A Recurrent Neural Network Based Subreddit Recommendation System

Cole MacLean MACLEAN.COLE@GMAIL.COM

Barbara Garza BARBI.GARZA@GMAIL.COM

Suren Oganesian OGANESIAN.SUREN@GMAIL.COM

Abstract

With the vast quantity of content available to consumers from a multitude of sources across the internet, users require tools for finding and filtering the available information into manageable quantities of interesting material. Community Curation and Recommender Systems have emerged as two tools available for content filtering. In this paper, we combine these two methods to solve the problem of Curation Community Discovery by developing a Recurrent Neural Network based recommender system for Reddit.com, a content curation and social website. The performance of Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU) and 2-Layer Stacked Bi-Directional network architectures are compared to inform the the discovery of promising architectures for subreddit recommender systems. The final model is validated using a novel 3-way validation procedure, using conventional Random and Most Popular model benchmarks, analogous recommender system comparisons using historical recommender systems competition data, and embedding space visualization inspection, providing a baseline, comparative and inspective means for recommender system model validation. The final model provides a nearly 5x performance improvement over the baseline Most Popular model, comparable to the performance gains of best-in-class recommender system models, showing Recurrent Neural Networks can be used as powerful models for subreddit recommender systems.

Keywords: Recommender Systems, Reddit, RNN, LSTM, GRU

1. Introduction

1.1. Problem Statement and Goals

With the advent of the internet, the marginal cost of content creation and distribution has approached negligibility, allowing anyone to become content creators capable of reaching billions of consumers. The problem of the consumer has now become one of filtering, as the fire-hose of information becomes debilitating and the effort required to parse through it becomes intractable for any individual. Two solutions have emerged for this problem of the Information Age: Community Curation and Recommender Systems. Community Curation depends on the collective interests of the inhabitants of a community that have come together under a specific theme to filter out any content not relevant to the communities collective interests. Recommender Systems rely on the historical patterns of users to predict

the preference of a user for new content, and work to suggest content most likely to be of interest to that individual user.

In this work, we combine these two methods of information filtering by creating a Recommender System designed to suggest new subreddits for a user to join, in an attempt to solve the problem of community discovery, where finding communities that successfully filter out uninteresting content for a given user can be difficult. Using the sequence prediction capabilities of Recurrent Neural Networks, we build different architectures using LSTM, GRU and stacked Bi-directional Networks to compare there results on the task of subreddit recommendation with the goal of building a system capable of suggesting interesting subreddits successfully at a rate comparable to other similar recommendation systems.

1.2. Reddit

Reddit, often called "The front page of the Internet", is an online community where users share and discuss their topics of interest. These entries are organized by areas of interest called "subreddits", which serve as sub-communities within the overall Reddit community. Reddit has 234 million unique users from 217 different countries and generates 8.19 billion views a month, making it the 7th most visited site in the United States and the 23th in the world Alexa. There exist over 11,464 active subreddits DMR (2016) which continue to grow every day.

1.3. Previous work

The sequence learning capabilities of Recurrent Neural Networks have been applied with state-of-the-art results Zachary C. Lipton (2015) in areas as diverse as phoneme classification Graves and Schmidhuber (2005), machine translation Kalchbrenner N. (2013), speech recognition Alex Graves (2013) , video captioning Haonan Yu (2016), and handwriting recognition Graves (2014), along with unreasonable effectiveness in character-level language models karpathy.

In the domain of Recommender Systems, the application of recurrent neural networks have been investigated for restaurant recommendations David Zhan Liu (2015) and user website session retail recommendations Balazs Hidasi (2016). The recurrent neural network architecture investigatory methodology of Liu et. al. largely informs the methodology used in this work to investigate the application of recurrent neural networks to the specific task of subreddit recommendations.

1.4. Model Hypothesis

The hypothesis of the recommendation model is, given an ordered sequence of user subreddit interactions, patterns will emerge that favour the discovery of particular new subreddits given that historical user interaction sequence. The intuition is, that as users interact with the Reddit ecosystem, they discover new subreddits of interest, but these new discoveries are influenced by the communities they have previously been interacting with. We can then train a model to recognize these emergent subreddit discoveries based on users historical subreddit discovery patterns. When the model is presented with a new sequence of user interaction, it "remembers" other users that historically had similar interaction habits and recommends their subreddits that the current user has yet to discover.

1.5. The CI Methods

1.5.1. Recurrent Neural Networks

The sequential view of user interaction/subreddit discovery is similar in structure to the problem domains listed in Section 1.2, and the wide-range of successes in these problems is supportive of the hypothesis that recurrent neural networks may be powerful in the specific domain of subreddit recommendations.

1.5.2. LSTM Cells

Long-Short term memory networks are a kind of RNN that have the ability of learning long time-dependencies Graves and Schmidhuber (2005). LSTM architecture was introduced by Hochreiter and Schmidhuber and it uses a memory cell to which information can be added or removed by its input and forget layers Olah. LSTMs provide effective results when dealing with sequential data Graves (2014) which is why using this architecture could work for subreddits sequences per user.

1.5.3. GRU Cells

The Gated recurrent units (GRUs) is a mechanism used in RNNs, which has a similar approach to the LSTM architecture. The GRU has been proposed by Cho et al. (2014), and like the LSTM it remembers and forgets its state depending on the input signal. The main difference of GRU is that it fully reveals its memory content each timestep and equilibrates between the previous memory content and the new memory content. This permits a GRU to disregard the previous hidden states each time it is believed necessary taking into account the previous hidden states and the current input Chung et al. (2015). For our recommendation system we chose the GRU to prevent the vanishing gradient problem and to compare it to the LSTM. In various tasks both models produce similar performance and modifying some of the parameters like layer size is usually more significant than choosing between the two. GRUs have less parameters and therefore might train a bit quicker or need less data to simplify. However, if you have sufficient data LSTMs may bring superior results Britz.

1.5.4. Bi-directional Stacked Networks

Bi-directional recurrent networks by Schuster and Paliwal (1997) uses information from the past and the future to produce an output based on better context. Previous experiments have shown that BRNNs outperforms regular RNNs and that multi-stack BNRRs outperforms single BRNNs David Zhan Liu (2015). For our subreddit recommendation system it makes sense to use forward and backward subreddit interactions as information for providing a final recommendation.

2. Results and Discussion

2.1. Dataset

The dataset was compiled using a python scrapper developed using Reddit's PRAW API. The raw data is a list of 3-tuples of [username,subreddit,utc timestamp]. Each row repre-

sents a single comment made by the user, with a total of 30,389,033 collected comments representing 49,812 unique users across 48,153 unique subreddits. Figure 1 depicts an example segment of the raw dataset.

	user	subreddit	utc_stamp
0	kabanossi	photoshopbattles	2016-12-26 10:24:59
1	kabanossi	GetMotivated	2016-12-26 10:23:14
2	kabanossi	vmware	2016-12-26 10:21:16
3	kabanossi	carporn	2016-12-26 10:20:18
4	kabanossi	DIY	2016-12-26 10:17:59

Figure 1: Example Dataset Segment

2.2. Exploratory Data Analysis

Before building the different RNN architectures, an exploration of the dataset is performed to build an intuition of the distributions in the dataset that will help guide the design decisions and parameter ranges of the final models, along with helping identify any missing or anomalous data. For brevity, this exploration is not included in this paper, but can be interacted with in the jupyter notebook supplied here.

2.3. Model Data Structure

To build the training dataset, the subreddit interaction sequence for each user can be ordered and then split into chunks representing different periods of Reddit interaction and discovery. From each chunk, we can randomly remove a single subreddit from the interaction as the "discovered" subreddit and use it as our training label for the interaction sequences. This formulation brings with it a hyperparameter that will require tuning, namely the sequence size of each chunk of user interaction periods.

There are also a couple of design decisions needed that will create inherent assumptions in the model. This includes whether the labeled "discovered" subreddit should be randomly chosen from each interaction sequence, or if there should be a more structured selection. The proposed model utilizes the distribution of subreddits existing in the dataset to weight the random selection of a subreddit as the sequence label, which gives a higher probability of selection to rarer subreddits. This will smoothen the distribution of training labels across the models vocabulary of subreddits in the dataset. Also, each users interaction sequence has been compressed to only represent the sequence of non-repeating subreddits, to eliminate the repetitive structure of users constantly commenting in a single subreddit, while providing information of the users habits in the reddit ecosystem more generally, allowing the model to distinguish broader patterns from the compressed sequences.

Four sequence processing parameters have been introduced to develop the final processed training dataset:

- Sequence Chunk Size - The number of non-repeating subreddit interactions to extract the training sequence/labels from. Final Model Parameter = 15
- Minimum Sequence Length - The minimum number of non-repeating subreddit interactions per training sequence/label pair after the selected label has been removed from the Sequence Chunk Sized processed sequences. Final Model Parameter = 7
- Minimum Subreddit Interaction Count - Filters out subreddits that have very rare interaction, reducing the models vocabulary size. Final Model Parameter = 250
- Maximum Subreddit Popularity - Filters out subreddits from being sequence labels that are highly represented in the dataset, removing the recommendation of very popular subreddits from the model. These subreddits are still used in the training sequences data. Final Model Parameter = 0.00075.

The final model parameters have been selected based on the results of the performed Exploratory Data Analysis. A limitation of the current work is an in-depth study of the effects of these parameters on the final model, and exists as an area of future work for our subreddit recommender system.

Figure 2 depicts the processed training data fed into the models. Each subreddit has been integer encoded, indexed by its position in the models vocabulary. Figure 3 shows some summary statistics for the processed dataset.

	seq_length	sub_label	sub_seqs
0	11	1957	[263, 339, 1, 339, 263, 339, 195, 339, 339, 37...
1	12	1318	[203, 14, 1, 10, 27, 641, 4094, 339, 557, 1504...
2	12	1623	[5, 16, 5, 4968, 5, 1137, 5, 5, 813, 26, 31, 5]
3	11	1488	[89, 92, 237, 419, 39, 1030, 1965, 107, 1336, ...
4	14	2041	[101, 24, 1, 24, 16, 8, 47, 350, 1, 10, 24, 47...

Figure 2: Example Processed Training Sequences

```
Model Vocabulary Size = 5027
Top Popular Filtered Subs = 219
Total Training Sequences = 766014
Training Label Coverage = 95.0
Average Samples per Sub = 152
```

Figure 3: Processed Training Dataset Statistics

These subreddit sequence/subreddit label pairs are then passed to various RNN architectures with an exploration of hyperparamter optimization to select the optimal model for recommending new subreddits of interest to reddit users.

2.4. Architecture Tuning and Results

The loss function utilized in training the models is categorical crossentropy Wikipedia (b). The hyperparameters and value ranges tuned in each model are:

- Learning Rate = [0.00005,0.001]
- Internal Units = [128,256]
- Dropout = [0.1,0.9]

The training/test datasets where split on an 85/15 basis.

Figures 4,5 and 6 depict the training and validation loss vs training step for initial models of single layered LSTM and GRU networks and a double-stacked bi-directional GRU celled network.

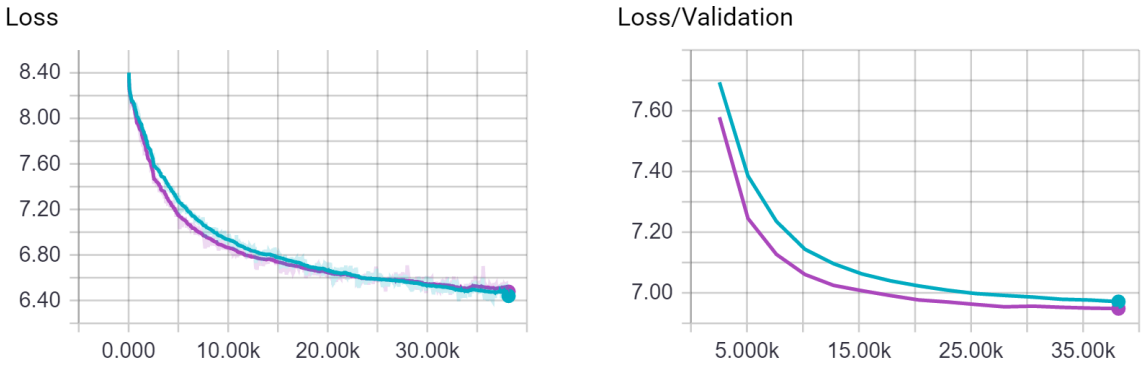

Figure 4: Single Layer LSTM Network Training Results

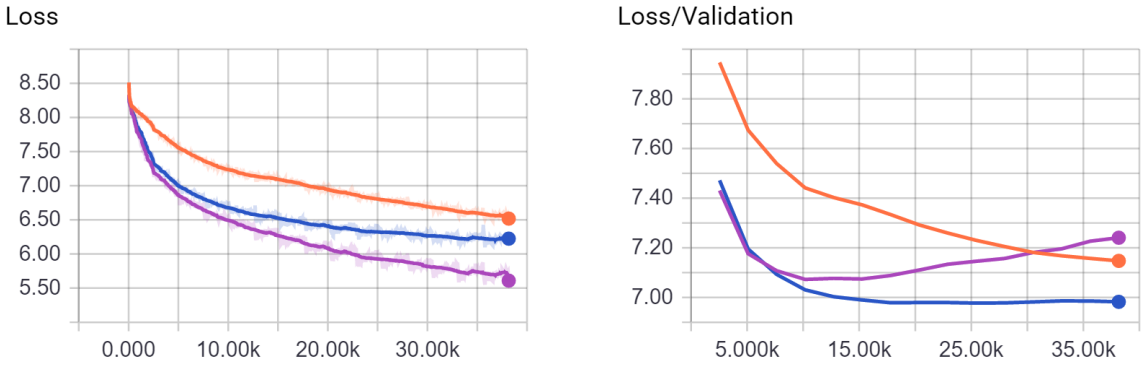

Figure 5: Single Layer GRU Network Training Results

An obvious limitation of the current work is the limited number of simulations for each Cell architecture, and there is likely opportunities for improvements to the final model with more refined hyperparamter tuning.

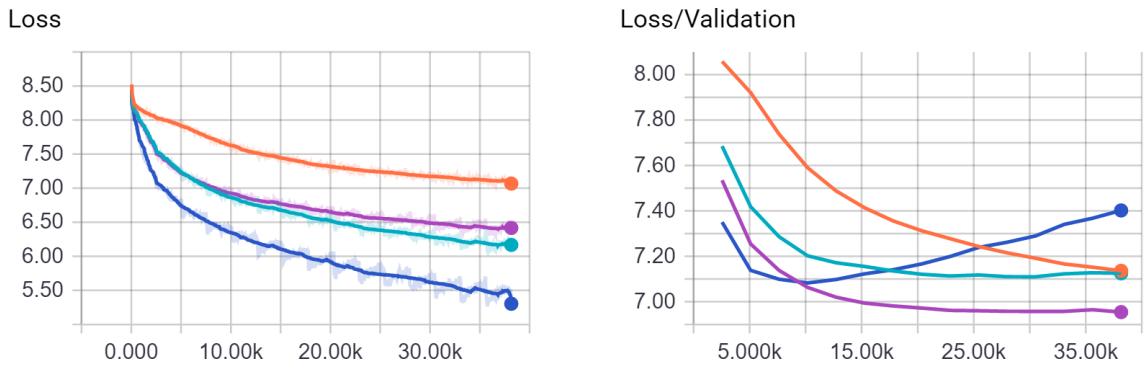

Figure 6: Double-Stacked Bi-Directional GRU Network Training Results

The final performances for the best model in each architecture are nearly identical, indicating that each architecture has sufficient expressiveness to capture the underlying patterns of the data, and the majority of performance gains are likely in finding the best set of hyperparameters to use for each architecture.

The purple line in Figure 6 represents the best performing model out of all Cell architectures, with a Learning Rate = 0.0008, Internal Units = 128 and dropout = 0.23 for both layers of the double-stacked bi-directional network. The final model is trained to 10 epochs, where the validation loss is minimal at a value of 6.96.

2.5. Final Model Validation

Categorical cross-entropy loss is a proxy for the recommender metric we are actually trying to optimize, which is some measure of the relevancy of predicted recommendations to the user. These are known as Ranking Metrics, the most common Zygmunt of which is Mean Average Precision Kaggle , which we use as the final validation metric in this work. It should be noted that it is difficult to directly optimize the discontinuous MAP function, but it has been shown that proxy ranking loss functions produce upper bounds of the measure based ranking errors Wei Chen (2009). Here, we utilize categorical cross-entropy as the proxy loss, but an investigation of the effects of using different ranking-loss functions on the final model should be performed as future work, similar to the approach developed in Hidasi et. al. The final validation set consists of subreddit interaction data for 1,187 users not included during training.

2.5.1. Baseline Validation

In order to evaluate the performance of the final model, we need a baseline performance to compare against and measure the improvements gained from the proposed solution. Common baseline models for recommender systems include Random, Most Popular, and Nearest Neighbor approaches Balazs Hidasi (2016). For our model validation, we forego the nearest neighbor baseline validation model due to the effort required in engineering a suitable comparative validation model, but provide further model validation through analogous recommender system comparison and visual validation in the following sections. Validation

of our model against conventional collaborative or content-based filtering Wikipedia (a) specific to subreddit recommendations is left for future work. Figure 7 summarizes the final baselined results.

MAP Metric	Random	Most Popular	Final Model Score	Random Improvement	Most Popular Improvement
MAP@5	0.0008	0.0201	0.0970	116.44	4.84
MAP@7	0.0006	0.0065	0.0773	132.29	11.87
MAP@200	0.0003	0.0081	0.0373	136.85	4.61
MAP@500	0.0003	0.0086	0.0409	153.53	4.74

Figure 7: Final Model Baselined Results

For the majority of tested MAP metrics, the final model's improvement above the most popular baseline model is about 5x, indicating the model is learning useful patterns in predicting subreddits of interest to a user based on his or her subreddit interaction history.

2.5.2. Analogous Recommender Systems

A benchmark for state-of-the-art subreddit recommendation systems is not available for direct comparison, but the metrics for the broader task of recommending items given user data do exist which we can compare against. The machine learning competition website, kaggle.com, provides a platform for teams to compete in various machine learning challenges, many of which are framed as the development of recommender systems. We compare our models improvement from the Most Popular model to the improvements of the winning teams submission from their competitions Most Popular model with each competitions specific MAP metric. Details for each specific MAP metric and benchmark models are provided in Appendix B. Figure 8 summarizes the analogous validation results.

Competition	MAP Metric	Kaggle Most Popular Improvement	Our Most Popular Improvement
Expedia Hotel Recommendations	MAP@5	10.04	4.84
Santander Product Recommendation	MAP@7	7.46	11.87
Event Recommendation Engine Challenge	MAP@200	1.49	4.61
Million Song Dataset Challenge	MAP@500	8.61	4.74

Figure 8: Final Model Analogous Results

Our model outperforms 2 of the comparison best-in-class models, and underperforms the other 2, but in all cases performances are comparable. This comparison gives context to the performance of our final model, with a 5x performance increase of the Most Popular model being competitive with best-in-class models for large scale recommender systems. Note that the MAP@7 metric's Most Popular model only recommends the highest ranked item for every prediction, where the other MAP metrics recommend a list of decreasing popular items for each recommendation, causing the difference in relative metric values for the Santander competition.

2.5.3. Visual Validation

As part of the learning process in Recurrent Neural Networks, a high dimension embedding space of the network vocabulary is developed to learn weights that clump more similar terms closer together in the embedding Hamid Palangi (2016). This high dimensional embedding can then provide direct visualization into the network using dimensionality reduction techniques to inspect the learned embedding Juliani. We can use this embedding visualization to validate by inspection if the model is learning human anticipated structures. Figures in Appendix A depict the learned embedding space and specific subsections highlighting regions that clearly depict that the model is learning structures that make intuitive sense by human inspection, indicating the model is learning at least partially useful patterns from the data. Although this method does not provide direct validation of the performance of the model, it does provide an intuitive avenue of model inspection that can help with the interpretation of the final model and guide directions for improving our proposed model.

3. Final Discussion

The data, infrastructure and model engineering required to develop a novel recommender system from custom data sources is non-trivial, and the methodology presented here, building on similarly structured systems, provides a powerful abstraction for conceptualizing recommendation tasks in the structure of sequence prediction. This sequence framework lends itself to the powerful modeling capabilities of recurrent neural networks, which we have shown perform competitively with best-in-class models from comparative recommender systems.

The 3-way method of baseline, analogous and visual validation provides a framework for providing model confidence without the need for the development of customized conventional recommender system models. The different perspectives of model performance allow greater confidence in the validity of the final model, and the direct comparison to analogous recommender systems eliminates the potential biases that can occur and the considerable engineering effort required when building meaningful benchmark models.

3.1. Limitations

Throughout the discussion of our work, limitations of the methodology have been identified that exist as points of improvement for further development of our proposed model. These include the need for tuning of data processing parameters, further tuning of each architecture's hyperparameters and the investigation of rank-loss functions as the MAP proxy objective to optimize. Other limitations include the scale of data being fed into the model, consisting of 30 million unique comments, or less than 15 days of Reddit interaction (averaging 2 million comments per day in 2015 Reddit). The limitations on the PRAW API limit the total comments per user to 1000, meaning that the model does not incorporate total user interactions, but a subset of their most recent 1000 interactions. The data scrapping process of iteratively selecting the most recent comments to find new users to scrape, biases the data to include subreddits or user profiles that have active discussions occurring during the scraping process. The final model is also limited by computational resources, forcing the final vocabulary to only represent 5,027 of the 48,153 scraped subreddits. Finally, a

true benchmark for comparing our system to the performance of other recommender systems applied to subreddit recommendations is needed to fully appreciate to what extent the application of recurrent neural networks to this domain improves upon existing techniques. Alternatively, an online validation method tracking the usefulness of our model's recommendations to live users would provide an even richer method of model validation.

3.2. Future Work

3.2.1. Research

Each noted limitation in Section 3.1 provides a jumping off point for further method and model improvement. Our intuition suggests two areas likely to provide the most exploitable model improvements: more data and model tuning.

Incorporating more data into the model is conceptually straight-forward, but will require proportionally more computational resources (and/or time). An investigation of the application of cloud computing platforms, such as Amazon's AWS service, may prove a good fit for improving the proposed model.

Neural Network Architecture design and hyperparameter tuning are time-devouring tasks, and proved to be the biggest limiting component of our work. Many methods of automating these portions of the machine learning pipeline have been proposed and are emerging, including Bayesian Optimization Jasper Snoek (2012), the use of Genetic Algorithms WHITLEY (1995), and the increasingly popular concept of meta-learning Marcin Andrychowicz (2016). The investigation and application of these techniques to the problem definition outlined in this work is likely to yield sizable improvements to the proposed model.

3.2.2. Application

The results indicate that the final model is capable of providing useful information about the Reddit ecosystem and subreddit recommendations for users. Founded on the proposed model and utilizing the learned embedding space for user interaction and self-discovery, a web application could be developed that shows a users current location within the Reddit network, and provide recommendations of where to explore next. The intuitive concept of subreddit distances learned in the embedding space could also be used to compare different users, showing their similar and different interests, and recommending distant and unexplored communities to users to provide perspectives outside their usual filter bubble.

4. Conclusions

The user interaction sequence abstraction for our subreddit recommender system has proven to be a powerful one when combined with the incredibly expressive RNN architecture, with a double-stacked Bi-directional network model slightly outperforming single layered LSTM and GRU cell architectures. The ability of our under-tuned final model to perform comparably with best-in-class models from analogous recommender systems indicates that RNN's may provide performance gains in other similarly structured recommender systems, which is encouraging considering the continuing flood of newly created content.

References

Geoffrey Hinton Alex Graves, Abdel-rahman Mohamed. SPEECH RECOGNITION WITH DEEP RECURRENT NEURAL NETWORKS. *arxiv*, pages 1–5, 2013. doi: https://arxiv.org/pdf/1303.5778.pdf.

Alexa. reddit.com traffic statistics. URL http://www.alexa.com/siteinfo/reddit.com.

Linas Baltrunas Domonkos Tikk Balazs Hidasi, Alexandros Karatzoglou. SESSION-BASED RECOMMENDATIONS WITH RECURRENT NEURAL NETWORKS. *arxiv*, pages 1–10, 2016. doi: https://arxiv.org/pdf/1511.06939v4.pdf.

Denny Britz. Recurrent neural network tutorial, part 4 implementing a gru/lstm rnn with python and theano. URL http://www.wildml.com/2015/10/recurrent-neural-network-tutorial-part-4-implementing-a-grulstm-rnn-with-python-and-thea

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Junyoung Chung, Caglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. Gated Feedback Recurrent Neural Networks. 2015.

Gurbir Singh David Zhan Liu. A Recurrent Neural Network Based Recommendation System. *arxiv*, pages 1–9, 2015. doi: https://cs224d.stanford.edu/reports/LiuSingh.pdf.

DMR. 60 amazing reddit statistics, 2016. URL http://expandedramblings.com/index.php/reddit-stats/.

A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5):602–610, 2005. doi: ftp://ftp.idsia.ch/pub/juergen/nn_2005.pdf.

Alex Graves. Generating Sequences With Recurrent Neural Networks. *arxiv*, pages 1–43, 2014. doi: https://arxiv.org/pdf/1308.0850v5.pdf.

Yelong Shen et al Hamid Palangi, Li Deng. Deep Sentence Embedding Using Long Short-Term Memory Networks. *arxiv*, pages 1–25, 2016. doi: https://arxiv.org/pdf/1502.06922.pdf.

Zhiheng Huang Yi Yang Wei Xu Haonan Yu, Jiang Wang. Video Paragraph Captioning Using Hierarchical Recurrent Neural Networks. *arxiv*, pages 1–10, 2016. doi: https://arxiv.org/pdf/1510.07712v2.pdf.

Ryan P. Adams Jasper Snoek, Hugo Larochelle. PRACTICAL BAYESIAN OPTIMIZATION OF MACHINE LEARNING ALGORITHMS. *arxiv*, pages 1–12, 2012. doi: https://arxiv.org/pdf/1206.2944.pdf.

Arthur Juliani. Visualizing deep learning with t-sne. URL https://medium.com/@awjuliani/visualizing-deep-learning-with-t-sne-tutorial-and-video-e7c59ee4080c#.3op41pt7t.

Kaggle. Mean average precision. URL https://www.kaggle.com/wiki/MeanAveragePrecision.

Blunsom P. Kalchbrenner N. Recurrent continuous translation models. *EMNLP*, pages 1700–1709, 2013. doi: http://anthology.aclweb.org/D/D13/D13-1176.pdf.

Andrej karpathy. The unreasonable effectiveness of recurrent neural networks. URL http://karpathy.github.io/2015/05/21/rnn-effectiveness/.

Sergio Gmez Colmenarejo et. al. Marcin Andrychowicz, Misha Denil. Learning to learn by gradient descent by gradient descent. *arxiv*, pages 1–17, 2016. doi: https://arxiv.org/pdf/1606.04474v2.pdf.

Christopher Olah. Understanding lstm networks. URL http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Reddit. Reddit in 2015. URL https://redditblog.com/2015/12/31/reddit-in-2015/.

Mike Schuster and Kuldip K. Paliwal. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, 45:26732681, 1997.

Yanyan Lan Zhiming Ma Hang Li Wei Chen, Tie-Yan Liu. Ranking Measures and Loss Functions in Learning to Rank. *NIPS*, pages 1–9, 2009. doi: https://arxiv.org/pdf/1511.06939v4.pdf.

D. WHITLEY. Genetic Algorithms and Neural Networks. *citeseerx*, pages 1–15, 1995. doi: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.18.2120&rep=rep1&type=pdf.

Wikipedia. Recommender system, a. URL https://en.wikipedia.org/wiki/Recommender_system.

Wikipedia. Cross entropy, b. URL https://en.wikipedia.org/wiki/Cross_entropy.

Charles Elkan Zachary C. Lipton, John Berkowitz. A Critical Review of Recurrent Neural Networks for Sequence Learning. *arxiv*, pages 1–38, 2015. doi: https://arxiv.org/pdf/1506.00019.pdf.

Zygmunt. Evaluating recommender systems. URL http://fastml.com/evaluating-recommender-systems/.

Appendix A. Embedding Space Visual Inspection

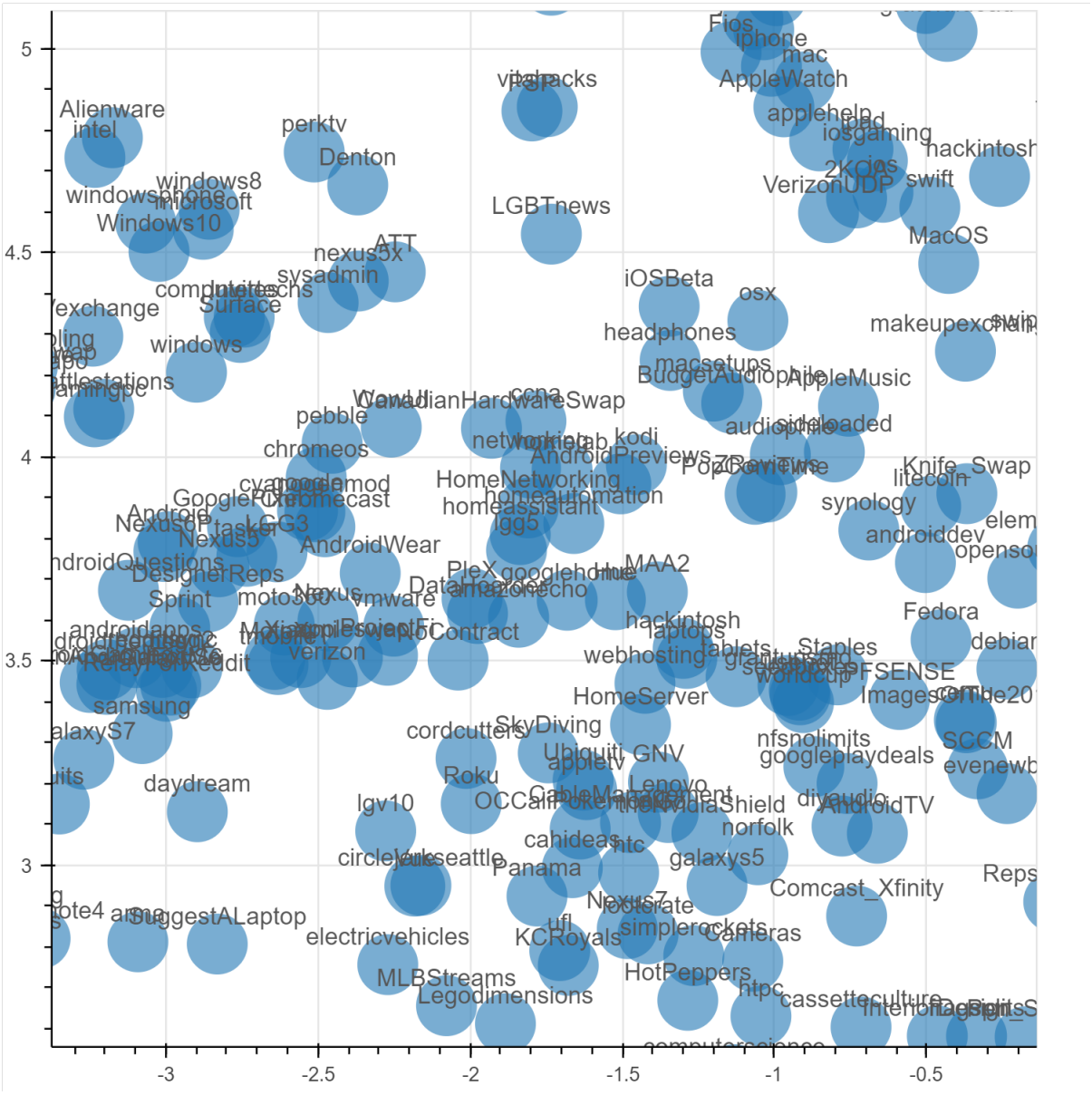

Figure 9: The Operating System Battlefield

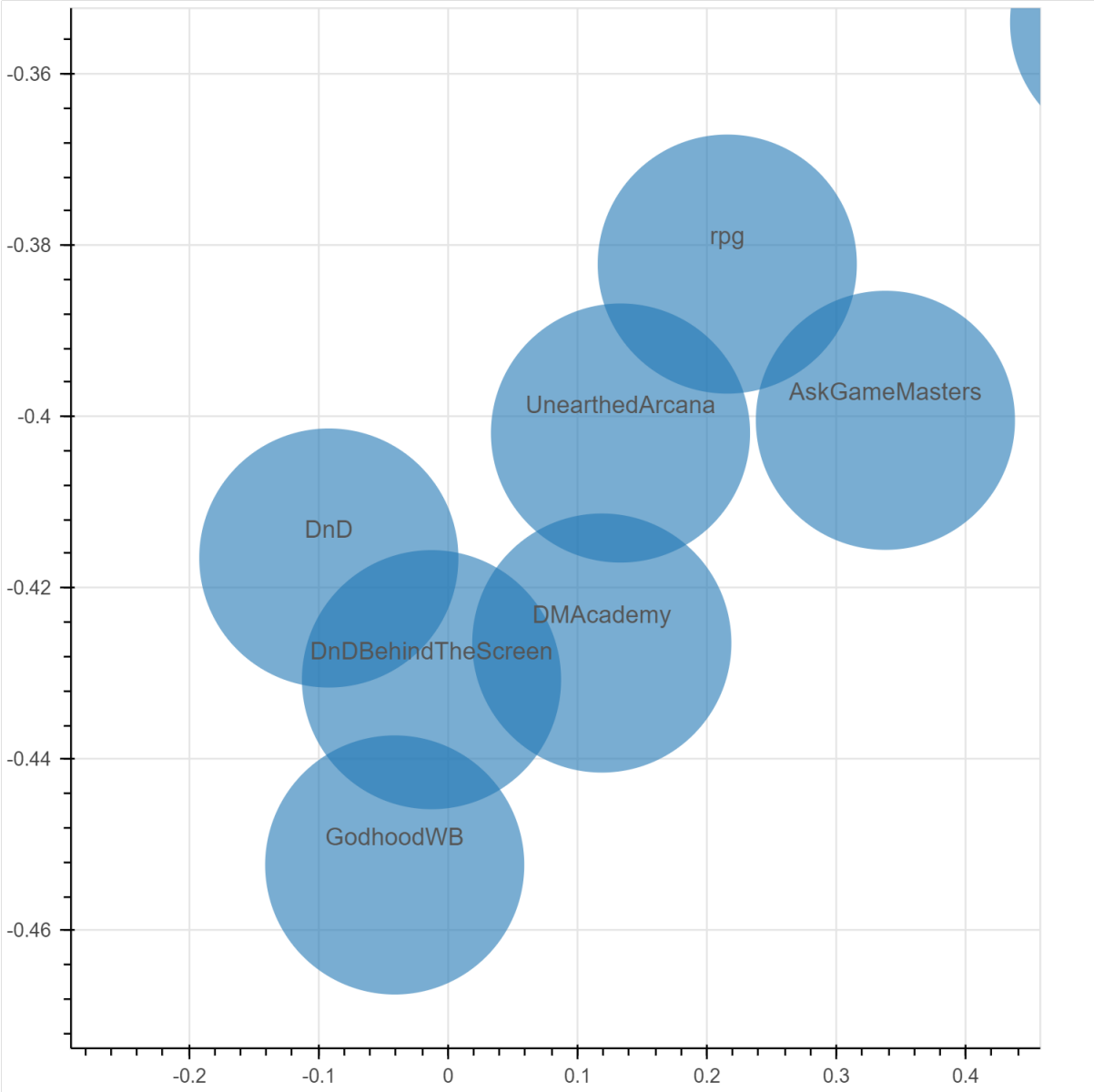

Figure 10: Dungeon Master's Corner

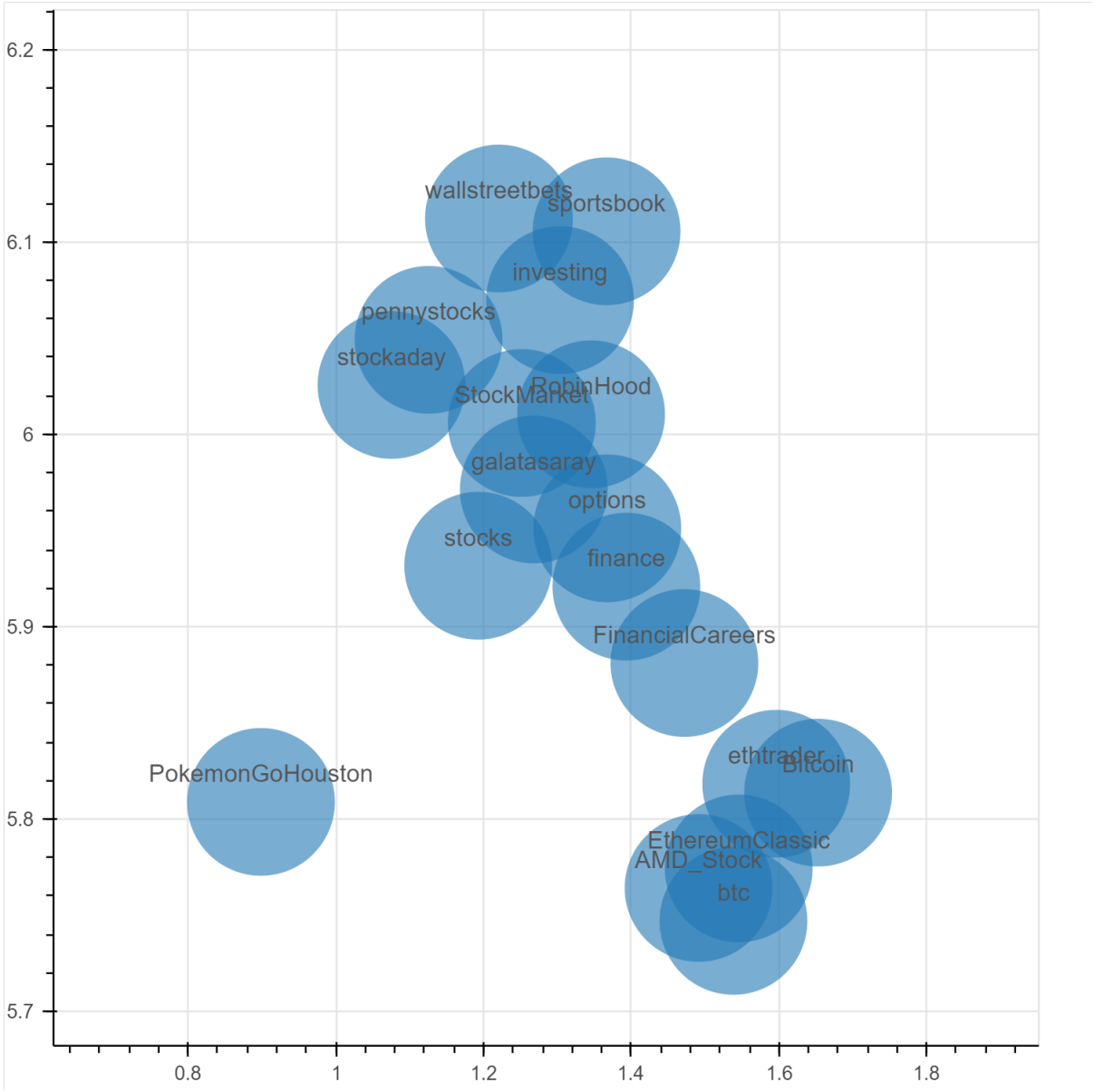

Figure 11: Reddit's White Collar Casino

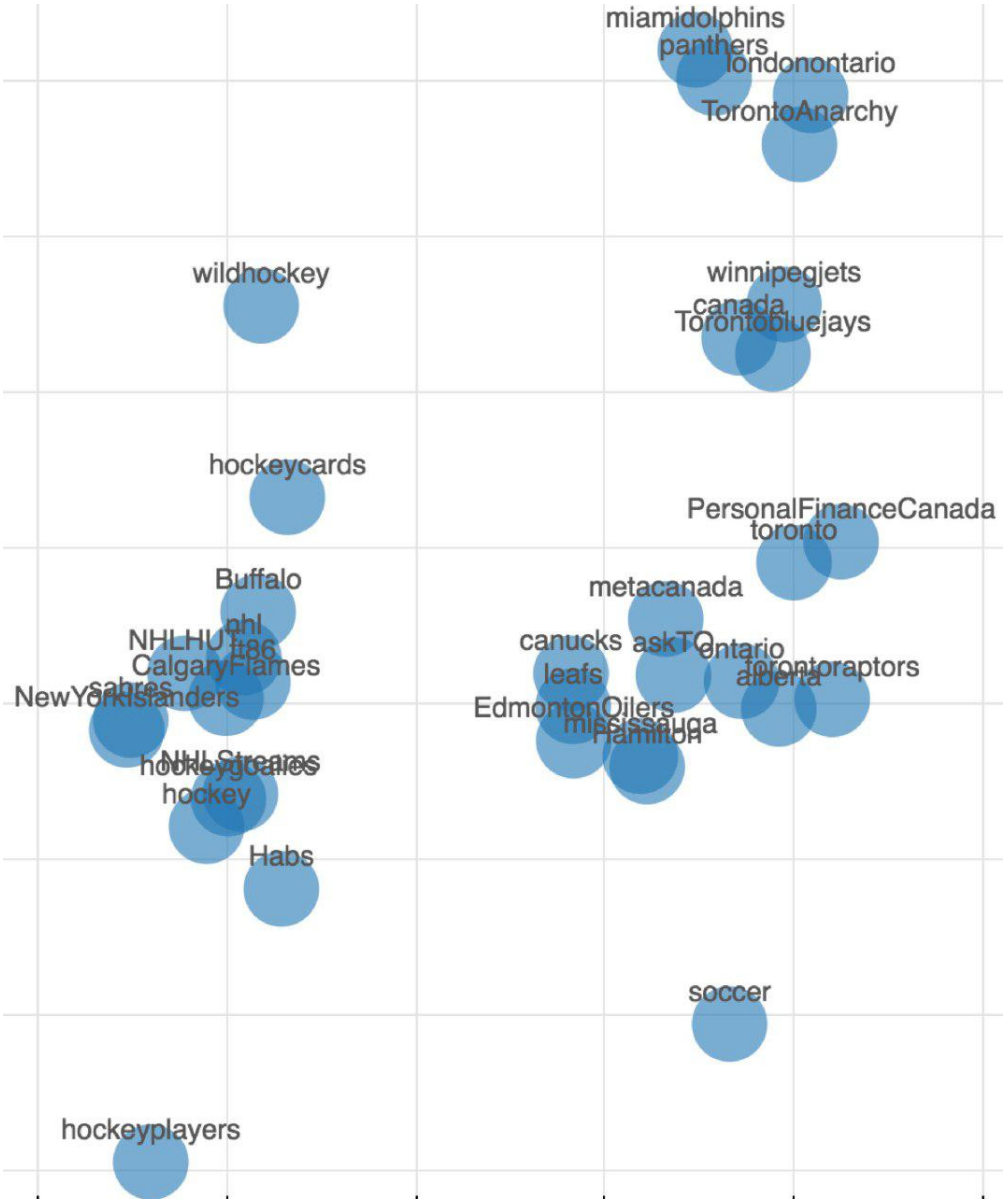

Figure 12: Team Canada

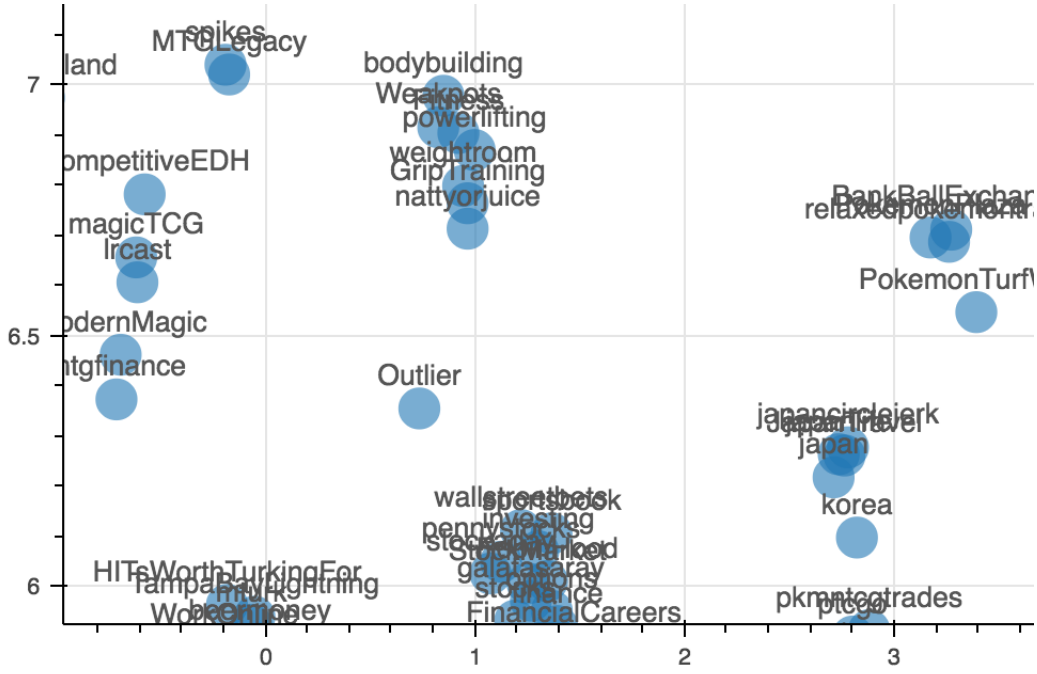

Figure 13: The Outlier outlier

Figure 14: Talken Politics

Appendix B. MAP Metric and Baseline Model Details

Competition	MAP Metric	Benchmark Info
Expedia Hotel Recommendations	MAP@5	Always predict the 5 most frequent hotel clusters across all training data
Santander Product Recommendation	MAP@7	Always Predict Top Most Popular Product
Event Recommendation Engine Challenge	MAP@200	For each user this benchmark recommends events in order from the most people responding "Yes" to the least people responding "Yes"
Million Song Dataset Challenge	MAP@500	Every user is recommended the same 500 songs, in the same order. Songs are ordered by decreasing popularity from the training set

Figure 15: Benchmark Details