Kaggle: Adzuna Data Mining Competition

Jordan Ell
University of Victoria, Victoria, British Columbia, Canada
jell@uvic.ca

***Abstract*—This paper outlines my attempt to tackle the Kaggle competition entitled "Job Salary Prediction" put on by Adzuna. I completed this competition roughly 8 months ago in a team of two. At the time of completion, I sat 108th (username: PickleFeathers) in the leader boards. I used a tool created by Yahoo! research called Vowpal Wabbit, along with many custom python scripts for data manipulation to examine various components of the training set. I focused largely on free text fields and the keywords that were contained in these fields in order to complete this challenge. This challenge has concluded on April 3rd 2013.**

I. Data Mining

Kaggle [1] is a website that is home to many competitions that solve very complex and intriguing problems. The competition that is described in this report is job salary prediction, put on by an advertisement search company called Adzuna [2]. By predicting the salary from job posts, Adzuna can provide more accurate ads to their customers. The Adzuna team provided all challengers of the Kaggle competition with data sets in CSV formats, one file for training and the other for test. Since the training set contained categorical fields as well as free text fields such as job description, I decided to focus my efforts on the free text fields which could offer better prediction than those of just categories which are used in standard data mining techniques

The training CSV file was over 244,000 records long so the use of long runtime algorithms was out of the question. This being the case, I decided to use a tool called Vowpal Wabbit (VW) created by Yahoo research. This tool uses a gradient descent algorithm with a bag-of-words approach to free text attributes for data mining. Several data transformation were required to get a correct format to run VW so I created many python scripts to aid in this conversion. Logarithmic and square root conversions were also used to normalize the data in terms of distribution of salary.

Aside from just running the out of the box algorithm from VW, I also decided to take it one step further by introducing the notion of keywords to the free text fields. I didn't want the algorithm to apply meaning to words such as "the" or "and" so I used the python library called "topia.termextract" [3] to extract only keywords from the free text fields. I had hoped that by limiting the algorithm to only keywords, better predictions would come about because importance will be placed on correct words such as "software" as opposed to words such as "the". Two keyword types were tested: global meaning the top 500 keywords across all job postings, and local meaning keywords local to each job description.

As can be seen in Table I results are expressed in mean absolute error (MAE), meaning the average "dollars o" between my estimate created using the data mining model and the actual answer known only by Kaggle.

TABLE I: Results from main data trial runs.

Free Text	Logarithmic	Square Root
None	12712.17	13293.12
All	7350.44	7175.04
Keywords (local)	8149.91	7744.84
Keywords (global)	8360.31544	8135.83

My results were quite surprising. as can be seen in the table above, the baseline test of just using all text performed better than any keyword implementation. I would have thought keywords would be of more use than just filler English text but I was wrong. The issue may lie in the fact that using keywords will break up key sentences that are more telling than keywords. Perhaps this would be worth exploring in the future. It is also interesting to see that local keywords performed better than global keywords. This may be a telling of how diverse the job postings can be as the top 500 keywords may not appear in every job posting so the free text fields get ignored. In future research, I would suggest, from the learnings of this experiment, that local investigations take place for free text fields to not only identify keywords, but key sentences. Can we summarize a job description with only 1 or 2 sentences? This may prove to result in a better result of data mining.

In conclusion, I experimented with various trial runs of gradient descent in VW with different data parsing on free text fields. I eventually discovered that the linear gradient descent algorithm works better with a larger collection of words instead of limiting to keywords.

[1] http://keggle.com
[2] http://adzuna.com
[3] 6https://pypi.python.org/pypi/topia.termextract/