening audience is the key concern of those who utilize social media. Businesses, political and social justice movements, even emergency response coordination relies on social media for the purpose of distribution of key messaging. Predicting whether the message will successfully reach its target audience or determining how to shape a message so that it reaches the broadest possible audience is why jiNx is a timely and useful product.

What Went Wrong

Core Implementation Phase I, jiNx was DiTTo. DiTTo in its first incarnation was intended to be a real-time streaming in-browser video captioning service. The first demo showed that the speech to text functionality was resource-inefficient and there was a near 33% observed inaccuracy of the transcript for word recognition. In addition, jiNx at the time of the demo was only compatible with video files and not streaming video. Given the announcement of the Google Captions Extension on March 18, 2021, it was collectively decided by the development team to shift focus in another direction.

For additional insight, please see:

Aquino, S. (March 18, 2021). Google Announces Live Caption For Chrome For Real Time Captioning. www.Forbes.com

https://www.forbes.com/sites/stevenaquino/2021/03/18/google-announces-live-caption-for-chrome-for-real-time-captioning/?sh=5eb9fdc36db5

After consideration, it was determined that it was not necessary to build a product that scales efficiently for high throughput. A generic or all-purpose approach was not what jiNx would provide in a landscape where it has long been discovered that one size does not fit all. The kind of analysis jiNx would be redesigned for would be infinitely more productive if it could offer user-specific insight after a single use. For those who argue that this is a poor business model, the response is simple, and eloquently devised by Cambridge Analytica:

"Communication has changed. Blanket advertizing no longer provides viable ROI for every campaign. Big data revolutionized the way organizations identify and locate their best prospects. But data alone isnďž˝t enough. Cambridge Analytica is building a future where every individual can have a truly personal relationship with their favorite brands and causes by showing organizations not just where people are, but what they really care about and what drives their behavior." - *https://cambridgeanalytica.org/*

A customer that could finetune jiNx's machine learning algorithm with their own data, is looking for a single outcome that will have a substantial impact across their entire business. jiNx could pivot from being a generic tool that spits out subtitles for any user who opts to download a Google extension to something far more useful.

This was not a popular decision among the original members of the DiTTo team. Three original core members chose not to work on this new product direction, electing to pursue another project entirely.

What Went Right

IBM Cloud Services

IBM Cloud provides APIs free of charge which made it possible for jiNx to be developed as a completely thin client. This means that jiNx can be run by any user, from anywhere without concern about meeting minimum hardware requirements on their local environment.

cloud.ibm.com

GPT2

OpenAI provides an already trained Machine Learning model with extensive support to get up and running with your first tests. Given the abbreviated development cycle, this was a particularly useful solution as there was not sufficient time to choose, train, and test an entirely new Machine Learning model.

https://www.geekslop.com/features/technology-articles/computers-programming/2020/what-is-gpt-2-and-how-do-i-install-configure-and-use-it-to-take-over-the-world

Python/Flask Vs Java/SpringBoot/JavaFx/Gradle/Camel:

During development it was elected to make a shift from a Java software development kit configured to utilize SpringBoot, JavaFx, Gradle and Camel to a Python software development kit utilizing Flask for quicker frontend development. Simply put, at the time jiNx pivoted from a captioning service to a video analysis prediction service, there was concern about technical debt and on-time completion. Developing in Python meant that development activities could proceed at a much faster pace comparatively speaking.

What We Would Do Differently

The true potential of jiNx as a prediction service is in its ability to leverage the power of a massive language prediction model. With an opportunity to utilize GPT3, jiNx could evolve from simply predicting views to something far more impactful for social media influencers. The addition of GPT3 would allow jiNx to move in the direction of predicting what specific content results in increased views. A key scenario would be for video content to be analyzed and the result of that analysis would be in the form of a suggested transcript for the user to read/perform in their video for a predicted percent increase in view count.

For additional insight, please see:

Dormehl, L. (June 12, 2020). OpenAI's GPT-3 algorithm is here, and it's freakishly good at sounding human. www.DigitalTrends.com
https://www.digitaltrends.com/features/openai-gpt-3-text-generation-ai/

and

https://gpt3examples.com/#examples

jiNx developer

Anthem Rukiya J. Wingate is an exceptionally polyvalent Bioinformatics Applications Engineer currently developing python, C++ and Java-based software customizations for the integration of biochemical analysis instrumentation and Chromatography Data Systems with SLIMS, a web-based Library Information Management System a product of Agilent

Technologies. Anthem has extensive Instrumental and Analytical Chemistry experience which has informed her current career path to develop Machine Learning and Data Science solutions for bioinformatics implementations.

https://www.linkedin.com/in/anthemrukiya/

References

Quotation Marks Stockphoto
https://media.istockphoto.com/vectors/quote-sign-icon-vector-id653858496?k=6&m=653858496&s=612x612&w=0&h=CG0IUEBNpUD5mgtMqvsbf59paD3zPTw-9ckEzimOlok=