CHARM: Inferring Personal Attributes from Conversations

Anna Tigunova, Andrew Yates, Paramita Mirza, Gerhard Weikum
Max Planck Institute for Informatics
Saarbrücken, Germany
{tigunova, ayates, paramita, weikum}@mpi-inf.mpg.de

Abstract

Personal knowledge about users' professions, hobbies, favorite food, and travel preferences, among others, is a valuable asset for individualized AI, such as recommenders or chatbots. Conversations in social media, such as Reddit, are a rich source of data for inferring personal facts. Prior work developed supervised methods to extract this knowledge, but these approaches can not generalize beyond attribute values with ample labeled training samples. This paper overcomes this limitation by devising CHARM: a zero-shot learning method that creatively leverages keyword extraction and document retrieval in order to predict attribute values that were never seen during training. Experiments with large datasets from Reddit show the viability of CHARM for open-ended attributes, such as professions and hobbies.

1 Introduction

Motivation. Personal Knowledge Bases (PKBs) capture individual user traits for customizing downstream applications like chatbots or recommender systems (Balog et al., 2019). A potentially automatic way to populate a PKB is to draw personal knowledge from the user's conversations in social media and dialogues on other platforms. These interactions are a rich source of personal attributes, such as hobbies, professions, cities visited, medical conditions (experienced by the user) and many more. Each of these would consist of key-value pairs, such as *cities visited:Paris* or *symptom:dizziness*. However, a large number of potential attributes and their respective values makes this a challenging task. In particular, there is little hope to have training data for each of these key-value pairs. Moreover, the textual cues in user conversations are often implicit and thus difficult to learn.

Example. Consider the user's utterance: *"I just visited London, which was a disaster. My hotel was a headache and I spent half the time in bed with a fever... So glad to be back home finishing the masts on my galleon."* As humans, we can infer the following attribute-value pairs: (a) *cities visited:London*, (b) *symptom:fever*, (c) *hobby:model ships*. Capturing such user traits is a daunting task, however, with both implicit and explicit signals present. We need to consider the context *"spent in bed with"*, to infer that *fever* relates to a disease (as opposed to *headache*). To predict the user's hobby *model ships*, we have to pay attention to the cues 'galleon' and 'mast'. Proper inference requires both deep language understanding and background knowledge (e.g., about ships, cities, etc.).

State of the Art and its Limitations. Explicit mentions of attribute-value pairs can be captured by pattern-based methods (e.g., Li et al. (2014); Yen et al. (2019)). Such methods are able to extract *London* from the the previous example by using the pattern *"I ... visited* ⟨*city_name*⟩*"*. Pattern-based approaches are limited, though, by their inability to consider implicit contexts, such as *"finishing the* ***masts*** *on my* ***galleon****"*. Question answering methods can be used to relax rigid patterns (e.g., Levy et al. (2017)), but still rely on explicit mentions of attribute values.

In this work we aim to extract attribute values leveraging both explicit and implicit cues, such as inferring *symptom:fever* and *hobby:model ships*. Additionally, we address the cases where there is a long-tailed set of values for such attributes as *hobby*. In principle, deep learning is suitable for such inference (Tigunova et al., 2019; Preoţiuc-Pietro et al., 2015; Rao et al., 2010), but it critically hinges on the availability of labeled training samples for every attribute value that the model should predict. Supervised training is suitable for a pre-specified limited-scope setting, such as learning personal interest from a fixed list of ten movie genres, but

it does not work for the situation with large and open-ended sets of possible values, for which there is little hope of obtaining comprehensive training samples. Therefore, we pursue a *zero-shot learning* (Larochelle et al., 2008; Palatucci et al., 2009) approach that learns from labeled samples for a small subset of labels (i.e., attribute values in our setting) and generalizes to the full set of labels including values *unseen* at training time.

Problem Statement. For a given attribute we consider the set of *known* values V, which can be drawn from lists in dictionary-like sources like Wikipedia. At training time, our method requires samples for a small subset of values $S \subset V$. Typically, the complement $V \setminus S$ is much larger than S: $|V \setminus S| \gg |S|$. For instance, S may consist solely of the popular values *sports, travel, reading, music, games*, whereas the complement includes hundreds of long-tail values, such as *beach volleyball*, *model ships*, *brewing*, etc. At inference time we need to predict values from all of V, although most of the values are *unseen* during training.

Approach and Contributions. We present CHARM, a Conversational Hidden Attribute Retrieval Model, for inferring attribute values in a zero-shot setting. CHARM identifies cues in related to a target attribute, which it then uses to retrieve relevant texts from external document collections, indicative of different attribute values. These external documents could be gathered by simple web search. They help CHARM to link the cues in the user's utterances to the actual attribute values to predict. CHARM consists of two components: *(i)* a cue detector, which identifies attribute-relevant keywords in a user's utterances (e.g., *galleon*), and *(ii)* a value ranker, which matches these keywords against documents that indicate possible values of the attribute (e.g., *model ships*).

To evaluate our approach, we conduct experiments predicting Reddit users' professions and hobbies based on their conversational utterances. We demonstrate that CHARM performs well when inferring unseen values and performs competitively with the best-performing baselines when predicting values seen during training. CHARM can easily be extended to other attributes with long-tail values, such as *favorite cuisine*, *preferred news topics* or *medication taken*, by providing a list of known attribute values, training examples for a subset of these values and access to external documents (e.g., via a Web search engine).

The salient contributions of this paper are: *(1)* a method for inferring both seen and previously unseen (zero-shot) attribute values from a user's conversational utterances; *(2)* a comprehensive evaluation for the *profession* and *hobby* attributes over a large dataset of Reddit discussions; and *(3)* labeled data and code as resources for later research.[1] [2]

2 Related Work

User profiling from utterances. There is ample prior work on classification models to predict a user's personal traits based on hand-crafted textual features (Preoţiuc-Pietro et al., 2015; Basile et al., 2017), or with embedding-based representations (Li et al., 2016; Bayot and Gonçalves, 2018; Tigunova et al., 2019). While classification models work well for inferring demographic attributes with a small set of values such as *age*, *gender* or *occupational class* (Preoţiuc-Pietro et al., 2015; Flekova et al., 2016; Basile et al., 2017) their dependence on seeing all attribute values in (sufficiently many) labeled training samples renders supervised classifiers inappropriate for open-ended attributes such as *profession* (Tigunova et al., 2019), *hobby* (Bando et al., 2019) or *favorite food* (Zeng et al., 2019), which are often modeled as a binary multilabel task predicting the presence of each attribute value (Welch et al., 2019). Similar to our approach, some studies map user input to Wikipedia concepts (Abel et al., 2011; Krishnamurthy et al., 2014) to predict interests or locations. However, this method requires explicit mentions of the entities.

Pattern-based approaches alleviate the problem of the lack of labeled entities for long-tail classes by employing information extraction techniques to obtain personal attribute values from users' utterances, using sequence labeling methods (Jing et al., 2007; Li et al., 2014) or context classification (Yen et al., 2019). However, their coverage is limited because they require crisp and explicit statements, like *"I am a **student**"*, which are infrequent in conversations.

Our approach is designed for handling attribute values that were never seen at training time. This is known as the *zero-shot learning* problem, which has been widely studied in the field of computer vision but less explored in NLP. We employ a technique similar to Ba et al. (2015) for visual classes,

[1] https://github.com/Anna146/CHARM

[2] https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/pkb

which builds image classifiers directly from encyclopedia articles without training images.

Most zero-shot studies for NLP (Wang et al., 2019) deal with machine translation, cross-lingual retrieval and entity/relation extraction (Levy et al., 2017; Pasupat and Liang, 2014), which are not suitable for our task, because they identify values that are explicitly mentioned rather than inferring them. Our task is similar to zero-shot text classification (Yazdani and Henderson, 2015; Zhang et al., 2019), where the class labels are represented as single-word embeddings. We consider a zero-shot BERT baseline (Devlin et al., 2018) that matches utterances with rich document representations.

Keyword extraction from conversational text. Notable applications of keyword extraction from conversational text include *just-in-time information retrieval* (Habibi and Popescu-Belis, 2015), with continuous monitoring of users activities (e.g., participation in meetings) and generating personalized tags for Twitter users (Wu et al., 2010) or search for relevant email attachments (Van Gysel et al., 2017). Prior work mostly pursued unsupervised approaches, e.g. *TextRank* (Mihalcea and Tarau, 2004) and *RAKE* (Rose et al., 2010), due to limited availability of training data. Exceptions use supervised learning, with feature-based classifiers (Kim and Baldwin, 2012) or neural sequence tagging models (Zhang et al., 2016).

Our neural approach lies in between, as we learn to identify salient keywords for a specific attribute (e.g., *profession*), without having training data of relevant keywords.

Information Retrieval in NLP. Most existing work leveraging Information Retrieval (IR) components to solve NLP tasks focused on Question Answering (QA) (Kratzwald and Feuerriegel, 2018; Wang et al., 2018; Guu et al., 2020) or dialogue systems (Feng et al., 2019; Luo et al., 2019), where the retrieval part is responsible for ranking the most appropriate answers or responses, given a question or chat session. As far as we know, we are the first to leverage a retrieval-based model for inferring attribute values without training samples.

3 Methodology

Overview. As illustrated in Figure 1, CHARM consists of two stages: *cue detection* and *value ranking*. As input CHARM receives a user's utterances $U = u_0..u_N$ that contain a set of terms $t_0..t_M$, for example, $U = \{$*"I stayed late at the* ***library*** *yesterday", "****Studied*** *for the* ***exam*** *so I could have better* ***grades*** *than my* ***classmates****"*$\}$. In the first stage, the term scoring model assigns a score to each term in the user's utterances, yielding $l_0..l_M$. The highest scoring terms are then selected to form a query $Q = q_0..q_K$ characterizing the user's correct attribute value, e.g., $Q =$"library studied exam grades classmates" for the *profession* attribute.

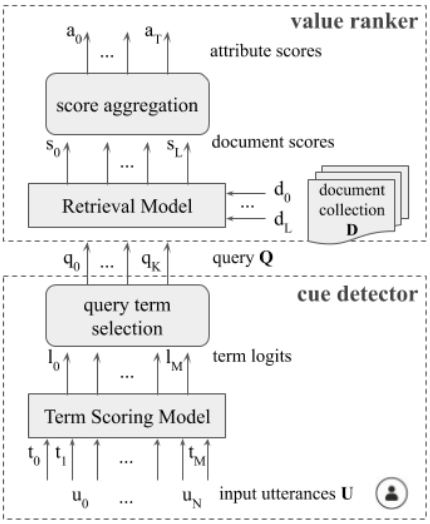


Figure 1: The pipeline of CHARM. The Term Scoring Model assigns scores $l_0..l_M$ to the terms in the input utterances $u_0..u_N$. The terms with the highest scores are passed to the Retrieval Model, which queries the document collection D. The document scores are aggregated to produce attribute value scores for predictions.

In the second stage, Q is evaluated against an external document collection $D = d_0..d_L$; each document in D is associated with possible attribute values. Documents such as *Wiki:Student* and *Wiki:Dean's List*, which are associated with the attribute value *student*, would score high with the example query. The score aggregator then ranks the attribute values based on the documents' scores $s_0..s_L$, for instance, yielding a high attribute score for *student* given our example utterances. The list of attribute values V is **known** in advance (e.g., taken from Wikipedia lists); however, potentially only a subset of values $S \subset V$ have instances **seen** during training.

3.1 Cue detection

The term scoring model δ evaluates how useful each word in a given user's utterances is for making a prediction, and assigns real-value scores $l_0...l_M$ to the terms accordingly. That is, $l_j = \delta(t_j|t_0,..t_M;W)$, where W denotes the parameters of the model. The term scores $l_0..l_M$ are then

used to select the words which will form the query for the value ranking component.

The term scoring model should produce high scores for terms that are descriptive of the user and of the attribute in general, instead of a specific attribute value. This means that it should be able to exploit background knowledge and a term's context to judge its relevance to the attribute. For instance, having seen the phrase *"stayed late at the hospital"* for the *physician* at training time, at prediction time an ideal model would correctly estimate the importance of the word 'library' in the phrase *"stayed late at the library"*, even if there were no instances of *student* in the training set.

BERT (Devlin et al., 2018) is well-suited for this requirement, because it is a sequential model that effectively uses word context and incorporates world knowledge.

For further description, let us suppose the cue detector picks the words $Q = q_0...q_K$ as our query terms for CHARM's value ranking stage. A typical query would consist of the terms associated with the correct attribute value (e.g., $Q =$"library studied exam grades classmates").

3.2 Value ranking

The second stage of the model consists of two steps: first, using the selected query terms to rank the documents in the external collection; and second, aggregating document scores to predict values.

Document ranking. The ranking component takes two inputs: query terms $Q = q_0...q_K$ resulting from the cue detector and an (automatically labeled) document collection $D = d_0...d_L$. The document collection could be a set of Web pages, where each page indicates a specific attribute value, $v_0...v_L$. For example, by generating a search-engine query "hobby ⟨value⟩" we can gather web pages related to specific hobbies.

The ranker $\rho(Q, d_k)$ evaluates the query Q, constructed by the cue detector, against each document d_k in the document collection to produce document relevance scores $s_0...s_L$. For the example query "library studied exam grades classmates", the document *Wiki:Dean's List* labeled with *student* will get a higher score than *Wiki:Junior doctor* (for *physician*). We consider two particular instantiations of the ranker: BM25 (Robertson et al., 1995) and KNRM (Xiong et al., 2017). BM25 is a strong unsupervised retrieval model, whereas KNRM is an efficient neural retrieval model that can consider semantic similarity via term embeddings in addition to considering exact matches of query terms.

Document score aggregation. The document scores $s_0...s_L$ obtained from the ranker are then aggregated to produce scores for each known attribute value. Depending on the document collection used, each attribute value may be represented by several documents. For example, the *student* attribute value may be associated with documents *Wiki:Dean's List*, *Wiki:Master's degree*, etc. In this case, the scores per document have to be aggregated to form the final scores $a_0...a_T$ for each attribute value in V. In our experiments, we consider the following aggregation techniques: *(i) average* (which allows multiple documents to contribute to the final ranking) and *(ii) max* (which may help when the document collection is noisy and we care only about the top-scoring document for each value). Having obtained the final attribute scores $a_0...a_T$, we sort them to get the top value as the model prediction.

3.3 Training

While predicting attribute values is not inherently a reinforcement learning problem, we utilize the REINFORCE policy gradient method (Sutton et al., 2000) to train the cue detector component because there are no labels indicating which input terms should be selected. This allows the cue detector to be trained based on the correct attribute values regardless of the non-differentiable `argmax` operation needed to identify the K top scoring terms from the scores it outputs.

When using the policy gradient method, the *state* in our system is represented by a sequence of input terms $t_0...t_M$. Each of the M input terms also represents an independent *action*. The term scoring model acts as the *policy*, which outputs the term selection probabilities based on the current state. Then a term is sampled (at training time) or the term with maximum probability is selected (at prediction time) and added to the query.

During training, we form the query sampling without replacement one word at a time. After sampling each term, we issue the current query and get intermediate feedback. The training episode ends when the query reaches its maximum length K. We define the reward r_τ for an intermediate query to be the normalized discounted cumulative gain (the nDCG ranking metric) of the correct attribute values' scores after aggregation at timestep

τ. The objective of REINFORCE is to maximize $J = \sum_{\tau=1}^{K} r_\tau * \log p_\tau$ by updating the weights of the policy network (where p_τ is the probability of selecting a term at timestep τ).

4 Dataset

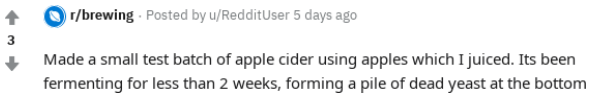


Figure 2: Example of an input utterance from Reddit.

The datasets used in our experiments cover two types of input: *(i)* users' utterances along with their corresponding attribute-value pairs (e.g., *hobby:brewing* from the example in Figure 2), and *(ii)* a collection of documents associated with each attribute value (e.g., documents describing *brewing* as a hobby). We consider two exemplary attributes: *profession* and *hobby*. We define lists of their attribute values based on Wikipedia lists[3].

4.1 Users' utterances

We consider publicly-available Reddit submissions and comments[4] from 2006 to 2018 as users' utterances. Given a Reddit user having a set of utterances $U = u_0..u_N$, we aim to label the user with a set of *profession* and *hobby* values, based on explicit personal assertions (e.g., *"I work as a **doctor**"*) found in the user's posts. To label the candidate users with attribute values we utilized the Snorkel framework (Ratner et al., 2017). We provide details on our data labeling using Snorkel in Appendix A.1.

For our experiments, we removed all posts containing explicit personal assertions that we used for labeling each user, because we want to test the ability of CHARM to predict attribute values based on inference, as opposed to explicit pattern extraction. The final dataset consists of 6000 users per attribute, with a maximum of 500 and an average of 23 users per attribute value. The number of attribute values for *hobby* and *profession* attributes is 149 and 71 respectively.

We evaluated the quality of Snorkel labeling on a held-out validation set, which we manually annotated. The validation set contains roughly 100 users per attribute, and was annotated with attribute values agreed by at least two out of three judges. The labeling obtained by Snorkel corresponded to 0.9 precision on the validation set. To demonstrate that Snorkel provides the same level of quality as crowdsourcing, we calculated the precision of human annotators on the same validation set by comparing the labels of each annotator against the agreement labels. The obtained precision scores were 0.91 for profession and 0.88 for hobby, demonstrating that Snorkel is a reasonable alternative.

4.2 Document collection

The scope of possible attribute values may be open-ended in nature, and thus, calls for an automatic method for collecting Web documents. In this work, we consider three different Web document collections; summary statistics on the number of documents per attribute value are provided in Table 1. Each document may be associated with multiple attribute values. To provide more diversity and comprehensiveness we augmented our pre-defined lists of known attribute values with their synonyms and hyponyms.[5]

Note that the approaches used to construct the document collections are straightforward and easily applicable for further attributes, such as favorite travel destination or favorite book genre.

Wikipedia pages (Wiki-page). To create this collection we take the lists of known attribute values and automatically retrieve a Wikipedia page corresponding to each value, which usually coincides with the article title (e.g., *Wiki:Barista*).

Wikipedia pages–extended (Wiki-category). This collection is an extension of Wiki-page that additionally includes pages found using Wikipedia categories. This allows us to include pages about concepts related to the attribute values, such as tools used for a profession and the profession's specializations. To construct Wiki-category, we identified at least one relevant category for each attribute value and included all leaf pages under the category (i.e., including no subcategories).

Web search. To create this collection we queried a Web search engine using attribute-specific patterns: *"my profession as ⟨profession value⟩"* and *"my favorite hobby is ⟨hobby value⟩"*. The collection consists of the top 100 documents returned for each value. Such patterns can be created with low effort by evaluating a few sample queries. Alternatively, patterns could be mined from a corpus or simplified to the generic form *"⟨attribute⟩ ⟨value⟩"*.

[3]Wikipedia pages: List_of_hobbies & Lists_of_occupations

[4]https://files.pushshift.io/reddit/

[5]Available at https://github.com/Anna146/CHARM

		min	max	avg	total
profession	Wiki-page	1	10	2	156
	Wiki-category	1	191	57	4,156
	Web search	71	100	92	6,688
hobby	Wiki-page	1	1	1	149
	Wiki-category	2	479	74	10,782
	Web search	54	100	82	12,312

Table 1: Document collection statistics.

5 Experimental Setup

We evaluate the proposed method's performance in two experimental settings. First, we consider a zero-shot setting in which the attribute values in the training and test data are completely disjoint (i.e., the test set only contains *unseen* labels). This setting evaluates how well CHARM can predict attribute values that were not observed during training. Second, we consider the standard classification scenario in which all attribute values are *seen* as labels in both training and test sets.This demonstrates that CHARM's performance in a normal classification setting does not substantially degrade because of its proposed architecture.

Experimental setup details differ for these two evaluation settings, which will be discussed in the following subsections. All our models were implemented in PyTorch; technical details are in Appendix B. The code and labeled datasets will be made publicly available upon acceptance.

Training and test data. For the *unseen* experiments, we perform ten fold cross-validation with folds constructed such that each attribute value appears in only one test fold. Each of the folds contains roughly the same number of users and approximately 2-4 unique attribute values.[6] We assigned the users having multiple attribute values to a fold corresponding to one of their randomly chosen values. For the experiments with *seen* values, we randomly split the users into training and test sets in a 9:1 proportion, respectively.

Hyperparameters. BERT, the term selection component, generates a contextualized embedding for each input term, which we process with a fully connected layer to produce a term score for each word in its context. Specifically, we use the pre-trained BERT base-uncased model with 12 transformer layers. To reduce BERT's computational requirements, we discard the last 6 transformer layers (i.e., we use embeddings produced by the earliest 6 layers) after observing in pilot experiments that this outperformed a distilled BERT model. (Sanh et al., 2019)

Following prior work (Hui et al., 2018), KNRM was trained with frozen word2vec embeddings on data from the 2011-2014 TREC Web Track with the 2009-2010 years for validation. We initialize KNRM with these pre-trained weights.

During training, we sample 5 negative labels (i.e., incorrect attribute values) to be ranked when calculating the nDCG reward. For each label, we sample a subset of 15 documents to represent the label (i.e., attribute value). If the document collection has fewer than 15 documents for a label (e.g., Wiki-page), we consider all the label's available documents. When making predictions, we consider all documents and all labels (values). In both settings, we truncate documents to 800 terms when using KNRM for efficiency and use the full documents with BM25. We use ten fold cross-validation on the training data to optimize the following hyperparameters in a grid search: (*i*) document aggregation strategy (*average* vs *max*); (*ii*) length of query; and (*iii*) maximum number of epochs. Further details on the hyperparameter search are in Appendix B.

Baselines. For the *unseen* experiments, we evaluate CHARM's performance against an end-to-end BERT ranking method and against a BM25 (Robertson and Zaragoza, 2009) *ranker* combined with two state-of-the-art unsupervised keyword extraction methods: TextRank and RAKE. We additionally include a baseline giving the user's full utterances as input to BM25 (baseline: *No-keyword*).

Following related work (Nogueira and Cho, 2019; Dai and Callan, 2019), we train the BERT IR baseline using a binary cross-entropy loss to predict the relevance of each document to the user's utterances (acting as queries). We use the same pre-trained BERT model as in CHARM. To fit both utterances and documents into the input size of BERT, we split both into 256-token chunks and run BERT on their Cartesian product. To obtain the final score for each utterances-document pair we average across all chunk pairs. Given N utterances and M documents, this baseline processes $N \times M$ inputs with BERT, whereas CHARM processes N inputs with BERT and M inputs with an efficient ranking method. This makes the BERT IR baseline very computationally expensive on the Wiki-category and Web search document collec-

[6] We used a greedy algorithm to approximate a solution to the NP-hard bin packing problem.

Model	*profession*						*hobby*					
	Wiki-page		Wiki-category		Web search		Wiki-page		Wiki-category		Web search	
	MRR	nDCG	MRR	nDCG	MRR	nDCG	MRR	nDCG	MRR	nDCG	MRR	nDCG
No-keyword + BM25	.15*	.32*	.17*	.37*	.11*	.28*	.16*	.42*	.13*	.35*	.06*	.22*
RAKE + BM25	.16*	.33*	.19*	.39*	.11*	.28*	.17*	.42*	.14*	.37*	.07*	.23*
RAKE + KNRM	.16*	.33*	.13*	.34*	.15*	.34*	.12*	.32*	.12*	.31*	.06*	.24*
TextRank + BM25	.21*	.39*	.26*	.45*	.15*	.32*	.21	.46	.20*	.42*	.10*	.28*
TextRank + KNRM	.21*	.38*	.18*	.36*	.20*	.40*	.15*	.36*	.16*	.36*	.11*	.31*
BERT IR	**.30**	.45	.28*	.44*	.26*	.38*	.22	.43*	.18*	.42*	.15*	.33*
CHARM $_{\text{BM25}}$	.29	**.46**	.28*	.47*	.28*	.45*	**.24**	**.47**	.21*	.43*	.11*	.30*
CHARM $_{\text{KNRM}}$	.27	.44	**.35**	**.55**	**.41**	**.59**	.22	.44*	**.27**	**.49**	**.19**	**.38**

Table 2: Results for *unseen* values. Results marked with * significantly differ from the best method (in bold) measured by a paired t-test ($p < 0.05$). As described in the experimental setup, BERT IR on Wiki-category and Web search must consider a subset of documents.

Model	Document collection	*profession*		*hobby*	
		MRR	nDCG	MRR	nDCG
N-GrAM	-	.13*	.43*	.11*	.40*
W2V-C	-	.09*	.39*	.08*	.32*
CNN	-	.20*	.52*	.14*	.43*
HAM $_{\text{2attn}}$	-	.32*	.59*	.33	**.55**
BERT	-	**.50**	**.68**	**.35**	**.55**
CHARM $_{\text{BM25}}$	Wiki-page	.42*	.57*	.31*	.51*
	Wiki-category	.38*	.56*	.32	.50*
	Web search	.49	.65	.31*	.51
CHARM $_{\text{KNRM}}$	Wiki-page	.37*	.54*	.28*	.46*
	Wiki-category	.43*	.62*	.31	.51*
	Web search	.49	.66	.31	.51

Table 3: Results for *seen* values. Results marked with * significantly differ from the best method (in bold face) measured by a paired t-test ($p < 0.05$).

	profession					
	barista (MRR=0.4, #sample=73)		**screenwriter** (MRR=0.65, #sample=52)		**airplane pilot** (MRR=0.64, #sample=14)	
CHARM	coffee	shop	script	story	pilot	flying
	starbucks	guitar	screenplay	film	flight	teacher
	store	student	screenwriting	films	training	fire
	school	customer	scripts	photo	fly	trading
	manager	college	writing	movie	pilots	military
TextRank	people	amp	first	hollywood	people	american
	first	love	people	tomorrow	first	lots
	coffee	things	thanks	time	things	guy
	today	starbucks	amp	second	today	time
	thanks	work	stuff	one	thanks	guys

Table 4: CHARM $_{\text{KNRM}}$'s top 10 terms per label for *profession* attribute, compared with TextRank keywords.

tions, which contain 4,000-12,000 documents. In order to run the baseline on these collections, we sample three documents per label; even with this change, BERT IR is 60x slower than CHARM. More details on the models' running time are in Appendix B. We use the full document collection with Wiki-page.

For the *seen* experimental setup, we compare CHARM with both state-of-the-art supervised approaches for inferring attribute values and a fine-tuned supervised BERT model that performs classification using its `[CLS]` representation. The Hidden Attribute Model (HAM $_{\text{2attn}}$) (Tigunova et al., 2019) is an attention-based neural classification model for inferring users' attribute values. N-GrAM (Basile et al., 2017) is a SVM classifier with n-gram features. W2V-C (Preoţiuc-Pietro et al., 2015) is a Gaussian Process (GP) classifier with embedding clusters as features. Finally, we include a neural CNN-based model (Bayot and Gonçalves, 2018). In this setup the baseline models are single-value, therefore, we split every multi-value user into several inputs through all their attribute values.

Evaluation metrics. Given the difficulty of inferring the correct attribute values for an attribute with many possible values, ranking metrics are the most informative and have been used in prior work (Tigunova et al., 2019; Preoţiuc-Pietro et al., 2015). We consider MRR (Mean Reciprocal Rank) and nDCG (normalized Discounted Cumulative Gain). Given that MRR assumes there is only one correct attribute value for each user, we calculate MRR independently for each attribute value before averaging. We average nDCG over users.

6 Results and Discussion

6.1 Quantitative Results

***Unseen* values (zero-shot mode).** The models' performance evaluated only on values that were not observed during training is shown in Table 2. Both CHARM variants significantly outperform all unsupervised keyword-extraction baselines for both attributes on all document collections. This suggests the importance of training the cue detector to identify terms related to the attribute, instead of

	hobby					
	baking (MRR=0.46, #sample=64)		**quilting** (MRR=0.26, #sample=27)		**model aircraft** (MRR=0.11, #sample=2)	
CHARM	cake	bread	sewing	way	cat	dimensions
	food	cream	quilting	game	plane	pilots
	recipe	cooking	quilt	metal	construction	song
	cheese	pasta	fabric	design	planes	steam
	baking	cook	music	playing	energy	music
TextRank	thanks	things	thanks	today	thanks	work
	first	work	first	science	german	elyrion
	amp	food	things	kids	steam	time
	people	time	people	time	tapjoy	purchase
	recipes	second	amp	lots	motorola	air

Table 5: CHARM$_{\text{KNRM}}$'s top 10 terms per label for *hobby* attribute, compared with TextRank keywords.

the more general keywords usually given by unsupervised keyword extractors. BERT IR performs similarly to CHARM for the Wiki-page dataset, but performs significantly worse for the remaining datasets while taking approximately 60x longer than CHARM$_{\text{KNRM}}$ to perform inference.

For both attributes, CHARM$_{\text{KNRM}}$ always outperforms the BM25 variant with Wiki-category and Web search collections. This may be related to the size of document collections which allow for more variations in the vocabularies that are captured well by embeddings with KNRM. Another observation is that for CHARM$_{\text{KNRM}}$, while Web search yields the best result for *profession*, Wiki-category is the best collection for *hobby*, possibly due to the noisy hobby-related documents from web search. CHARM$_{\text{BM25}}$ on Wiki-page does not require any additional inputs and consistently performs as well as or better than the baselines across both attributes. Wiki-category performs significantly better than all baselines for both attributes, making it a reasonable choice when Wikipedia categories are available.

To demonstrate that the collections are resilient to inaccuracies in their automatic construction, we conducted an experiment where some percentage of the documents' attribute values were randomly changed. We found that randomly changing 20% of the documents' labels resulted in approximately a 15% MRR decrease for CHARM$_{\text{KNRM}}$ on Web-search and Wiki-category. The performance decrease on these collections was roughly linear. This indicates that noise in the document collection does not severely damage CHARM's performance.

***Seen* values (supervised mode).** In this experiment we evaluate CHARM's performance in the fully supervised setting (i.e., all labels are seen during training). In Table 3 we observe that CHARM's performance is competitive compared to HAM$_{\text{2attn}}$ (i.e., the best-performing attribute value prediction method from prior work) and the state-of-the-art BERT model. The fully supervised BERT model consistently performs the best for both attributes, though these increases are not statistically significant over all CHARM configurations. Furthermore, BERT and HAM$_{\text{2attn}}$ are trained with full supervision in this experimental setting, whereas CHARM still uses a policy gradient. In this experiment, the Web search collection consistently performs best, suggesting that the collection's shortcomings are mitigated when all labels are observed.

6.2 Qualitative Analysis

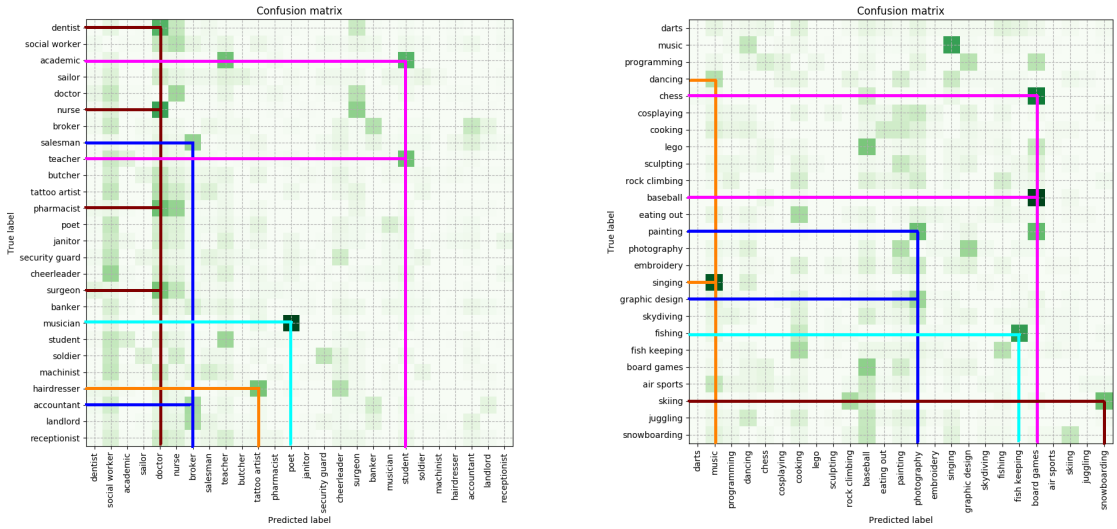
Analysis of selected terms For each attribute value, we gathered all query terms that were selected for the users predicted as having the attribute value, together with the scores given by the cue detector. We then averaged the scores for each term within an attribute value, and selected top 10 terms as the representative ones. Terms were extracted using CHARM$_{\text{KNRM}}$ with Wiki-category on *unseen* experiments. We performed the same method for TextRank keywords, because this was the best performing keyword-based baseline in the *unseen* experiments. The comparison of selected terms by CHARM vs TextRank is reported in Table 4 and Table 5 for selected attribute values of *profession* and *hobby*, respectively.

We can observe that, regardless of the small sample size for some values like *airplane pilot*, CHARM can still detect meaningful words. For *barista*, CHARM did not even consider the term 'barista', but rather focuses on words such as 'coffee' and 'starbucks'. Choosing terms like 'screenplay', 'scripts' and 'screenwriting' helps the model to distinguish *screenwriter* from other film-related professions like *director*.

Picking the terms like 'cake', 'baking' and 'bread', helps the model to distinguish between *baking* and *cooking* hobbies more effectively. Note, that even for rare unusual hobbies like *quilting*, CHARM manages to pick indicative terms. This essentially shows that the model can easily be used for large lists of attribute values, with long tail.

Finally, as opposed to CHARM, TextRank keywords rarely make sense. This suggests that unsupervised keyword detectors are not capable of producing useful attribute-value-related keywords from users' utterances.

Misclassification Study To conduct error analysis,



(a) *profession* (b) *hobby*

Figure 3: Confusion matrix for *profession* and *hobby* with CHARM $_{\text{KNRM}}$ on *unseen* experiments, with some values removed for brevity. Unseen values are aggregated across folds. Darker cells indicate more misclassifications. The lines illustrate misclassifications of interest.

profession		***hobby***	
firefighter (MRR=0.46)	**investor** (MRR=0.52)	**knitting** (MRR=0.68)	**ice hockey** (MRR=0.68)
Firefighter	Index_fund	Yarn_over	Extra_attacker
Firefighter_assist_and_search_team	Venture_capital	Brioche_knitting	Ice_hockey_rules
Calvert_County_Fire-Rescue-EMS	Treasury_management	Combined_knitting	Neutral_zone_trap
Firefighter_arson	Buy_side	Flat_knitting	Playoff_beard
Fire_captain	Sovereign_wealth_fund	Tunisian_crochet	Line_(ice_hockey)

Table 6: CHARM $_{\text{KNRM}}$'s top 5 retrieved documents per attribute value.

we plotted confusion matrices of CHARM $_{\text{KNRM}}$ on *unseen* experiments, which are shown in Figure 3a and 3b for *profession* and *hobby*, respectively.

We observe that medical professions such as *dentist, nurse, pharmacist* and *surgeon* are often confused to *doctor* in general. Professions associated with studying (*academic, teacher* and *student*), beauty (*hairdresser* and *tattoo artist*) and art (*musician* and *poet*) are often confused with each other. *Salesman* and *accountant* are confused to *broker*, because of the common financial terms used.

Hobbies associated with music (*dancing, singing* and *music*) and images (*painting*, *graphic design* and *photography*) are often mixed up. Hobbies in which the term 'game' is profusely used like *chess* and *baseball* are confused to *board games*; similarly, *fishing* and *fish keeping*, as well as *skiing* and *snowboarding* are confused due to the common lexicon used.

Analysis of top ranked documents For each attribute value, we collected all documents that were returned for a user with the given value as the ground-truth label. We then averaged the scores for each page and select the top 5 retrieved pages from Wiki-category, shown in Table 6 for selected *profession* and *hobby* attribute values.

It is interesting to observe, that in spite of the common lexicon for some similar values, the model manages to retrieve documents which are relevant to a particular value, e.g., documents for *investor* are distinct from other financial-related professions, like *broker* or *salesman*. It is also worth mentioning that the retrieved pages for *investor* and *ice hockey* are rather the pages for related lexicon (*venture capital*, *playoff beard*), which shows the power of CHARM's cue detection.

7 Conclusion

We presented the CHARM method for inferring personal traits from conversations. CHARM differs from prior work by its zero-shot ability to predict attribute values that are not present in the training samples at all. We demonstrated the viability of CHARM for inferring users' unseen attribute values by comprehensive experiments with Reddit conversations, leveraging document collections from Wikipedia and web search results for CHARM's retrieval component.

References

Fabian Abel, Qi Gao, Geert-Jan Houben, and Ke Tao. 2011. Analyzing user modeling on twitter for personalized news recommendations. In *ACM UMAP'11*, pages 1–12. Springer.

Jimmy Ba, Kevin Swersky, Sanja Fidler, and Ruslan Salakhutdinov. 2015. Predicting deep zero-shot convolutional neural networks using textual descriptions. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4247–4255.

Krisztian Balog, Filip Radlinski, and Shushan Arakelyan. 2019. Transparent, scrutable and explainable user models for personalized recommendation. In *Proceedings of SIGIR'19*, pages 265–274.

Koji Bando, Kazuyuki Matsumoto, Minoru Yoshida, and Kenji Kita. 2019. Twitter user's hobby estimation based on sequential statements using deep neural networks. *International Journal of Machine Learning and Computing*, 9(2).

Angelo Basile, Gareth Dwyer, Maria Medvedeva, Josine Rawee, Hessel Haagsma, and Malvina Nissim. 2017. N-GrAM: New Groningen Author-profiling Model—Notebook for PAN at CLEF 2017. In *CLEF 2017 Evaluation Labs and Workshop – Working Notes Papers*.

Roy Khristopher Bayot and Teresa Gonçalves. 2018. Age and gender classification of tweets using convolutional neural networks. In *Machine Learning, Optimization, and Big Data*, pages 337–348, Cham. Springer International Publishing.

Zhuyun Dai and Jamie Callan. 2019. Deeper text understanding for ir with contextual neural language modeling. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 985–988.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ethan Fast, Binbin Chen, and Michael S Bernstein. 2016. Empath: Understanding topic signals in large-scale text. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 4647–4657. ACM.

Jiazhan Feng, Chongyang Tao, Wei Wu, Yansong Feng, Dongyan Zhao, and Rui Yan. 2019. Learning a matching model with co-teaching for multi-turn response selection in retrieval-based dialogue systems. In *Proceedings ACL'19*, pages 3805–3815.

Lucie Flekova, Jordan Carpenter, Salvatore Giorgi, Lyle Ungar, and Daniel Preoţiuc-Pietro. 2016. Analyzing biases in human perception of user age and gender from text. In *Proceedings of ACL'16 (Volume 1: Long Papers)*, pages 843–854.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909*.

M. Habibi and A. Popescu-Belis. 2015. Keyword extraction and clustering for document recommendation in conversations. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(4):746–759.

Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2018. Co-pacrr: A context-aware neural ir model for ad-hoc retrieval. In *WSDM 2018: The Eleventh ACM International Conference on Web Search and Data Mining*.

Hongyan Jing, Nanda Kambhatla, and Salim Roukos. 2007. Extracting social networks and biographical facts from conversational speech transcripts. In *Proceedings of ACL'07*.

Su Nam Kim and Timothy Baldwin. 2012. Extracting keywords from multi-party live chats. In *Proceedings of PACLIC'12*, pages 199–208.

Bernhard Kratzwald and Stefan Feuerriegel. 2018. Adaptive document retrieval for deep question answering. In *Proceedings of EMNLP'18*, pages 576–581.

Revathy Krishnamurthy, Pavan Kapanipathi, Amit P Sheth, and Krishnaprasad Thirunarayan. 2014. Location prediction of twitter users using wikipedia.

Hugo Larochelle, Dumitru Erhan, and Yoshua Bengio. 2008. Zero-data learning of new tasks. In *AAAI*, volume 1, page 3.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. *arXiv preprint arXiv:1706.04115*.

Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model. In *Proceedings of ACL'16 (Volume 1: Long Papers)*.

Xiang Li, Gökhan Tür, Dilek Z. Hakkani-Tür, and Qi Li. 2014. Personal knowledge graph population from user utterances in conversational understanding. In *Proceedings of IEEE Spoken Language Technology Workshop (SLT)*.

Liangchen Luo, Wenhao Huang, Qi Zeng, Zaiqing Nie, and Xu Sun. 2019. Learning personalized end-to-end goal-oriented dialog. In *Proceedings of AAAI'19*.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of EMNLP'04*, pages 404–411.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*.

Mark Palatucci, Dean Pomerleau, Geoffrey E Hinton, and Tom M Mitchell. 2009. Zero-shot learning with semantic output codes. In *Advances in neural information processing systems*, pages 1410–1418.

Panupong Pasupat and Percy Liang. 2014. Zero-shot entity extraction from web pages. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 391–401.

Daniel Preoţiuc-Pietro, Vasileios Lampos, and Nikolaos Aletras. 2015. An analysis of the user occupational class through twitter content. In *Proceedings of ACL/IJCNLP'15 (Volume 1: Long Papers)*, pages 1754–1764.

Delip Rao, David Yarowsky, Abhishek Shreevats, and Manaswi Gupta. 2010. Classifying latent user attributes in twitter. In *Proceedings of SMUC'10*, pages 37–44.

Alexander Ratner, Stephen H. Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel. *Proceedings of the VLDB Endowment*, 11(3):269–282.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at trec-3. *Nist Special Publication Sp*, 109.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. *Text mining: applications and theory*, 1:1–20.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063.

Anna Tigunova, Andrew Yates, Paramita Mirza, and Gerhard Weikum. 2019. Listening between the lines: Learning personal attributes from conversations. In *Proceedings of WWW'19*, pages 1818–1828.

Christophe Van Gysel, Bhaskar Mitra, Matteo Venanzi, Roy Rosemarin, Grzegorz Kukla, Piotr Grudzien, and Nicola Cancedda. 2017. Reply with: Proactive recommendation of email attachments. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 327–336.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerry Tesauro, Bowen Zhou, and Jing Jiang. 2018. R 3: Reinforced ranker-reader for open-domain question answering. In *Proceedings of AAAI'18*.

Wei Wang, Vincent W Zheng, Han Yu, and Chunyan Miao. 2019. A survey of zero-shot learning: Settings, methods, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–37.

Charles Welch, Verónica Pérez-Rosas, Jonathan K. Kummerfeld, and Rada Mihalcea. 2019. Look who's talking: Inferring speaker attributes from personal longitudinal dialog. In *Proceedings of CICLing*, La Rochelle, France. Springer.

Wei Wu, Bin Zhang, and Mari Ostendorf. 2010. Automatic generation of personalized annotation tags for twitter users. In *Proceedings of NAACL-HLT'10*.

Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of SIGIR'17*.

Majid Yazdani and James Henderson. 2015. A model of zero-shot learning of spoken language understanding. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 244–249.

An-Zi Yen, Hen-Hsen Huang, and Hsin-Hsi Chen. 2019. Personal knowledge base construction from text-based lifelogs. In *Proceedings of SIGIR'19*, pages 185–194.

Zhaohao Zeng, Ruihua Song, Pingping Lin, and Tetsuya Sakai. 2019. Attitude detection for one-round conversation: Jointly extracting target-polarity pairs. In *Proceedings of WSDM'19*, pages 285–293.

Jingqing Zhang, Piyawat Lertvittayakumjorn, and Yike Guo. 2019. Integrating semantic knowledge to tackle zero-shot text classification. *arXiv preprint arXiv:1903.12626*.

Qi Zhang, Yang Wang, Yeyun Gong, and Xuanjing Huang. 2016. Keyphrase extraction using deep recurrent neural networks on twitter. In *Proceedings of EMNLP'16*, pages 836–845.

Appendices

A Data

All datasets used in the experiments are available at https://github.com/Anna146/CHARM. We provide IDs and texts of the posts used as training and test data for CHARM. All users are anonymized by replacing usernames with IDs. Additionally, we provide the posts containing explicit personal assertions, which have been used for ground truth labeling with the Snorkel framework.

A.1 Labeling users' utterances with Snorkel

Our data consists of submissions on Reddit, which are: *(1)* authored by users having 10-50 posts, *(2)* 10-40 words long, and *(3)* containing a personal pronoun (except for 3rd person ones. Requirements *(1)* and *(2)* were derived from observing the distributions on the full dataset. Requirement *(3)* comes from the assumption that posts containing personal pronouns are most likely to contain personal assertions. These restrictions allow us to select posts that look more similar to the real conversation (i.e., relatively short and containing references to the speakers with personal pronouns). In addition, we did not consider the following subreddit types: *(i) dating*, which may provide plenty of personal information but no real conversation to infer from, and *(ii) fantasy/video games* (for the *profession* attribute), because users may refer to gaming personalities. We took only users whose utterances contain at least one mention of attribute values, resulting in around 250K and 500K candidate users for profession and hobby, respectively.

We used the Snorkel framework (Ratner et al., 2017) that allows data labeling using weak supervision, relying on the inference that combines multiple *labeling functions*, which are manually specified and can be potentially noisy. Given a user's utterance set U, an attribute a and a possible attribute value v, Snorkel will decide on *positive/negative* label–denoting the user as having/not having personal trait a:v–or *abstain* label. We have separate labeling models for each attribute a, and defined two labeling functions which consider: *(LF1)* the existence of *attribute-specific patterns*, and *(LF2)* the weighted count of the words belonging to the *value-specific lexicon*.

LF1: Attribute-specific patterns. We compiled a list of positive and negative patterns for each attribute (see Table 7), e.g., *"my hobby is ⟨hobby-value⟩"* vs *"I hate ⟨hobby-value⟩"* as positive vs negative patterns for hobby. LF1 labels a user with a *positive/negative* label for each attribute value v if there exist at least one positive/negative pattern in the user's utterances U, and *abstain* otherwise.

LF2: Value-specific lexicon. For each attribute-value pair, we used *Empath* (Fast et al., 2016)–pre-trained on the Reddit corpus–to build a lexicon of *typical words* (e.g., 'cider' and 'yeast' for *hobby:brewing*). Given seed words, Empath builds lexical categories by means of an embedding model. As our value-specific lexicon, we took the union of Empath terms for a specific attribute value and all its synonyms; each typical word is weighted by embedding similarity to the seed words. Given a user's utterance set U and an attribute value v, LF2 yields a *positive* label if the weighted count of typical words of v is above an empirically-chosen threshold, and *abstain* otherwise.

Given a pair of user's utterance set U and a possible attribute value v, the Snorkel probabilistic labeling model utilizes our labeling functions to predict a confidence score for the *positive* label, i.e., the user is labeled with attribute value v. As our labeled dataset, we took only the user-value pairs with confidence scores above a specific threshold.

To determine the threshold of confidence scores, we manually annotated a held-out validation set containing 100 users per attribute. Given a post and a set of attribute values mentioned explicitly in the post, the annotators must identify whether the candidate user traits truly hold. For instance, from *"My dad bought me a **chess** board even though I enjoy **video games** more"*, *hobby:video games* is correct while *hobby:chess* is not applicable. The final annotation for each post consists of attribute values agreed by at least two out of three judges. The selected confidence threshold corresponds to the 0.9 precision of the model on the validation set. After thresholding, we obtained 13.5k users labeled with profession values and 11.7k users with hobby values.

Finally, for practical reasons, for each attribute we sorted the labeled users by confidence scores and cropped the set to maximum 500 users per attribute value and 6000 users in total. Note that users might have multiple values for each attribute (e.g., having *brewing* and *swimming* as hobbies); there are 605 such users for profession and 245 for hobby.

	positive	negative
profession	i am/i'm a(n) my profession is i work as my job is my occupation is i regret becoming a(n)	(no/not/don't within pos. patterns)
hobby	i am/i'm obsessed with i am/i'm fond of i am/i'm keen on i like i enjoy i love i play i take joy in i adore i appreciate i am/i'm fan of i am/i'm fascinated by i am/i'm interested in i fancy i am/i'm mad about i practise i am/i'm into i am/i'm sucker for my interest is my hobby is my passion is my obsession is	i hate i dislike i detest i can't stand (never/not/don't within pos. patterns)

Table 7: Positive and negative patterns used in the labeling function LF1 of the Snorkel labeling model. Each pattern must be followed by possible attribute values within a context window of 2 terms.

B Training details and hyperparameters

In our experiments we used the server with 32 cores (2x Intel Xeon Gold 6242, 16C/32T 22MB) and 2 GPU NVIDIA Corporation GV100 [Tesla V100]. On this server the running time of our models was fast, compared to the baseline BERT IR architecture as shown in Table 8. BERT IR inference is slow because for a single utterance-document pair it makes several passes through BERT for each chunk combination, which is repeated for every document. CHARM runs BERT once on each utterance only, independent of the number of documents. Using BM25 as a ranker is slower because it requires iteration through the query-document inputs to calculate term frequencies, whereas KNRM uses efficient vectorized representations of the inputs. However, it is possible to speed up BM25 inference, by providing a precomputed inverted index.

	train (10.000 instances)	**test** (100 instances)
CHARM $_{\text{KNRM}}$	31.8	1.2
CHARM $_{\text{BM25}}$	54.4	10.9
BERT IR	56.2	72.7

Table 8: Running time of the models given in minutes. The train time is a sum of the times across all training epochs, all times are averaged across folds in the unseen experiment.

The numbers of parameters in CHARM $_{\text{KNRM}}$ model are shown in Table 9. We used manual tuning to search for the hyperparameters, running about 280 search trials per attribute and collection combination. Several hyperparameters were fixed across different setups (across attributes, document collections and rankers) and some we tuned to each setup individually. The bounds for each hyperparameter and the best parameters are in Tables 11, 10. The best parameters were chosen based on the MRR score. Additionally we performed some experiments on changing the policy gradient training setup, adding discounting factor to the reward after each sampled query term and changing the reward from nDCG to MRR. We found that the results after these modifications did not significantly change.

	Number of parameters (e+3)
BERT embeddings	23,832.6
word2vec embeddings	882,366
BERT parameters	43,118.6
KNRM parameters	0.4

Table 9: Number of model parameters. CHARM $_{\text{KNRM}}$ uses all parameters mentioned in the table, while CHARM $_{\text{BM25}}$ and BERT IR use only parameters related to BERT.

Parameter	Options	*hobby*						*profession*					
		CHARM$_{BM25}$			CHARM$_{KNRM}$			CHARM$_{BM25}$			CHARM$_{KNRM}$		
		Wiki-page	Wiki-category	Web search	Wiki-page	Wiki-category	Web search	Wiki-page	Wiki-category	Web search	Wiki-page	Wiki-category	Web search
aggregation type	avg, max	avg	avg	max	avg	avg	avg	avg	max	avg	max	avg	avg
training epochs	1-50, step 2	19	23	21	23	21	21	17	23	15	43	27	17
query length	10-25 step 5	15	25	10	10	15	15	10	15	15	10	15	10

Table 10: Hyperparameter search for specific configurations.

Parameter	**Search bounds** (low; high; step)	**Best configuration**
BM25: k1	(0.75; 2.0; 0.25)	2.0
BM25: b	(0.25; 1.0; 0.25)	0.75
batch size	(2; 4; 1)	4
negative labels sampled	(5; 15; 5)	15
documents sampled per label	(3; 9; 2)	5

Table 11: Common parameters across all attributes and document collections. The last two parameters refer to the number of negative labels used during training for one instance and number of documents sampled for each selected label.