Filter, Rank, and Transfer the Knowledge: Learning to Chat

Sina Jafarpour
Department of Computer Science
Princeton University
Princeton, NJ 08540
sina@cs.princeton.edu

Chris Burges
Microsoft Research
One Microsoft Way
Redmond, WA 98052
cburges@microsoft.com

Alan Ritter
Computer Science and Engineering
University of Washington
Seattle, WA 98195
aritter@cs.washington.edu

Abstract

Learning to chat is a fascinating machine learning task with many applications from user-modeling to artificial intelligence. However, most of the work to date relies on designing large hard-wired sets of rules. On the other hand, the growth of social networks on the web provides large quantities of conversational data, suggesting that the time is ripe to train chatbots in a more data driven way. A first step is to learn to chat by ranking the response repository to provide responses that are consistent with the user's expectations. Here we use a three phase ranking approach for predicting suitable responses to a query in a conversation. Sentences are first filtered, then efficiently ranked, and then more precisely re-ranked in order to select the most suitable response. The filtering is done using part-of-speech tagging, hierarchical clustering, and entropy analysis methods. The first phase ranking is performed by generating a large set of high-level grammatical and conceptual features, exploiting dictionaries and similarity measurement resources such as wikipedia similarity graphs, and by ranking using a boosted regression tree (MART) classifier. The more precise (conceptual) ranking is performed by designing more conceptual features obtained from similarity measurement resources such as query refinement and suggestion systems, sentence paraphrasing techniques, LDA topic modeling and structural clustering, and entropy analysis over wikipedia similarity graphs. The sentences are then ranked according to the confidence of a Transfer AdaBoost classifier, trained using transfer-learning methods in which a classification over a large corpus of noisy twitter and live-journal data is considered as the source domain, and the collaborative ranking of actively collected conversations, which are labeled in an online framework using user feedback, is considered as the destination domain. We give results on the performance of each step, and on the accuracy of our three phase ranking framework.

1 Introduction

The notion of designing automated chatbots is at least as old as Artificial Intelligence, since the Turing test requires that machines be endowed with conversational skills. However, most chatbots designed to date (such as ELIZA and A.L.I.C.E), are rule-based agents; a set of rules which drives the system's responses is hard-wired into the system. Although rule-based chatbots are suprisingly successful at engaging users for a while, they are brittle, with one false move starkly revealing their limitations, and they are rigid, with no way of displaying the kind of understanding and creativity that a truly engaging conversation requires.

Even just a decade ago, collecting large amounts of conversational data was impractical. Today however there is a flood of accessible conversational data, arising from social networks such as Twitter, from Live-Journal blogs, and from free semantic web applications. Moreover, new machine learning and ranking algorithms have been designed to manipulate large data sets in practical applications such as information retrieval, natural language processing, and web search ranking [1, 2]. In this paper, we investigate a framework for exploiting these large conversational data sets, using machine learning techniques with the objective of *learning* to chat.

Learning to chat can be regarded as a collaborative filtering problem, in which the suitability of a response is evaluated according to the extent that user likes or dislikes it, and our aim is to come up with an automatic ranking framework consistent with user evaluation. Having provided the system with a large corpus of millions of conversations (in our case twitter and live-journal data), we use an algorithm with three phases in order to rank the whole corpus and to provide a suitable response to the user query[1]. The first phase, which we call "filtering", filters the whole database of twitter responses, shrinking the set of candidate responses from two million to approximately two thousand.

We discuss what characteristics define a good filtering function, and we compare three filtering approaches based on part-of-speech tagging, hierarchical clustering, and entropy analysis. After identifying the filter set, following standard information retrieval approaches, we use a first-phase ranking in order to select more relevant candidate responses from the filter set. We designed a set of 300 features, including high-level features, grammatical features, and concept-based features, for the first phase ranker. The ranking is done using boosted regression trees (the MART classifier)[1].

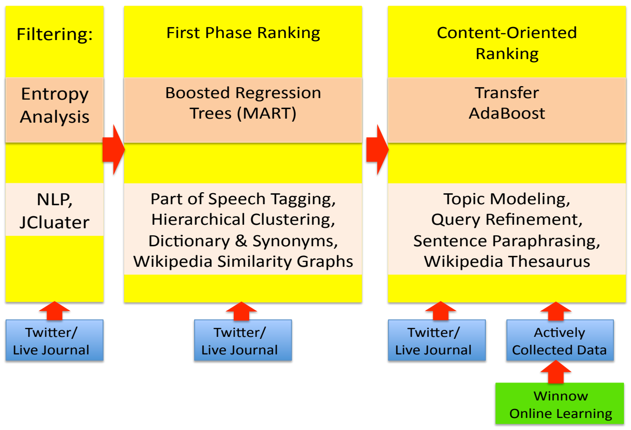

Figure 1: Three ranking phases for generating a response.

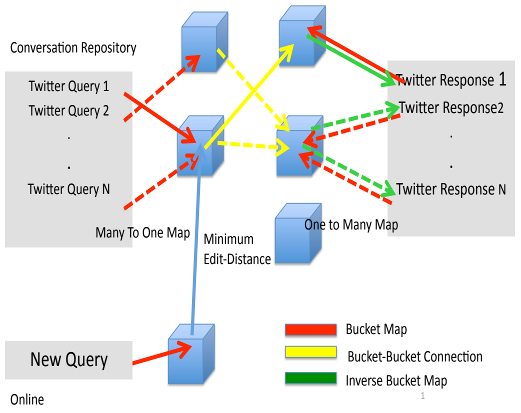

Figure 2: Mapping sentences into buckets for filtering.

Approaching the task of learning to chat by finding the best response in a large database of existing sentences may thus be viewed as a ranking problem. The next step (which we do not address in this paper) would be to alter the top ranked response to generate a better response, for example, by leveraging a model of the user's interests, or of the interests of users who had similar conversations. Here, we attempt to improve the selection of responses using a collaborative filtering step. Thus first, the unlabeled (and very noisy) twitter and live-journal datasets are used in the related classification task of distinguishing the *exact* response following a query, from a random response. A smaller data set is then actively collected and labeled by users, for which we designed an online-rating web-application for collecting data. We then use a transfer learning method (here, Transfer AdaBoost) [3, 4], in order to learn the destination (ranking) domain, exploiting the large number of instances available in the source (classification) domain. Having incorporated all two million source instances, we show that by then in addition exploiting the noise-free destination data, the accuracy of the transfer learning ranker increases on the task.

2 Filtering and First-Phase Ranking

Given a query and a corpus of millions of candidate sentences, a mechanism is needed to efficiently remove irrelevant sentences to reduce the set of candidate responses. A filtering is a many-to-one mapping from sentences to some

[1]We use the term "query" to denote the current sentence in a conversation, for which a response is desired, in analogy with the ranking task in IR.

string patterns known as *buckets*. Intuitively, one measure for selecting appropriate filtering methods is the extent to which either semantically related queries are mapped to the same bucket or their following responses are mapped to the same bucket. This intuitive argument can be made more precise using McDiarmid's inequality and tight concentration arguments:

Theorem 1 *For each query q^*, let $S_{q^*} = \{(q_1, r_1), \cdots, (q_n, r_n)\}$ be a set of appropriate responses r_i to q^*, coupled with the true preceding queries q_i in the repository, and let $\hat{r}_{q^*}$ be the most relevant response in S_{q^*}. Suppose M queries $\langle q_1^*, \cdots q_M^* \rangle$ are sampled iid from a distribution $\mathcal{D}$ for filtering, and for each q_i^*, ℓ samples $\langle r_1^i, \cdots, r_\ell^i \rangle$ are drawn iid from a distribution $\mathcal{D}_{q_i^*}$. Finally let $\mathcal{B}$ denote the bucket map, and suppose there exists a positive θ such that for all query q_i^*,*

$$\max \left\{ \Pr_{r_j^i} \left[\mathcal{B}(r_j^i) = \mathcal{B}(\hat{r}_{q_i^*}) \right], \Pr_{{r_j}^i} \left[\mathcal{B}(q_j^i) = \mathcal{B}(q_i^*) \right] \right\} \geq \theta.$$

Then with overwhelming probability, on average, each query has $\left(\theta - \sqrt{\frac{\log M}{\ell M}} \right) \ell$ suitable responses captured by mapping .

We investigated three heuristics for selecting the filtering function. In the first approach we used Part of Speech Tagging and Natural Language Processing methods [5]. Each sentence is first mapped to its chunked POS tag, and is then encoded to a binary string using Huffman coding. We use Huffman coding because it provides a hierarchical representation of the tags, i.e. tags with the same frequency (as a heuristic for similarity) have similar binary encoding. The second heuristic uses JCluster [6]. JCluster is a recursive algorithm for generating hierarchical clustering of the words. Let ABC denote trigrams in the corpus and b denotes the cluster of B. At each iteration JCluster splits each cluster to two sub-clusters so that $P(C|b)$ has minimal entropy. For instance, the words "you" and "u" are considered similar by JCluster. The 1-JCLuster bucket-map of each sentence can then be computed efficiently as follows: for each word of the sentence, the corresponding JCLuster index is extracted, and truncated to be limited to its most significant bits. The reason that 1-JCluster bucket-mapping is used is first to obtain a sufficiently large number of candidate sentences and second, we observe empirically that there exist a large number of similar queries that have the same 1-JCLuster representation but whose 2-JCLuster representation differs, which decreases the quality of the candidate responses.

The last heuristic combines an entropy analysis with hierarchical clustering. The major difference between this approach and the previous approaches is that in this approach a bucket map is many-to-many mapping, i.e. a set of binary strings, while in the previous approaches a bucket-map was always a single binary string. In this approach we first remove the common, low entropy words from each sentence. These words are then mapped to their JClusters indices which are truncated to their seven most-significant bits. Experimental results suggest that the third approach almost always provides more suitable responses comparing to the first two heuristics. Having performed the filtering to remove a large fraction of irrelevant sentences, a boosted regression tree classifier (MART) is used for finding suitable responses. A set of informative features were designed to train the MART classifier so that the ranking of the candidate filtered responses for a new query can be done using the confidence of the classifier. Our feature set incorporated the grammatical properties of the (query/response) pair, such as part of speech tagging, the parsing tree, the hierarchical clustering, and high-level features (such as whether the query is a question and the response is a statement, etc), and content-based features exploiting public resources such as dictionaries and synonyms, and also wikipedia similarity graphs.

The final problem is to provide labeling for training examples. We use the following method for labeling instances: for each query, its true response (the sentence following that in the twitter conversation) is labeled positive, and a random response is selected and labeled as negative. The labeled set is then divided uniformly into training, cross validation and test set. Having run the MART algorithm with 200 rounds, we observed that the test error continuously drops from 50% to 32% (which is 36% relative improvement); furthermore, no sign of over-fitting is observed. However, having tried the same learning task with 10 random negative responses, rather than just one, for each query, the test error dropped from 9% of random guess to 7.5% (which is only 16% relative improvement). Hence, although the test error still drops, which is a good news for the first-phase ranking, the improvement over the random guess decays as the number of random negative instances increases. The main reasons for this behavior are (1) the twitter data is very noisy; there are lots of generic responses, outliers, inconsistencies and errors in this labeling approach, and (2) chatting is a different task from distinguishing a random twitter response from the true response (although the two tasks are related). We tried to solve the first difficulty by designing an active data-collecting web application, and the second issue by using transfer learning methods.

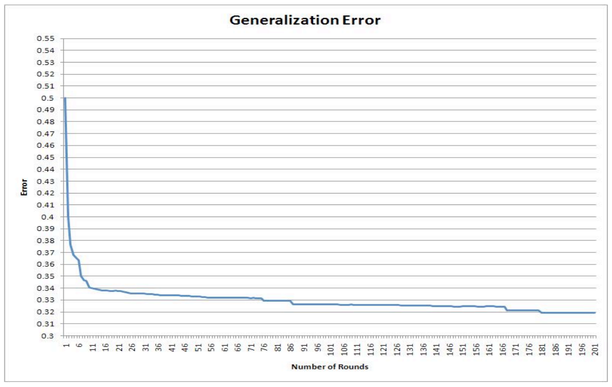

Figure 3: Test error vs. number of trees for MART(first phase ranker)

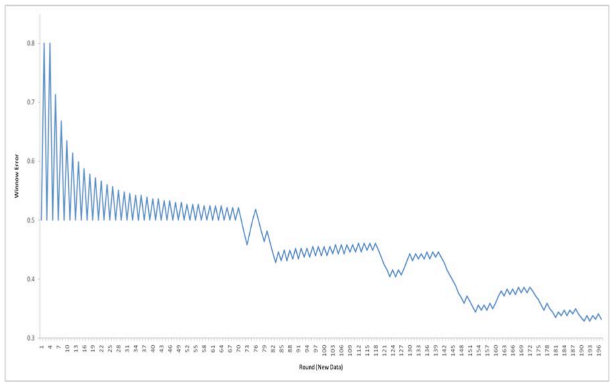

Figure 4: Winnow (online learning) generalization error vs. round

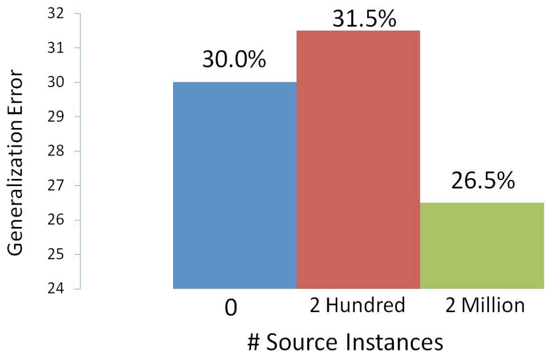

Figure 5: TrAdaBoost(transfer learning)

Figure 6: Generalization error of different phases of the ranking process

3 Collaborative Filtering, Online Learning, and Transfer Learning

To overcome the difficulties with Twitter and Live-Journal data in terms of noise and outlier effects in labeling, we designed a web-application game in order to actively collect less noisy training data. In this game, a query (with some conversation history), and a candidate response is provided to the user. Note that the main goal of this game is just collecting more accurate conversational data for the content-oriented learning phase. The web application provides a query and a candidate response at real-time. As a result, even though using MART might provide better responses, due to the necessity of real-time ranking and online learning requires fast and *online* learning methods, the *winnow* algorithm [7], an online multiplicative update algorithm which tends to be both fast and robust, is used for online candidate generation. The user then presses the like or dislike button. If dislike is pressed, the online classifier is updated and another response is generated, and if the user dislikes three such responses, then the user is asked to type a new, suitable response. If on the other hand the user liked one of the three responses, then the previous query is added to the chat history, and the new response becomes the next query. The online learning classifier is always updated after the user's action. Since the data is collected during an interactive game with labelers, collecting a large number of labeled examples requires a server managing the winnow algorithm and serving all human labelers simultaneously. This is where the term *collaborative learning* comes from. The Winnow is updated from all human labelers at the same time, and managing parallel sessions not only helps in speeding up the data collection rate, but it also provides advantages in training the winnow classifier, making the game more fun for the users. We enlisted the help of our local colleagues to play the game and as a result collected one thousand labeled samples.

Table 1: Some Examples of Automatically Generated Responses

Query	Suggested Response
I play football a lot	how long have you been playing ? i 've been on 2 years
I like Oranges	unless you eat like 10 acre of them at one time ... no
How is the weather in Seattle?	perfect for a day off ...

The question naturally follows: is it possible to exploit the large number of twitter instances, together with the data from the web game, in order to train a more accurate classifier? We propose a knowledge transfer solution for this question using the Transfer AdaBoost algorithm. Transfer learning tries to exploit the available information provided in one domain, known as the *source* domain, in order to improve or facilitate the task of learning another domain, called *destination domain*. In our chat-bot example, the source domain is the set of twitter and live-journal data, and the destination domain is the set of actively collected conversations.

Having identified the source domain, and the destination domain, we design a set of content-oriented features in order to capture as much information as possible between a query and the possible responses. The features include the first-phase ranking features as well as features learned from the topic modeling of a fresh twitter corpus using HMM and LDA Analyses [8], and extracted from query refinement tables and sentence paraphrasing techniques [9]. To train the TrAdaboost, we use 400 positive and 400 negative actively sampled conversations as the destination domain, and 2 million twitter conversations as the source domain. The destination set is then randomly divided to training and test sets with the same size. It turns out that with 2000 random twitter data the test error of the TrAdaBoost algorithm increases. However, whenever the whole twitter corpus is used, the error over the test set drops to 26.4%, comparing to 30.2% of not using knowledge transfer on a set of 200 positive and 200 negative human-labeled instances. Having trained TrAdaBoost, given a new query, the response is generated by first filtering the twitter corpus, then picking 100 sentences for which the trained MART algorithm has the highest confidence, and among them finding the sentence for which the TrAdaBoost has the highest confidence.

4 Conclusion

In this paper, we modeled the learning-to-chat problem as a collaborative ranking task, and we designed a triphase ranking framework consisting of filtering, first-phase ranking using MART, and conceptual ranking using Transfer AdaBoost. At each step, designing informative but not restrictive features play a key role, and to accomplish this a large variety of different similarity resources have been used. Like any other machine learning task, availability of a large corpus of conversational instances will lead to better ranking results and possibly more suitable responses. The Twitter data is extremely noisy and our work leads us to believe that training chat-bots with less noisy, massive training sets should further improve performance. A complementary task to collaborative filtering is user modeling, which is likely to provide valuable information regarding the response to be chosen. How to build such user models in response to the user's interaction with the chatbot is an interesting direction for future research.

References

[1] J. Friedman, "Greedy Function Approximation: A Gradient Boosting Machine," *Annals of Statistics, Vol. 29, PP. 1189-1232*, 1999.

[2] C. Burges, T. Shaked, E. Renshaw, M. Deeds, N. Hamilton, and G. Hullender, "Learning to rank using gradient descent," in *ICML, pp. 89-96*, 2005.

[3] S. Pan and Q. Yang, "A Survey on Transfer Learning," *IEEE Transactions on Knowledge and Data Engineering (IEEE TKDE)*, 2009.

[4] W. Dai, Q. Yang, G. Xue, and Y. Yu, "Boosting for Transfer Learning," *ICML'07, PP. 193 - 200*, 2007.

[5] A. Bies, M. Ferguson, K. Katz, R. Macintyre, M. Contributors, V. Tredinnick, G. Kim, M. Marcinkiewicz, and B. Schasberger, "Bracketing Guidelines for Treebank II Style Penn Treebank Project," 1995.

[6] J. Goodman, "JCLUSTER, a fast simple clustering program that produces hierarchical, binary branching, tree structured clusters.," *http://research.microsoft.com/en-us/um/people/joshuago/*, 2009.

[7] N. Littlestone, “Learning Quickly When Irrelevant Attributes Abound: A New Linear-threshold Algorithm,” *Machine Learning PP. 285-318*, 1998.

[8] D. Blei, A. Ng, and M M. Jordan, “Latent Dirichlet Allocation,” *JMLR, Vol. 3, PP. 993-1022*, 2003.

[9] S. Cucerzan and R. White, “Query suggestion based on user landing pages,” *ACM SIGIR*, 2007.

[10] S. Young, M. Gasic, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, and K. Yu, “The Hidden Information State Model: a practical framework for POMDP-based spoken dialogue management,” *Computer Speech and Language. http://mi.eng.cam.ac.uk/ farm2/papers/ygkm09.pdf*, 2009.

[11] K. Church, “A stochastic parts program and noun phrase parser for un- restricted texts,” *Proceedings of the Second Conference on Applied Natural Language Processing*, 1988.

[12] S. DeRose, “Grammatical category disambiguation by statistical optimization,” *Computational Linguistics, Vol. 14*, 1988.

[13] T. Hufmann, “Unsupervised Learning by Probabilistic Latent Semantic Analysis,” *Machine Learning Journal, Vol. 42, PP. 177-196.*, 2001.

[14] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, 2003.

[15] J. Weizenbaum, “ELIZA a computer program for the study of natural language communication between man and machine,” *Communications of the ACM, Vol.9 PP. 36-45*, January 1966.

[16] R. Wallace, “From Eliza to A.L.I.C.E,” *http://www.alicebot.org/articles/wallace/eliza.html*, 2004.

[17] Y. Freund and R. Schapire, “Game theory, On-line prediction and Boosting,” *In Proceedings of the Ninth Annual Conference on Computational Learning Theory, PP. 325-332*, 1996.

[18] Q. Wu, C. Burges, K. Svore, and J. Gao, “Ranking, Boosting, and Model Adaptation,” in *Tecnical Report, MSR-TR-2008-109*, October 2008.

[19] C. Burges, “A tutorial on Support Vector Machines for pattern recognition,” *Data Mining and Knowledge Discovery, Vol 2, PP. 955-974*, 1998.

[20] R. Caruana, S. Baluja, and T. Mitchell, “Using the Future to Sort Out the Present: Rankprop and Multitask Learning for Medical Risk Evaluation,” *Advances in Neural Information Processing Systems (NIPS ’95)*, 1995.

[21] G.Cavallant, N. Cesa-Bianchi, and C. Gentile, “Linear classification and selective sampling under low noise conditions,” *Advances in Neural Information Processing Systems*, 2009.

[22] N. Craswell and M. Szummer, “Random walks on the click graph,” *SIGIR ’07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval), pp. 239-246*, 2007.

[23] W. Dolan and C. Brockett, “Automatically Constructing a Corpus of Sentential Paraphrases,” *Third International Workshop on Paraphrasing (IWP2005), Asia Federation of Natural Language Processing*, 2005.