Tuesday, October 9, 2018

Kai Kuspa H1 Coding Challenge

This will be a short writeup of my process and code.

- **The Challenge as I saw it**

  - The first thing I did after reading the challenge instructions was to conduct a literature review on Author Disambiguation. I compared different approaches, and found a paper on Random Forests (Treeratpituk, 2009) that claimed to successfully disambiguate authors with >95% accuracy.
  - Random Forests seemed like a good choice because I know that the model does not require large amounts of training time as the dataset or dimensionality increases. Also, intuitively, RF seemed to fit the problem well.
  - Initially I thought to train a model to recognize different authors and classify based on features selected from the datasets provided. However, the large number of class variables would have made training difficult and certain names had less data than others. Furthermore, the prediction task set included class labels (names) of people who were not in the training dataset! So a classical prediction model would not work best in this scenario.
  - In fact, when I saw that none of the labelled names in training data corresponded to any of the target names in the test data, I started to consider unsupervised approaches. I developed a way to extract variance from features generated on pairs of assets. My plan was to compare every ambiguous asset to every other asset and cluster the results per name. The problem was I didn't have the cluster labels! So I thought I would peak into the data to find a couple of assets I were sure belongs to the person_id, view them in the cluster, select all assets in that cluster and make my predictions. This approach worked for rare names, but the number of pairs to consider was N choose K, which doesn't scale.
  - I decided to adapt my previous unsupervised approach with a semi-supervised method. The script "disambiguate.py" in my project will read the training data and generate a model based on pairs of assets. I knew which assets_ids in the ambiguous set belonged to the target person_ids so I randomly sampled pairs containing one AID from target and one AID not in the target set for the negative labels ("0") and one pair with two AIDs belonging to the target PID for the positive label ("1").

- The intuition behind this labeling schema was to train a model that could discriminate between pairs of the same author and pairs of different authors. I would then take this model and use it to predict, for completely separate dataset of authors and assets, which ones belonged to our predict list identities.
- Unfortunately, the problem still remained in where I did not know if a predicted match was the *particular* author I was interested in or if it was a pair from a different author.
- So I manually created a short list of assets to PIDs of which I was certain belonged to the identity I was interested in. I created feature vectors based on pairs of that asset, and every other ambiguous asset. If I predicted a match, I would know that "unknown" asset belonged to my target author of interest.

- **Data Challenges**
  - Given the size of the data, I make extensive use of hash tables. I knew I would be searching a list of records about ~3M long several times, so leveraged the hash table's O(1) lookup time in order to drastically improve computation time. For building the list of ambiguous assets tied to a "NameKey", a concatenation of the first initial and last name, the hash table improved the run time over the brute force method from 201 days to 231 seconds!
  - 231 seconds was still a little slow so for my large dictionary hash tables I used cPickle to serialize the byte stream so that I didn't have to rebuild the hash tables each time.
  - Other challenges included variable type mixing, handling sparse data, removing punctuation, and other general data cleaning/quality tasks.
- Feature Selection
  - Out of the 22 features outlined in (Treeratpituk, 2009) I used 10 in my similarity profile plus one email feature I designed. Many features given were specific to the Medline database used in the paper, but I adapted for this dataset. They are:
    - *auth_fst*: One-hot vector of first name similarity
    - *auth_mid*: One-hot vector of middle name similarity
    - *auth_lname_idf*: Inverse Document Frequency of last name
    - *aff_jacc*: Sum of Jaccardian Similarity of matching affiliation token words
    - *keyw_share*: number of keyword matches
    - *keyw_idf*: Sum of Inverse Document Frequencies of matching keywords
    - *jour_shared_idf*: Sum of Inv. Doc. Frequencies of matching journal keywords

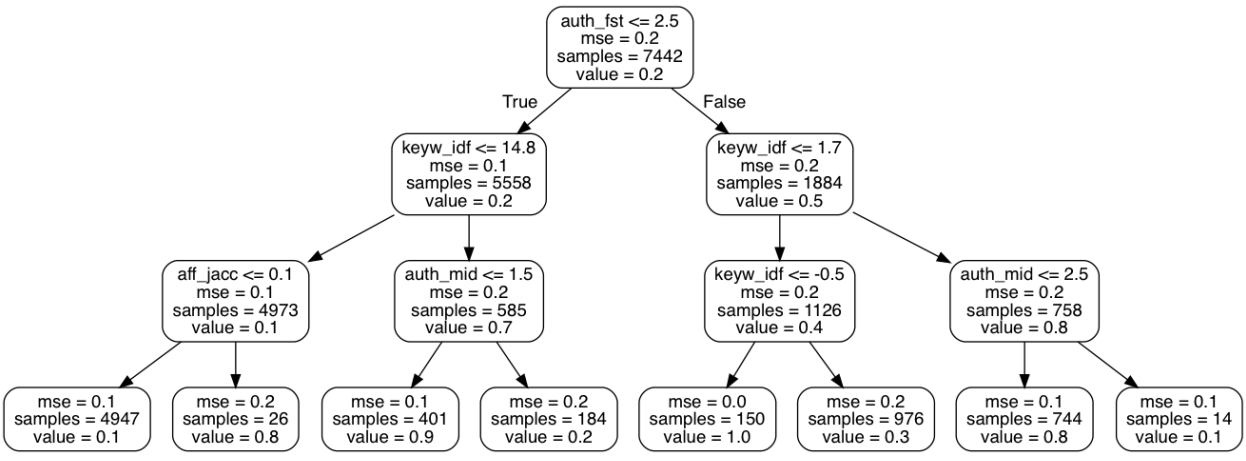
- *jour_date_diff*: Difference in days of publish
- *title_jacc*: Jaccardian similarity of title token words.
- *email_sim*: One-hot vector of email similarity.

• These features were tested for the Gini importance index, shown later.

- **Pseudocode**

1. Read training data into data frames.
2. Hashify training resources.
3. Generate histograms for IDF features.
4. For Name in training_Identities:
    1. Generate positive training samples
        1. Randomly sample up to 100 match pairs
    2. GenerateNegative training samples
        1. Use N choose K to limit random samples
        2. Randomly sample up to 400 non-match pairs
5. For each pair generate similarity profile.
6. If features cannot be created, skip this pair
7. Label each similarity profile.
8. Instantiate Random Forest Model from Sklearn
    1. Best prediction acc with 2000 estimators, max_feat=3
    2. max_features chosen for log2(num_features + 1)
9. Split features into training features (75%) and test features (25%) to get mean error.
10. Read test data into data frames
11. Hashify training resources.
12. Generate histograms for IDF features.
13. For Name in test_identities (manually curated set containing all target PIDs)
    1. Pair one asset from curated set to ambiguous asset
    2. Generate similarity profile
14. Use RF model to predict if pair is a match
15. For each prediction:
    1. If prediction > 0.75 add asset_id, PID, and prediction value to csv output file

- **Results**



- The above tree is a visualization of the random forest classifier with 20 estimators and a max depth of 3. The “true” model is about 1GB and wouldn’t open in my word processor. It’s included in the “img” folder along with this “small tree”.
- Error:
  - Mean accuracy of **RF model** on the test data was **91.5%.** Different iterations sample different random pairs, giving mean error fluctuations of about 0.7%
  - A **baseline** prediction that naively matched assets to PIDs based on if at least one coauthor had the same last name yielded **74%** accuracy.
  - We can speculate on the gap between my random forest implementation and (Treeratpituk, 2009) who had a top accuracy of 95%. Treeratpituk had more features, and used more data, but he also had a highly structured database within Medline, which is a professionally curated database. Other factors like data quality and sparsity can come into play. To improve our model accuracy, we can continue tuning parameters of the random forest, get more training data, get more sophisticated methods to handle sparsity, and to implement more features. One I would particularly like to build if I had more time is “Term Frequency-Inverse Document Frequency” for keyword matches. We can also use the abstracts to generate new features. Abstracts took too much time to generate features.
  - Gini importance:

Feature	Gini Importance (higher is better)
keyw_share	0.2913659854307419
keyw_idf	0.2814123825719411
auth_fst	0.1803618174711446
auth_mid	0.17165630977446272
jour_date_diff	0.040592175164979705
aff_jacc	0.018946545166632564
auth_lname_idf	0.010535416548838415
jour_shared_idf	0.004250732214035139
email_sim	0.00048467226888150653
title_jacc	0.00039396338834227 34

We can see that the keyword features were most significant, followed by name features, then affiliation, journal attributes, email similarity, and followed by title similarity. This makes intuitive sense because the same author will write about similar topics, and name comparisons are important as well. I am surprised by the email similarity having such low importance, but looking at the data we can see that only 24,783 emails are listed for 2,905,250 asset_author records. If only 1% of authors have their email on an asset then most features will be a negative match.

- **Conclusion**

  - I used a semi-supervised random forest classifier to predict which assets in the test set belong to each of the 30 test identities. My output file is named “predictions.csv” and it includes the model prediction on a scale from 0 to 1. Lower numbers show the model is less certain about it’s prediction.
  - My random forest classifier is training to output 1 if two assets were written by the same author and 0 if they are different.
  - Using this classifier we achieved 91.5% accuracy on a test set split containing 834 records.
  - This challenge was difficult because I had never encountered disambiguation problems before, the dataset was large, there were a large number of class labels, and the target and training identities did not overlap.
  - If I had more time, I would:
    - Set up better visualization tools to explore the data. Initially I wanted to build a vis.js tool to visualize the connection between authors and their papers.
    - Explore more features
    - Train the model with more data than 500 pairs per identity. Some ambiguous names have over 700,000,000 possible pair combinations!
    - Use more of the data. identity_institutions_(train/test).csv were not even loaded in my scripts.
    - Apply the model to more real world scenarios. Real databases may have more metadata than the set provided and could provide useful features. Also real databases are large, and I’m interested to test the scalability of this model.

Thank you for reading!