All files and scripts used in this report are available on Github(https://github.com/F1-mania/ADAPT) or can be found on festvox (http://festvox.org/). Replicating the steps used should lead to similar results as those outlined in this report.

Due to problems with language models in the latest version of NLTK, I am using NLTK version 2.0.4. If you already have NLTK it might make sense to set up a python virtual environment in order to install the required version of NLTK. This method will also require installation of PyEnchant.

The below is a comprehensive guide as to the steps taken in building a list of prompts from twitter, with reasoning given behind decisions taken at each stage. If you simply wish for a list of commands, they are available down the bottom.

Collecting the Tweets

The tweets were collected using the getTweetsWS.py script. The getTweetsWS script collects the 3200 most recent tweets from the selected twitter handle and returns them as .txt file, each tweet being on a new line. In total, this amounted to 309,850 tweets from 93 separate handles. The tweets are organised in the tweets_db directory in separate files distinguished by the respective handle they were retrieved from.

Cleaning the Tweets for prompt selection

The first step in generating a set of tweets for prompt selection is running join.py. This script goes through each file in /tweets_db and generates a single file (joinedtext.txt) comprising of all the tweets present within these files. This script also takes a rudimentary cleaning step of removing all links within tweets. Due to an unexplained quirk when writing our output file, the first line often has to be removed manually in order to advance to the next stage.

The next step in cleaning the tweets is running sentence_checker.py. This script will first prompt for an input file, so select the file generated by join.py. sentence_checker will output the final file required for prompt selection, selected_sents.txt. This script selects those prompts by iterating through the tweets in the following way:
It first checks whether the selected tweets are between 5 and 15 words long. These are the figures used when generating the cmu_arctic database, and were therefore considered a good baseline to use for differentiating between tweets.
Next it checks whether there are multiple sentences present within the tweet. Multiple sentences mean that festvox will generate multiple utterances for that tweet. This makes tweet removal, as required at a later point of this process, more difficult. Therefore tweets with multiple sentences are simply discarded.
Following this, the script will check for any heteronyms present within tweets. Heteronyms are words with multiple pronunciations, and thus we consider these unhelpful for the purposes of training a voice. Therefore, any tweets with heteronyms are removed from our set of tweets at this stage. The set of heteronyms is taken off the internet, and may not be complete.

Following this, we want to check whether all the words present in the tweets are correctly spelled English words. The script uses the PyEnchant library to do this, and any tweets containing non-english or incorrectly spelled words are removed at this point. Due to the nature of tweets, prompts will often contain '#' or '@' symbols to denote hashtags and tweets directed at another user. Whilst removing these characters from the word, or simply removing these words from the tweets can turn the tweets into meaningful sentences, neither of these approaches are reliable, and thus tweets with either of these are removed from our list of possible prompts. PyEnchant won't recognise any of these words as correct English words, so these are removed at this point.
Any tweets containing the word "RT" (retweet) are removed at the next stage. Tweets containing RT aren't, by themselves, usually a normal English sentence, so these can be discarded from our set of prompts. Again, it could be possible to simply remove the "RT" from the tweet to gain a meaningful sentence.

At this point, the original set of tweets has been reduced to a set of size 7097. However, it is still possible that we have sentences with unusual syntax or sentence structure in our tweets. In an attempt to limit this, we train a trigram langauge model. The trigram model is based off the gutenberg and brown corpora that are delivered within NLTK. Each of these corpora are considered a relatively good representation of the English language, and thus they are used in order to evaluate the quality of the tweets in our revised set.
To do this, the entropy of each of the tweets is calculated. In this case, entropy can be interpreted as a measure of how unexpected a sentence in our set is, based on what we know about English from the gutenberg and brown corpora. Tweets with higher entropies are considered to be less expected than those with lower entropies. As such, the tweets are sorted based on their entropies, and the top 20% of these tweets are removed.
Finally, the remaining tweets are written to the file selected_sents.txt. In this case 5,677 of the original 309,850 tweets were written to be considered for prompt selection.

There are no input parameters for either of these scripts, and thus any changes must be made within the scripts themselves. Obviously, simple changes such as increasing the sentence length, or changing the threshold at which point tweets are removed based on their entropies will make small changes on the final set of tweets, mainly affecting the size of the set. Further changes that could be made are removing the # from hashtags in order to have a larger set, or simply removing the RT from a tweet and not entirely removing it from our set. In this case, the decision was made against these measures, as I found the quality of the prompts suffered greatly as a result.

There is a section in the script that is commented out. The purpose of this section is to remove tweets that add no words that have not been seen before. By changing the number in the if statement (if word_count[word] ==) the number of times a word must be seen before it's considered to have been seen multiple times can be changed. If all the words have been seen this number of multiple times, we can disregard the tweet, as we can consider it to be adding nothing overly new to our list of prompts. This section of the script hasn't been used in this case, as I considered the reduction of tweets to 5,677 adequate, and any further reduction in this number may end up with a loss of coverage when doing our

prompt selection. However, our final set is large, this section may be helpful in smaller set with a similar word coverage.

Prompt selection from cleaned tweets

split_prompts.py generates a separate file for each tweet in our set of cleaned tweets, in a directory named split_prompts (note, split_prompts cannot already exist before running this script).
For the purposes of this section, I will assume the festvox scripts text2utts, dataset_select and dataset_subset have been copied into the split_prompts directory. Carrying out the following commands will give us a final set of prompts. Festvox here utilizes phoneme bigram maximisation, in order to reduce a our training set to a set that has the same phoneme coverage.

```
$ ./text2utts -dbname tweets_ *.txt -o tweet.data
$ ./text2utts -level Segement -itype data tweet.data -o tweet.seg.data
$ ./dataset_select tweet.seg.data
$ ./dataset_subset tweet.data tweet.set.data.selected >tweet1.data
```

Upon retrieving the prompt set, all tweets in this set were removed from the tweet_prompts directory and a second pass was performed. In order to get the filenames to remove, get_filename.py can be run and the filenames can be retrieved from tweet_prompts/filenames.txt. After removing the files, the above commands are simply run, changing the output filename in the final command (in my case I used tweet2.data).

To retrieve the final list of prompts from the generated .data files, create_text_from_labels.py can be used. This script has been written for the two passes as used here, but can be easily adapted for a greater or fewer number of passes. The final list of prompts are saved into prompt_list.txt

Manual removal/editing of prompts

After running the scripts, it is still necessary to read over the list of tweets produced in prompt_list.txt, as some editing may be required. Tweets taken from news agencies are often headlines, and thus it may be necessary to add determiners for a more natural reading. Numbers can also be added to the start of some tweets(e.g 21 things you should do on your first day of work), again to add for a more natural reading. It is possible that certain elements of language have not been removed whilst cleaning the tweets (ellipses being the most common), and these should also be removed manually. Finally, tweets were edited for content, removing those that contained expletives or similar.

Normalisation of text

In order to help with the reading of the prompts, text should be normalised. This means capitalisation of the opening words of the sentences, lowercasing further capitalised words, adding full stops where none are present, and normalising any further punctuation (! to . and so on).

Analysis of Prompts

For the purposes of analysis, the prompts obtained have been compared with the cmu_arctic database, designed for unit selection speech synthesis. In order to compare the two databases of prompts, we consider the bigrams of phonemes present, with a larger number of distinct bigrams considered to be better. In the below table, HTS-tweets is a naive attempt to build a set of prompts, taken from one twitter handle without designated prompt selection, whilst Prompt tweets is the set generated from the above steps.

0	CMU_ARCTIC	HTS-Tweets	Prompt Tweets
Number of phonemes	38856	48560	39603
Total distinct bigrams	1327	1160	1290

As can be seen, there are slightly more phonemes in our set of tweets, whilst the number of bigrams is slightly lower in comparison to the CMU_arctic database. However, this difference in coverage can be considered negligible, whilst there is a marked improvement when compared to our naive approach. As such, the set generated can considered to have good coverages for the purposes of training a speech synthesis system.

List of Commands

http://xchange.adaptcentre.ie/api/tweetdownloader/?apikey=241231232130952&handle= (with appropriate handle)

```
$ python cleantweets.txt
```

With tweets from previous step in directory /tweets_db

```
$ python sentence_checker.py
$ python split_prompts.py
$ ./text2utts -dbname tweets_ *.txt -o tweet.data
$ ./text2utts -level Segement -itype data tweet.data -o tweet.seg.data
$ ./dataset_select tweet.seg.data
$ ./dataset_subset tweet.data tweet.set.data.selected >tweet1.data
$ python get_filename.py
$ rm {filenames from get_filename}
```

Repeat festvox scripts with different final file name

```
$ python create_text_from_labels.py
```