Google RIPS 2015 Project:
Personalized Local Recommendations

Industry Mentors: Brian Milch (brian@google.com),
Russ Howes (rhowes@google.com)

Introduction:
Google has a number of consumer websites, such as YouTube, Google Play Music, and Google News, for which accurate prediction of consumer preferences can lead to more satisfied customers. For instance, YouTube recommends videos that a user might want to watch, and Google News offers a "Suggested for You" section. These websites, as well as other businesses such as Amazon, Netflix, and iTunes, have systems for predicting which items a user will like. These systems are often based on *collaborative filtering*: using collected feedback from many users to make predictions about the interests of another user.

A rich, publicly available source of data for investigating personalized recommendations is the Yelp Challenge Dataset, which includes 1.6 million reviews of local businesses. This dataset includes not just star ratings, but the text of the reviews and additional attributes of users and businesses. Thus, it offers an opportunity to go beyond traditional collaborative filtering by making use of text and attribute data. It also presents the challenges of geographical diversity (the dataset covers 10 cities) and diverse types of businesses. In this project, the RIPS team will develop and evaluate recommendation algorithms that address these challenges.

Technical Background:
A common mathematical approach to collaborative filtering is the *linear factor model*. This model assumes that a user's rating for an item (e.g., a local business in the Yelp setting) is based on a number of latent factors. The number of factors, K, is chosen by the modeler; it is typically between 5 and 50. The weights that a particular user i puts on these factors are represented by a K-dimensional vector u_i. Each item j is associated with a K-dimensional vector v_j, representing its scores on each of the factors. The rating that user i would give to item j is then modeled as $u_i \cdot v_j$.

This model can be expressed concisely as a form of matrix factorization. Suppose there are N users and M items. Then the linear factor model factorizes the $N \times M$ rating matrix R into a $N \times K$ user-factor matrix U (with the u_i vectors as its rows) and a $K \times M$ factor-item matrix V (with the v_j vectors as its columns):

$$R = UV$$

In practice, all we get to observe is ratings by particular users for particular items -- a sparse subset of the entries in R. Based on these entries, we need to estimate the matrices U and V. We can then predict other entries of R, telling us which items the users are likely to prefer.

There are a number of approaches for estimating the matrices *U* and *V* from sparse observations of *R*. The recommended reading list below can serve as a starting point for investigating these methods. The first task for the team will be to implement one or more of these methods and evaluate its accuracy in predicting a held-out subset of the Yelp Challenge ratings. The next phase of the project will investigate how the diversity of this dataset affects the prediction task, addressing a few of the following possible questions.

Geographic diversity: The Yelp Challenge Dataset includes data from 10 cities in four countries. We expect that for most users, there is one city that accounts for the vast majority of their ratings (and each business location is in just one city). So the collaborative filtering task will *almost* decompose into a separate task for each city -- the only coupling between the tasks would be created by users who traveled or moved between the cities. This leads to questions such as:

- Is there enough cross-city data that the latent factors end up consistent across cities? Are predictions for ratings outside of a user's home city just as accurate as the rest?
- Review text should be generally consistent across the 8 English-speaking cities in the dataset. Are cross-city predictions particularly improved by adding review text to the model, as done (for example) by McAuley & Leskovec (2013)?
- Does a single model for all the cities perform better or worse than separate models trained on the individual cities? Does this change when text is included?
- Do some of the same results apply across neighborhoods within a city (which are also included as business attributes in the dataset)? Can we improve predictions for sparsely reviewed businesses by estimating per-neighborhood prior distributions on the v_j vectors?

Business type diversity: The dataset includes ratings for a wide range of businesses, from restaurants to plumbers to clothing stores. Category labels are included among the business attributes. One might think that the factors involved in a user's preferences for restaurants might be very different from those involved in clothing preferences, but perhaps there are common factors such as price or trendiness.

- Suppose that for each of the latent factors, we find the businesses in various categories with the highest estimated scores for that factor. Do these businesses have anything in common? Can we understand what the factor represents?
- Can we improve predictions by estimating *category-specific* models for certain popular or broad categories, such as restaurants or local service providers?
- Can we improve predictions within a single model by learning, say, category-specific importance weights on the factors? How about category-specific prior distributions on the v_j vectors?

Special Requirements:

There are no requirements for a specific programming language or computer hardware. Many commercial and open-source libraries and packages might be appropriate for this project,

including ones developed for C++, MATLAB, or Python. Here are some examples of tools that might prove useful and are freely available on the Web:

scikit-learn: a collection of tools for data mining and analysis in Python, compatible with other Python packages like NumPy and matplotlib.
Weka: a Java-based suite of algorithms for data processing and classification.
CMU Matlab Toolkit: a set of C and Matlab functions that implement several collaborative filtering algorithms.

Python is probably the most commonly used language for data analytics at this point, so we recommend building on scikit-learn. But the team will be able to choose the language and toolkit that they are most familiar with or would most like to learn.

The primary dataset for the project will be from the Yelp Dataset Challenge (http://www.yelp.com/dataset_challenge), which offers 1.6 million reviews of businesses, and information on the 360,000 users who wrote those reviews. Depending on the problem chosen, group members may augment the Yelp data with other data from publicly-accessible webpages such as Google or Wikipedia.

Expectations:
By the end of the project, the team should have a good understanding of several methods of collaborative filtering, have implemented a few such methods on their chosen problem, and selected an appropriate candidate method through thoughtfully-designed experiments. Deliverables will include the code used to generate and/or implement their methods, a written report outlining methods and results, and a presentation at Google's offices in Venice.

Recommended Reading:

Koren, Y.; R. Bell; C. Volinsky. "Matrix factorization techniques for recommender systems". *IEEE Computer*, 2009. http://www2.research.att.com/~volinsky/papers/ieeecomputer.pdf
(These authors were part of the team that won the Netflix Prize, an earlier collaborative filtering competition; see http://www2.research.att.com/~volinsky/papers/chance.pdf).

McAuley, J.; J. Leskovec. "Hidden factors and hidden topics: Understanding rating dimensions with review text". In *Proc. 7th ACM Conf. on Recommender Systems*, 2013. http://dl.acm.org/citation.cfm?id=2507163 (also at http://cseweb.ucsd.edu/~jmcauley/pdfs/recsys13.pdf)

Schafer, J.B.; D. Frankowski; J. Herlocker; S. Sen. "Collaborative filtering recommender systems". Chapter 9, *The Adaptive Web*, Springer-Verlag, 2007. http://rd.springer.com/chapter/10.1007%2F978-3-540-72079-9_9