Project 2: Geolocation of Twitter Users

October 1, 2014

1 Introduction

The objective of Project 2 was to build a geolocation classifier for Twitter users, based on the content of their Tweets (ASCII strings between 1 and 140 characters in length). A feature vector, containing 386 distinct words, and a large corpus of American Tweets was supplied as part of Project 2. Token-level and semantic generalisation feature engineering methods and Bayesian classification algorithms were explored in Project 2.

2 Preprocessing Input Data

Several heuristics were employed in the preprocessing of the the feature vector attributes and the training, development and test twitter files to improve the performance of the classification algorithms explored in Project 2. The preprocessing steps were first employed on the given feature vector attributes, before being employed on the Twitter data, to produce new training, development and test instances.

The reduction in the size of the feature vector instance files, due to the preprocessing steps outlined below, are summarised in Table 4.

2.1 Unimelb English Social Media Normalisation Lexicon - Twitter Text Normalisation

Normalising text into a canocial form increases the consistency of the text and reduces the problem sample space (by reducing the number of different tokens to consider). Good text normalisation techniques are highly document dependent, there is no all purpose normalisation procedure.

The supplied Twitter data was normalised using the University of Melbourne's (Unimelb)English Social Media Normalisation (ESMN) Lexicon. The Unimelb ESMN Lexicon mapped OOV words to IV words, therefore a simple (and quick) linear search and substitution through the vector attributes and twitter data was all that was required to (somewhat) normalise the vector attributes and twitter data.

The ESMN Lexicon was chosen due to it offering a fast, lightweight and easy-to-use solution that was suitable for high-volume microblog (Twitter) pre-processing. Further, the ESMN Lexicon had a large volume of normalisation, containing over 40,000 entries [1].

2.2 Word Stemming

Stemming is the process of reducing inflicted words to their stem (base form) [2]. A Porter Stemmer algorithm was used in the preprocessing of the location and tweet files.

The Porter Stemmer algorithm was found to further distort a subset of misspelled words within the tweet bodies, however the Porter Stemmer was also capable of (partially) correcting miss-spelled location words. Further, the Porter Stemmer reduced the sample space of the problem by introducing consistency in tweet words and reducing the number of words in the preprocessed tweets.

A sample of differences between tweet and feature vector strings with and without the Porter Stemmer preprocessing step can be seen in Table 1.

Table 1: Stemming Tweets

	Tweet Text	
No Porter Stemming	yelled egypt phone today true story	family friends atlanta motor speedway first time experience
Porter Stemming	yell egypt phone today true stori	famili friend atlanta motor speedway first time experi

2.3 Removal of Stop Words And Excess White Space

A large amount of tweets and given features contained words which were one character in length and words which were extremely common in the English language (especially after preprocessing stage **??**). Common words and words of one-character length carried little entropy[3] in determining descriptive features for geolocating a Twitter user (Table 2), and therefore were removed from the vector attributes and corresponding twitter data.

It is worth noting that stop words can cause problems when a tweet contains informative 'geolocating' phrases that include them. However, as a 'bag of words' model was used for modelling the twitter document, information contained within phrases was already lost.

The list of stop words, contained within the NLPK library for python, was used to remove stop words from the locations and tweet texts.

Table 2: Undesirable Features Before Stop Word Removal

Removed Feature Token	**Example Tweets**
'i', 'n', 'so'	omg accid on i n so scari httpyfrogcomhrwadj
'my', 'me'	my mom just told me to updat my websit next i suppos my grandma will tell me my tweet are shit

2.4 Per User Feature Vectors

Initially, a feature vector instance was supplied for each individual tweet. However, as Project 2 involved the geolocation of Twitter USERS, the supplied training, development and test data was restructured to contain a feature vector for each individual Twitter user.

The restructuring of the model vectors to be per-user had the advantage of reducing the number of vectors that had to be processed. The potential disadvantage of the reduction in data-points in the new per-user model was offset by the information being retained in the now richer instance vectors (Table 3).

Table 3: Feature Vector Restructuring

	Training and Development Files	
	Per Tweet	**Per User**
Number of Instance Vectors	958,359	45,572
Vectors With At Least 1 Non-Zero Dimension	748,851	44,089
	78.14%	96.75%

Table 4: Preprocessing Input Data Size Reduction

	Attributes	**Training Instances**	**Development Instances**
Raw Feature Vectors	387	766,841	191,518
Processed Feature Vectors	366	22,993	22,579

3 Geolocation Classification Algorithms And Results

All of the string approximation matching algorithms used in Project 1 used very little memory with worst case memory complexity of $O(nm)$, where n is the character length of a location query and m is the length of a tweet. Therefore, the following analysis will compare and contrast algorithm run times and matching effectiveness.

3.1 Sub-Distance: Needlemen-Wunsch and Smith-Waterman Combined

The runtime and matching performance of the Smith-Waterman algorithm was a definite step forward in finding an edit-distance based solution to the string approximation task of Project 1. However, the disadvantage of S-W matching location strings anywhere within a longer tweet string, including mid-word, without penalty resulted in additional false positives. The Sub-Distance algorithm was developed to address this undesirable S-W characteristic.

Sub-Distance involved implementing two small changes to the N-W Whole String Matching algorithm to significantly improve its matching performance:

1. The initialisation of the row associated with the empty character in the location name and the tweet text was altered to have an edit distance score associated with the depth of a character in a word NOT the whole string (Table **??**). The intuition behind this change is that location (query) matches that start at the start of a word should not be penalised, however location matches that start mid-word should be penalised in relation to how deep into the word said match is.

2. Once the scores of the bottom row had been calculated, the lowest score was taken as the edit-distance cost, rather than the score in the rightmost corner. This meant that we didn't care about how many characters we skipped after we stopped matching the location name within the tweet text (a reasonable assumption).

Sub-Distance had a slightly smaller runtime than S-W and N-W Whole String, 2.83 seconds. Sub-Distances runtime result was expected as both N-W whole string and S-W have the same complexity as Sub-Distance, $O(nm)$.

Sub-Distance still generated more false-positives compared to N-W Tokenised Words, as Sub-Distance still suffered from not penalising location matches that matched the prefix of a larger word (similar to S-W). However, Sub-Distance achieved closer matching to the N-W Tokenised Words baseline than S-W, as it was less likely to match location names that started mid-word (Table **??**, **??** and **??**).

4 Conclusion

X methods were considered for the task of geolocating Twitter users. By employing clever initialisation and exit parameters to the N-W algorithm, the Sub-Distance algorithm, gave the best overall results, .

All algorithms, explored as part of Project 1, often exhibited false positives when considering location names that could also be used in language when NOT referring to a location. Therefore, a future area of research would include algorithms that take into account the context in which words are used.

Further, the shortest runtime for a single tweet to be compared to all queries was 2.8 seconds (Sub-Distance). Therefore, to process all 3.6 million preprocessed tweets against all of the preprocessed locations would take over 116 days. If the whole Location-Tweet sample space had to be analysed faster methods of string approximation, such as the use of Tries or N-gram matching algorithms, are an area for further research.

References

[1] Bo Han, Paul Cook, and Timothy Baldwin. Automatically constructing a normalisation dictionary for microblogs. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 421–432, Jeju Island, Korea, July 2012. Association for Computational Linguistics.

[2] Mirko Popovic and Peter Willett. The effectiveness of stemming for natural-language access to slovene textual data. *Journal of the American Society for Information Science*, 43(5):384 – 390, 1992.

[3] Anand Rajaraman and Jeffrey David Ullman. *Mining of Massive Datasets [electronic resource]. / Anand Rajaraman, Jeffrey David Ullman.* Cambridge : Cambridge University Press, 2011., 2011.