MINI PROJECT 2

Scenario: Learn a Distance Function between Documents

A conference has n papers accepted. Our job is to learn a distance function between each pair of papers. The distance will be low for papers that are fit to be in the same session and high for papers that will have minimum conflict (i.e., few people interested in both). The distance needs to be between 0 and 1.

Input:

The text (converted from pdfs) for all the accepted papers.

Output:

A matrix of distance functions for papers (that can be input to Project 1). Each distance function should be between 0 and 1.

Training Data:

The schedule from the past conferences can be used as training data. Since, someone manually scheduled the conference, we can assume that papers in a single session are very low distance and papers in parallel sessions are high distance. This information can be used to train a distance function or a "fit to be in the same session" binary classifier.

Testing:

To test your distance function we will run the software on a different AI conference. Given any manually constructed schedule we can find a set of paper triplets (p1,p2,p3) such that d(p1,p2) < d(p1,p3). This happens when p1 and p2 are in the same session and p1 and p3 are in parallel sessions. For each such known triplets we will test what fraction of those inequalities that are correctly predicted by your learning algorithm. This constitutes your final learning-score.

Algorithms and Ideas:

1. Information Retrieval Ideas
    a. Cosine similarity on the word incidence vectors
    b. Cosine similarity on the word-frequency vectors
    c. Cosine similarity on word tfidf scores
    d. Other similarity metrics, e.g, Jaccard index, Dice's coefficient, Sorensen Similarity, etc.
    e. Similarity based on Latent Semantic Analysis of papers
2. Text pre-processing

    a. Remove stop words: http://www.cs.washington.edu/education/courses/cse573/12sp/A2/stopword.list
    b. Remove infrequent words
    c. Stemming of the words. Use Porter Stemmer: http://tartarus.org/~martin/PorterStemmer/
    d. Annotate the words by their part of speeches: http://opennlp.apache.org/cgi-bin/download.cgi
    e. Bigrams or Trigrams instead of unigrams. Keep the frequent bigrams. Alternatively, try the noun-phrase chunks from the NP chunker: http://opennlp.apache.org/cgi-bin/download.cgi
3. Supervised Classification: use the training data as supervised training to build a logistic regression classifier. The concept can be whether two papers are in the same session. Use the probability of the concept as the similarity function. Use Weka for an open-source implementation of the learning algorithms: http://www.cs.waikato.ac.nz/ml/weka/index_downloading.html
    a. A basic set of features is the word count differences in the two papers.
    b. Use L1 or L2 regularization to reduce the number of active features.
    c. Design other features to improve on classification.
    d. Try other algorithms such as SVMs, Perceptrons.

Other Exploratory Algorithms/Suggestions:

1. Directly fit a distance function, say as a linear combination of features. The training will come in the form of distance rankings (given by taking three papers, two in same and one in parallel session). The weight updates can be error-driven as in perceptron learning.
2. Use references (conference names, authors, etc) as additional source of evidence for which kind of AI subfield a paper may belong to.
3. Try advanced IR algorithms. Examples include probabilistic LSA, Latent Dirichlet Allocation (LDA). Often, the code for these algorithms is widely available.
4. Label more training data as required.

Implementation:

You are being provided sample code that can take in the input and generate the output in Java. You may choose to not use this code. You may program the software in any language (or even in multiple languages).

Also, you may choose to write the algorithms yourself or use existing software. You could use any piece of existing software if that helps you. Of course, please describe in your paper what software you used and the source.

The sample code can be downloaded from the course website. Readme.txt explains the files and code.

Timeline

You are to work in teams of up to 2. The project has two milestones.

Milestone 1 (due May 16th noon): Write a one page document summarizing your current implementation, and cross-validation results from those and ideas to explore in the future.

Milestone 2 (due May 30th noon): Write a 4-8 page report describing: (1) your algorithms, features, etc (2) the analysis and comparison of the algorithms/features along any metrics that make sense (e.g., learning-score or cross-validation accuracy, etc). Also note any external resources used and the names of other students you discussed with through the project. Submit the best performing code from your team for the final competition.

Grading (25 points)

You will be graded for the following:

- [5 points]: The writeup for Milestone 1. These are mostly free points and intended to check that you start making progress early on.
- [15 points]: Study at least three different approaches/enhancements to generate distance function. Analyze the impact of different features/enhancements/approaches along cross-validation accuracy. These 15 points are awarded based on choice of algorithms/features, carefulness of evaluation, validity of final results and clarity of the paper. This is like evaluating a research paper.
- [5 points]: Final performance of your best system compared to other teams' software. This is a competition and better performing systems get more points. For this we will run your system on the papers from a different AI conference. The evaluation metric will be the learning-score.
- [Extra credit]: If your ideas are particularly creative and novel. Or if your experiments and extremely thorough and detailed.

Submission Instructions

You should submit a zip file containing all relevant parts of the assignment for each milestone. At Milestone 1, this will just be your one page writeup. For Milestone 2 this will be your report, your source code, and a README describing how to run your system. You should submit your assignment via dropbox. We will post a link on the website that you can use to access dropbox.

Extra Credit [upto 5 bonus pts, though this part is really not about the points ☺]

Couple your both systems (project 1 and project 2) together. Analyze the quality of the final schedule. Do individual sessions make sense? Do the parallel sessions make sense? If not, what is missing? What might it take to get to a good working system? You may even add some of these insights in your system to get better results.