Advertisement detection system using machine learning techniques.

Interim Report

DT211c

BSc in Computer Science (Infrastructure)

Jack Duggan

C16350866

Brian Gillespie

School of Computer Science

Technological University, Dublin

09/12/19

Abstract

This project revolves around the detection of advertisements using machine learning means, creating a system that can apply these means and the evaluation of that system. The motivation behind this system is the turmoil in advertisement deliverance. The use of advertisements to track users as they navigate the internet and data privacy concerns that arise because of this. Further research into this space has revealed advertisement blockers use the same technology to complete their task. New methods of completing this task should be developed.

This report describes how this system was designed and how the design will be implemented. The back-end will utilise two main machine learning algorithms, word2vec and the inception model. The system will be used through a chrome plugin implemented in JavaScript. The middle layer will then connect these algorithms and front-end.

The final system will be a chrome plugin that highlights advertisements on a website and presents a metric for how sure the system is.

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

____[illegible]_________

Jack Duggan

09/12/19

Acknowledgements

I would like to acknowledge the support I have received during the undertaking of this project.

My supervisor, Brian Gillespie, for guiding this project.

My family for their ability to translate my English into readable English.

And finally, my fellow students for the sense of comradery and support during my time at TUDublin.

Table of Contents

1. Introduction

1.1. Project Background

In January of 2019, Google proposed a change in their extension manifest version (v3) (1). The point of this update is to make extensions safer by adding greater permission controls. This however had the knock-on effect of “killing” ad block support according to Raymond Hill, the maintainer of uBlock origin and uMatrix (2). A conclusion therefore can be made that extensions cannot be used reliably to protect privacy while browser creators have a vested interest in creating targeted ads.

On the 25th of May 2018, the EUs General Data Protection Regulation (GDPR)(3) came into effect. GDPR has made it mandatory for websites to ask for consent to use the user’s data for personalizing content and ads. The problem here is often if you do not agree, you are unable to access the content on the website. In other countries the user is not given the option and consent is implied by the user using the service. This makes adblock necessary to protect the privacy of users who want to use such websites.

Obseravtions were made of the turmoil between advertisement suppliers and advertisement blockers, as a result research to find other solutions to the adblock problem commenced. Pi-hole, a dns sinkhole for advertisements appeared as a promising solution. This program is simple to install but requires an always on server. The problem with this solution is it required prior knowledge of computing where as browser plugins are a 1-click install meaning they are far simpler for the greater public to use.

1.2. Project Description

To solve the problems mentioned above, a solution was proposed where the system uses a convolutional neural network model to Identify images in a website. The output will then be compared against what the text on the page is about. To compare the text and image classification output, word2vec a word embedding 2-layer neural network will be used decide what the text is about. If there is a miss match, we can assume that the suspect image is an advertisement. As time goes on advertisement providers are getting craftier or have the ability to remove the impact of the from ad blocking software, as a result more ads are slipping through the cracks. This system would get around these issues by not having ad lists that must be maintained.

1.3. Project Aims and Objectives

The goal of this project is to decide if a website has adverts or not. To achieve this objective the images and text on the page will be classified. The outcome of these two machine learning algorithms will then be compared against each other. This will be done by grouping these words based on what they are. For example, a dog is in the canine category were as classes will be in an item category. These are two separate groups, so it is likely the image is an advertisement. In this project, the concept will be refined and tested as a valid method of detection.

To achieve this a number of objectives must be completed. Step one is to gather relevant technology's and test their suitability for the systems application.

1.4. Project Scope

This project is about ad detection. Ad removal is out of scope for this project, the system will inform the user on how likely an image is or is not an advertisement. For the purpose of this system, the scope will be narrowed to the type of advertisements and text the system will deal with. This can be expanded depending on how accurate the system is at predicting the likely hood of the image being an advertisement. Starting with a small number of advertisement types will allow testing and validate if this is a relevant method of checking for advertisements.

As an application the system must be lightweight and fast. The interface should follow the concepts of universal design.

1.5. Thesis Roadmap

This section will provide a summary of each of the chapters covered in this report.

Research

This chapter delves into existing methods of ad detection. Following this, an examination into current/existing technologies and data that will enable the completion of this project. Finally, this chapter will discuss any other research done for this project.

Design

This chapter explains the design and overall layout of the system. An explanation each layer of the system will be explained. This will include UML diagrams.

Development

The development chapter goes through the entire development process of the prototype system. The middle-tier and backend systems that where discussed in the Design and Research chapters are implemented here.

Testing and Evaluation

In this chapter I discuss the testing methodologies used and define the parameters for success.

Redevelopment

This chapter goes onto explain what steps must be taken to incorporate the technologies used in the prototype into the final system.

Conclusions and Future Work

This chapter reflects on the completed prototype system and research completed. The conclusions drawn from this work will be used to propel the development of this system.

2. Literature Review

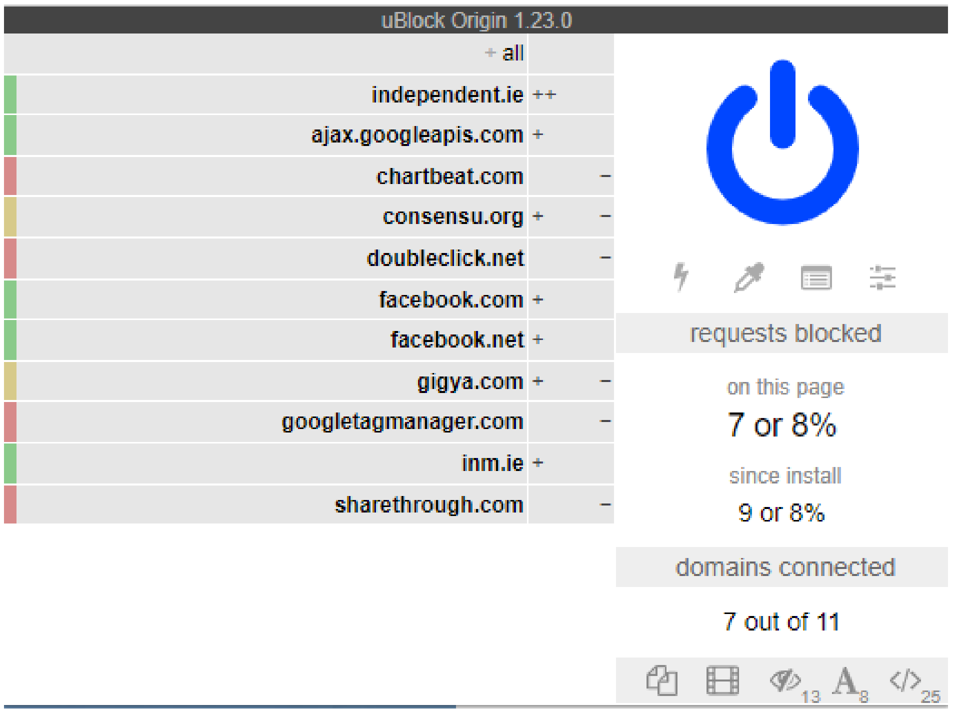
2.1. Introduction

In this chapter, key technologies to the success of the system will be discussed. A Comparison to alternative existing solutions to this problem will be made.

2.2. Alternative Existing Solutions to Your Problem

uBlock Origin

An efficient blocker: easy on memory and CPU footprint, and yet can load and enforce thousands more filters than other popular blockers out there (4). This is a complex interface for an app of this type. This is because the domains of each ad that are blocked are displayed. Control is limited to an on and off button. The app also has html element removing functionality.



uBlock Origin 1.23.0

	+ all		
🟩	independent.ie	++	
🟩	ajax.googleapis.com	+	
🟥	chartbeat.com		−
🟨	consensu.org	+	−
🟥	doubleclick.net		−
🟩	facebook.com	+	
🟩	facebook.net	+	
🟨	gigya.com	+	−
🟥	googletagmanager.com		−
🟩	inm.ie	+	
🟥	sharethrough.com		−

requests blocked

on this page: 7 or 8%

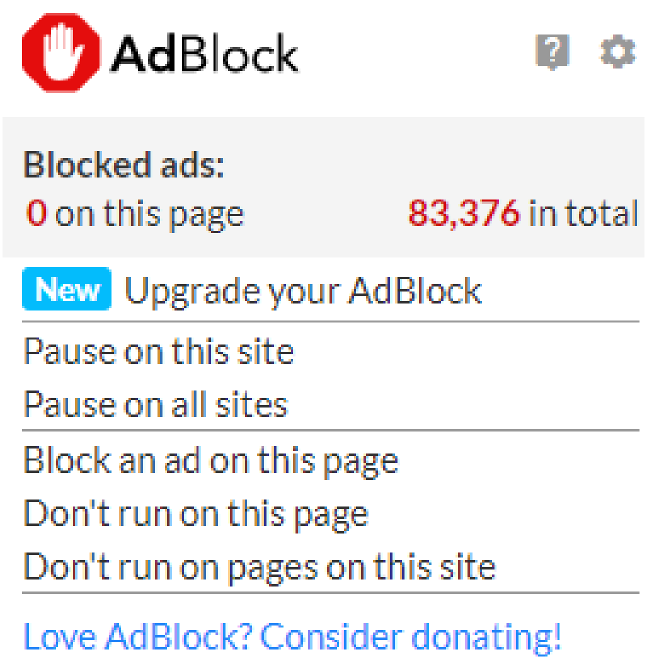
since install: 9 or 8%

domains connected: 7 out of 11

Figure 1- uBlock origin interface

AdBlock

AdBlock (5) does exactly what it says on the tin. It supports the mainstream web browsers as well as android and iOS. The user interface displays the least information but allows the users to pause block and control running in one simple page.


AdBlock

Blocked ads:
0 on this page 83,376 in total

New Upgrade your AdBlock

Pause on this site
Pause on all sites

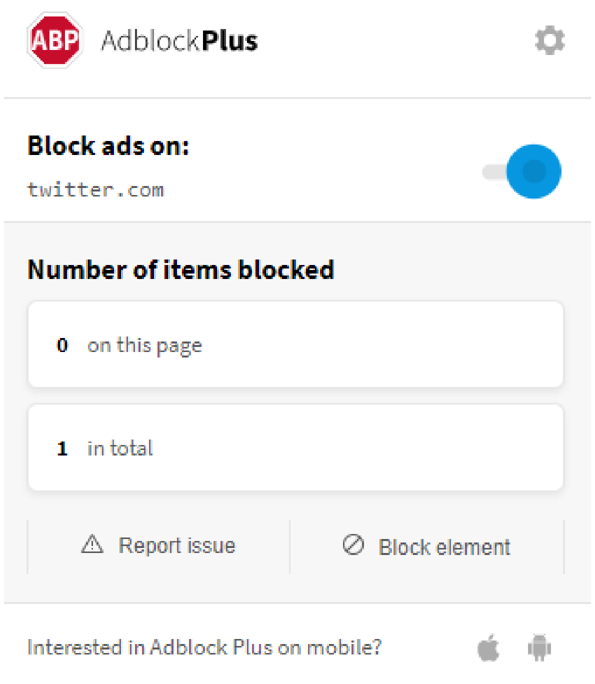
Block an ad on this page
Don't run on this page
Don't run on pages on this site

Love AdBlock? Consider donating!

Figure 2- AdBlock interface

AdBlock Plus

AdBlock plus (6) provides a distraction free browsing experience in the same was as others in this list. AdBlock plus is open-source and gives users access to privacy controls. The interface is very simple, this is a toggle for blocking ads on the current website and a display for advertisements blocked.


AdblockPlus

Block ads on:
twitter.com

Number of items blocked

0 on this page

1 in total

Report issue Block element

Interested in Adblock Plus on mobile?

Figure 3- AdBlock Plus

Pi-hole

Pi-hole (7) is a DNS sinkhole that is used to provide network wide adblocking. This is another system that uses static lists to find and block advertisements. The system is harder to setup then its counterparts as it is hosted on an internal server and client DNS setting must be changed to point to the internal server. The system can also double as a DHCP server.

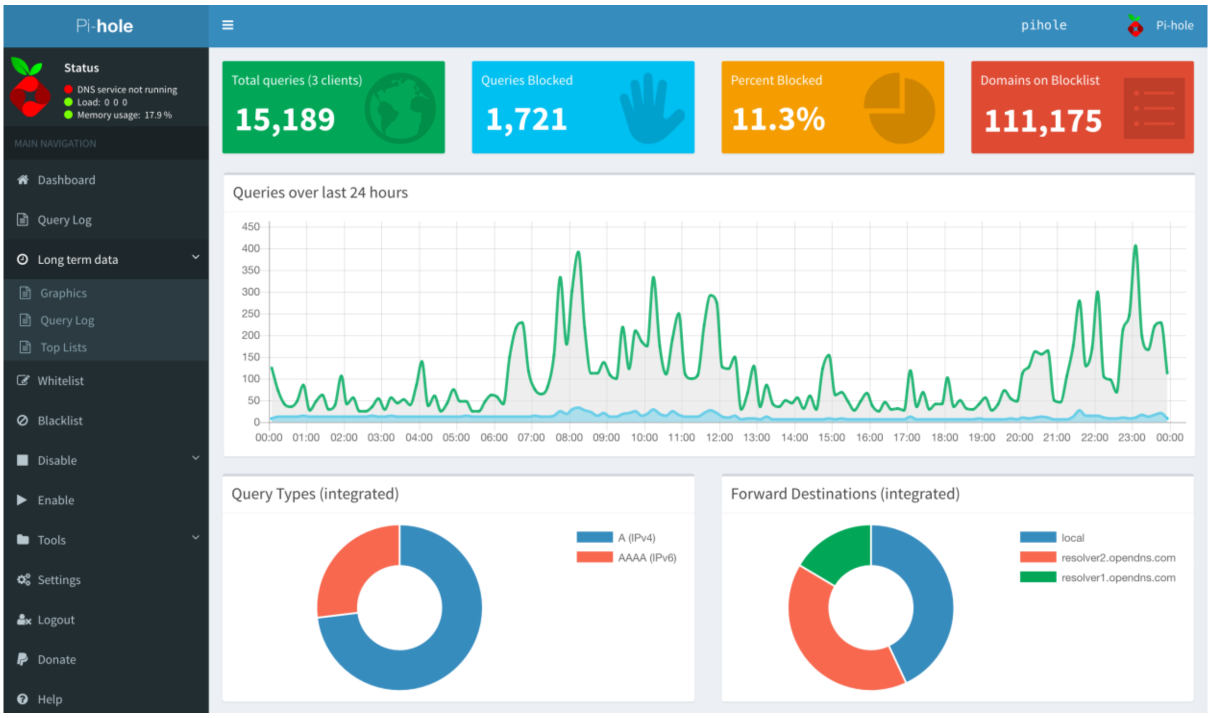


Figure 4- Pi-hole dashboard

Conclusion

These programs all use a variation of static lists to find adverts. The most common are plugins for common browsers i.e. Firefox, chrome, etc. All these solutions have very simple interfaces that allow the user to see the total number of advertisements blocked on the site. They also allow the user to enable and disable on the fly. All the plugin mentioned have 10,000,000+ users on google chrome making this a very sought-after functionality.

2.3. Technologies you've researched

The following are possible platforms:

Chrome

Chrome is a popular web browser (8). Chrome currently has about 60 percent of the browser market according to statcounter and w3counter. These statistics are collected from all websites that use W3Counters or statcounter, these websites collect telemetry using JavaScript allowing insights into the website's performance. Google are a major supplier of advertisements, so have a vested interest in crippling the current systems (9).

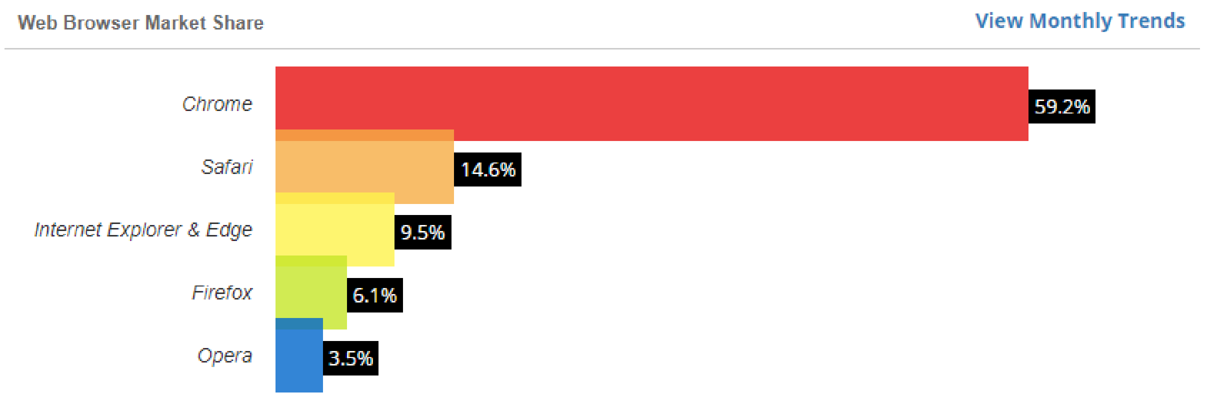


Figure 5- W3counter report

Firefox

Firefox (10) does not have anywhere near the popularity of chrome. Firefox has a reputation for user privacy and in 2019 have rolled out features such as a VPN and have removed plugins that collect excessive user data. By default, 3rd party tracking cookies and crypto miners are blocked. Firefox is backed by a non-profit.

Conclusion

In conclusion, chrome is the market leader for web browsers. A method that does not uses static lists will be most useful on this platform as this is where the advertisement skirmish is happening. Firefox actively supports user privacy meaning that if the system is successful it should be ported to this platform in the future.

The following are possible programming languages:

Java script

This is the go-to language for extension development along with html and CSS. Java script can be used to modify the DOM on websites this will be used to highlight or completely remove the ads.

Python

Python is a high-level interpreted language that emphasised code reliability. Python supports many open source frameworks, library's and development tools. Python also allows for rapid prototyping due to the small size of the code. Applications like conda or virtualenv allow for python programs to be isolated from other projects and help manage their dependencies.

C++

C++ is a general-purpose compiled language. C++ is used mainly in systems programming and embed systems because of its features and security. C++ has support for library's and frameworks.

Conclusion

In conclusion, python's ability to produce rapid prototypes and mature library's make it a good language for this system. Python is often used in machine learning and web development tasks.

The following are possible library's and models:

Word2vec

This is a method of converting words to a machine-readable format. Normally to deal with categorical features like words one must embed them into a numerical format but in doing so it loses how the words relate to each other. Word2vec preserves the relationship between the words. Word2vec comes in two flavours, the skip-gram method of word2vec weighs nearby context words more heavily than more distant context words and Bag of words which the order of context words does not influence prediction. (11)

Inception

Inception is a convolutional network that classifies images. Unlike its competitors it goes wider instead of deeper by having multiple convolutions on the same level. Using a pre trained model as opposed to building a new model makes sense when trying to solve this problem as these models are optimized and trained well past anything one person can manage on their own. (12)

WordNet

WordNet® is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. (13) The linking of the items in the wordnet database and our ability to see what Synset the word is part of will allow the system to make meaningful comparisons between the output of the convolutional neural network and text analysis systems

TensorFlow Lite

This allows TensorFlow models on mobile, embedded, and IoT devices (14). Lite does this by using pre-trained models. Theses models are designed to execute efficiently on mobile and other embedded devices with limited compute and memory resources. TensorFlow Lite has a python implementation this should allow me to use python as the backend of my extension/website.

The following are possible web frameworks:

Flask and flask restful

Flask (15) is a micro web framework for python that provide the means to create a web application. It is free and open source. Flask restful builds on this functionality allowing developers to create REST APIs. A rest API may be necessary for this system depending on what portions of this project can be ran in browser. Flask is a micro framework meaning that functionality like logging in and Restful APIs are added in separate libraries or must be implemented by the developer.

Django

Django is a web framework for python that includes all the functionality needed to scale and secure a website. Django like flask is free and open source but is more feature rich out of the box.

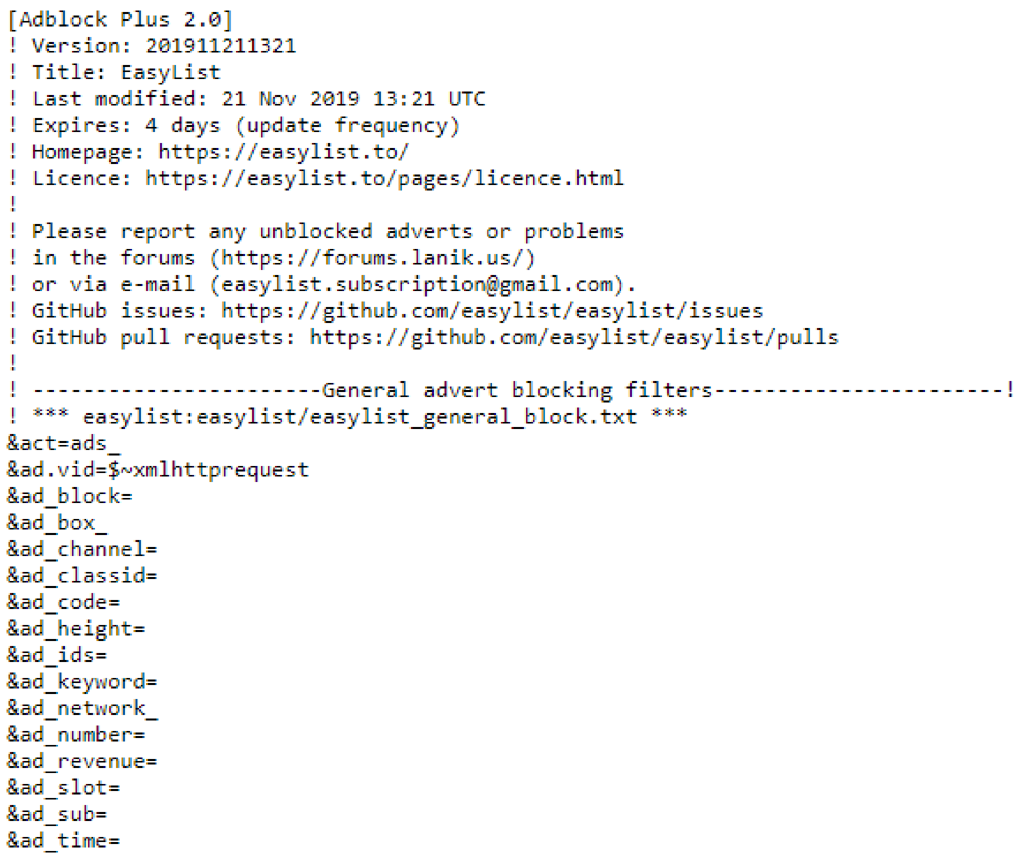
Conclusion

Flask is a far simpler and more flexible as it sets out to be a mini web framework and nothing more. Where as Django aims to be an all-inclusive platform. As of now I only require a method of communicating between the middle layer and backend because of this the lighter simpler platform will be better for this project.

2.4. Other Research you've done

Ad lists

These are lists made and maintained by the AdBlock community. A popular Ad list, EasyList (16) comes pre-packed and/or tested with many adblockers (AdBlock Plus, AdBlock and uBlock Origin). EasyList is a primary filter list for removing most adverts from international webpages, including unwanted frames, images and objects. It is the basis for many other filter lists.



```
[Adblock Plus 2.0]
! Version: 201911211321
! Title: EasyList
! Last modified: 21 Nov 2019 13:21 UTC
! Expires: 4 days (update frequency)
! Homepage: https://easylist.to/
! Licence: https://easylist.to/pages/licence.html
!
! Please report any unblocked adverts or problems
! in the forums (https://forums.lanik.us/)
! or via e-mail (easylist.subscription@gmail.com).
! GitHub issues: https://github.com/easylist/easylist/issues
! GitHub pull requests: https://github.com/easylist/easylist/pulls
!
! -----------------------General advert blocking filters-----------------------!
! *** easylist:easylist/easylist_general_block.txt ***
&act=ads_
&ad.vid=$~xmlhttprequest
&ad_block=
&ad_box_
&ad_channel=
&ad_classid=
&ad_code=
&ad_height=
&ad_ids=
&ad_keyword=
&ad_network_
&ad_number=
&ad_revenue=
&ad_slot=
&ad_sub=
&ad_time=
```

Figure 6- EasyList

2.5. Existing Final Year Projects

Here are some final year projects from previous years that are relevant to this project. These projects are both machine learning projects and as such share some similarities to this project in their development style, technologies and methods used.

Title: Euro Coin Classification. Using Image Processing & Machine Learning

Student: Yumin Chen

Description (brief):

This project aims to investigate the visual features of euro coins and develop models that can represent different natural images. The objective of this project is to investigate the use of machine learning specifically in the context of euro coin classification.

What is complex in this project

Classifying the coin is the difficult part of the project. This is done using both image processing and machine learning. The student implements different algorithms and evaluates them.

What technical architecture was used

OpenCV, Cordova, machine learning and data mining techniques.

Explain key strengths and weaknesses of this project, as you see it.

The student tested a multitude of ml algorithms and evaluated them against each other. I see this as a strength. The student uses Rapid Application Development methodology, prototypes were made quickly but the problem of immediately starting to code is an issue. This weakness in the development style, extra care should be taken in the planning stage for this sort of project.

Title: Detecting Bot Twitter Accounts using Machine Learning

Student: Emmet Hanratty

Description (brief):

In this project the student creates a system to detect and evaluate whether a user account is a bot. The system manages good accuracy, and this will be increased as the data set expands. They system also creates graphs on the data used in prediction.

What is complex in this project

Training a sentiment model based on tweet data. This decides if a tweet is positive, natural or negative.

What technical architecture was used

Python using the Django framework, MySQL, Twitter API and Git

Explain key strengths and weaknesses of this project, as you see it.

The project is not multi-threaded meaning it is slow in the pre-processing stages. Twitter have a limit on the amount of calls you can make to their API, meaning after a certain number of requests the system is unusable for the rest of the day.

The system has good accuracy when predicting the state of an account and this accuracy will grow over time.

2.6. Conclusions

From looking at the current popular solutions for advertisement blocking we can see they all employ a similar approach for this task. With certain companies having a vested interest in curtailing this technology for their own profit and said companies having the means of doing so other solutions should be developed. To develop a solution to this problem possible technologies were researched and existing Final year projects were looked at to provide guidance.

3. Prototype Design

3.1 Introduction

In this chapter, possible UIs, overviews and back-ends for this system will be designed.

3.2. Software Methodology

Waterfall Methodology

The waterfall model (17) is a phase by phase method of developing software. The current phase must be completed before commencing the next phase. Once a phase is finished it cannot be revisited till the end of the project.

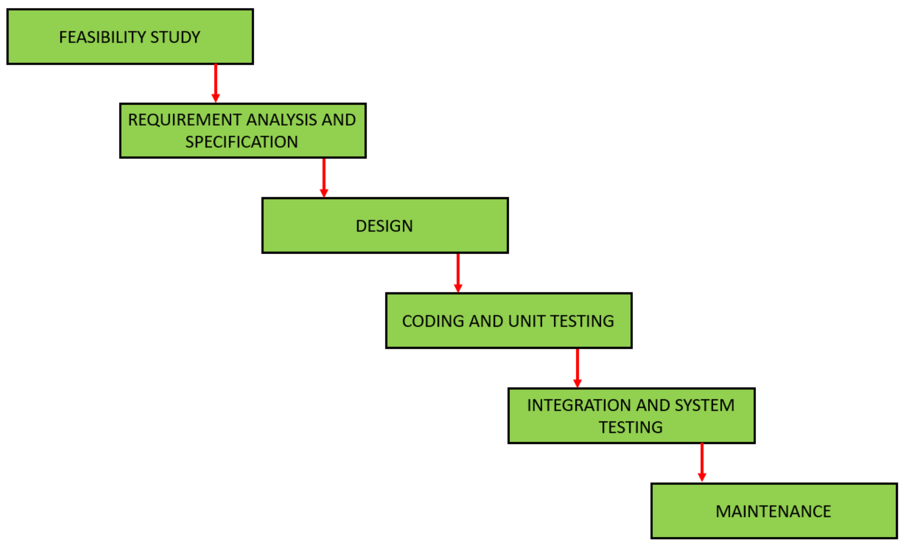

FEASIBILITY STUDY
REQUIREMENT ANALYSIS AND SPECIFICATION
DESIGN
CODING AND UNIT TESTING
INTEGRATION AND SYSTEM TESTING
MAINTENANCE

Figure 7- Waterfall model

An advantage of this model includes clear and understandable milestones. This makes the stages of the process and advancements very clear.

A disadvantage of the waterfall model is that is difficult to redevelop and redesign when the goalpost changes. Once started you are locked into this mode of development till the maintenance phase. On completion of the project, the application may or may not be suitable for its purpose.

Rapid application development

Rapid application development (18) is a form of agile software development. This methodology will allow rapid testing and creation of new approaches.

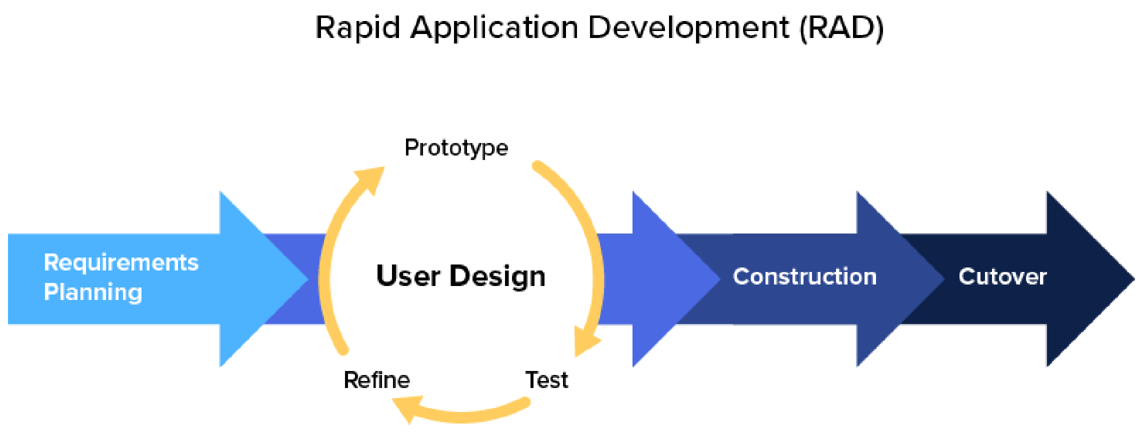


Figure 8- Rapid Application Development

The main advantage of the RAD methodology is that time between prototypes and their next iteration is short. This allows for rapid development over a shorter period.

The disadvantages of RAD are mainly based around applying the methodology to larger teams. As this is a solo project this is not a problem. Another disadvantage of this methodology is it is not suitable for larger projects.

3.3. Overview of System

Using the Rapid application development method of software design the first step is define what the system aims to accomplish, this has been completed in part 1.3 of this report. The second stage of RAD is to prototype designs and develop these prototypes. The general approach to the prototype phase was to divide the system into manageable pieces. These pieces were then built, refined and tested. After Unit testing which will serve as the feedback, these pieces will be grouped together to create the final system.

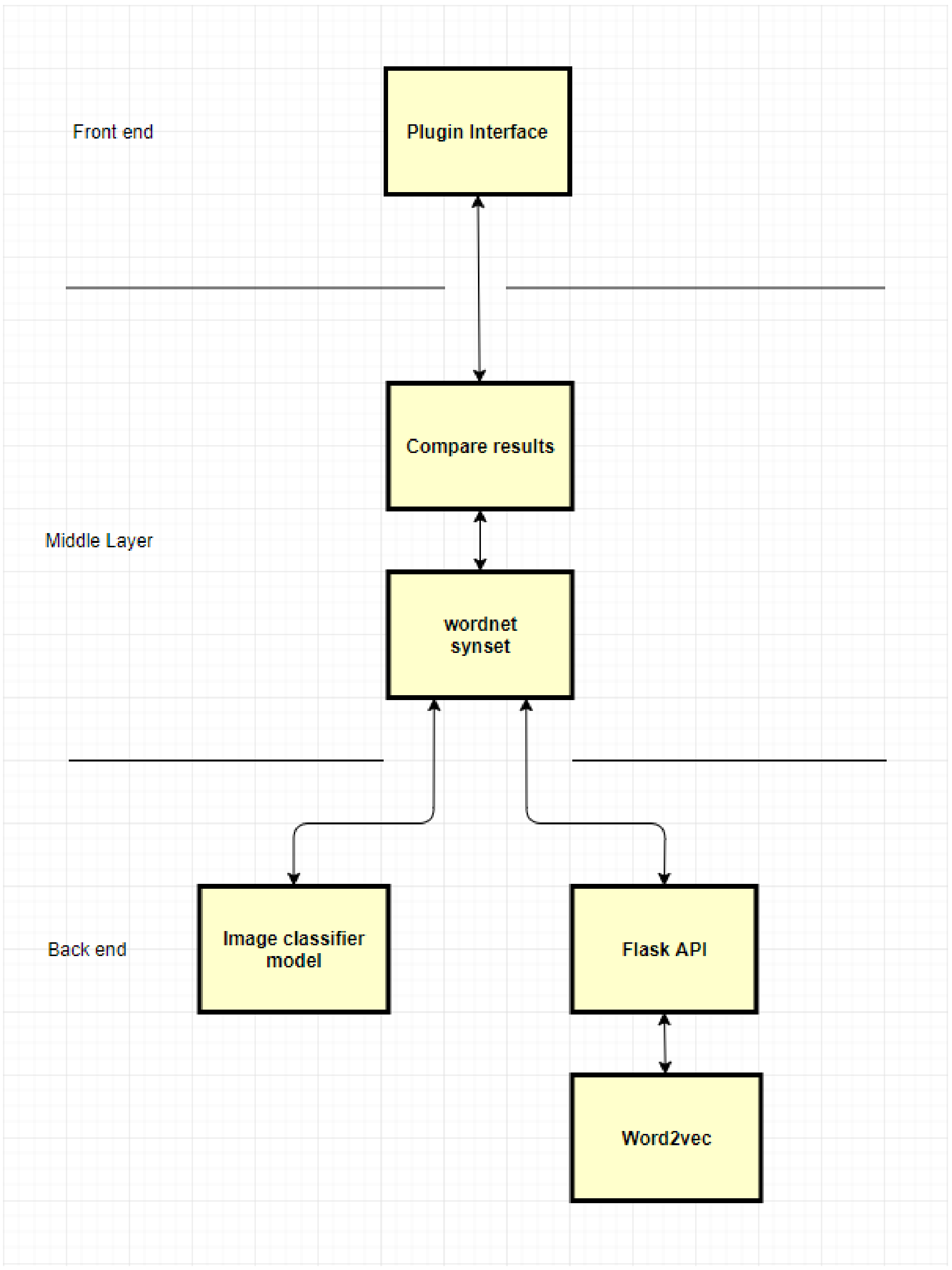


Figure 9 - Overview of the system

3.4. Front-End

The final system will have a front-end but for the purposes of the prototype this is not the case. The final system will have a similar front end to its alternative solutions. This will be written in html, CSS and JavaScript as these are the default languages used to write extensions. Across these solutions, the number of ads blocked, a way to stop and restart the application are universal. Below are possible designs for the interface of the application.

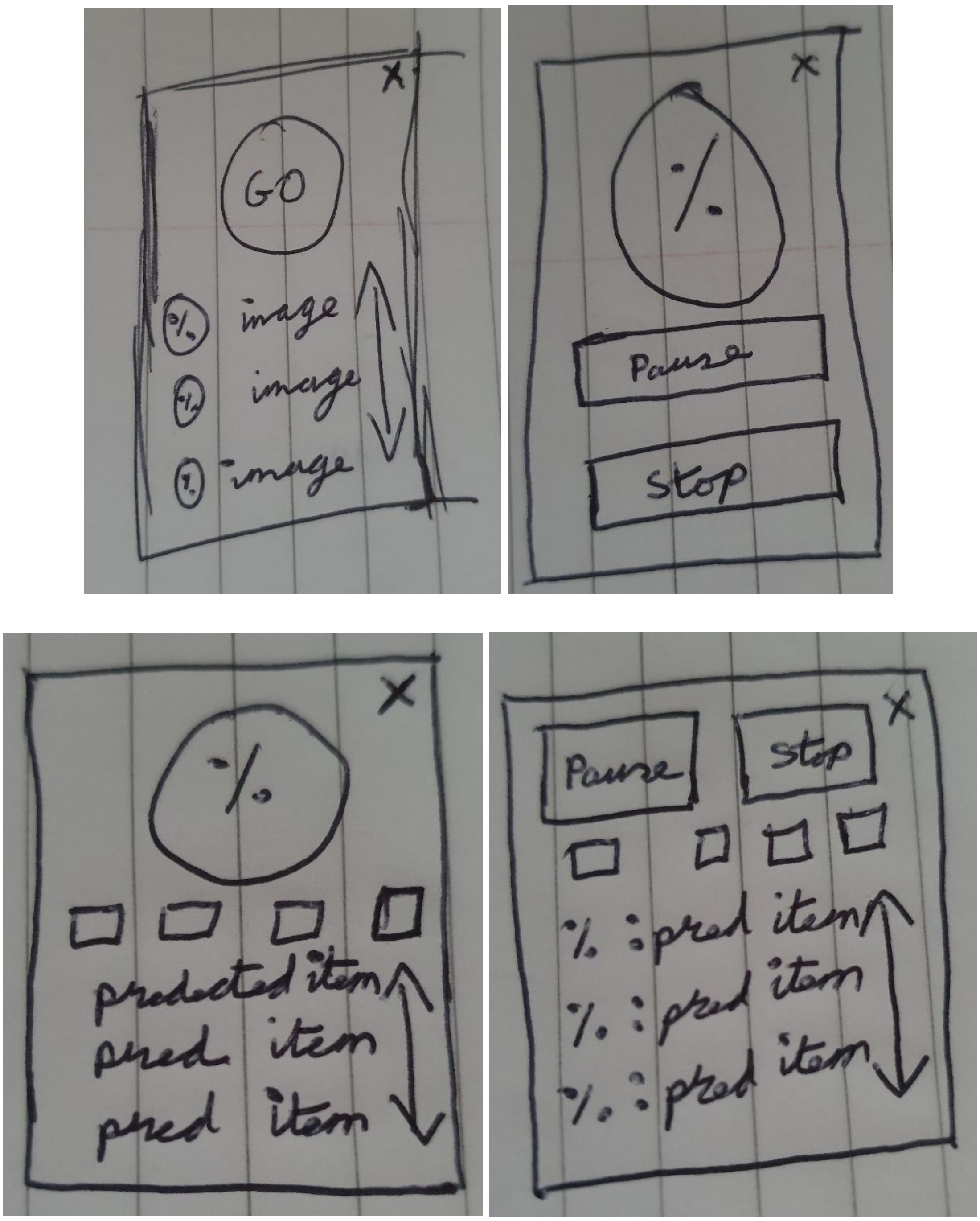


Figure 10 – Paper prototype

The next step was to create medium fidelity prototypes. The front end will we simple with one to two different pages. The prototype displays the functionality of the interface and its overall layout. A simple easy to understand interface will server this system the best as it will need to be accessible to all types of users.

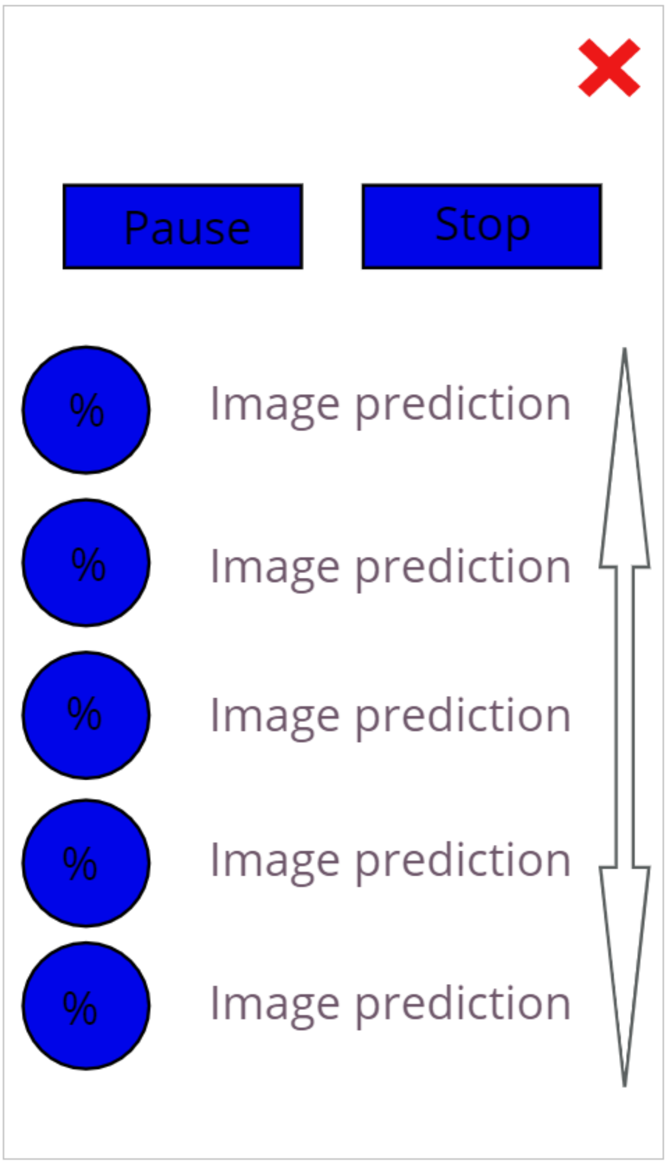


Figure 11 - Medium Fidelity Prototype

3.5. Middle-Tier

This layer will be responsible for parsing the websites sent in by the user and passing the necessary data to the backend. This software will grab the main paragraphs and advertisements on the website, it will do this by using the Beautiful Soup library or some other means. Beautiful Soup is a Html parser and possesses functions to find specific elements. Most advertisements are served through JavaScript and this functionality will allow the system to get the required images.

The middles tier will also compare the outputs of the backend systems using wordnets database and generate a value for how sure the system is that the images on the site are or are not advertisements. Flask is a web framework for python, this library will allow the word2vec implementation to be queried from the client.

3.6. Back-End

The backed will consist of 2 systems, image classification and word embeddings.

Image classifier

The image classification system will be based on a retrained inception mode inception. This model is already trained on ImageNet making its predictions already quite accurate, retraining this model for even more common items that we see in advertisements will give an even better result. Training for logos is another option letting us specialize the classifier even more.

Below is a possible structure for the image prediction. In the final version this will run in browser using TF lite.

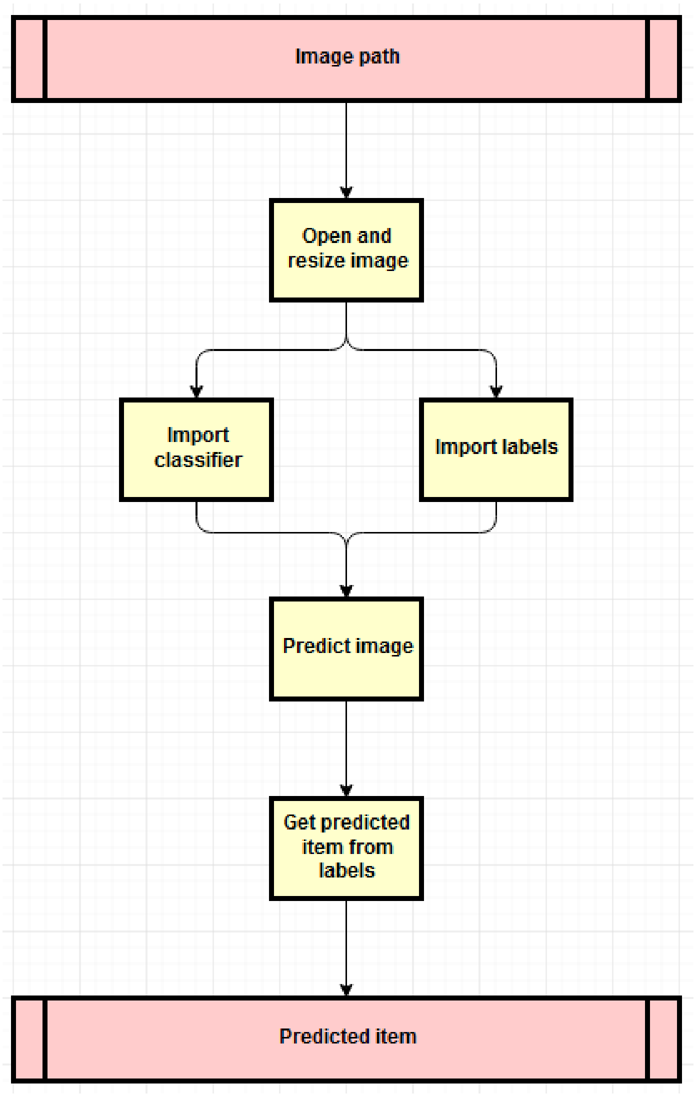


Figure 12 - Image Classification Design

Word2vec

The word2vec (19) portion of the back end will run on a server as it is to intensive to run on the clients. Each new document will have to be trained separately as each document will produce a different set of vectors. Word2vec outputs word embeddings by predicting the neighbouring words given a word. Word2vec is a 3-layer neural network which consists of an input layer, a hidden layer and an output layer.

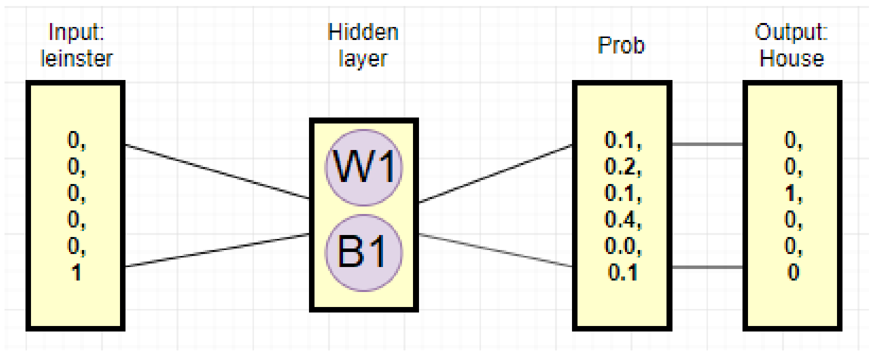


Figure 13- Word2vec structure

Training the model to predict the next word will never give high accuracy at the task. But the hidden layer which is the vectors used to predict the outcome give us the relationship of the words to each other.

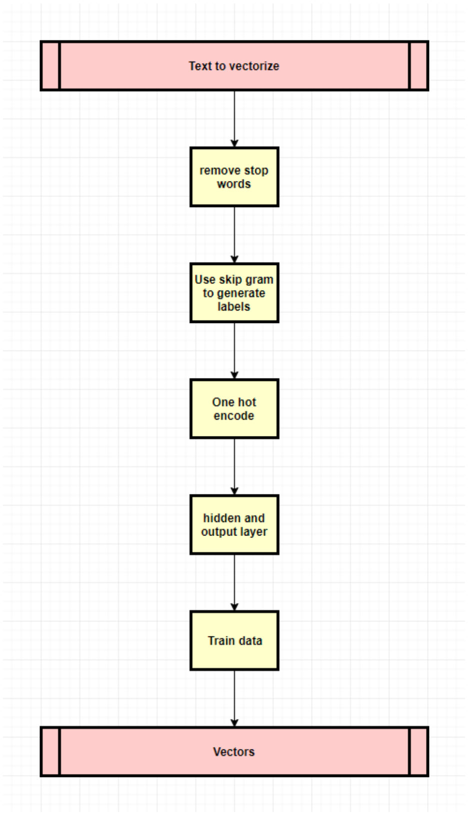


Figure 14 - word2vec Design

3.7. Conclusions

In this chapter, the two main back end technologies were discussed and an overall design was given for each of them. The middle tier will be responsible for connecting the backend to the frontend and translating the gathered information into meaningful data. The design of the UI was explored and fleshed out.

4. Prototype Development

4.1. Introduction

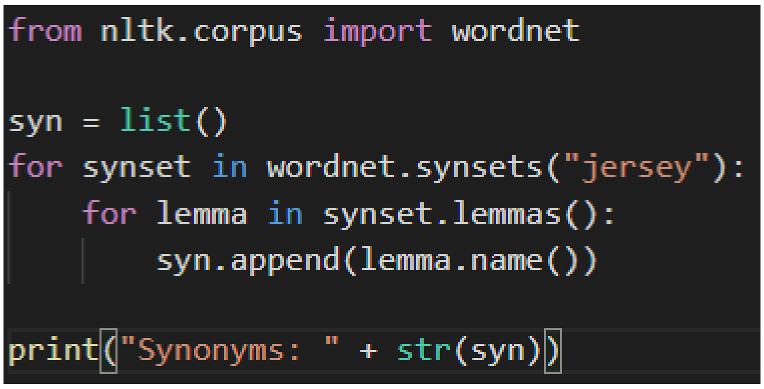
From my research and design I have decided on a few technologies to use for this system. These technologies may change depending on how the prototype develops.

4.2. Prototype Development

The first step to developing any system is to set up version control. This project is being managed through GitHub.

4.3. Middle-Tier

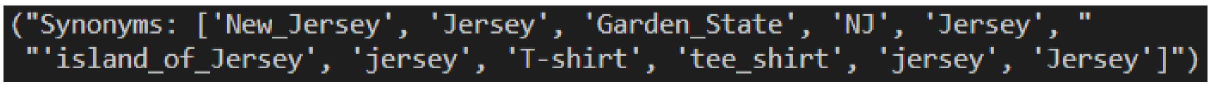
Word net has an implementation in pythons NLTK library. Using this library, the system will be able to get the Synonyms of words allowing comparison between the word2vec server and the prediction of the image.



```
from nltk.corpus import wordnet

syn = list()
for synset in wordnet.synsets("jersey"):
    for lemma in synset.lemmas():
        syn.append(lemma.name())

print("Synonyms: " + str(syn))
```

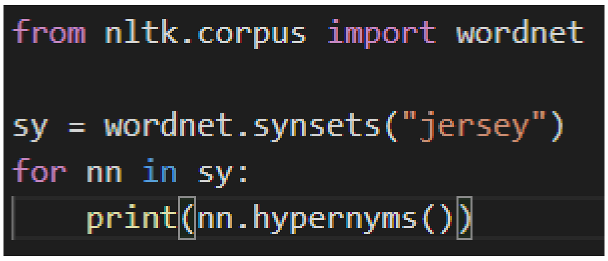


```
("Synonyms: ['New_Jersey', 'Jersey', 'Garden_State', 'NJ', 'Jersey', "
 "'island_of_Jersey', 'jersey', 'T-shirt', 'tee_shirt', 'jersey', 'Jersey']")
```

Figure 15- wordnet implementation and output

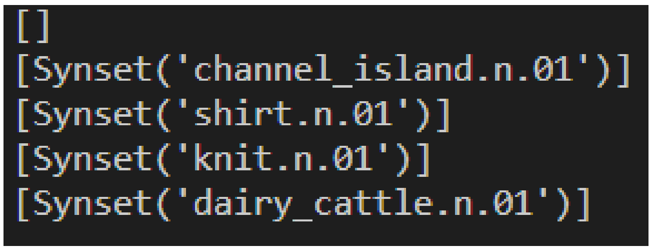
As you can see in the above samples, wordnet allows the system to know that Jersey is both a shirt and a place. This functionality will allow for better comparisons between the backend systems.

Where synonyms fail a fall-back solution could be synsets, these are the semantic relations of the words that are arranged hierarchy.



```python
from nltk.corpus import wordnet

sy = wordnet.synsets("jersey")
for nn in sy:
    print(nn.hypernyms())
```



```
[]
[Synset('channel_island.n.01')]
[Synset('shirt.n.01')]
[Synset('knit.n.01')]
[Synset('dairy_cattle.n.01')]
```

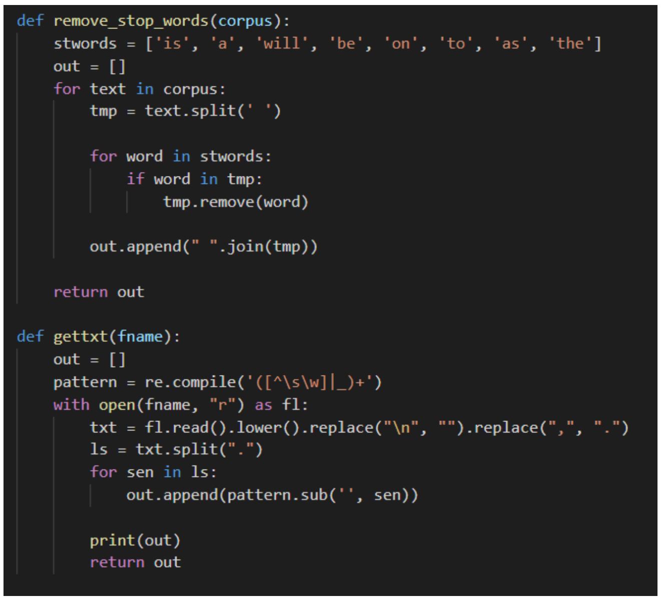
Figure 16 - Synset example output

4.4. Back-End

Word2vec

The systems current Word2vec implementation works as follows:

1. Format, clean and remove stop words from text.



```python
def remove_stop_words(corpus):
    stwords = ['is', 'a', 'will', 'be', 'on', 'to', 'as', 'the']
    out = []
    for text in corpus:
        tmp = text.split(' ')

        for word in stwords:
            if word in tmp:
                tmp.remove(word)

        out.append(" ".join(tmp))

    return out

def gettxt(fname):
    out = []
    pattern = re.compile('([^\s\w]|_)+')
    with open(fname, "r") as fl:
        txt = fl.read().lower().replace("\n", "").replace(",", ".")
        ls = txt.split(".")
        for sen in ls:
            out.append(pattern.sub('', sen))

        print(out)
        return out
```

Figure 17 - Stop words and clean text

2. Convert unique words to int and generate skip gram data.
3. Make hidden layer.
4. Train model.
5. Assemble vectors to words.

Image classification

The current Image classifier implementation is using the inception v3 model. By using this model with no extra training on my part the effectiveness of inception at identifying items.

1. Open and format image.

```
image = Image.open("top.jpg").resize(IMAGE_SHAPE)
image = np.array(image)/255.0
```

2. Import classifier and labels.

```
classifier_url = "https://tfhub.dev/google/tf2-preview/inception_v3/classification/4"
classifier = tf.keras.Sequential([
    hub.KerasLayer(classifier_url, input_shape=IMAGE_SHAPE+(3,))
])
```

```
labels_path = tf.keras.utils.get_file('ImageNetLabels.txt','https://storage.googleapis.com/download.tensorflow.org/data/ImageNetLabels.txt')
imagenet_labels = np.array(open(labels_path).read().splitlines())
```

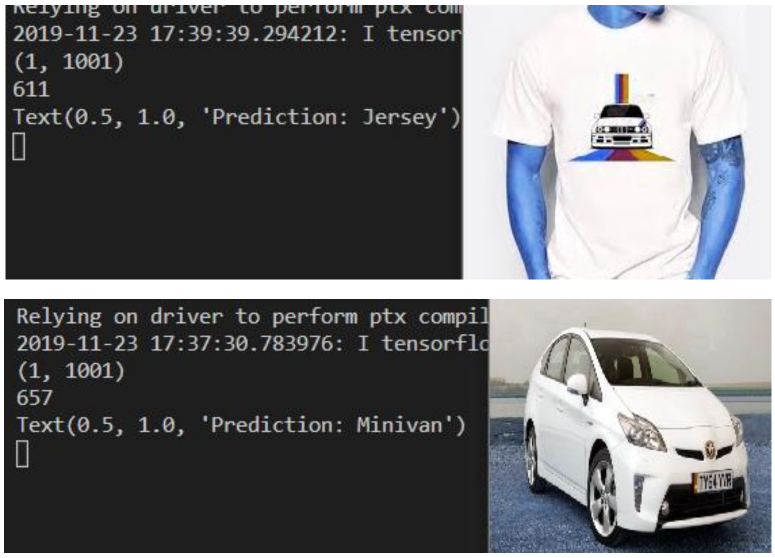
3. Predict image.

```
result = classifier.predict(image[np.newaxis, ...])
predicted_class = np.argmax(result[0], axis=-1)
```

4. Match predicted value with label.

```
predicted_class_name = imagenet_labels[predicted_class]
_ = plt.title("Prediction: " + predicted_class_name.title())
print(_)
```

Example output:



```
Relying on driver to perform ptx com
2019-11-23 17:39:39.294212: I tensor
(1, 1001)
611
Text(0.5, 1.0, 'Prediction: Jersey')
```

```
Relying on driver to perform ptx compil
2019-11-23 17:37:30.783976: I tensorflo
(1, 1001)
657
Text(0.5, 1.0, 'Prediction: Minivan')
```

Figure 18 - Example output of image classifier

4.5. Conclusions

In conclusion, the prototype development has allowed to test and refinement of core parts of the system. In the continuation of this project, building on the technologies developed here should allow for a successful project.

5. Testing and Evaluation

5.1. Introduction

This section will explain the testing and evaluation of the final system. This project will be tested using black box, grey box and white box testing. It will go into depth into the use of unit tests and integrated testing. The overall system will be tested using Black box testing.

5.2. Plan for Testing

As the project progresses, white box testing will be done on each of the modules. If necessary, these modules will be strung together to related components of the program. Otherwise the module tests will happen as a stand-alone entity. This is a rapid prototyping strategy, allowing for fast iterations and refinement of the system.

Extensive unit testing will save time in the long run. An automated approach will allow the testing to be more extensive. There are multiple python library's and external programs that automate unit testing.

When the system progresses to an integrated version integration testing will be done. Testing the single units as a group in a bottom up approach will allow for the lowest risk as modules tested independently will not allow function correctly together. The bottom up approach to integration testing means that the lowest level will be tested first. The problem with this method is most modules must be ready at the same time, but this method allows the testing of the most important and complex modules first.

To test the entire system, Black box testing will be utilized. Black box testing is a type of testing where the tester does not know the internal structure of the application. This method of testing is useful for all types of testing but here it will be used for Acceptance testing. This type of testing determines if the system reaches its requirements.

Test plan

Test no.	Description	Expected output	Outcome
1	Access and get website content	The system will get advertisement images and main text content.	
2	Convert to synonyms and synsets	They system can convert words to synonyms or word net synsets	
3	Quire word2vec API	They system can post to python flask webserver	
4	Clean text from website	Remove stop words and un-needed characters from text	
5	Word2vec implementation	Test text on word2vec	
6	Image classification	Test image prediction correctness on advertisements.	

5.3. Plan for Evaluation

This system must be evaluated in two ways. First, it will be verified if this method of advertisement detection is fit for purpose. Secondly, Is the system usable and functional as an application.

To complete this evaluation the system will be ran against cherry picked websites. As this system is a demonstration of the idea of using convolutional neural networks to detect advertisements and as I do not have the processing power to train the model against all advertisement types, the system should be evaluated against a small set of websites.

Evaluating if the application is usable. If the UI is clear and follows universal design principles. The system should be well documented and preform to an acceptable level.

5.4. Conclusions

This chapter reviewed the types of testing that will be completed in order to ship a functional system. This testing includes unit testing and integrated version testing which will happen throughout and on competition of the system. There is also a test plan for the unit tests.

6. Issues and Future Work

6.1. Introduction

This chapter will go into the future issues and risks associated with the further development of this system. These will range from possible technical problems to other possible issues that could arise. The chapter then goes onto explain the plans and future work that will be completed in order to finish the system. A GANTT chart is included showing both the past and future deadlines that should be completed.

6.2. Issues and Risks

There are many things that can go wrong in the development of a system. Good research and testing can mitigate many issues. Many risks cannot be so easily ruled out.

One such risk is the idea while solid in the prototyping, testing and eval phases may not scale well. This could happen in two ways; the system may not be able to handle the number of users or the CNN model may not be able to predict a large number of advertisements accurately.

Another risk is the time taken to retrain the model. As Technological University Dublin does not have GPU resources the model will be trained on my own hardware. A mitigation for this issue could be finding people with capable hardware or purchasing my own.

In addition, after retraining this model it may not preform as well as hoped. This could be because the model ends up with lower than expected accuracy. Another cause may be the new training data may not be high enough quality.

Even after these risks are dealt with CNN models can be easily fooled by injecting data into the image. Currently the system does not have any method to protect against this. This is a larger risk as it would effectively stop this system before it was even launched.

Another problem that could arise is the use TfLite, as this is a new technology the implementation and development experience may not be fully in place. The python implementation was first released in October of 2019 (20).

More mundane issues could arise. For example, the ability to complete milestone may not be met if the developer gets ill or other work arises.

6.3. Plans and Future Work

As the prototyping phase continues, the following technology must get through this process.

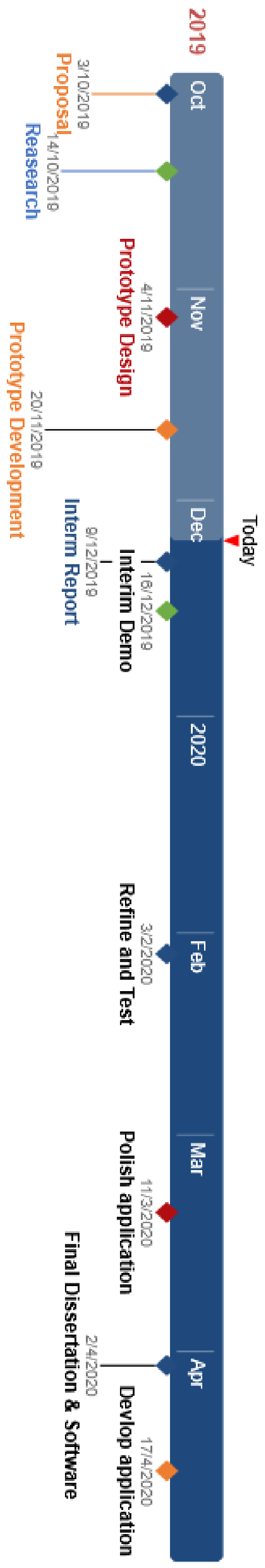
The inception model must be retrained for advertisement specific items. As mentioned in the issues and risks section there are many things that could go wrong here. This is a necessary step to the development of the project as the system lacks the ability to discern logos.

Implementing an API for a word2vec server will be completed. Because of the nature of the word2vec network it is necessary to train the model each time a new text is submitted. The trained model is effectively the word embeddings of said text and these word embeddings must be returned for the system to function.

Implementing a method of determining how sure the system is that an image is an advertisement will be completed. The system will consider the relationships presented in the word2vec model and the score of the image classifier. If the top results of the image classifier are within a certain percentage it may be necessary to compare the top number of results to improve accuracy.

In the design portion of this report both low and medium fidelity diagrams of possible user interfaces were presented. The final system will include a functional implementation of this user interface. In the coming months this user interface will be implemented and improved upon using the RAD mythology of software design. As mentioned in the design chapter, this UI will follow the principles of universal design.

6.3.1. GANTT Chart



Bibliography

1. Manifest V3 [Internet]. Google Docs. [cited 2019 Sep 29]. Available from: https://docs.google.com/document/d/1nPu6Wy4LWR66EFLeYInl3NzzhHzc-qnk4w4PX-0XMw8/edit?usp=embed_facebook

2. 896897 - Extensions: Implement Manifest V3 - chromium - An open-source project to help move the web forward. - Monorail [Internet]. [cited 2019 Dec 7]. Available from: https://bugs.chromium.org/p/chromium/issues/detail?id=896897&desc=2#c23

3. General Data Protection Regulation (GDPR) – Official Legal Text [Internet]. General Data Protection Regulation (GDPR). [cited 2019 Sep 29]. Available from: https://gdpr-info.eu/

4. Hill R. gorhill/uBlock [Internet]. 2019 [cited 2019 Dec 7]. Available from: https://github.com/gorhill/uBlock

5. AdBlock. AdBlock [Internet]. [cited 2019 Dec 5]. Available from: https://getadblock.com

6. Adblock Plus | The world's # 1 free ad blocker [Internet]. [cited 2019 Dec 7]. Available from: https://adblockplus.org/en/

7. telekrmor. Home [Internet]. Pi-hole®: A black hole for Internet advertisements. [cited 2019 Dec 7]. Available from: https://pi-hole.net/

8. Google Chrome - The New Chrome & Most Secure Web Browser [Internet]. [cited 2019 Dec 7]. Available from: https://www.google.com/chrome/

9. W3Counter: Global Web Stats - November 2019 [Internet]. [cited 2019 Dec 5]. Available from: https://www.w3counter.com/globalstats.php?year=2019&month=11

10. Firefox Privacy Promise [Internet]. Mozilla. [cited 2019 Dec 7]. Available from: https://www.mozilla.org/en-US/firefox/privacy/

11. Rong X. word2vec Parameter Learning Explained. arXiv:14112738 [cs] [Internet]. 2016 Jun 5 [cited 2019 Dec 5]; Available from: http://arxiv.org/abs/1411.2738

12. Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. Rethinking the Inception Architecture for Computer Vision. arXiv:151200567 [cs] [Internet]. 2015 Dec 11 [cited 2019 Dec 5]; Available from: http://arxiv.org/abs/1512.00567

13. WordNet | A Lexical Database for English [Internet]. [cited 2019 Dec 7]. Available from: https://wordnet.princeton.edu/

14. TensorFlow Lite [Internet]. [cited 2019 Dec 7]. Available from: https://www.tensorflow.org/lite

15. Flask-RESTful — Flask-RESTful 0.3.7 documentation [Internet]. [cited 2019 Dec 7]. Available from: https://flask-restful.readthedocs.io/en/latest/

16. EasyList - Overview [Internet]. [cited 2019 Dec 7]. Available from: https://easylist.to/

17. Software Engineering | Classical Waterfall Model [Internet]. GeeksforGeeks. 2018 [cited 2019 Dec 7]. Available from: https://www.geeksforgeeks.org/software-engineering-classical-waterfall-model/

18. Rapid Application Development: Definition, Steps, Advantages and Case Study [Internet]. 2018 [cited 2019 Dec 7]. Available from: https://kissflow.com/rad/rapid-application-development/

19. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J. Distributed Representations of Words and Phrases and their Compositionality. In: Burges CJC, Bottou L, Welling M, Ghahramani Z, Weinberger KQ, editors. Advances in Neural Information Processing Systems 26 [Internet]. Curran Associates, Inc.; 2013 [cited 2019 Dec 7]. p. 3111–3119. Available from: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf

20. WANG 王振华 Zhenhua. tflite 2.0.0.post2 : A package to parse TFLite models (*.tflite) [Internet]. [cited 2019 Dec 9]. Available from: https://pypi.org/project/tflite/#history