Harnessing Twitter for Answering Opinion List Queries

Ankan Mullick, Pawan Goyal, Niloy Ganguly, and Manish Gupta

***Abstract*— Opinion list (OL) queries like "valentines day gift ideas" and "best anniversary messages for your parents" are quite popular on web search engines. Users expect instant answers comprising of a list of relevant items (OL) for such a query. Surprisingly, current search engines do not provide any crisp instant answers for queries in this critical query segment. To the best of our knowledge, we present the first system that tackles such queries. Although such social factors are heavily discussed on online social networks like Twitter, extracting such lists from tweets is quite challenging. The challenges lie in discovering such lists from tweets, rank the discovered list items as well as handle lists with very low cardinality (tail OLs). We present an end-to-end system that: 1) identifies these "OLs" from a large number of Twitter hashtags using a classifier trained using novel task-specific features; 2) extracts suitable list answers from relevant tweets using carefully designed regex patterns; 3) uses the learning to rank framework to present a ranked list of these items; and 4) handles tail lists using a novel algorithm to borrow list items from similar lists. Crowd-sourced evaluation shows that the proposed system can extract OLs with a good accuracy.**

***Index Terms*— List item extraction, opinion lists (OLs), opinion mining, opinion queries, opinion ranking, opinion search.**

I. Introduction

WEB content and web search have matured significantly over the past two decades. Beyond the 10 blue links interface, search engines have now become entity aware, show instant answers, show multimodal heterogeneous answers, and so on. In the past few years, searching one particular type of queries that has increased significantly is the segment of opinionated queries. Opinionated queries seek a list of items as an answer and are often related to social topics. A few examples include "innovative marketing ideas for managers," "how to wish good night," "fun events in Miami," and so on. Users put up such queries in the hope of obtaining innovative, witty, popular, and informative answers from opinions and experiences expressed on matching webpages. While some of these queries seek information about social events (e.g., "ways to do wedding decorations" and "how to organize a concert"), many others are related to social situations and moods (e.g., "late night conversations," "tricks to feed your baby," and "common lies people say"). Fig. 1 shows percentage of "opinion list (OL) queries" on Bing since 2011. It shows a significant increase of 36.7% and 25.9% in 2015 and 2016, respectively[1] making it almost 2% of entire Bing query.

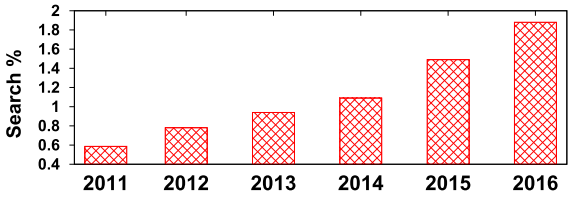


Fig. 1. Yearwise percentages of OL queries.

Search engines are very effective at displaying instant answers for factoid queries like "weather New York" or "temperature Singapore." However, they hardly support instant answers for such OL queries. In Fig. 2, we show top Google results for the opinion query "8th grade memories."[2] We find that Google does not provide instant answers. Fig. 3 shows a snapshot from the first Google result, which is a Prezi presentation.

It requires significant efforts to extract useful list answers from the first Google result page: "chromebooks," "first day at school," and so on from the flash presentation. Similarly, the next result provides some list items but in the form of a flash video. For a large number of other such queries, no good list item can be obtained even after browsing through the top few results.

One of the primary reasons behind the explosion of such queries is proliferation of such content in social media such as Twitter. Twitter is, by its nature, a social platform for people to express their opinions, choices, and suggestions through tweets and hashtags [1]. Consequently, Twitter hashtags can be clearly exploited toward shortlisting items for such OL queries. A few tweets for "8th grade memories" are shown

Manuscript received April 8, 2018; revised September 8, 2018; accepted November 4, 2018. Date of current version December 3, 2018. *(Corresponding author: Ankan Mullick.)*

A. Mullick is with Microsoft, Hyderabad 500032, India, and also with the Department of Computer Science and Engineering, IIT Kharagpur, Kharagpur 721302, India (e-mail: ankan.mullick@microsoft.com).

P. Goyal and N. Ganguly are with the Department of Computer Science and Engineering, IIT Kharagpur, Kharagpur 721302, India (e-mail: pawang@cse.iitkgp.ernet.in; niloy@cse.iitkgp.ernet.in).

M. Gupta is with Microsoft, Hyderabad 500032, India (e-mail: gmanish@microsoft.com).

Digital Object Identifier 10.1109/TCSS.2018.2881186

[1]To calculate these numbers, we took the Bing query log for the first day of December of every year. We filtered this list using patterns for list type queries. Then, we randomly selected 200 queries out of this filtered set. We manually labeled the queries in this sample were as "OL queries" or not. The graph shows the fraction of queries meeting the filter criteria among all the queries multiplied by the fraction of queries which were labeled as "OL queries" in our random sample.

[2]Bing also displays similar results

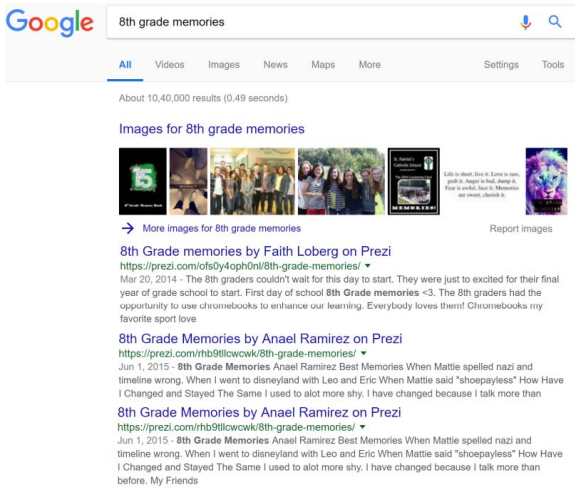


Fig. 2. Top google results for "8th grade memories," retrieved on March 4, 2018.

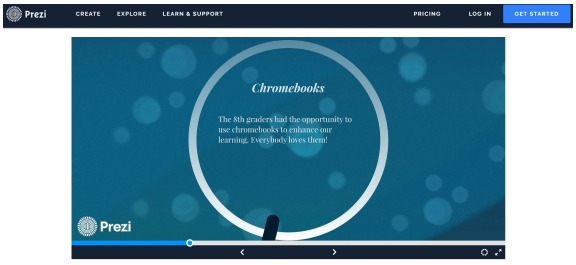


Fig. 3. Snapshot from the Prezi presentation corresponding to the first google result for "8th grade memories."

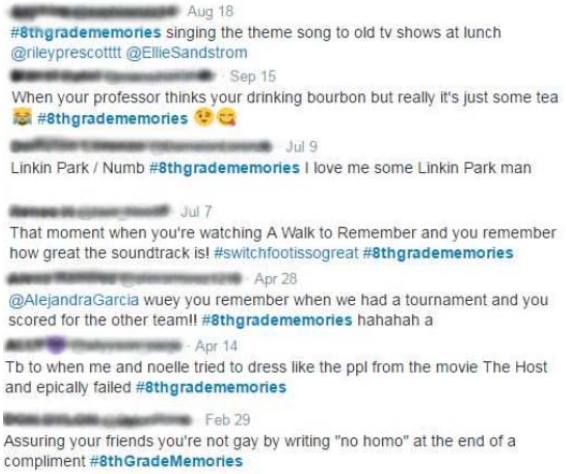


Fig. 4. Some tweets for "8th grade memories."

in Fig. 4. As is evident, very interesting list items can be quickly extracted from such tweets: "singing the theme song to old TV shows," "watching some movie/listening to some song," "scoring for the other team in a tournament," and so on. Subsequently, these results can be presented in a list form as search output. The aim of this paper is to retrieve structured information from those discussions and produce a list of items corresponding to each OL. We implement the methodology on Twitter data.

On Twitter, among different types of conversational tweets, a specific category of tweets discusses "social topics"—we refer to the hashtags that anchor them as OL-hashtags. Examples of OL-hashtags include #tipsforinteriordesign, #10thingspeoplelovetodo, #adviceforjunorstudents, and so on. Formally, we define OL-hashtags as a particular category of hashtags, where users talk about: 1) a list of things, specific to the corresponding social topic or a list of list of things mentioned in tweets and 2) among a random sample of some tweets containing this hashtag, a good fraction contain the list items specific to this opinionated hashtag. OL-hashtags can be broadly classified into two categories: 1) objective—that contains factual objects as list items, e.g., names of places in #thingstoseeinbarcelona or movie names in #3movieshatmakeyoucry and 2) subjective—that contain list items conveying people's views on a topic, e.g., #schoolmemories expresses subjective list items conveying school life memories. OL-hashtags are not similar to traditional lists[3] or Twitter idioms.[4]

However, we need to overcome many difficulties to build a system as follows.

OL-hashtags

1) constitute a very small percentage of all hashtags, hence, detecting these sparse identities is challenging.
2) Tweets are unstructured and noisy, so extraction of valid list items from the unstructured texts and URLs (contained within tweets) is not trivial.
3) The derived valid list items may not be relevant. For example, corresponding to "late night party ideas" query, the derived item *We need #latenightpartyplans* provides no relevant lists, hence, deriving relevant items from the valid list is challenging.
4) Finally, many OL-hashtags occur only in few tweets leading to retrieval of only a small number of relevant list items.

It is challenging to populate such sparse tail lists with relevant list items from other similar hashtags.

Toward overcoming the above-mentioned challenges, we make the following contributions.

1) We designed a classifier to extract OL-hashtags from Twitter using the very specific hashtag and tweet-level features. Our recall optimized classifier provides a precision of 75.5% at a recall of 95.3% (Section IV).
2) We extract list items from the tweets containing the OL-hashtags using regular expressions with an accuracy @10 of 66.12% (Section V).

[3]Traditional list items are mostly factual, not related to people's opinion or experiences. For example, "Oscar winners 2017."

[4]Idioms are defined as tags representing a conversational theme on Twitter, consisting of a concatenation of at least two common words, while the concatenation does not include names of people or places and the full phrase cannot be a proper noun itself. We observe that while all the OLs qualify this definition, all idioms are not query type hashtags where the user expects a list of answers. OL-hashtags are a subset of idiom set; for instance, from the idiom list of five—#africanproblems, #aboutmenight, #awkwardcompanynames, #musicmonday, and #childhoodfeels and only three—#africanproblems, #awkwardcompanynames, and #childhoodfeels qualify as opinion list hashtags.

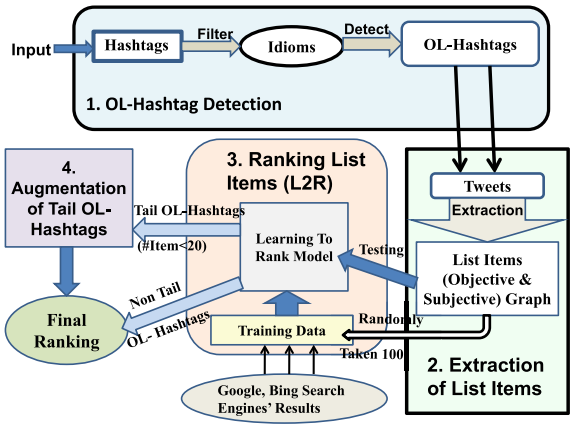


Fig. 5. Overall framework.

3) Furthermore, using a learning-to-rank approach, we promote the relevant settings to a higher rank. This provides a Precision@10 of 91.31% and Precision@20 of 90.72% (Section VI).
4) We also designed mechanisms to handle tail OLs (Section VII).

The overall framework of these steps is illustrated in Fig. 5. To enrich the data set (Section III), we manually annotated 1001 OL-hashtags, along with manually evaluated list items.

Note that our classifier to extract OL-hashtags from Twitter was first proposed in a prior study [2]. This paper extends our prior work as follows. First, we extract the list items from the tweets containing these OL-hashtags and use a learning-to-rank approach to promote the relevant settings to a higher rank. Second, we also design mechanisms to handle the tail opinion lists.

II. Related Work

We review related work on the categorization of Twitter hashtags, opinion mining, and structured data extraction.

A. Categorization of Twitter Hashtags

Romero *et al.* [3] worked on the problem of categorizing Twitter hashtags into various categories such as celebrity, sports, idioms (conversational theme marker hashtags), and so on. They also discussed as to how hashtags of different types and topics exhibit the different mechanism of spreading. Lee *et al.* [4] classified Twitter trending topics/hashtags across multiple categories in real time.

Naaman *et al.* [5] characterized and categorized Twitter trend variations across different geographical areas. Bhattacharya *et al.* [6] identified various topical groups in Twitter and analyzed their characteristics. Zubiaga *et al.* [7] explored various hypotheses for trending hashtags toward early detection of trending topics and also developed a real-time classifier to classify Twitter trending topics into different categories. Tsur and Rappoport [8] explored how Twitter hashtags cause information diffusion considering each tweet as an idea and predicted characteristics of spreading in the community. Maity *et al.* [9] built a classifier with 86.9% accuracy, decent precision, and recall to detect idiom hashtags. Their proposed two-stage framework also predicts the popularity of Twitter idiom hashtags in the different time frame after the birth of the hashtags. Rudra *et al.* [10] filtered the hashtag idioms based on community detection on Twitter. They also showed how users oriented to Twitter idiom are clustered into one community while topical users get clustered into another. They also discussed the trending dynamics of idiom users. None of these works, however, have focused on OL-hashtags. We present an approach to identify OL-hashtags with high accuracy.

B. Opinion Mining

Opinion mining is a major field of study for the last two decades. Researchers have explored opinions in various dimensions. Kim and Hovy [11] presented the methodology to detect an opinion with its holder and topic given a sentence in online news media texts. Qadir [12] worked on opinion mining in customer reviews and product features. Scholz and Conrad [13] explored the field of opinion tonality identification. Yu and Hatzivassiloglou [14] developed a model to identify opinions and classify positive and negative opinions in a question answering system. Wiebe and Riloff [15] distinguished subjective and objective sentences using sentence level features. Asher *et al.* [16] detected categorical and subcategorical opinions. Rajkumar *et al.* [17] identified opinions and facts from news articles using hyperlink-induced topic search (HITS) algorithm in a graphical framework. Mullick *et al.* [18] extended this HITS framework to detect diverse opinions and then built classifiers to detect opinion subcategories by designing Opinion Diversity Algorithm (OP-D). Mullick *et al.* [19] developed an opinion-fact classifier to detect opinionatedness in online social media. Mullick *et al.* [20] identify opinion and fact subcategories from social web. However, none of the previous work focused on exploring opinionated queries to find list items as possible answers, or even extracting opinions from Twitter and putting these as lists.

C. Extraction of Structured Content From Web and Social Media

Extraction of structured content from the web is a well-studied problem. Liu *et al.* [21] developed a novel and effective technique on mining contiguous and noncontiguous data record on the web based on string matching algorithm. Gatterbauer *et al.* [22] approached the problem of domain-independent information extraction from web tables by shifting our attention from the tree-based representation of web pages to a variation of the 2-D visual box model used by web browsers to display the information on the screen. Cafarella *et al.* [23] attempted to build a comprehensive Web database by running multiple domain-independent extractors in parallel over a Web crawl, then combining their outputs into a single large entity-relationship database.

Miao *et al.* [24] presented a novel approach to extract data records from Web pages. Their method first detects the visually repeating information on a Web page and then extracts the

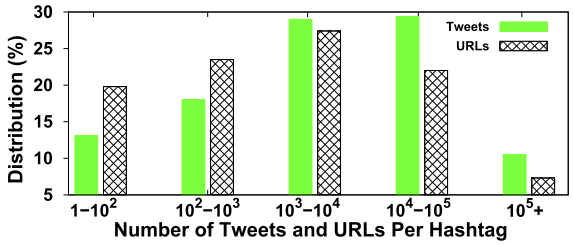


Fig. 6. Distribution of tweets and URLs for our data set.

data records. Hybrid approaches for generalized list extraction have also been proposed by Fumarola *et al.* [25], [26]. Finding high-quality content in social media has also been well studied by Agichtein *et al.* [27]. Rafiei and Li [28] and Wang *et al.* [29] worked on extracting information from the web using the wildcard and other queries.

Zhang *et al.* [30] worked on extraction of top-k lists from some specific types of webpages. However, this approach assumes a given structure of the webpages and is not applicable, in general, on noisy social media data. We study the problem of collecting list items from the noisy tweet data and providing top-ranked list items.

III. Data Set

To prepare our data set, we collected ∼4M Twitter hashtags each with a minimum frequency of 20 from January 2015 to June 2015 using the Twitter Streaming application programming interface (API).[5] Recall that the set of OL-hashtags is a subset of idioms. Hence, we first removed nonidioms from this set. The state-of-the-art idiom detection algorithm [9] resulted in identification of ∼67K idioms from the ∼4M hashtags. Furthermore, these idioms were manually annotated to identify 1001 OL-hashtags. The annotation guidelines were to select only those hashtags as OL-hashtags, which can lead to list type answers. While selecting the OL-hashtags, the annotators also labeled these as "subjective" or "objective." 589 OL-hashtags were labeled as subjective, while 412 were labeled as objective. To be able to train a classifier to detect OL-hashtags from idioms, we also annotated another 1001 hashtags which are not OL-hashtags to obtain a balanced data set of total 2002 hashtags.

English tweets corresponding to all 2002 hashtags were also collected. Often times, such tweets contain relevant URLs. The data set contains ∼204.43 million tweets and ∼85.07 million URLs, respectively. Fig. 6 shows detailed statistics of the tweets and URLs distribution for the OL-hashtags. In the figure, the x-axis represents various ranges of tweets and URLs for any OL-hashtag, and the y-axis represents the percentage of total hashtags, which have tweets/URLs in this range. For example, only 13.1% of total hashtags contain tweets within range (1–100) while 19.3% of total hashtags contain URLs in range (1–100).

IV. OL-Hashtag Detection

In this section, we discuss the design of a classifier that can identify OL-hashtags from the rest. We first perform the following preprocessing steps.

1) Segmentation: Not all hashtags are in CamelCase style (e.g., #BeTheHope) so it is not trivial to identify constituent words from a hashtag. We segment each hashtag using a modified version of the Viterbi Algorithm [31] and the Google n-gram corpus.[6]
2) Part of Speech (POS) Tagging: We use Carneigie Mellon University (CMU) part of speech (POS) tagger for tweets [32] to identify different POS tags, @-mentions, URLs, and so on.

The data preprocessing step is followed by extracting the following hashtag and tweet features.

A. Features

All features can be categorized into three broad feature subsets—language features, search features, and Tweet features.

Language Features

a) *Hashtag Length:* Number of characters in the hashtag.

b) *Number of Words:* After segmenting the hashtag, we count the number of words.

c) *Presence of Days:* This is a binary feature that captures the presence of names of days of the week (e.g., Sunday, Monday, and so on).

d)–i) *Presence and Count of POS Tags:* After doing Part of Speech Tagger (POS) tagging by CMU POC tagger [32], the presence of some POS tags is considered as binary features: preposition (d), interjection (e), whereas some are considered as numeric features: the count of nouns (f), pronouns (g), adjectives (h), and verbs (i).

j) *POS Tag Entropy:* Entropy of the hashtag denotes POS tag diversity. After POS tagging the hashtag with the CMU POS tagger, we take 14 POS tags (noun, pronoun, verb, and so on) to calculate the entropy using the standard definition as follows:

$$\text{Entropy}(i) = -\sum_{j=1}^{14} p_{ij} \times \log(p_{ij}) \tag{1}$$

where i denotes the respective hashtag and j denotes the 14 POS tags.

k) *Vocabulary Ratio*: Ratio of the count of vocabulary words to out-of-vocabulary words.

l) *Presence of Numbers:* This is a binary feature based on the presence of numbers in the hashtag.

m) *Presence of Plurals:* If the noun in the segmented hashtag is plural then this feature is set to 1 otherwise 0. For example in #vdaygiftideas, "ideas" is the plural form of "idea." Similarly, in #fishingtips, "tips" is the plural form of "tip."

n) *Presence of Category Match:* We find that many of the opinion lists follow some specific patterns. It is a binary feature that captures the presence of any such patterns as follows: 1) wh word-*-verb: #howtolearnfromyourmistakes, #whatdreamsaremadeof, #10peoplewhomeanalottome; 2) **-in-⟨num⟩-words: #senioryear in4words, #lovestoryin5words; 3) top-⟨num⟩*-adv*-adj*-noun: #top3romanticmovies; 4) presence of plurals at the beginning or end:

[5]https://dev.twitter.com/streaming/firehose

[6]http://books.google.com/ngrams

TABLE I
CATEGORIZATION OF FEATURES FOR OL DETECTION

Category	Features
Language Features	(a) Hashtag Length, (b) No. of Words, (c) Presence of Days, Presence of (d) Prepositions and (e) Interjections, Count of (f) Nouns, (g) Pronouns, (h) Adjectives, (i) Verbs, (j) POS Tag Entropy, (k) Vocabulary Ratio, Presence of (l) Number and (m) Plurals, (n) Category Match
Search Features	(o) Page Body Coverage, (p-q) Title Coverage in Top 10 and Top 20 Search Engine Results
Tweet Features	(r) Duration, (s) #Timespans, #co-occurring hashtags = (t) 0, (u) 1, (v) 2, (w) 3, (x) 4, (y) 5

#unsolvedmysteries, #thingsthatmakeyoucoolinschool; and 5) presence of superlative adjectives (best, worst, most, and so on): #besttimeoftheyear, #worstcustomerservices.

Search Features

Some hashtags are quite interesting—e.g., "howtoloseaguyin10days" looks like an OL-hashtag but actually is a movie name. An important way to detect it is by querying in search engines. Intuitively, if the titles of returned webpages contain phrases with different cardinality of list items, it is an OL-hashtag, else it is not. For example, all search results for "how to lose a guy in 10 days" refer to exactly 10 days. However, a search for "10 birthday day gift ideas" leads to results with five ideas, 20 ideas, and so on in the title. Accordingly, we have extracted the following features by querying in Google and Bing with segmented hashtags as follows.

o) *Page Body Coverage:* Number of times the segmented hashtag appears in the top 10 webpages.

p)–q) *Title Coverage in Top* 10 *and Top 20 Search Engine Results:* Number of titles in top 10/20 results which contain the hashtag but with a different number. For hashtags that do not contain the number, this feature is set to the number of titles containing the hashtag.

Tweet Features

Processing the English tweets related to OL-hashtags, following features are extracted.

r) *Duration:* Time duration for which the hashtag was popular.

s) *#Timespans:* Number of contiguous time chunks for which the hashtag was popular.

t)–y) *Distribution of Other Cooccurring Hashtags:* While event related hashtags may cooccur frequently with other hashtags, other cooccurring tags are expected to have a relatively low frequency for OL-hashtags. We encode this intuition as a set of features which provide the distribution over tweets (containing the current hashtag) with 0–5 other hashtags.

A summary of all the features is presented in Table I.

B. Classification Results

After the features have been extracted for the hashtags, various classifiers: Naïve Bayes (NB), Logistic Regression (LR), Support Vector Machine (SVM), Local Deep SVM (LDSVM), Binary Neural Network (BNN), Gradient Boosted Tree (GBT), Averaged Perceptron (AP), and XGBoost Binary Classification (XGBBC) were used to detect OL-hashtags. Average across 10 rounds of 10-fold cross-validation test results in terms of precision (P) and recall (R) for OL-hashtags, overall accuracy (A), area under curve (AUC), and respective standard deviations (SD) are shown in Table II.

TABLE II
OL-HASHTAG DETECTION: AVERAGE ACROSS 10 ROUNDS OF 10-FOLD CROSS-VALIDATION RESULTS AND RESPECTIVE SD OF PRECISION (P), RECALL (R), ACCURACY (A), AND AUC FOR VARIOUS CLASSIFIERS

Classifier	P, SD(P)	R, SD(R)	A, SD(A)	AUC, SD(AUC)
NB	78.81, 5.73	89.47, 5.13	82.67, 3.75	92.2, 1.51
LR	86.17, 4.29	85.6, 3.32	**86.12**, 2.27	**94.1**, 1.07
SVM	86.05, 8.28	74.7, 1.41	80.76, 4.29	92.2, 1.58
LDSVM	84.47, 5.2	84.3, 2.92	84.33, 2.1	90.9, 1.36
BNN	81.3, 3.85	80.1, 5.04	80.77, 1.75	90.6, 1.17
GBT	85.05, 3.81	85.81, 4.33	85.38, 2.41	93.25, 1.39
AP	83.79, 4.02	83.17, 4.01	83.44, 1.8	92.7, 1.23
XGBBC	85.27, 3.77	84.53, 5.31	84.94, 3.25	92.67, 1.77

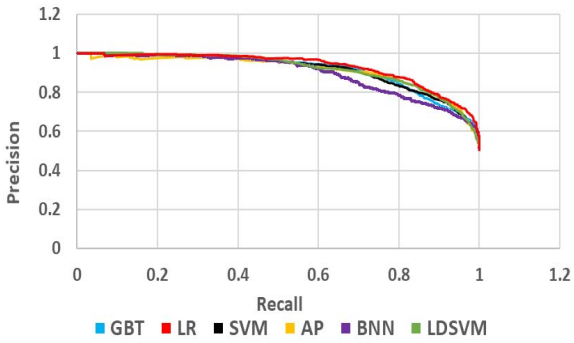


Fig. 7. AUC for various classifiers.

Table II shows that LR provides the best accuracy and AUC among all the classifiers. Tuning various parameters for LR classifier, we have found the best AUC with regularization weights L1 = 0.5, L2 = 0.1, and initial weight = 0.5. For the proposed task, an ideal classifier would be one with very good recall and reasonable precision. A lower precision would simply amount to indexing of some extra OL-hashtags, which may never be used. Hence, we tune the LR classifier to provide a high recall of 0.953 with a precision of 0.755. We perform further analysis using this recall-optimized LR classifier. Fig. 7 shows precision–recall curves for classifiers having the best and least three AUC. Fig. 7 shows that LR classifier is the best recall optimized classifier with highest AUC.

C. Feature Importance

We explored the accuracy obtained using various broad feature subset combinations to measure the feature importance for all the combinations of language features, search features, and Tweet features. The best results are obtained when we use all the three feature subsets together.[7]

To understand individual feature importance, we calculate the information gain (IG) and one attribute evaluation (OAE) accuracy results for each feature. Detailed analysis is presented in [2], which shows that that "a) hashtag length," "f) count of nouns," "j) POS tag entropy," and "m) presence of plurals" are very effective features with high IG and high OAE values.

[7]Please refer to [2] for further details.

TABLE III
REGULAR EXPRESSIONS FOR LIST ITEM EXTRACTION AND THEIR INTERPRETATION

No.	Expression	Interpretation
1	(?<=[#\s][\d] [\)\.\:] [\s])(.*?)(?=[#] [\d][\)\.\:][\s])	Identifies if the tweet contains one or multiple items
2	[#\s](\d|ix|iv|v?i{0,3}) [\)\.\:\,\-] [\s]	Separates individual lists from the tweet set
3	(#[0-9A-Za-z_] + |@[0-9A-Za-z_] + |[\d]{10}[|]{3}|RT|\s* \:\s*|[\d][\.\)]\s*)\Z	Removes all the unnecessary hashtags, RT, mentions from the end of the input tweet
4	\A(#[0-9A-Za-z_]+|@[0-9A-Za-z_]+ |[\d]{10}[|]{3}|RT| \s*\:\s*|[\d][\.\)]\s*)	Removes all the unnecessary hashtags, RT, mentions from the start of the input tweet

TABLE IV
EXAMPLE TWEETS CONTAINING OBJECTIVE AND SUBJECTIVE LIST ITEMS

No.	Type	Example Tweet
1	Objective	@Username1 #3thingsthatmakeyousmile #1) livingleondre #2) Leondre and Charlie #3) chocolate. please notice me it's my birthday
2	Subjective	He liked food. She liked Instagram. together, they posted annoying food pictures. @WeAreMumbai #MumbaiLoveStories
3	Subjective	RT @Username2: Remember when I almost fought haji @Username3 @Username4 @Username5 #8thgrademomories

We apply this classifier on the top frequent 0.1 M hashtags in our data set to get 0, 4, 47, 601, and 6993 OL-hashtags from the top frequent 10, 10^2, 10^3, 10^4, and 10^5 hashtags, respectively.

V. EXTRACTION OF LIST ITEMS

After identifying the OL-hashtags, we extract candidate list items from the tweets containing these hashtags. Besides extracting from tweets, we also extract list items from URLs mentioned in these tweets.

A. Extracting List Items From Tweets

Identifying list items from tweets is quite challenging because there is no unique structure in the way the list items are mentioned. We, therefore, carefully observed the patterns of list items in the tweets, and came up with various regular expressions (regex) as listed in Table III. We explain our algorithm using three tweets shown in Table IV.

Our algorithm to extract list items from each tweet depends on matching the tweet text with regular expression patterns. The algorithm works as follows.

First, we match each tweet to Regex-1 to find out if the tweet contains multiple list items, in which case this expression gives two matches, otherwise, the list does not return any match. If there are at least two matches, we consider it as an objective list based on our observation that objective list items generally occur in multiples in a single tweet, otherwise, it is considered as a subjective list. Thus, using Regex-1, we mark Tweet-1 in Table IV as objective (two matches) and Tweet-2 and Tweet-3 as subjective (0 matches each). These are further processed using the other regexes as follows.

For objective lists: Using Regex-2 on Tweet-1, we split the list into meaningful units after basic noise removal. The following noise removal steps are performed.

1) The first part, which contains mention and hashtags, is ignored as it is the part before the list starts.
2) The last element in the tweet might contain some trailing unwanted text which we remove by recognizing the emoticons, the URLs and hashtags.
3) We clean each answer by removing unnecessary padding spaces and hex code.

For Tweet-1, the final output consists of three list items: 1) livingleondre; 2) Liondre and Charlie; and 3) chocolate.

For subjective lists: Regex-3 and Regex-4 remove all the unnecessary hashtags, Retweet Delimiters ("RT"—as shown in the third example of Table IV), mention at the beginning and the end of the tweet, respectively. Subsequently, unnecessary non-ASCII characters and words are removed, and the remaining text is provided as the final answer. The final extraction is as follows.

Tweet-2: "He liked food. She liked Instagram. together, they posted annoying food pictures."

Tweet-3: "Remember when I almost fought haji."

TABLE V
PERCENTAGE DISTRIBUTION OF VARIOUS TYPES OF URLS

URLs/Tweet - 45.45					
Working Urls - 72.58					NW
Social - 62.41			Shopping	General	27.42
Text	Pic	PicText	5.25	32.34	
19.23	6.47	74.3			

B. Extracting List Items From URLs

In our data set, 45.45% tweets contain a URL of which around 72.58% were still active. Active URLs are of three main types as follows.

1) Social Network Pages: Mainly Twitter pages fall in this category. 62.41% of active URLs are social pages. Furthermore, these pages are of three types: 1) containing only text (19.23%); 2) containing only picture (6.47%); and 3) containing both text and picture (74.3%). We did not process the pictures and focus only on text.

2) Shopping Website Pages: These are e-commerce webpages and form 5.25% of active URLs.

3) General Webpages: 32.34% of the active URLs are neither social nor shopping pages. The statistical distribution of URLs in our data set is shown in Table V. Examples of different URLs are shown in Fig. 8.

If the tweet contains a URL, our algorithm tries to extract list items from the corresponding webpage content. If the URL is a social network page, it uses the method mentioned in

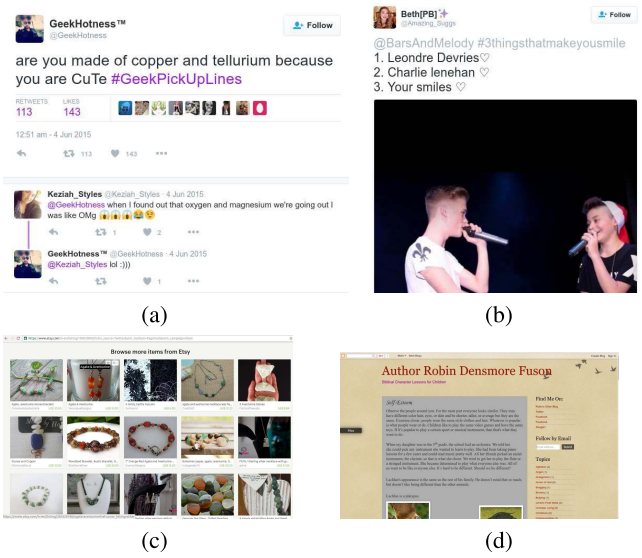


Fig. 8. Examples of different types of URLs. (a) Only text. (b) Picture and text. (c) Shopping URL. (d) General URL.

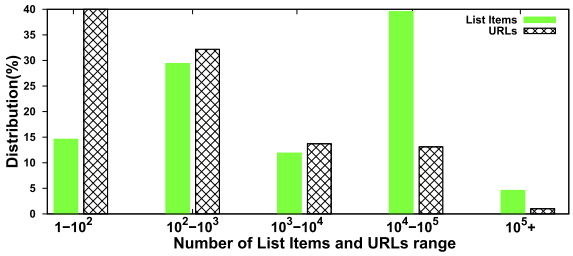


Fig. 9. Statistics of list items and URLs.

Section V-A to process the tweet text and extract list items. If the URL is from a shopping website or a general webpage, it uses a generalized version of the top-k list extraction algorithm from webpages [30].

Given our data set, we were able to collect ∼1.4M list items. The distribution of list items and URLs across all the OL-hashtags is shown in Fig. 9 where the x-axis denotes various ranges and the y-axis denotes the percentage of hashtags, for which that many list items/URLs are found.

C. Evaluation

We perform a crowd-sourced driven quality evaluation of a random sample of 20 extracted list items for a random sample of 58 OL-hashtags using Amazon Mechanical Turk (AMT). The AMT workers were provided with detailed guidelines and examples and were asked to judge the quality of the extracted list items for a given OL-hashtag using one of the three options: A) answer contains the list item and the list item is considered to be good with respect to the OL-hashtag; B) answer contains the list item but the list item is not good; and C) answer is bad, i.e., unrelated to the OL-hashtag.

Each list item was evaluated by three different AMT workers. We compute two metrics as follows.

1) *Weak Accuracy:* For each list item, we choose answers "A" and "B" as positive and "C" as negative.
2) *Strong Accuracy:* For each list item, we choose only "A" as positive and "B" and "C" as negative.

We found that the average weak and strong accuracy values for randomly collected 10 items are 66.12% (weak@10) [for objective list 54.3% and subjective list 74.1%], and 49.1% (strong@10) [for objective list 39.3% and subjective list 56.1%]. The result shows that the accuracy is decent with every one of two list items being accurate. In order to bubble up to the relevant list items from this set, we develop a learning to the rank framework in Section VI.

VI. Ranking the List Items (L2R)

Although there exist multiple papers on ranking tweets [33], [34], but none of them ranks list items extracted from tweets. In this section, we consider several features of the list items and relate them to an importance score. To generate distant supervision data for training, the importance score is semiautomatically derived using search engines. Subsequently, we execute a supervised learning algorithm to rank the list items.

A. Feature Extraction

We use the following features for ranking the list items.

1) *Frequency (freq)*: Number of times the list item appears in tweets. The intuition is that a good list item would occur in many different tweets.
2) *Average Follower Count (foll)*: For the list item, we collect follower counts of all users who posted tweets containing this item and use the average follower count as a feature. The intuition here is that more influential users might post better list items.
3) *Timestamp (time)*: We use the latest timestamp of posting the list item as a feature, with the intuition that the newer list items might be more relevant.
4) *Cooccurring Items Count (cooc)*: We use a number of cooccurring items across all the tweets, with the intuition that good list items generally occur with many other list items, especially for objective list items.
5) *PageRank Scores (page)*: For each list, we first construct a graph where each list item is a node. We construct separate graphs for objective and subjective lists. We then use the PageRank scores of the list items with the intuition that good list item will be related to many other good items in this graph. We explain the graph construction process in detail in the following.

1) Graph Construction for PageRank Score Computation: We construct the graph differently depending on whether the list items are subjective or objective.

a) Cooccurrence graph for objective lists: We observe that objective list items generally occur together in the same tweet. Therefore, we construct a graph for all the list items such that if two items cooccur in a tweet, then an edge exists between them and the edge weight depends on the number of tweets they cooccur in, normalized on a logarithmic scale. For example, if "smile" and "chocolate" both appear in four different tweets, then frequency $n = 4$. Similarly, we collect

TABLE VI

RANKING LIST ITEMS: 10-FOLD CROSS VALIDATION RESULTS OF CC, MAE, AND RMSE FOR DIFFERENT REGRESSION METHODS—LIN-REG, SVM, DT, AND BG

Model	Lin-Reg	SVM	DT	Bagging
CC	0.140	0.150	0.210	**0.460**
MAE	0.183	0.177	0.181	**0.161**
RMSE	0.238	0.243	0.235	**0.213**

all the possible frequencies and let N denote the sum of all the frequencies. The normalized edge weight is given by $\log(n+1)/\log(N)$.

b) Similarity graph for subjective lists: We observe that subjective list items do not cooccur, however, we find that good items have some similarity on the surface level. Therefore, we construct the graph for all the subjective list items, and an edge exists between two nodes if cosine similarity (CS) between them is above a predefined threshold (we choose positive similarities) value after removing stop words.

We combine the above-mentioned features using a regression-based learning to rank framework. The learning to rank framework requires some labeling of comparisons or relevance scores to the set of items. To avoid manual labeling, we use the following observation to provide the labeled data set for learning to rank in a semiautomated manner.

B. Semiautomated Ranking Using Search Results

We observe that while for most of the list items, search engines do not provide good results, for some, the search engines provide good webpages, which contain the answers. We select 100 random OL-hashtags, for which a reasonable number of results come up, and collect top 10 results using both Google and Bing APIs. After implementing the generalized version of the top-k list extraction algorithm [30], we collect list items from these webpages.[8] Furthermore, we manually check each list item and remove a few incorrect results (those coming from ads or other parts of the webpages, not containing the list items), or added missing list items. Score for a list item l is then computed as $\text{score}(l) = \sum_{i=1}^{10}(11-i) \times f(i)$ where $f(i)$ denotes the frequency of list item l on webpage at position i.[9] We normalize this score using max-normalization and use it as a label to train the learning to rank model using the five features. Thus, we approximate the learning to the rank problem using a pointwise approach. Table VI compares various regression mechanisms such as SVM, Linear Regression (Lin-Reg), Decision Tree (DT), and Bagging (Bg) using 10-fold cross validation using three metrics: correlation coefficient (CC), mean absolute error (MAE), and root mean squared error (RMSE). Bg performs the best. After training the model on 100 OL-hashtags, we use it to rank the list items extracted for the rest (901) of the OL-hashtags.

[8]The method proposed in [30] works only for specific types of webpages; we generalized it to collect list items from more webpage types.

[9]The weight functions are linear—for both Google and Bing search engines. If we get the list item from the first website, then the weight is put to 10, and it is decreased linearly for further positions. However, the same item may be mentioned in multiple webpages and also multiple times on the same webpage. We, therefore, provide a weight to this item by multiplying the frequency of occurrence in a given webpage with the weight of the webpage, and summing it over all the webpages. For example, for the OL-hashtag "#giftideas," we get "ring" as a list item four times (once in first and third page, and twice in the fourth webpage), wedding ring (three times—once in the first and twice in the sixth webpage), birthday ring (two times—once in the second and tenth webpage). Now, after clustering, the frequency of ring is 9 $(4+3+2)$, wedding ring is 3, birthday ring is 2, and aggregated weight of ring is 62 $(1\times10+1\times8+2\times7+1\times10+2\times5+1\times9+1\times1)$, wedding ring is 20 $(1\times10+2\times5)$, and birthday ring is 10 $(1\times9+1\times1)$.

TABLE VII

RANKING LIST ITEMS: INDIVIDUAL FEATURE IMPORTANCE AND FEATURE ABLATION RESULTS

Method	Weak@10	Weak@20	Strong@10	Strong@20
Random	65.14	64.10	50.24	49.76
L2R All	**92.14**	**91.45**	**79.23**	**77.12**
L2R All-page	86.12	83.54	73.48	72.87
L2R All-foll	84.32	82.71	72.23	70.96
L2R All-cooc	83.56	80.65	70.75	68.86
L2R All-freq	80.63	78.42	68.45	67.31
L2R All-time	79.56	77.23	67.31	66.52
time	70.65	70.61	61.92	59.67
freq	70.64	68.50	60.30	58.23
cooc	68.27	67.47	59.25	57.54
foll	65.35	64.76	55.23	53.61
page	64.12	62.32	53.72	51.42

C. Evaluation

We perform a crowd-sourced evaluation through AMT for top 20 list items for randomly selected 120 OL-hashtags from the remaining 901 OL-hashtags using three different annotators. 50 of these 120 lists overlap with the ones labeled for random extraction evaluation (Section V). Annotators mark each list item as "A," "B," or "C."[10] The evaluation shows that we obtain a high strong accuracy of *79.23%* for Prec@10 [for objective list 70.3% and subjective list 85.5%] and *77.12%* for Prec@20; and very high weak accuracy of *92.14%* for Prec@10 [for objective list 88.5% and subjective list 94.6%] and *91.45%* for Prec@20. This is much better compared to the accuracy of randomly chosen 20 list items before ranking (Section V).

Furthermore, to evaluate feature importance, we learn regression models using individual features and also perform feature ablation experiments. The AMT results are shown in Table VII where it is easily seen that using learning to rank (L2R) framework helps in improving the performance by a huge margin. Note that for the "Random" method, the average is computed over 50 overlapping lists but for other methods, we use 120 OL-hashtags. Clearly, all the five features are important; leaving any of the features reduces the performance by a good margin.

VII. AUGMENTATION OF TAIL OPINION LISTS

Several OL-hashtags produce just a few number of list items because: 1) the hashtag is contained in small number of tweets; 2) number of *English* tweets containing the hashtag are few; or 3) items repeat across tweets. We define them as "tail" OL-hashtags (producing less than 20 list items). Fig. 10(b) shows the cumulative distribution of OL-hashtags

[10]Interannotator agreement Fleiss κ is 0.61

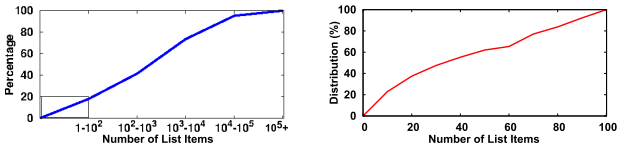


Fig. 10. Distribution of list items produced by (a) all OL-hashtags (left) and (b) hashtags with #List Items<100 (right).

containing a number of list items between 0 and 100. The x-axis denotes the number of list items and y-axis denotes cumulative percentages of lists with that many list items. Note that 23% lists have list items in the range 0–100 [Fig. 10(a)] and among them 38% are in range 0–20, i.e., tail OL-hashtags as shown in Fig. 10(b). The idea is to enrich tail lists by borrowing items from similar lists.

Some examples of "tail" OL-hashtags are: "moviesthatmakeyoucrysobad," "whydopeoplehateairlines," having 3 and 7 list items only. For the first hashtag, there are other similar hashtags, e.g., "moviesthatmakemencry," "moviesthatmadeyoucry" with 79 and 15 list items, respectively. Tail OLs can be augmented by borrowing list items from similar lists. We hypothesize that *two lists are similar if they denote a similar concept and contain similar list items.* Based on this hypothesis, we identify features to measure the similarity between the names of the hashtag pair as well as the similarity between the list item set corresponding to the pair. Furthermore, we use these similarity measures as features to build a model for learning similarity value for any pair of OL-hashtags. Finally, we borrow items from the similar list set to augment tail lists and verify the accuracy of augmentation.

A. Similarity Measures

1) Hashtag–Hashtag Similarity: We segment the OL-hashtags into constituent words and then compute three different similarity values described as follows.

1) CS: This captures CS between term frequency-inverse document frequency (TFIDF) vectors for the two OL-hashtags.
2) Concept-Based Similarity: Using CMU POS tagger [32], we retain only nouns, verbs, and adjectives for both the list hashtags. Next, we compute CS between the 300-D word2vec [35] vector for every word pair containing a word from each of the two lists. For every word in the first list, we find the best matching word in the second list with the highest similarity. Finally, concept-based similarity is defined as the average CS across matched word pairs. Thus, e.g., although the CS between "breakupin4words" and "breakuplines" is low, concept-based similarity is very high.
3) Pattern-Based Similarity: We manually created a dictionary of list patterns. This similarity is set to 1 if both lists contain one of the dictionary patterns, else 0. For example, "describethe90sin4words" and "explainthe90sin4words" contain the same pattern "in4words." Since "in4words" is in the dictionary, the similarity is set to 1.

2) Item–Item Similarity: We consider a list hashtag pair as a candidate only if their CS value is above a threshold (we fix it to 40%). Below this threshold, the similarity is considered to be zero. This ensures that we deal with only promising candidates making the computation efficient. We create a list item pair for comparison by taking the first item from list A and second from list B such that (A, B) is a candidate list pair. List items could be objective or subjective. While the syntactic match is good for objective items, subjective items need a semantic match. Hence, we compute the following similarity values.

1) Item–Item CS: After removing the stop words, we calculate CS between the list items.
2) DSSM/CDSSM Score: We use deep semantic similarity model (DSSM) and convolutional DSSM (CDSSM) [36] to compute semantic match between list items.

TABLE VIII

ITEM–ITEM SIMILARITY: 10-FOLD CROSS VALIDATION RESULTS FOR DIFFERENT REGRESSION METHODS

Model	Lin-Reg	SVM	DT	Bagging	RF
CC	0.800	0.794	0.833	**0.850**	0.829
MAE	0.237	0.238	0.152	**0.135**	0.136
RMSE	0.302	0.305	0.277	**0.264**	0.281

TABLE IX

HASHTAG–HASHTAG SIMILARITY: 10-FOLD CROSS-VALIDATION RESULTS FOR DIFFERENT REGRESSION METHODS

Model	Lin-Reg	SVM	DT	Bagging	RF
CC	0.863	0.856	0.925	0.933	**0.942**
MAE	0.209	0.198	0.07	0.059	**0.045**
RMSE	0.252	0.264	0.189	0.179	**0.169**

B. Models for Learning List Similarity

We learn two regression models at two levels to weigh different similarity measures and come up with a combined similarity measure between OL-hashtags. The first model tries to learn a combined measure for item–item similarity while the second model incorporates hashtag–hashtag similarity signals along with the output of the first model to learn a combined similarity measure for hashtag–hashtag similarity.

To learn the *item–item similarity model*, 250 matching item–item pairs and 250 dissimilar pairs were identified by annotators (different from the authors). We then experimented with Weka [37] implementations of various regression models: Lin-Reg, SVM, DTs, Bg, and Random Forest (RF) with the three item–item similarity measures (cosine, DSSM, and CDSSM) as features. Table VIII shows that the 10-fold cross-validation results in terms of several standard measures like CC, MAE, and RMSE. Note that Bagging provides the best CC and least MAE and RMSE, so we use Bagging to generate the item–item combined similarity score for the rest of the data set.

To learn the *hashtag–hashtag similarity* model, 150 matching hashtag–hashtag pairs and 150 dissimilar pairs were identified by annotators (different from the authors). To run the regression model, we used the three hashtag–hashtag similarity values (cosine, concept, and pattern based), and the aggregated item-based hashtag–hashtag similarity value as

features. Table IX shows the 10-fold cross-validation results in terms of CC, MAE, and RMSE of various regression models: Lin-Reg, SVM, DTs, Bg, and RF. The RF regression model provides the best CC and least MAE and RMSE, hence, that is used subsequently. Feature importance analysis using the ReliefAttribute Evaluator with Ranker-based search method shows the following ranking of the features: item–item similarity score, concept-based similarity score, CS score, and pattern-based similarity score. It is interesting to note that the underlying list items provide the best measure for establishing similarity between two list hashtags.

Once the hashtag–hashtag similarity measure is obtained, we perform clustering [38] to generate hashtag clusters. For our data set of 1001 list hashtags, we obtain 134 clusters covering 791 hashtags; 210 hashtags remain isolated. For example, "primaryschoolmemories," "secondaryschoolmemories," "teenagememories," "middleschoolmemories," and "childhoodmemories" form a single cluster. Another cluster includes hashtags like "sophomoreyearin5words," "senioryearin5words," "junioryearin5words," and "freshmanyearin5words." However, some hashtags such as "georgiastatejokes," "4favoritevillains," and so on remain isolated.

After clustering, we can augment the tail lists by borrowing "good" list items from other lists in the same cluster. Let i be a borrowed list item into a tail list l. Let L be the set of lists in the same cluster as l. Rank score of i for l is defined as a normalized weighted sum of rank scores (as computed in Section VI) of i in other lists in L where weight corresponds to the combined hashtag–hashtag similarity between l and other list $l' \in L - \{l\}$.

C. Evaluation

To evaluate the accuracy of augmentation of tail OLs, we perform an automated and a crowd sourced-based evaluation.

In automated evaluation, we randomly select k elements of list items from a tail list set and check what fraction of them can be recovered from the item sets within its cluster. We only consider the top-k ranked elements from each item set. The intuition is that a high recovery would mean that the list sets within a cluster are similar, and consequently, it can be assumed that augmentation of the tail list would be done with relevant lists. We find that the accuracy of augmentation is 69.2% and 54.7% list items after removal of 10% and 20% items, respectively, from "tail" OL-hashtags containing only 1–20 list items. For example, the list "moviesthatmakeyoucrysobad" contains only three list items ("Miracle in Cell No. 7," "The Dark Knight," and "Shawshank Redemption") but other similar lists contribute items such as "The Pianist," "Schindlers List," "127 Hours," and so on.

For crowd-sourced experiment, we augmented 30 tail OL-hashtags, each with 20 list items. Furthermore, using AMT, the augmented list items are labeled as "A," "B," or "C." We found that the weak and strong accuracies @20 are 86.56% and 73.81%, respectively (which is slightly lower than what we achieve for normal OL-hashtags), showing the effectiveness of our augmentation approach.

VIII. Error Analysis

In this section, we discuss detailed error analysis of different parts of our proposed framework to gain more insights into the difficult cases.

A. OL-Hashtag Detection

Most OL-hashtags contain a plural noun in them. For instance, the OL-hashtags, "girlfriendsbelike," "mexicanmomsbelike," and "90svideos." Hence, the feature "presence of plural" has been one of the distinguishing features in classification. However, some OL-hashtags such as "crazygirlfriendbelike" do not contain any plural noun and, hence, do not get detected. OL-hashtags of a particular pattern "ifyou," e.g., "ifyouaremygirl," "ifyouknowmewell" also do not get detected because of their similarity with many non-OL-hashtags such as "ifnotformusic," "ifyouneed," "gotofftwitterif," and so on. Finally, abstract OL-hashtags such as "getabetterwatch" which expect very subjective list items, suffer from very sparse feature vector representation and, hence, get labeled as non-OLs incorrectly.

B. Extraction of List Items

Our extraction algorithm leads to errors in the following cases.

1) Collecting list items containing exactly k words, e.g., from the tweet "#3bestwordsever thanks a lot...after getting help," we get "thanks a lot after getting help" as the list item but the correct answer should be "thanks a lot" (three words). The patterns should be able to make use of this information in the hashtag.
2) In some cases, while removing unnecessary mentions and hashtags, a sentence may become incomplete. For example, from the tweet "#bigEconomicEvents2016 @Modi declares demonetization," removal of hashtags and mentions results in a partial answer, "declares demonetization."

C. Ranking the List Items (L2R)

We analyzed some of the list items with high scores but labeled as "C" by all the AMT workers. In all these cases, we found that the underlying tweets are not exactly on the hashtag but related and since it has been highly retweeted or has been tweeted by highly popular people, it scored high. For example, for "#breakfastofchampion," list item "GAME DAY http://t.cogQykUKE5Y7" received a lot of retweets. Similarly, for "#describeyourcrush," list item "Tweet me with the hashtag and I'll retweet. Gooo" has come within top 10 due to its high average follower count.

D. Augmentation of Tail Lists

We found that once good clusters are formed, augmentation is usually good. However, there are two problems: 1) among total 98 "tail" OL-hashtags, 25 remained isolated, hence, their performance cannot be improved through this scheme and 2) some clusters are of low quality. Two lists could get incorrectly clustered together if there are a lot of matching

words in the hashtags (but a keyword mismatches), and the two lists also share a lot of list items but are conceptually different. For example, "myteamsyearin5words" and "mylastwordsin5words" get incorrectly assigned to the same cluster as "sophomoreyearin5words," "senioryearin5words," "junioryearin5words," and "freshmanyearin5words."

IX. Discussion

A. Categorization of Opinion Lists

For each of the 1001 opinionated hashtag that we manually identified, we did further classification across multiple perspectives as follows.

Some OLs are closed while others are not. We define a list as closed if it has list items from a finite set. For example, "worldsbiggestbookclub," "placestovisitinindia," and "20favouritefootballers" are all closed lists. On the other hand, lists such as "lovestoryin5words," "thingsthatmadeyoucoolatschool," "8thgrademomories," and "iftwitterdidntexistanymore" are not closed lists. Among the 1001 OLs, we found that only 242 were closed while people expressed themselves socially in the other 759 lists. This showcases the social nature of the lists extracted from Twitter which is very different from lists extracted from general webpages or traditional lists.

Furthermore, lists could be classified as ambiguous versus nonambiguous. Ambiguous lists are those where the list name does not restrict the type of items in the list leading to a possibility of different users focusing on list items of different types. An example of an ambiguous list is "23goldenyearsofsrk." This list could contain names of movies in which Shah Rukh Khan (SRK) has worked, or it could also contain names of awards he has won, or it could describe his acting style. An example of nonambiguous list is "bestsrkmovies." We observed that our 1001 list data set contains 101 ambiguous and 900 nonambiguous lists. To classify ambiguous hashtags from normal hashtags, we identify various features. General features: 1) POS tagging features for the hashtag: the presence of nouns, adjectives, verbs, pronouns, plural, prepositions, and adverbs. 2) Tweet features: different tweet features—a) ngram represents presence of n hashtags in that tweet including the OL-hashtag where values of n are from 1 to 6 denoting unigram, bigram, trigram, tetragram, pentagram, and hexagram; b) number of tweets; and c) time difference and time span. We experimented with various classifiers—NB, LR, SVM, repeated incremental and pruning (Rip), AdaBoost(AB), RF, and Bg as shown in Table X. Bg produced the best results among all classifiers in terms of precision, recall, accuracy, and area under receiver operating characteristic (ROC) (AU-ROC). By ranking the features using IG, we found the following as the most important features—presence of plurals (0.109), presence of adjectives (0.1), presence of pronouns (0.071), bigram (0.044), time difference (0.43), and presence of nouns (0.04). Furthermore, tweets corresponding to individual subtopics of the ambiguous OL-hashtag can be assigned to individual subtopics using topic modeling. For example "bestofsrk" subtopics for the ambiguous OL-hashtag can be award names, best movie names, best song names, pictures, best movie dialogues, and so on. Using Named Entity Tagger [39] and modified search features (e.g., presence of words like "lyrics" or "songs" along with the searched string), one can identify the topic of the ambiguous OL-Hashtag's tweet text. We plan to include this feature in our future model.

TABLE X
Ambiguous OL-Hashtag Detection: 10-Fold Cross-Validation Results of Precision (P), Recall (R), Accuracy (A), and AU-ROC for Various Classifiers

Classifier	P	R	A	AU-ROC
NB	80.7	81.2	81.2	72.2
LR	93.1	93.6	93.60	80.6
SVM	69.1	82.7	82.68	49.8
Rip	85.9	86.8	86.82	62.6
AB	84.9	84.6	84.65	76.1
RF	93.1	92.4	92.42	87.6
Bg	**93.9**	**93.7**	**93.63**	**87.8**

Finally, OLs could also be categorized as subjective or objective. Our data set contained 589 subjective and 412 objective lists.

B. Temporal Aspect of Opinion Lists

While a large number of lists are invariant with respect to time, some are temporal. Some of them are explicitly temporal such as "heroineoftheyear." Some other lists are implicitly temporal because users usually discuss recent candidate items. For example, for the list "bestplayerintheworld," users tend to talk about recent players which are fresh in their memory. In case of Football, users tweeted more about "Manuel Neuer" in January 2015, about "Lionel Messi" in first week of May 2015, and about "Christiano Ronaldo" in the fourth week of May 2015 events. Similarly, in 2017, users who tweeted about tennis tweeted a lot about "Nadal" during French Open 2017 (May–June 2017) and about "Roger Federer" during Wimbledon 2017 (July 2017). Another such example is "bestmobileplans." We found that in April 2015, users tweeted about Google and Next Helsinki. While in May 2015, Carphone Warehouse and Tata Docomo were more popular. Thus, Twitter-based OLs can help us to get not only just the popular list items right now but also popular list items across time.

Time-based features on tweets have low IG and low OAE accuracy for most of the OL-hashtags such as "breakupin4words," "highschoolmemories," "adviceforyoungjournalists," and so on but time-based features are important for some OL-hashtags, which show temporal variations with respect to time. Therefore, based on different time features, we can classify OL-hashtags as volatile (OL-hashtags showing temporal variations over time) or nonvolatile (OL-hashtags which do not exhibit temporal variations over time).

C. Instant Answers

Traditional search engines do not always provide instant answers for opinionated queries. We have developed a system which provides ranked answers for queries corresponding to OL-hashtags. For example, opinionated query like "8th grade memories" is similar to the "#8thgrademories" and

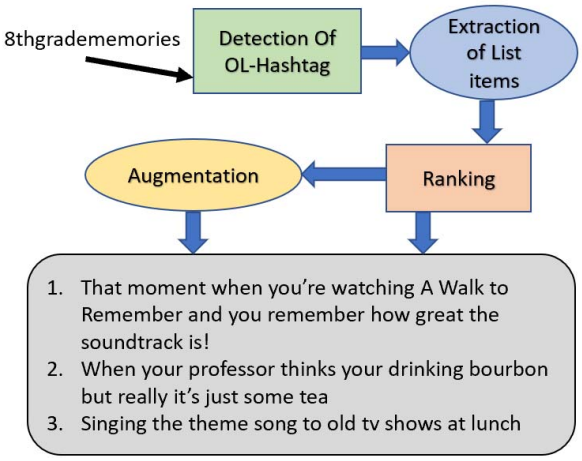


Fig. 11. Generating answers for opinionated queries.

respective tweets are shown in Fig. 4. When this query is input to our system, after detecting it as an OL-hashtag, different list items are extracted. Thereafter, ranking of list items and augmentation with other list items of similar OL-hashtags (e.g., #primaryschoolmemories, #secondaryschoolmemories, and so on) are done to get the ranked answers instantly as shown in Fig. 11. We plan to include other social media data including Reddit, Quora, Facebook, and so on and scale up the system to allow large-scale OL finding. We also plan to publish this database of OL items extracted from various social media and provide an interactive Web service.

X. Conclusion

It is generally believed that Twitter is a treasure trove of opinionated phrases—the main contribution of this paper lies in exploiting that to lay foundation for an efficient search system. We identified that there is a special category of hashtags, called "OL-hashtags" (e.g., #TipsforInteriorDesign), which can be leveraged to collect relevant OL answers (e.g., "Hang artwork at the right height," "Pick the paint color last," and so on). The indexed answers can then be used to return results for a search query. This is a challenging task as a very small percent of hashtags in the Twitter pool are OL-hashtags. We proposed multiple hashtag- and tweet-level features and learned an LR model that provides 75.5% precision at 95.3% recall. Furthermore, we studied various patterns through which the list items for these OL-hashtags are reported and formulated regular expressions to capture the social items. However, there is no uniform way in which Twitter users provide these list items, and hence, extraction led to a modest accuracy of only 66.12%. The precision is enhanced by applying a learning to rank framework whereby the Precision@10 achieved is 91.31% and Precision@20 is 90.72%.

While working with the data, we make several interesting observations. We find that the URLs present in a tweet corresponding to OL-hashtags are mainly pointing to either social network pages or shopping website pages which perhaps also provide the cue about the type of queries this indexing scheme would cater to. In case of names of OL-hashtags, we interestingly observe that the presence of plural word is an important discriminatory feature to identify OL-hashtags (#5wordsbeforebreakup). We also observe that corresponding to several OL-hashtags, there are very few tweets and, hence, very few OL items. We realize that there are also similar hashtags, the item set of which can be borrowed to populate the sparse list. We develop an augmentation scheme to implement that.

There are, however, several areas for improvement; we have noted some of the limitations of the process in the *error analysis* section. Besides this, the importance of several of the list items may depend on time—hence, considering the temporal aspect in ranking them is necessary. Also, hashtags can be ambiguous leading to the possibility of different communities focusing on producing tweets of different types. For example, the list "23goldenyearsofsrk" could contain names of movies in which SRK, a famous Indian actor, has worked, or it could also contain names of awards he has won, or it could describe his acting style. All these observations and limitations would be addressed in the near future. Furthermore, the current system is based only on Twitter data.

Nowadays, hashtags are being commonly used across all the social platforms—Facebook, Instagram, Tumblr, Pinterest, and so on. Therefore, a generic system can be developed based on the use of hashtags in these social media platforms. Some other social media platforms such as Quora, Reddit, and Huffington Post also exhibit similar features but on such platforms, questions, and respective answers are present instead of hashtags and tweets. For example, in Quora, opinionated questions are like OL-hashtags and answers are like tweets. While OL-hashtags are short but opinionated, Quora questions are normally long. For example, corresponding to the OL-hashtag "weekendplans," there are different Quora questions such as "What are your weekend plans to relax?," "What are the best things to do on a weekend?," and so on. Thus, one of our future plans is to extend the current system to support other social media data including Reddit, Quora, Facebook, Pinterest, and so on including the pictorial representations and scale up the system to large-scale OL finding. Another plan is to develop a runtime Web API to produce instant top-k answers for opinionated search queries.

References

[1] M. R. Morris, J. Teevan, and K. Panovich, "What do people ask their social networks, and why? A survey study of status message Q&A behavior," in *Proc. SIGCHI*, 2010, pp. 1739–1748.

[2] A. Mullick, P. Goyal, N. Ganguly, and M. Gupta, "Extracting social lists from Twitter," in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining*, 2017, pp. 391–394.

[3] D. M. Romero, B. Meeder, and J. Kleinberg, "Differences in the mechanics of information diffusion across topics: Idioms, political hashtags, and complex contagion on Twitter," in *Proc. WWW*, 2011, pp. 695–704.

[4] K. Lee, D. Palsetia, R. Narayanan, M. M. A. Patwary, A. Agrawal, and A. Choudhary, "Twitter trending topic classification," in *Proc. ICDM Workshops*, 2011, pp. 251–258.

[5] M. Naaman, H. Becker, and L. Gravano, "Hip and trendy: Characterizing emerging trends on Twitter," *J. Assoc. Inf. Sci. Technol.*, vol. 62, no. 5, pp. 902–918, 2011.

[6] P. Bhattacharya *et al.*, "Deep Twitter diving: Exploring topical groups in microblogs at scale," in *Proc. 17th ACM Conf. Comput. Supported Cooperat. Work Social Comput.*, 2014, pp. 197–210.

[7] A. Zubiaga, D. Spina, R. Martínez, and V. Fresno, "Real-time classification of Twitter trends," *J. Assoc. Inf. Sci. Technol.*, vol. 66, no. 3, pp. 462–473, 2015.

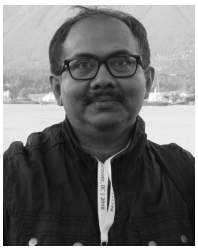
[8] O. Tsur and A. Rappoport, "What's in a hashtag?: Content based prediction of the spread of ideas in microblogging communities," in *Proc. 5th ACM Intl. Conf. Web Search Data Mining*, 2012, pp. 643–652.
[9] S. K. Maity, A. Gupta, P. Goyal, and A. Mukherjee, "A stratified learning approach for predicting the popularity of Twitter idioms," in *Proc. ICWSM*, 2015, pp. 642–645.
[10] K. Rudra, A. Chakraborty, M. Sethi, S. Das, N. Ganguly, and S. Ghosh, "# FewThingsAboutIdioms: Understanding idioms and its users in the Twitter online social network," in *Proc. PAKDD*, 2015, pp. 108–121.
[11] S.-M. Kim and E. Hovy, "Extracting opinions, opinion holders, and topics expressed in online news media text," in *Proc. Workshop Sentiment Subjectivity Text (ACL)*, 2006, pp. 1–8.
[12] A. Qadir, "Detecting opinion sentences specific to product features in customer reviews using typed dependency relations," in *Proc. eETTs*, 2009, pp. 38–43.
[13] T. Scholz and S. Conrad, "Opinion mining in newspaper articles by entropy-based word connections," in *Proc. EMNLP*, 2013, pp. 1828–1839.
[14] H. Yu and V. Hatzivassiloglou, "Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences," in *Proc. EMNLP*, 2003, pp. 129–136.
[15] J. Wiebe and E. Riloff, "Creating subjective and objective sentence classifiers from unannotated texts," in *Proc. CICLingLing*. Berlin, Germany: Springer, 2005, pp. 486–497.
[16] N. Asher, F. Benamara, and Y. Y. Mathieu, "Appraisal of opinion expressions in discourse," *Lingvisticæ Invest.*, vol. 32, no. 2, pp. 279–292, 2009.
[17] P. Rajkumar, S. Desai, N. Ganguly, and P. Goyal, "A novel two-stage framework for extracting opinionated sentences from news articles," in *Proc. TextGraphs*, 2014, pp. 25–33.
[18] A. Mullick, P. Goyal, and N. Ganguly, "A graphical framework to detect and categorize diverse opinions from online news," in *Proc. Workshop Comput. Modeling People's Opinions, Personality, Emotions Social Media (PEOPLES)*, 2016, pp. 40–49.
[19] A. Mullick *et al.*, "A generic opinion-fact classifier with application in understanding opinionatedness in various news section," in *Proc. 26th Int. Conf. World Wide Web Companion*, 2017, pp. 827–828.
[20] A. Mullick *et al.*, "Identifying opinion and fact subcategories from the social Web," in *Proc. ACM Conf. Supporting Groupwork*, 2018, pp. 145–149.
[21] B. Liu, R. Grossman, and Y. Zhai, "Mining data records in Web pages," in *Proc. KDD*, 2003, pp. 601–606.
[22] W. Gatterbauer, P. Bohunsky, M. Herzog, B. Krüpl, and B. Pollak, "Towards domain-independent information extraction from Web tables," in *Proc. WWW*, 2007, pp. 71–80.
[23] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang, "WebTables: Exploring the power of tables on the Web," *Proc. VLDB Endowment*, vol. 1, no. 1, pp. 538–549, 2008.
[24] G. Miao, J. Tatemura, W.-P. Hsiung, A. Sawires, and L. E. Moser, "Extracting data records from the Web using tag path clustering," in *Proc. WWW*, 2009, pp. 981–990
[25] F. Fumarola, T. Weninger, R. Barber, D. Malerba, and J. Han, "Extracting general lists from Web documents: A hybrid approach," in *Proc. Int. Conf. Ind., Eng. Appl. Appl. Intell. Syst.*, 2011, pp. 285–294.
[26] F. Fumarola, T. Weninger, R. Barber, D. Malerba, and J. Han, "HyLiEn: A hybrid approach to general list extraction on the Web," in *Proc. ACM 20th Int. Conf. Companion World Wide Web*, 2011, pp. 35–36.
[27] E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne, "Finding high-quality content in social media," in *Proc. WSDM*, 2008, pp. 183–194.
[28] D. Rafiei and H. Li, "Data extraction from the Web using wild card queries," in *Proc. CIKM*, 2009, pp. 1939–1942.
[29] D. Z. Wang, M. J. Franklin, M. Garofalakis, and J. M. Hellerstein, "Querying probabilistic information extraction," *Proc. VLDB Endowment*, vol. 3, nos. 1–2, pp. 1057–1067, 2010.
[30] Z. Zhang, K. Q. Zhu, H. Wang, and H. Li, "Automatic extraction of top-*k* lists from the Web," in *Proc. ICDE*, 2013, pp. 1057–1068.
[31] G. Berardi, A. Esuli, D. Marcheggiani, and F. Sebastiani, "ISTI@ TREC Microblog track 2011: Exploring the use of hashtag segmentation and text quality ranking," in *Proc. TREC*, 2011, pp. 1–9.
[32] K. Gimpel *et al.*, "Part-of-speech tagging for Twitter: Annotation, features, and experiments," in *Proc. ACL-HLT*, 2011, pp. 42–47.
[33] Y. Duan, L. Jiang, T. Qin, M. Zhou, and H.-Y. Shum, "An empirical study on learning to rank of tweets," in *Proc. COLING*, 2010, pp. 295–303.
[34] X. Zhang, B. He, T. Luo, and B. Li, "Query-biased learning to rank for real-time Twitter search," in *Proc. CIKM*, 2012, pp. 1915–1919.
[35] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. NIPS*, 2013, pp. 3111–3119.
[36] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck, "Learning deep structured semantic models for Web search using clickthrough data," in *Proc. CIKM*, 2013, pp. 2333–2338.
[37] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: An update," *ACM SIGKDD Explorations Newslett.*, vol. 11, no. 1, pp. 10–18, 2009.
[38] X. Yin, J. Han, and P. S. Yu, "CrossClus: User-guided multi-relational clustering," *Data Mining Knowl. Discovery*, vol. 15, no. 3, pp. 321–348, 2007.
[39] K. Bontcheva, L. Derczynski, A. Funk, M. A. Greenwood, D. Maynard, and N. Aswani, "TwitIE: An open-source information extraction pipeline for microblog text," in *Proc. Int. Conf. Recent Adv. Natural Lang. Process. (RANLP)*, 2013, pp. 83–90.

Ankan Mullicke received the bachelor's degree in computer science and engineering from Jadavpur University, Kolkata, West Bengal, India, and the master's degree in computer science and engineering from IIT Kharagpur, West Bengal, India.

He is a Data Scientist with Microsoft, Hyderabad, India. His areas of interest include machine learning, natural language processing, data mining, and deep learning.

Pawan Goyal received the B.Tech. degree in electrical engineering from IIT Kanpur, West Bengal, India, in 2007, and the Ph.D. degree in computing and engineering from University of Ulster, Belfast, U.K., in 2011.

He was then a Post-Doctoral Fellow at INRIA Paris Rocquencourt. He is an Assistant Professor with the Department of Computer Science and Engineering, IIT Kharagpur, West Bengal, India. He has published around 75 research papers in international conferences and journals including ACL, NAACL, EMNLP, KDD, SIGIR, CIKM, JCDL, CSCW, WWW, IEEE, and ACM Transactions. His research interests include natural language processing, text mining, information retrieval and Sanskrit computational linguistics.



Niloy Ganguly received the Ph.D. degree from the Indian Institute of Engineering Science and Technology, Shibpur, India, and the bachelor's degree in computer science and engineering from IIT Kharagpur, West Bengal, India.

He has been a Doctoral Fellow with the Technical University of Dresden, Germany. He is a Professor with the Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur, India, where he has worked in the EU-funded project Biology Inspired techniques for Self-Organization in dynamic Networks (BISON). He currently focuses on dynamic and self-organizing networks, especially peer-to-peer networks, online social networks, and delay tolerant networks.

Manish Gupta is a Principal Applied Researcher at Microsoft India R&D Private Limited, Hyderabad, India. He is also an Adjunct Faculty at the International Institute of Information Technology, Hyderabad, and a Visiting Faculty at the Indian School of Business, Hyderabad. His areas of interest include web mining, data mining, and deep learning.