

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

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

Christopher J.C. Burges
Microsoft Research
One Microsoft Way
Redmond, WA 98052
cburges@microsoft.com

July 12, 2010

Abstract

We propose a discriminative approach for automatically training chatbots to provide relevant and interesting responses. In contrast to most prior work, our approach is not based on hard-wiring response rules, but rather relies on machine learning. We set ourselves the task of ranking a repository of responses to find the most suitable response. This work is just a first step towards the more general goal of then modifying the result to form a more appropriate response. We use a large corpus of public Twitter and LiveJournal conversations as training data for the learning task. Selecting an appropriate response from this repository, given new input from a user, is done in three phases. First, a fast filtering approach removes most irrelevant sentences. Second, a boosted tree ranker (using features that are very efficient to compute) further shrinks the set of candidate responses. Finally a more precise content-oriented ranking framework is used to output the final response. In addition to our offline repository of dialogs, we also exploit a smaller repository of human-generated and labeled instances. These data are collected through a web-application in which human users interact with the system and provide suggestions and feedback regarding the responses. The response selection is mainly based on content-oriented features and uses the “winnow” multiplicative weight online learning approach. Having a large corpus of noisy offline Twitter and LiveJournal data as a source knowledge domain, and a moderate repository of less noisy, labeled online conversations as a destination knowledge domain, we use a transfer learning method (Transfer AdaBoost) to build a classifier that outputs the final response. We show both qualitative and quantitative results on the performance of the various approaches used in the project.

Microsoft Research Technical Report MSR-TR-2010-93

*This work was done while visiting Microsoft Research, Redmond.

Contents

| | | |
|----------|---|-----------|
| 1 | Introduction | 2 |
| 2 | Basic Definitions and Chat Terminology | 3 |
| 3 | Corpus of Offline Conversational Data | 3 |
| 4 | Filtering | 5 |
| 4.1 | Part of Speech Tagging | 5 |
| 4.2 | JCluster Filtering | 5 |
| 4.3 | Content-Oriented JCluster Filtering | 5 |
| 5 | First-Phase Ranking | 6 |
| 5.1 | Feature Selection | 6 |
| 5.2 | Multiple Additive Regression Trees | 8 |
| 6 | Online Learning and Active Data Collection | 8 |
| 6.1 | Active Data Collection and Labeling | 8 |
| 6.2 | Content-Oriented Features | 11 |
| 7 | Transfer Learning | 13 |
| 8 | Conclusion | 14 |

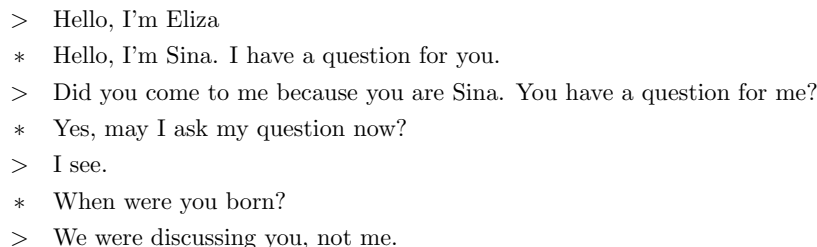
List of Tables

| | | |
|----|---|----|
| 1 | Examples of Twitter Conversations | 4 |
| 2 | The dependency between the frequency, and the length of the conversation in Twitter dataset. | 4 |
| 3 | Examples of a LiveJournal Conversation | 4 |
| 4 | Example of Filtering via Part of Speech Tagging and Hoeffman Coding | 5 |
| 5 | Example of Filtering via 1-bit and 2-bit JCluster indexings | 5 |
| 6 | An example of content-oriented JCluster Mapping | 6 |
| 7 | The most similar words for the words “Barack Obama” and “Hiking” extracted from wikipedia articles using a backward random walk | 7 |
| 8 | Examples of generic and hard to interpret Twitter dialogs | 8 |
| 9 | Substitution and paraphrasing results for a Twitter sentence | 11 |
| 10 | Examples of the words clustered in the topics named “book” and “film” using an LDA topic model of our corpus of Twitter data | 12 |
| 11 | Suggested Responses for Query “I like oranges”: | 15 |
| 12 | Suggested Responses for Query “I play football a lot”: | 15 |
| 13 | Suggested Responses for Query “How is the weather in Seattle?”: | 15 |

1 Introduction

The task of designing automatic chatbots is at least as old as the invention of the “Artificial Intelligence Principles” [1]. In 1923, Boring, in a somewhat circular definition, defines intelligence as *that which is measured by intelligence tests*, and intelligent human-computer chat can be regarded as one of the most natural and fundamental of intelligence tests. The importance of designing intelligent chatbots is further highlighted by the Turing Test, as a measure of achieving artificial intelligence, proposed by Alan Turing in 1950 [2].

Since the Turing Test was proposed, many have attempted to design chatbots with the goal of being able to at least partially pass the test. In 1966, Joseph Weizenbaum designed ELIZA [3], a rule-based chatbot which works by examining a user’s typed comments for keywords. If a keyword is found, a rule is applied which transforms the user’s text and the resulting sentence is returned. If a keyword is not found, ELIZA responds with either a generic riposte or by repeating one of the earlier comments. In addition, Weizenbaum developed ELIZA to replicate the behavior of a Rogerian psychotherapist, allowing ELIZA to be *free to assume the pose of knowing almost nothing of the real world* [3]. Even with these very simple techniques, Weizenbaum’s program was able to temporarily convince some people that they were talking to a real person. However, in most cases, and especially in more wide ranging conversations, ELIZA is brittle. Figure 1 illustrates an example of a case where ELIZA rapidly failed.



> Hello, I'm Eliza
* Hello, I'm Sina. I have a question for you.
> Did you come to me because you are Sina. You have a question for me?
* Yes, may I ask my question now?
> I see.
* When were you born?
> We were discussing you, not me.

Figure 1: Example of a simple conversation in which Eliza quickly failed the Turing Test

Since then several more robust and complicated rule-based chatbots such as Parry [4], HIS system [5], and ALICE (Artificial Linguistic Internet Computer Entity) [6] have been designed. ALICE applies heuristic pattern matching rules (specified in AIML, “Artificial Intelligence Markup Language”) to the human’s input. However, ALICE is also brittle, due to its rule-based nature. Rule based systems simply cannot capture the complexity and novelty of a human conversation.

The time seems ripe to leverage a host of powerful machine learning and information retrieval methods, such as learning to rank [7, 8] and transfer learning [9, 10], together with the availability of large (but noisy) public datasets, such as blog and social network data, to attempt to construct a compelling chatbot. This paper is a first step in that direction.

Section 2 introduces the terminology used in this paper. Section 3 describes the two large sources of conversational data we used: Twitter conversations, and LiveJournal blog posts. Section 4 discusses the problem of filtering sentences in order to quickly remove irrelevant sentences. The filtering paradigm is explained and three different heuristics for filtering are introduced and compared. Section 5 explains the first-phase ranking, which further shrinks the candidate set. We describe the boosted tree classifier (MART) we use, as well as the feature generation methods which exploit natural language processing, information retrieval and text mining techniques for generating high-level, informative features. In Section 6 we propose a user-friendly web-application, capable of interacting with users in real-time with the purpose of collecting less noisy labeled conversational data, via an online game with users. The Winnow online learning algorithm [11] was used to suggest candidate sentences in the game. Finally, having collected a large corpus of offline instances, and a moderate repository of online labeled examples, in Section 7 we show how transfer learning and knowledge transfer methods can be used to train a classifier exploiting the less noisy conversations while ignoring the noisy or content-free conversations. Finally we present experimental results on the performance of the transfer learning approach.

2 Basic Definitions and Chat Terminology

In this section, we give the terminology we use to describe the task. Here we clarify the ingredients of the chat modeling (although in this work, we do not address the user modeling part).

- **Agent (Chat-bot)** The chat-bot is the automated agent which responds to the human’s input.
- **User** The human or machine that the chat-bot is talking to. Each chat consists of a sequence of conversational responses between the chat-bot and the user.
- **Dialog Repository** The main repository of conversations that the chat-bot uses to learn how to chat, as well as to choose an appropriate (possibly modified) sentence for the real-time conversation.
- **User Model** The user model is the offline knowledge the system has about the user before starting the chat. A successful chat session will usually require some understanding of the user’s interests, knowledge, etc. The user model may be constructed, for example, from a social network, from the user profile in company data, or from the user’s search behavior on the web.
- **Chat History** The chat history provides the online information about the user and is complementary to the user model. As the chat proceeds, more information is revealed about the user, their interests, their current situation, etc.

With the user model and chat history in hand, a well trained chat-bot using a sufficiently rich dialog repository and a reasonably large set of chat features should be able to generate an appropriate response and propel the conversation. We can also split every chat between two entities into three parts:

- **Context (History):** Context encapsulates the history, i.e the set of previous sentences, in a chat. For instance, in a greeting between two people Mindy and Sam, the context may be:

Mindy: “Hi, how are you doing?”

Sam: “I’m fine, and you?”

Mindy: “Thanks, how was your weekend?”

Sam: “It was great! We went hiking!”

- **Query (Current Sentence):** The *query* is the last (current) statement in a chat. We use the term query by analogy with the information retrieval task, since a core component of our system is the retrieval of a suitable sentence from the dialog repository to use as a response. For instance, in the above chat between Mindy and Sam, the query might be

Mindy: “Awesome! Where did you go?”

- **Response (Next Sentence):** The response, as its name suggest, is the sentence following the query, and is generated by the agent. For instance, a typical response to Mindy’s query may be

Sam: “We went to Mt. Ranier.”

3 Corpus of Offline Conversational Data

The first step in training a discriminative chat-bot is collecting a large corpus of training examples. In this project we used two sources of public datasets. A crawler was used to collect more than two million Twitter conversations (see Table 1 for four examples of query/response pairs). We consider a Twitter post as a conversation if it has at least one comment. The sequence of comments are then treated as the conversation in a chat. We observed that most Twitter conversations are short. Table 2 shows the dependency between the conversation length (which is defined as the number of comments plus the query itself) and the frequency of that length conversation.¹

The second source of instances we used (though not as large as the Twitter data) is a LiveJournal dataset. Here, each blog post (with a threshold on its length) was treated as the starting point of a chat, and then the corresponding comments were treated as the following sentences in the conversation. Table 3 indicates an example of one LiveJournal query and its corresponding response.

¹Numeric rounding is the cause for the 1% truncation error.

Table 1: Examples of Twitter Conversations

| | |
|----------|--|
| Query | -usr- lunchtime latte it is then . wanki frappucino here i come. after noodles. -smi- |
| Response | wanky, wanky, wanky, what a word |
| Query | i'm a bit disfunctional today, did n't sleep very well. would a coffee help? |
| Response | a coffee or tea almost always helps me, but i am a hopeless caffeine junkie |
| Query | has exciting plans for today ! one of which is cleaning up cat sick . |
| Response | -usr- sorry. seems i helped ruin you and henry. :((((|
| Query | can't do school night drink anymore. even if there is no school the following day :(. |
| Response | -usr- get a job |

Table 2: The dependency between the frequency, and the length of the conversation in Twitter dataset.

| Conversation Length | Frequency |
|---------------------|-----------|
| 2 | 46% |
| 3 – 5 | 38% |
| 6 – 9 | 12% |
| 10 and more | 3% |

Table 3: Examples of a LiveJournal Conversation

| | |
|------------|--|
| Query | -usr- fantastic message for the new beginning .. thank you . |
| Response 1 | Wish you and your family a happy and prosperous year ahead :-) |
| Response 2 | Wishing you great timing ahead. Godspeed. |

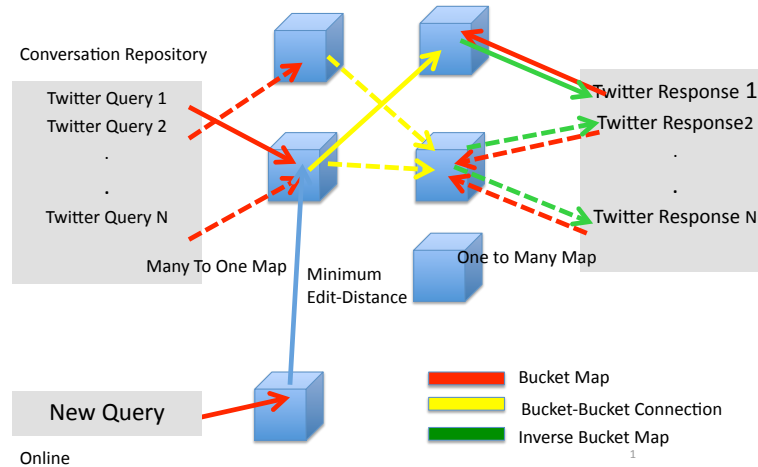


Figure 2: A high-level overview of filtering. Filter maps are shown with red-arrows. For time efficiency, the inverse bucket map is also computed offline for response sentences, which is demonstrated by green arrows. Two buckets are connected if one contains a query and the other contains its corresponding response. Connections between the buckets are shown by yellow arrows.

4 Filtering

The goal of filtering is to efficiently reduce the set of candidate responses from the whole corpus to a set of sentences that still contains a reasonable number of suitable responses. The filtering process is performed through a filter map, a many-to-one function that maps the sentences to binary strings called “buckets”. Two buckets are connected if there exists a pair (query, response), mapped to those buckets respectively.

We investigated three filtering techniques: part of speech tagging, JCluster filtering, and content-oriented JCluster filtering. Our final system used only the latter filtering method: we describe all three and the shortcomings of the first two here.

4.1 Part of Speech Tagging

Part-of-Speech-Tagging (POS tagging) [12, 13] is the task of generating a grammar tree for sentences of a corpus. POS tagging methods often construct a Hidden Markov Model (HMM), and use maximum likelihood decoding methods such as the Viterbi algorithm in order to find the most probable tagging of sentences and resolve the ambiguities. As the first filtering approach, we used the state of the art Penn Treebank Part of Speech Tagger [14] as our filter map. Each sentence is first mapped to part of speech tags, and then the tag is converted to a binary string. Table 4 shows an example of a sentence, its part of speech tagging, and binary (bucket) representation. The buckets are then expanded using edit-distance as the notion of similarity so that the filtering provides a reasonably large (around two thousand) candidate responses.

Table 4: Example of Filtering via Part of Speech Tagging and Hoeffman Coding

| Sentence |
|---------------------------------|
| I should have eaten more apples |
| PR MD VB VB JJ NN |
| 100 10100 0 0 1011 11 |

Table 5: Example of Filtering via 1-bit and 2-bit JCluster indexings

| Sentence |
|---------------------------------|
| I should have eaten more apples |
| 0 0 1 0 0 1 |
| 00 01 10 01 01 11 |

4.2 JCluster Filtering

Jcluster [15] is a recursive algorithm for generating the hierarchical clustering of the words in a corpus. At each iteration, every selected word in the document is clustered based on the entropy of the words following it. Initially, all words to be clustered are partitioned into two sets. The words in each set are then partitioned again into two sets, and the process is repeated recursively, until a maximum depth is reached.

As an example, having observed that the words *you* and *u* are followed by the similar words (such as *are*) in many sentences, JCluster assigns the index 100110000010 to “u” and the index 100110011000 to “you”. The α -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 α most significant bits. The corresponding binary strings are then concatenated to construct the final bucket-map. Table 5 shows the bucket-mapping of a sentence using hierarchical clustering with $\alpha = 1$ (only the most significant bit).

We found that $\alpha = 1$ gave the best results, for two reasons. First, it was necessary in order to obtain a sufficiently large number of candidate sentences. Second, we observed that 2-Jcluster splits similar sentences across buckets: we found a large number of similar sentences that have the same 1-JCluster representation but whose 2-JCluster representation differs.

4.3 Content-Oriented JCluster Filtering

The POS and JCluster filtering approaches had a tendency to prefer generic, short, and high-level responses. These responses were applicable to many queries, and hence in many cases were uninformative. This was of course not very suitable for designing creative chat-bots. We observed that the notion of concepts and meaning must be incorporated in the filtering task and cannot simply be postponed to later ranking phases. As a result, we used a variation of JCluster filtering called “Content-Oriented JCluster Filtering”.

In this approach, we use an entropy analysis to remove the common and meaningless words from a sentence, leaving only the informative words. These words are then mapped to their JCluster indices which are truncated to their seven most-significant bits. The bucket maps corresponding to the sentence are then the set of extracted seven-bit binary strings.

Two buckets in this model are connected, if there exists a query and its following response such that the query contains an informative word whose truncated JCluster index is the first bucket, and the response contains an informative word whose truncated JCluster index is the second bucket. In order to make this approach efficient, all the mapping and connection tables were computed offline. Table 6 illustrates an example of a content-oriented JCluster filtering. As an example, suppose the sentence *did you go hiking at weekend ?* is mapped to two buckets 1111000 and 1101100 (*hike* and *weekend*). The response to this sentence is *yes, we had a long trip towards the peak* which is mapped to 0011010 and 0111010. Consequently the buckets 1111000 and 1101100 are both connected to both buckets 0011010 and 0111010.

Table 6: An example of content-oriented JCluster Mapping

| |
|---------------------------------|
| Sentence |
| I should have eaten more apples |
| {eat, apple} |
| {0010001, 0111001} |

5 First-Phase Ranking

5.1 Feature Selection

Having used filtering to remove a large fraction of irrelevant sentences, we now use a more precise but still fast method for finding suitable responses. To obtain these, a set of more informative features are needed by which a trained ranker can rank the candidate filtered responses for a new query.

Here we used an automatic labeling approach for labeling the Twitter examples. For each query, we label the pair “query and the response following it” as positive (+1), and we pick a response uniformly at random from the corpus and label the pair “query and the random response” as negative (−1).

Note that the resulting labels are noisy. There exist generic responses suitable for most queries: for example the response *lol* is frequent and may be picked by random sampling. Moreover, not all responses to a query in the Twitter data are necessarily appropriate responses for a chat conversation. However, as we will show, this labeling approach is nevertheless useful for extracting some key features, and can be exploited later using knowledge transfer methods. For this part of the project we used one million Twitter dialogs, from which we obtained one million positive instances and one million negative instances.

Having identified the training examples, the next step is to design a set of suitable features which can be computed efficiently. We designed 300 features from the following categories:

- **Rule-based Features:** This set of features has the most similarity with the hardwired rules provided in AIML languages used in A.L.I.C.E. For instance, one binary feature corresponds to whether the query contains a pronoun such as “I” and the response contains the pronoun “you”; another to whether the query contains a question keyword and the answer contains “yes/no”; another to whether the query length is smaller than a threshold and the answer is just “lol”. These features turned out to play key roles as weak learners. We found that they are mainly useful for quickly removing very irrelevant sentences, however their contribution on recovering suitable sentences is not very significant.
- **Punctuation-Based Features:** These features are based on the lexical properties of sentences. One example is a feature which indicates that the query is a question, and the response is a statement. Another example is a query that contains certain words and a response that ends with an exclamation mark. Interestingly, one of the useful features in this set is a statement (non-question) query, followed by a compound response.

Miss : misses, missing, main-in-space-soonest, state road and tollway
 , authority, woman, women, girl, girls, man in space soonest
 , senorita, signorina, failure, young lady, young woman, lose
 , losing, young ladies, lost, missy, fille, overlook, lassoed
 , slip, blunder, misfire, fumble, lass, maid, error, fail
 , misfires, teacher, address, mistakes, neglects, slips, drop
 , mistook, female, gal, maiden, misstep, mishap, absence
 , trips, skip, desire, bypass, pretermite, overleap, omit, leave out
 , lack, abort, avoid, yearn for, forego, long for, overrun
 , fumbled, mishit, false step, foul ball, miscue, damsel
 , delinquency, oversight, daughter, loss, miscues, neglecter
 , not catch, abstain, crave, give up, muffing, escape, mademoiselle.

Figure 3: The set of synonyms and dictionary-based related words for the word “miss”

- **JCluster Features:** Here we exploit the hierarchical structure of the JCluster indexing in more detail. For example some features rely on all 12 bits of the JCluster indexing, some features count the number of hits in the indexing of query and response, some features use a metric other than the Hamming distance, etc.
- **Parse Tree and Part of Speech Tagging Features:** We also used the grammar parse trees, part of speech tagging, and other natural language processing characteristics to generate features for the first phase ranking. We used two POS tagging algorithms: a natural language processing program “NLPWin” implemented by the Natural Language Processing group at Microsoft Research, which was faster, and an implementation of the Penn Treebank Tagging algorithm, which was slower but which provided a more detailed grammar tree. The extracted NLP information for the query and the response were then combined using general similarity matching methods (such as edit-distance, hit-counts, etc), as well as more specialized NLP similarity kernels. For instance, in one measure of similarity two trees were considered similar if their roots have the same type, and each had a subsequence of children (not necessarily consecutive) with the same types. The value of the similarity depends on how many such subsequences exist, and how spread out they are.
- **Dictionaries and Synonyms:** Dictionaries, synonyms, and the set of related words were found to be very useful. Indeed, a rich dictionary can capture the relation between the keywords in the query and response, and in all experiments, we found that the dictionary features were the dominant features in the prediction and ranking task (although we emphasize that we found that the other sources of information were required to improve accuracy). Figure 3 shows a subset of synonyms for the word “miss”.
- **Wikipedia Similarity Features:** The large corpus of wikipedia articles can be exploited to provide a resource of similarity between words and especially between named entities. The similarity between two wikipedia articles can be measured by the number of common anchor links to other wikipedia articles or to wikipedia categories [16]. These anchor similarities must be regularized in order to prevent over-fitting to very common articles². Table 7 shows the most similar results for the words “Barack Obama” and “hiking” computed in this manner.

Table 7: The most similar words for the words “Barack Obama” and “Hiking” extracted from wikipedia articles using a backward random walk

| | | | | |
|--------------|-----------|---------------|---------|------------------|
| Barack Obama | President | United States | McCain | Economy |
| Hiking | Mountain | Sports | Running | Horseback Riding |

²For instance the article “united states” is related to large number of different articles.

5.2 Multiple Additive Regression Trees

Multiple Adaptive Regression Trees (MART) [17] is a family of boosted-tree learning algorithms that are trained using gradient descent. We trained a MART binary classifier, which outputs probability estimates. We used two million samples, and we divided them into three sets, for training, cross-validation, and test data. The cross validation set was used to prune the regression trees to avoid overfitting. Given a new query, the set of candidate responses was then extracted by computing the confidence of the classifier on all responses, and outputting the responses with highest confidence.

In the first experiment, from one million Twitter dialogs, one million positive instances (with true response for each query), and one million negative examples (with random response for each query) were generated. The instances were then randomly divided into 700,000 training samples, 700,000 cross-validation samples, and 600,000 test samples. We trained 200 trees, each with 15 nodes, and we used a learning rate of 0.1. Figure 4 plots the test error of the MART classifier with respect to the number of iterations (trees). We observed that although the trained classifier gave high scores to the correct responses, it also gave high scores to many incorrect responses. This is not surprising as the labeling is noisy and the data contains a large number of generic responses. Consequently, we repeated the experiment with ten (instead of one) random responses for each query, and divided the dataset into 4 million training examples, 4 million cross-validation examples, and 3 million test examples. Figure 5 shows the test error of the trained classifier. Here we observe that the test error is only 1.5 percent better than a guessing ‘negative’ every time (which would give a 9% error rate); clearly this is a hard classification task.

There are two important issues regarding this classification approach: first, the Twitter data is very noisy. It is not even easy for human users to distinguish a true response from a random response for many Twitter queries. There are many generic responses, there are interconnections between many Twitter posts, and some Twitter comments can be irrelevant to the Twitter post (See Table 8). Second, note that the problem of distinguishing a random response from a true response is just one initial approach to solving the problem of learning to chat. These problems have interconnections, but they are very distinct problems.

We designed a web application and used human-labeled data and transfer learning methods to address these issues; we describe this next.

Table 8: Examples of generic and hard to interpret Twitter dialogs

| | |
|----------|---|
| Query | oh no! I looked at youtube so many times. I broke it ! :(|
| Response | -usr- lol |
| Query | One daughter moved in, two more to go *whew* |
| Response | -usr- tell misa I said hi :-) |
| Query | California here I come. Right back where i started from ... (in the north bay!) |
| Response | -usr- yay!!! |
| Query | ta da -url- |
| Response | awesome |

6 Online Learning and Active Data Collection

6.1 Active Data Collection and Labeling

The results above show that relying on the automatically generated labels is not sufficient to make much headway. In response we designed an interactive web game between human users and the chatbot in order to generate more accurate labels. The game works as follows:

First a Twitter conversation with more than 5 comments is selected. The last comment is taken to be the query, and the previous 5 sentences are taken to be the context (history) of the chat, in order to provide the user with some information about what the chat is about. Using the “winnow” online learning algorithm [11] the best response (for which the classifier has the highest confidence) is elected, and is shown to the user (as demonstrated in Figure 6). The

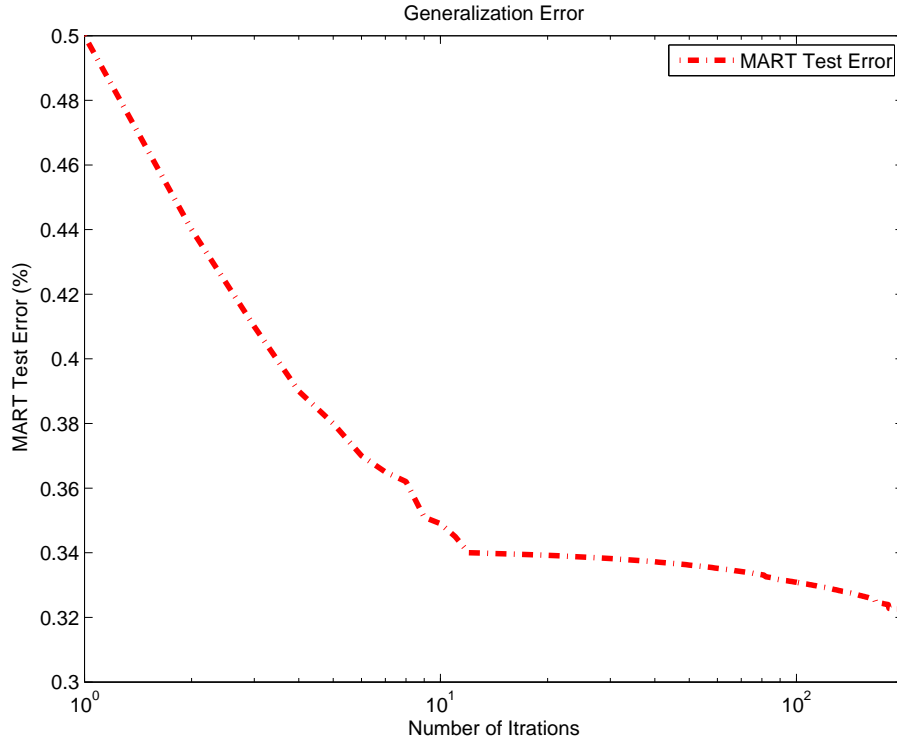


Figure 4: test error of the MART classifier, with 200 iterations, on 700,000 training examples, 700,000 cross-validation examples, and 600,000 test examples with equal number of positive and negative labeled examples.

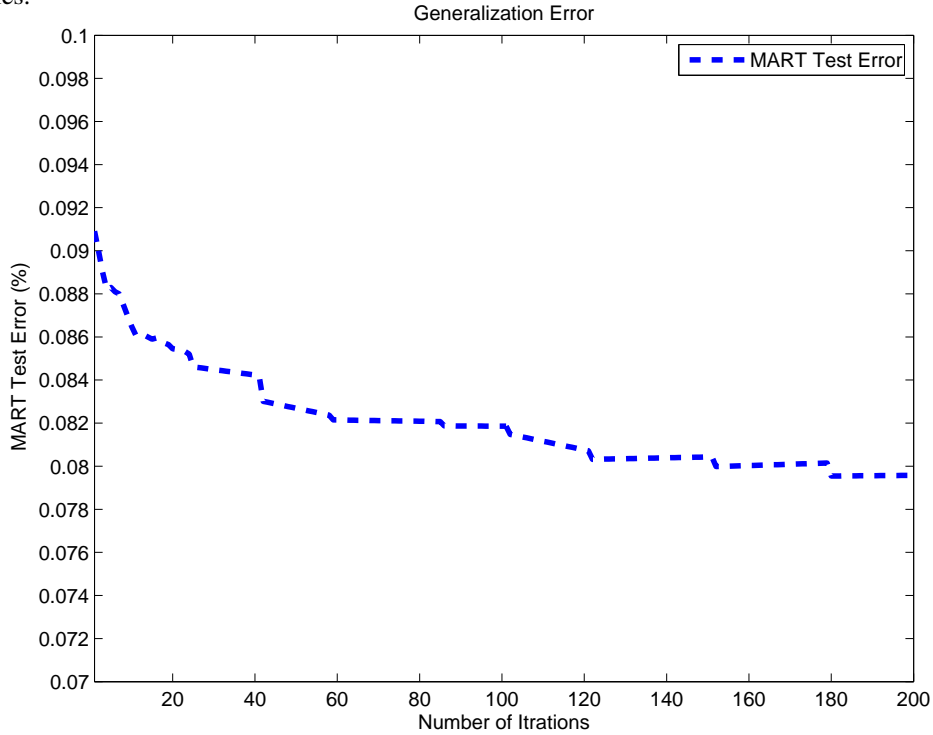


Figure 5: test error of the MART classifier, with 200 iterations, on 4,000,000 training examples, 4,000,000 cross-validation examples, and 3,000,000 test examples with nine times more negative examples.

Question 0 out of 10
 Previous Sentences :
 rt -usr- i say we celebrate #michaeljackson every year on august 29 (michael jackson day) . in honor of the day the world received mj .
 -usr- yeah i agree august 29th michael jackson day =] king of pop total legend ! ! & hope i make it into music indust :-) day haha x
 -usr- i 'll pray that everything will work out for you . -smi-
 -usr- thank & i 'll pray you get that movie role =] woah :p but when you come to uk we so have 2 have a night on the town girl :-]
] x
 -usr- thank you and i want to visit within the next year . that 's my goal . -smi-
 -usr- well you 've got a buddy here :-] im from leeds in uk :-] so how come most us people allways want to visit uk ? w/b xx .

ChatBot Query: -usr- i like to travel period . ca n't speak for everyone else -smi-

Suggested Responses:

-usr- thanks dude wish u were with us last night . safe travels man .

☐ Like ☐ Moderate ☐ Dislike

Submit base of verbier . i love climbing summit finishes after trans atlantic travel . haha . -url-

☐ Like ☐ Moderate ☐ Dislike

Submit can i tell ya ... ca n't ever say no 2 dogs (& peeps) ...

☐ Like ☐ Moderate ☐ Dislike

Submit

New Response:

because uk has lots of interesting places to visit

Next

Question 2 out of 10
 Previous Sentences :
 -usr- hey that 's me !
 -usr- what 's you ... ? lol i 'm so tired !
 -usr- oh i was just making a joke , you put up a picture of joe jonas and you said i look like him , so i said hey that 's me !
 -usr- oh ! haha , you should 've said that on the picture then ! you got me confused !
 -usr- oh sorry to confuse you !
 -usr- no it 's no problem (: so how was your day ?

ChatBot Query: -usr- it 's been 65 degrees in my office all week . freezing . that being said ... i 'd still rather have my problem than yours .

Suggested Responses:

-usr- lmao .

☐ Like ☐ Moderate ☐ Dislike

Submit -usr- no . really you do n't . the nights are 30 degrees and all dreams are of the flames of hell .

☐ Like ☐ Moderate ☐ Dislike

Submit

Next

Figure 6: Web interface for collaborative data collection and labeling

user then can select if he/she likes the sentence, dislikes it, or is neutral about it³. If the response is a *dislike* or *neutral* option, the online learning algorithm updates the classifier, and outputs another response. If the user selects the *dislike* or *neutral* option three consecutive times, the application instead asks the user to type an appropriate response. This response is then used with a large weight to update the online classifier. If the user presses the “like” button, or types a new sentence after three “dislike/neutral” attempts, the conversation continues with the response as the new query, and the previous query as the latest sentence in the conversational context (history). The oldest (5th) sentence is removed from the chat context to ensure that the chat history contains 5 sentences. The game is repeated 10 times and the labeled data provided by the user is added to the repository.

6.2 Content-Oriented Features

Table 9: Substitution and paraphrasing results for a Twitter sentence

| | |
|--------------|---|
| Sentence | How long ago did you send a request ? |
| Paraphrase 1 | How long ago did you throw a query ? |
| Paraphrase 2 | How long ago did you forward a proposal ? |

In order to get the users involved in the active data-collection game, it is crucial to output as many relevant and suitable candidate responses as possible. Designing key features capturing the semantics and conceptual relations between the query and response sentences is then an essential task. Consequently, in addition to the most important features revealed in the Twitter classification step, we included a large set of content-oriented features, which are derived from the web recommendation application itself, semantic natural language processing, and information retrieval. In addition to the features explained in Section 5, we used the following sources for generating content-oriented features:

- **Sentence Paraphrasing and Substitution:** Paraphrasing is the task of restating a sentence differently and with new words while preserving the conceptual meaning of the sentence. Paraphrasing provides a useful set of conceptual features, as it deals with the semantics of the sentences abstractly. Here we used the paraphrasing algorithms generated in the Natural Language Processing group at Microsoft Research which are based on statistical models that have been trained on paraphrase sentence pairs. We incorporated the paraphrasing method as a means for generating content-oriented features. Table 9 shows an example of a Twitter sentence, and two substitution candidates provided for it.
- **Query Refinement and Suggestion Systems:** Recently, White and Cucerzan [18] proposed an approach to generate query suggestions based on the pages where many users who submit a particular query end up through post-query browsing. This collaborative filtering approach captures the “wisdom of the crowds” and provides rich semantic similarity tables. For instance, the query refinement method suggests the words “crab, seafood, lobster, steak” as the most relevant words for the query “shrimp”. We used the suggestion systems thesaurus as an additional source for generating content-oriented features.
- **Topic Modeling:** We can also use LDA topic modeling as an additional method for clustering words and capturing similarities among them. A “topic” is a cluster of words that are conceptually related to each other and that frequently occur together. Topic modeling methods [19, 20] try to learn these topics through a Bayesian approach, based on the idea that each document is a mixture of topics. Topic modeling is a generative approach in which first a set of topics are generated according to some “latent” process, and then the words in the documents are generated from the topics according to word generating processes [21].

We generated a topic model using our Twitter corpus. Ritter et. al [22] further extended the learned topics to provide an unsupervised model of the Twitter conversations. The topics and corresponding clusters of words are then used as a content-oriented feature set for the learning task. Table 10 demonstrates examples of two topics and the related clustered words.

³Mostly this case occurs when the response is a generic response.

- **Wikipedia Article Names:** Multi-word wikipedia article names provide another source of information, especially for named entities. The relation between two words in these sets of features relies on the wikipedia article names that contain the two words simultaneously. For instance, the most relevant extracted words for the word “software” are “Windows, Game, and India” and the relation between the words software and India was not highlighted by the previously mentioned features.

Table 10: Examples of the words clustered in the topics named “book” and “film” using an LDA topic model of our corpus of Twitter data

| | |
|------|---|
| Film | Trailer, Horror, Scream, Love |
| Book | Harry Potter, Library, Reading, Published |

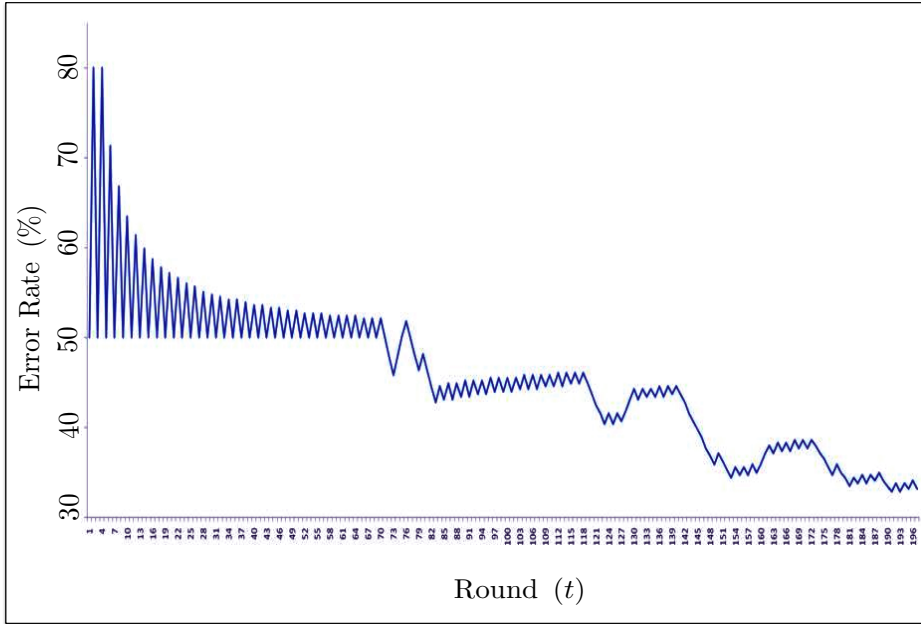


Figure 7: Error rate of the winnow algorithm.

Figure 7 plots the error rate of the online learning algorithm with 200 rounds encompassing 100 positive instances and 100 negative examples fed to the algorithm one by one manually (one positive is followed by one negative). The horizontal axis shows the round t , and the vertical error illustrates the error rate defined as

$$\text{error rate}(t) \doteq \frac{\# \text{mistakes up to round } t}{t}.$$

As we can see from the figure, at the beginning, the classifier is not predicting better than random guessing, and the error rate is $\frac{t/2}{t}$ if t is even and $\frac{t/2+1}{t}$ if t is odd⁴. However, as time passes, the classifier starts learning and the (smoothed) error rate decreases.

⁴This is the reason for the initial oscillation.

7 Transfer Learning

In the previous section we saw how we can collect labeled dialog data through a collaborative game with human users. By construction, this data should be both more relevant to the task, and equipped with cleaner labels, than the automatically labeled Twitter data used previously. Our next step is to exploit this collected data together with other sources of dialogs in order to increase the accuracy of the ranking process. The number of available instances used in our study was limited to 800 pairs.

The Twitter and LiveJournal data on the other hand are not labeled by users, and are very noisy. Nevertheless, this does not mean that we have to throw away all the Twitter and LiveJournal data. There are plenty of informative and useful dialogs among these datasets as well. Hence, our objective is to get the best of both resources and to exploit the active and passive datasets simultaneously, in order to improve the accuracy of learning. We used a transfer learning approach to attempt to leverage the Twitter and LiveJournal instances while ignoring the useless or irrelevant sentences.

Algorithm 1 Transfer AdaBoost Algorithm

Inputs

The source domain dataset S of size m , and destination domain dataset D of size n .

A base learning algorithm **base-learner**, and number of iterations T .

- 1: Initialize the weight vector $w_1 = (w_1(1), \dots, w_1(n+m))$ arbitrarily or based on some prior knowledge.
- 2: **for** $t = 1, \dots, T$ **do**
- 3: Set $P_t \leftarrow \frac{w_t}{\sum_{i=1}^{n+m} w_t(i)}$.
- 4: Call **base-learner**, providing it the combined training set $C = S \cup D$ with the distribution P_t over C . Then, get back a hypothesis h_t .
- 5: Calculate the error of h_t on D :

$$\epsilon_t \doteq \sum_{i=1}^n \frac{w_t(i) |h_t(x_i) - y_i|}{\sum_{i=1}^n w_t(i)}$$

- 6: Set $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$, and $\beta = \frac{1}{1+\sqrt{2 \ln m/T}}$.
- 7: Update the new weight vector

$$w_{t+1}(i) = \begin{cases} w_t(i) \beta_t^{-|h_t(x_i) - y_i|} & 1 \leq i \leq n \\ w_t(i) \beta^{|h_t(x_i) - y_i|} & n+1 \leq i \leq n+m \end{cases}$$

- 8: **end for**
- 9: Output the final hypothesis

$$H(x) = \begin{cases} 1 & \text{if } \prod_{t=T/2}^T \beta_t^{-h_t(x)} \geq \prod_{t=T/2}^T \beta_t^{-\frac{1}{2}} \\ 0 & \text{otherwise} \end{cases}$$

Transfer learning [9] exploits 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 the *destination* domain. In other words, the goal of transfer learning is to transfer the knowledge from the source domain to the destination domain. Here our source domain is the set of Twitter and LiveJournal data, with noisy labeling in which for each query, the response following that is labeled positive and a random response is labeled negative. Our destination domain is the set of actively collected dialogs, which are labeled by human users in an interactive environment. Here the source domain has a much larger number of available instances, but the labeling is noisy, while for the destination domain a smaller number of instances is provided but the labeling is more robust.

We used a modification of the AdaBoost algorithm known as Transfer AdaBoost (TrAdaBoost), which was originally proposed by Dai *et. al* [23]. The algorithm is shown in Algorithm 1. At each iteration the algorithm gives more weight to source instances that were predicted correctly, and in this way exploits the source knowledge. It also gives more weight to destination instances predicted incorrectly, and hence follows the boosting paradigm of challenging the learner with the hardest destination instances [24]. Note that since the goal of transfer learning is learning

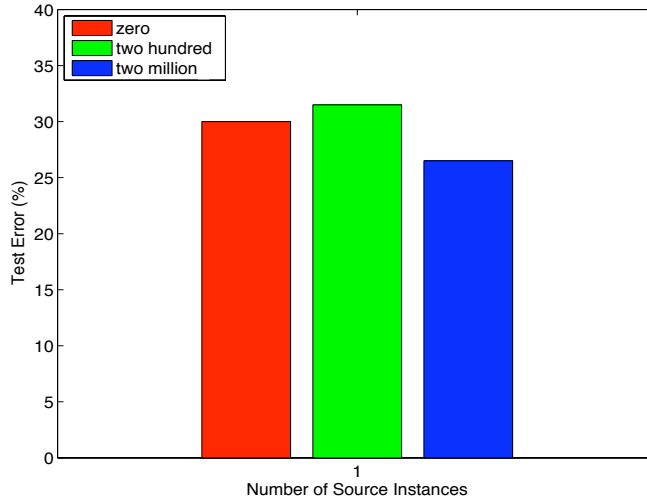


Figure 8: Dependence of the generalization error of the Transfer AdaBoost algorithm on the number of source instances.

the destination domain with the highest accuracy, the parameters are updated just based on the error on destination instances.

The MART algorithm with only 20 iterations is used as the base-learner, and the Transfer AdaBoost used 200 iterations. The source data consist of Twitter and LiveJournal dialogs, and the destination domain is the set of actively labeled dialogs. Figure 8 plots the test error of the Transfer AdaBoost algorithm versus the number of source instances. As we see from the figure, with 200 random source instances, the error actually increases. However, if all 2 million source instances are used, an almost 4% drop in the generalization error of the classifier is achieved.

Figure 8, as a first step, shows that the idea of transfer learning, provided that the source domain is rich enough, can facilitate the task of training the chat-bot. Here we emphasize that this experiment should also be considered as a first step for using transfer learning in chat learning. By incorporating more specialized transfer learning methods, it may be possible to strengthen the final ranker further.

Having trained the Transfer AdaBoost classifier, the content-oriented ranking phase is done. The final response is then the sentence for which TrAdaBoost has the highest confidence. Tables 11 to 13 list a few responses the chat-bot provides to three queries issued by users.

8 Conclusion

In this paper, we proposed a discriminative model for learning to chat. The model introduced consists of three phases: filtering, which is done by entropy analysis methods, first phase ranking, which uses carefully designed features and the MART boosted tree classifier, and content-oriented ranking, which uses more content-oriented features, online learning, active data-collection, and knowledge transfer methods.

Our approach was driven by the desire to see, as a first step, how far we could get using publicly available, large (but noisy) datasets, using off-the-shelf machine learning methods, and using a game-like web interface to gather cleanly labeled data. This holistic approach provided some interesting insights, such as the value of transfer learning methods when small amounts of cleanly labeled data are available. However we emphasize that this work was really more of an engineering exercise than a scientific investigation. In particular, in this work, responses are restricted to being sentences that are already in the database, whereas a compelling chat bot will likely have to generate content based on models of the user, the world, and the context of the conversation. Nevertheless, our basic premise is that machine learning, and the availability of large datasets, have progressed sufficiently that the problem of learning to

Table 11: Suggested Responses for Query “I like oranges”:

R: unless you eat like 10 acre of oranges at one time ... no
 R: and you know how i love swerving for oranges
 R: florida make oranges lol
 R: made jam with ripe apricots , canned peaches , brown sugar , a tab of water and a tad bit of honey . it is awesome try it!
 R: bananas , passion fruit , tangerines , and mangos are in season
 R: seaswine flu ?
 R: can't wait to redo our kitchen . . fista colors ! !it 'll be reds , oranges , yellows , blues , and greens ... yay ! !!
 R: north shore is my favorite place on the island , way to spend a day ! -smi-
 R: been filming , but right now i 'm checking stubhub , ticketmaster , orbitz , mapquest & travelocity . guess why ?

Table 12: Suggested Responses for Query “I play football a lot”:

R: how long have you been playing ? i've been on 2 years
 R: don't forget the final test match
 R: from vegas : hawes may not play in summer league . westphal says he 'll come mon . to “support ... teammates”
 R: i want to play over the line
 R: i got to see that pic
 R: i know , but you said you wanted a tattoo with it . if it makes you feel any better i'll now be in the rain waiting for a bus
 R: you are so far behind on that one
 R: you work late
 R: omg -creams- do you know what i would do to see them live gaah ! i really do n't know why i creamed but i did
 R: area -usr- you rock

Table 13: Suggested Responses for Query “How is the weather in Seattle?”:

R: perfect weather for a day off
 R: wish i was there ! i need some cooler weather ! !95 in south tx ! you ready ? =)
 R: snow in august and tornadoes in february would n't be surprising here
 R: cause i did n't bring my umbrella ! :(
 R: rain falling , thunder storming , lightning striking
 R: the rain is so nice ...but why do people complain already ? we 've had record dry spells
 R: cold , stressed , bored and can't find a bathroom . are these the best conditions to sit an exam ? no i think not
 R: yes hurt because of friggin the sims 3 ! my guy and gurl sims have finally moved in together ! no cheats , i swear . :p
 R: feeling tired and under the weather today . been napping but disturbed dreams

chat now presents a fascinating challenge that may be conquerable. We hope that this work will spark the interest of others in taking another look at the problem of learning to chat.

Acknowledgements

We thank Alan Ritter for providing the Twitter data, and Misha Bilenko and U Kang for providing similarity data based on the Wikipedia graph. We thank Chris Brockett for providing the synonym generator and dictionaries, and for providing a prototype sentence paraphrasing system. We thank Silviu Cucerzan for providing the similarity data based on user session behavior, and for providing a trie-based edit-distance calculator to compute similarities. We thank Matt Richardson for his helpful comments on designing the web interface. We also thank Misha Bilenko, U Kang, Alan Ritter, Chris Brockett, Bill Dolan, Silviu Cucerzan and Colin Cherry for helpful discussions.

References

- [1] S. Russell and P. Norvig, “Artificial Intelligence: A Modern Approach,” *Prentice Hall*, 2003.
- [2] A. Turing, “Computing Machinery and Intelligence,” *Mind*, vol. 236, pp. 433-460, October 1950.
- [3] 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.
- [4] M. Bowden, “Mind As Machine: A History of Cognitive Science,” *Oxford University Press*, 2006.
- [5] S. Young, M. Gasić, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, and K. Yu, “The Hidden Information Statemodel: A practical framework for POMDP-based Spoken Dialogue Management,” *Computer Speech & Language*, vol. 24(2), pp. 150-174, 2009.
- [6] R. Wallace, “From Eliza to ALICE,” Article available at <http://www.alicebot.org/articles/wallace/eliza.html>, 2004.
- [7] C. Burges, T. Shaked, E. Renshaw, M. Deeds, N. Hamilton, and G. Hullender, “Learning to rank using gradient descent,” *International conference on Machine learning*, vol. 119, pp. 89-96, 2005.
- [8] Q. Wu, C. Burges, K. Svore, and J. Gao, “Ranking, Boosting, and Model Adaptation,” *Technical Report, MSR-TR-2008-109*, October 2008.
- [9] S. Pan and Q. Yang, “A Survey on Transfer Learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 99(1), 2009.
- [10] 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*, 1995.
- [11] N. Littlestone, “Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm,” *Machine Learning*, vol. 2(4), pp. 285-318, 1998.
- [12] K. Church, “A stochastic parts program and noun phrase parser for unrestricted texts,” *Second conference on applied natural language processing*, pp. 136-143, 1988.
- [13] S. DeRose, “Grammatical category disambiguation by statistical optimization,” *Computational Linguistics*, vol. 14(1), pp. 31-39, 1988.
- [14] 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,” *University of Pennsylvania*, 1995.
- [15] J. Goodman, “JCLUSTER, a fast simple clustering program that produces hierarchical, binary branching, tree structured clusters,” available at <http://research.microsoft.com/en-us/um/people/joshuago>, 2009.

- [16] N. Craswell and M. Szummer, "Random walks on the click graph," *International ACM SIGIR conference on research and development in information retrieval*, pp. 239-246, 2007.
- [17] J. Friedman, "Greedy Function Approximation: A Gradient Boosting Machine," *Annals of Statistics*, vol. 29, pp. 1189-1232, 1999.
- [18] S. Cucerzan and R. White, "Query suggestion based on user landing pages," *International ACM SIGIR conference on Research and development in information retrieval*, pp. 875-876, 2007.
- [19] M. Steyvers and T. Griffiths, "Probabilistic Topic Models," *Handbook of Latent Semantic Analysis*, 2007.
- [20] T. Hofmann, "Unsupervised Learning by Probabilistic Latent Semantic Analysis," *Machine Learning Journal*, vol. 42, pp. 177-196, 2001.
- [21] D. Blei, A. Ng, and M. Jordan, "Latent Dirichlet Allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993-1022, 2003.
- [22] A. Ritter, C. Cherry, and B. Dolan, "Unsupervised Modeling of Twitter Conversations," *To appear in proceedings of ACL*, 2010.
- [23] W. Dai, Q. Yang, G. Xue, and Y. Yu, "Boosting for Transfer Learning," *24th Annual International Conference on Machine Learning*, pp. 193 - 200, June 2007.
- [24] R. Schapire, "The strength of weak learnability," *Machine Learning*, vol. 5(2), pp. 197 - 227, 1990.