CS4248 Assignment № 2

TONG Haowen Joel, A0108165J, National University of Singapore 23/10/2015

Algorithm Modifications

This assignment implements a supervised classifier determining if an unlabelled text should be disambiguated as labels **word1** or **word2**, with a logistic regression classifier. The gradient ascent algorithm finds the (local) maximum point iteratively, by determining each weight w_i individually. The stochastic gradient ascent algorithm and its termination condition are given by lecture notes as:

$$w_{i,k+1} = w_{i,k} + \alpha {x_i}^j \left(y^j - \frac{1}{1+\exp(-\mathbf{w}_k \cdot \mathbf{x}^j)} \right) \quad , \alpha = \text{LEARNING\_RATE}$$

$$\Delta w_i \leq \text{TERMINATION\_THRESHOLD}$$

A stochastic gradient ascent algorithm was chosen over the batch method to reduce learning complexity and hence timings.

The gradient ascent works by a hill-climbing algorithm. Hence, it may remain stuck in a local maximum, rather than the desired global maximum. This implies that trained weights might not reach its fullest potential. Hence, the following were implemented:

Decaying Learning Rates

Earlier iterations were given higher learning rates, while later iterations were given finer learning rates. [1] This allows a coarse version of the global maximum to be reached in the earlier iteration stage, while later iterations finetuned the hill-climbing to that precise maximum. This was implemented via a decay algorithm, where γ is the decay value used and k is the iteration number:

$$\alpha = \gamma^k \cdot \text{MAX\_LEARNING\_RATE} \quad , \gamma < 1$$

Learning and model instance selection via N fold cross validation

N-fold validation results were used to train various model instances, of which the best was selected for further training. Records were first shuffled. Then, 3-fold validation evaluated the best model for training. The model instance with the highest accuracy

was then re-trained with the entire training set as the final model. Precision stored was in Double form. It is noted that the BigDecimal / storing as fractions might be more accurate when writing the model file. For hyper-parameter optimization, a test driver was written to optimize parameters for model training in CSV format.

Features

Logistic Regression classifiers degrade in higher dimensions, and take longer to compute. Hence the following features were used: (1) Word tokens, (2) N-Gram Collocations. Further 70% - 80% dimension reductions were made via computing a histogram and eliminating all sparse features ($n < 3$). This dimensionality reduction greatly reduced computational time.

Feature Stop words

Undistinct top words were removed from word tokens. This is necessary to reduce dimensions and reduce instances of overfitting noise, and reduce chance of sparse but similar data. An example notes that **bought him a toy** and **bought her a toy** should be unified as **« toy**. Hence, only stop words with the most distinct word count are preserved. The following describes the algorithm used.

Listing 1: Stop Words Removal Algorithm

```
For each word in stopWords:
    if word_freq_label1 - word_freq_label2 < wordDiffMinThreshold
        remove word from tokens
return tokens
```

Feature Collocations

Collocations both before and after the confusable word appeared to be most useful, as the confusable word is often used immediately before and after other connectives. Take for example the instance in **brought**. In this case, bigram **« down** in the collocation is a determinating feature:

```
...the war which >> brought << down the regime...
```

Furthermore, it is noted that long collocations are unhelpful as they appear minimally in text and contribute to sparse data. Hence, the collocation was obtained after redundant stop words were removed before chunking into N-gram features. In the case where

Table 1: Training and testing from file

Filename	Word1	Word2	Total	Time(s)	NumFeatures	OrigFeatures
peace_piece	0.84	0.78	0.81	5	870	4923
bought_brought	0.88	0.78	0.83	30	1587	7960
adapt_adopt	0.82	0.72	0.77	31	1678	5994

Table 2: Training and testing from model in memory

Filename	Word1	Word2	Total	Time(s)
peace_piece.train	0.84	0.96	0.9	5.329
bought_brought.train	0.76	0.92	0.84	30.156
adapt_adopt.train	0.82	0.84	0.83	35.547

required start or end collocations did not exist, collocations fell back to the remaining tokens. Similarly, Ngrams fell back to (N-1)Grams if nonexistent within the collocation.

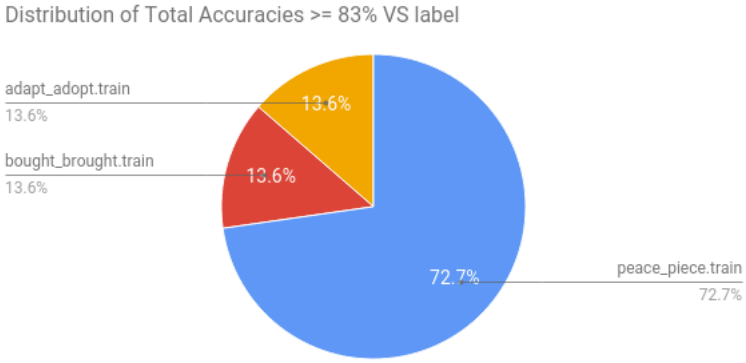
Tri-grams were preferred over bigrams as they provided more context, yet were not as sparse as full collocations. Furthermore experimentally, it appeared that Tri-grams were more common at 85% - 90% scores as compared to 7-grams, commonly found in 60% accuracy scores. Collocations were restricted to $C(-4,4)$ to reduce computational time, as longer collocations did not appear to significantly improve accuracy. Both before and after collocation indexes were used, as confusing words (1) could be commonly used in conjunction with those words, and (2) Some labels were used near the start or end of the sentence, hence one index was unusable.

Results and Discussion

Experiments were test-run on a remote server, before best results were arranged by descending total accuracy. The optimal parameters found were: **FeatureCountMin (minimum feature occurence)=3, numFolds=3, nGramSize=3, learningRate=2, learningDecay=0.8, terminationThreshold=1.0E-10, wordDiffMinThreshold (measure of distinctiveness between stop words)=20, and collocation=(-4,4)**. It is also noted that accuracies of the corpus consistently followed in decreasing order: (1) Piece/-Peace, (2) Brought/Bought, (3) Adapt/Adopt. (1) Piece/Peace took significantly shorter to train, at 2 - 5 seconds. They are as shown in the table above.

Piece/Peace had a smaller dimension size and performed much better. This could explain the shorter train timings. Furthermore, reduced features lessen occurrences of high-dimensionality performance degradation. It is noted that all three texts have similar token counts (30610,29831,29932). Therefore this discrepancy is likely due to the

distribution of features. It implies that Piece/Peace has a more sparse dataset, concentrated around several features. For instance, it is noted that » **piece** « **of** appears 307 times, rather than twice in » **peace** « **of**. This is because **peace of** has a more limited context and refers to a country, while **piece of** can refer to differing nouns. Having a large number of these distinct, noise-irrelevant features contributes heavily to the success of this confusable set. Testing from file was less accurate, possibly due to truncation error from storing and printing from double.



Further work

Higher scores may be possible when trained with a POS tagger. For instance, the features « **two dollars** and « **three dollars** for **bought** can be unified as « **<CD> dollars** instead of two separate features. This will increase accuracy, reduce feature dimensionality and possibly computations.

If time taken for training is not a major consideration, then batch gradient ascent could be used. Batch training considers all other examples in the corpus for training. This increases likelihood of a global max being found, although time complexity is an issue.

The least distinctive tokens should also be removed. In order to minimize issue of over-fitting a given dataset, various unique words (for example, **Florida** in label **bought**) should be removed from the datasets. Having such unique words may result in bought being associated with Florida, leading to reduced accuracy when testing in a different context.

References

[1] Daniel T Larose. *Discovering knowledge in data: an introduction to data mining*. John Wiley & Sons, 2014.