Overview

This file includes instructions for replicating the dynamic topic modeling results described in the paper:

```
Exploring the Political Agenda of the European Parliament Using a Dynamic
Topic Modeling Approach. Derek Greene, James P. Cross. Political Analysis,
2016.
```

For further details, please contact derek.greene@ucd.ie or james.cross@ucd.ie

Software Requirements

This code has been tested with Python 2 (version 2.7.11), with the following third party modules installed. These can be installed via Pip or Anaconda:

- scikit-learn = 0.16
- gensim >= 0.12.2
- nltk >= 3.0.5
- prettytable >= 0.7.2

Please note that the scikit-learn implementation of NMF was re-implemented in version 0.17 of the package, which can result in marginally different results for the topic models described in our paper. We recommend using 0.16 if seeking to reproduce our results exactly.

Data

We provide the following data files for use with the software. The data is available from http://erdos.ucd.ie/europarl/downloads

- *europarl-data-speeches.zip*: Archive of 211,302 English language European Parliament speeches in plain text format, one file per speech, where the filename is the unique speech identifier from the Europarl website. Speeches are arranged in sub-directories based on their date.
- *europarl-documents-metadata.tsv*: Tab-separated file containing metadata for all speeches, linked by the unique speech identifiers.
- *europarl-meps-metadata.tsv*: Tab-separated file containing metadata for MEPs who delivered the speeches.
- *europarl-word2vec-model.bin*: A pre-trained Word2vec model, generated on the set of all text files, which was created using Gensim using the default parameter values (i.e. CBOW model with window size of 5 and vector size of 100). This model is stored in standard Word2vec binary format.

Experimental Process

Step 1: Text Pre-processing

The first step is to process the text files for each time window, creating one pre-processed dataset per time window:

```
python prep-text.py europarl-documents-metadata.tsv europarl-data-speeches --tfidf --norm --minlen 50 -o data
```

Step 2: Window Topic Modeling

Once the data has been pre-processed, the next step is to generate the window topics, where a topic model is created by applying NMF to each the pre-process data for each time window. For each dataset, we run NMF a range of values for the number of window topics k, and use the pre-built Word2vec model to select the optimal value for k. The relevant output files will be stored in the sub-directory results/windows-selected.

```
python find-window-topics.py data/*.pkl -k 10,25 -o results -m europarl-word2vec-model.bin
```

Step 3: Dynamic Topic Modeling

Next, using the window topics selected in Step 2, we apply dynamic topic modeling for a range of values of k. Again we use the pre-built Word2vec model to select the optimal value for k. In the case of our data, this results in the selection of k=57. The relevant output files will be stored in the sub-directory results/dynamic-selected.

```
python find-dynamic-topics.py results/windows-selected/*.pkl -k 30,80 -o results -m europarl-word2vec-model.bin
```

Step 4: Inspecting the Results

To display a summary of the top terms for the final dynamic topics:

```
python display-topics.py results/dynamic-selected/europarl-dynamic_k57.pkl
```

To track the individual topics from each window that contribute to the overall dynamic topics, run the script 'track-dynamic-topics.py', specifying the file path for the output of dynamic topic modeling, following by the paths for all of the individual window

topic models (ordered by time window).

```
python track-dynamic-topics.py results/dynamic-selected/europarl-
dynamic_k57.pkl results/windows-selected/*.pkl
```