Improving semantic topic clustering for search queries with word co-occurrence and bigraph co-clustering

Jing Kong*, Alex Scott, and Georg M. Goerg

Google, Inc.

Abstract

Uncovering common themes from a large number of unorganized search queries is a primary step to mine insights about aggregated user interests. Common topic modeling techniques for document modeling often face sparsity problems with search query data as these are much shorter than documents. We present two novel techniques that can discover semantically meaningful topics in search queries: i) word co-occurrence clustering generates topics from words frequently occurring together; ii) weighted bigraph clustering uses URLs from Google search results to induce query similarity and generate topics. We exemplify our proposed methods on a set of Lipton brand as well as make-up & cosmetics queries. A comparison to standard LDA clustering demonstrates the usefulness and improved performance of the two proposed methods.

keywords: search queries, topic clustering, word co-occurrence, bipartite graph, co-clustering

1 Introduction

With the increasing size and popularity of online search, exploring information embedded in search queries has become a remarkable resource for valuable business insights. For instance, analyzing search queries related to a brand can inform a company what constitutes the brand search volume and what customers associate with their brand. Clustering similar individual search queries into meaningful buckets is often beneficial prior to more advanced analysis. For example, aggregating search volume across search queries from misspellings, plural forms and variations of a single concept increases signal-to-noise ratio and speeds up algorithms due to smaller dimensionality. At a high level, finding common topics yields insights beyond individual queries. From a product category point of view, say 'beauty products', grouping similar queries together simplifies the analyses in looking for new and trending topics.

Latent dirichlet allocation (LDA) [2] and probablistic latent semantic analysis (PLSA) [5] are widely used techniques to unveil latent themes in text data. Basically, mixture models of topics are imposed on documents, where a topic is a probability distribution over words. These models learn the hidden topics by implicitly taking advantage of document level word co-occurrence patterns [3, 10].

Short texts however – such as search queries, tweets or instant messages – suffer from data sparsity, which causes problems for traditional topic modeling techniques. Unlike proper documents, short text snippets do not provide enough word counts for models to learn how words are related and to disambiguate multiple meanings of a single word [6]. As a demonstrative example, Figure 1 and 2 show 2,000 Lipton brand queries which are naturally decomposed into a 'tea' and a 'soup' cluster. By treating each query as a document and unique words as terms in the LDA setting, these Lipton queries yield a document-term matrix with 99% sparsity. Since each query usually covers either the 'tea' or the 'soup' concept, we set the number of clusters to $K = 2$ and $\alpha = 0.1$.[1] Figure 1 shows that LDA can recognize the two topics to a large extent. However, we can easily identify several words, marked in red boxes, falling in the wrong category. In contrast, our proposed *word co-occurrence clustering* does not suffer from this shortcoming (Figure 2).[2] Section 2.1 explains the

*Corresponding author: jingkong@google.com

[1] The hyperparameter α specifies the Dirichlet prior topic distribution of each document in LDA. A low α encodes the prior belief that a document may contain just a few –or even only one – topics.

[2] We don't treat Lipton company related words including 'job', 'song', 'stock' as misclustered. The reason is that Lipton is better known for its tea products, therefore queries like "*lipton tea job*" and

underlying methodology in detail.

A common way to reduce sparsity is to aggregate short texts into lengthy pseudo-documents. For example, Weng et al. (2010) [11] collapse tweets belonging to one user into documents before training LDA. However, this usually makes it much harder to find common and *meaningful* associations for search queries. Suppose a user is a heavy searcher for beauty products, then they are likely to search for several products or brands within beauty category over time. Collapsing all their queries into a single 'document' adds lots of wrong product $\leftrightarrow$ attribute relationships because attributes of one product are mixed with characteristics of a completely unrelated product or brand. Moreover, a large number of users have too few search queries to make aggregation helpful.

Yan et al. (2013) [13] proposed biterm topic model to learn the topics by directly modeling the generation of unordered word pairs (biterms) in the whole corpus. This method is effective to deal with short text by aggregating word co-occurrence patterns, but the drawback is that all non-stop terms are considered equally in forming the biterms [12].

This paper introduces two novel topic clustering methods for search queries, namely *word co-occurrence* and *weighted bigraph* co-clustering. Word co-occurrence clustering first applies text data processing to canonicalize variations including misspelled and plural forms and to remove stop words, then generates topics by looking for words that co-occur frequently with topic anchors in the given set of queries. It can alleviate the sparsity problem as topic anchors are selected to be words more likely to be searched in the context of search queries than usual. This also effectively encourages the algorithm to emphasize on interesting words within the queries, rather than indistinctly forming biterms among all possible non-stop words. Creating bipartite graph for document clustering is proposed in Dhillon (2001) [4], where documents and words from these documents constitute the two sets of nodes for the graph with edge weights being document term frequency. However, this does not offer additional information beyond the scope of words, documents and their interaction. We introduce *weighted bigraph co-clustering* which extends the idea of Befferman and Berger (2000) [1] and constructs a weighted bipartite graph using query and *URL pairs, – as new source of data to enrich short queries* –, in organic

Figure 1: LDA clusters, top for 'tea' cluster and bottom for 'soup' cluster, with red boxes marking wrongly classified words.

"*lipton tea stock*" exist in our collection of Lipton brand queries as opposed to "*lipton soup job*" or "*lipton soup stock*".

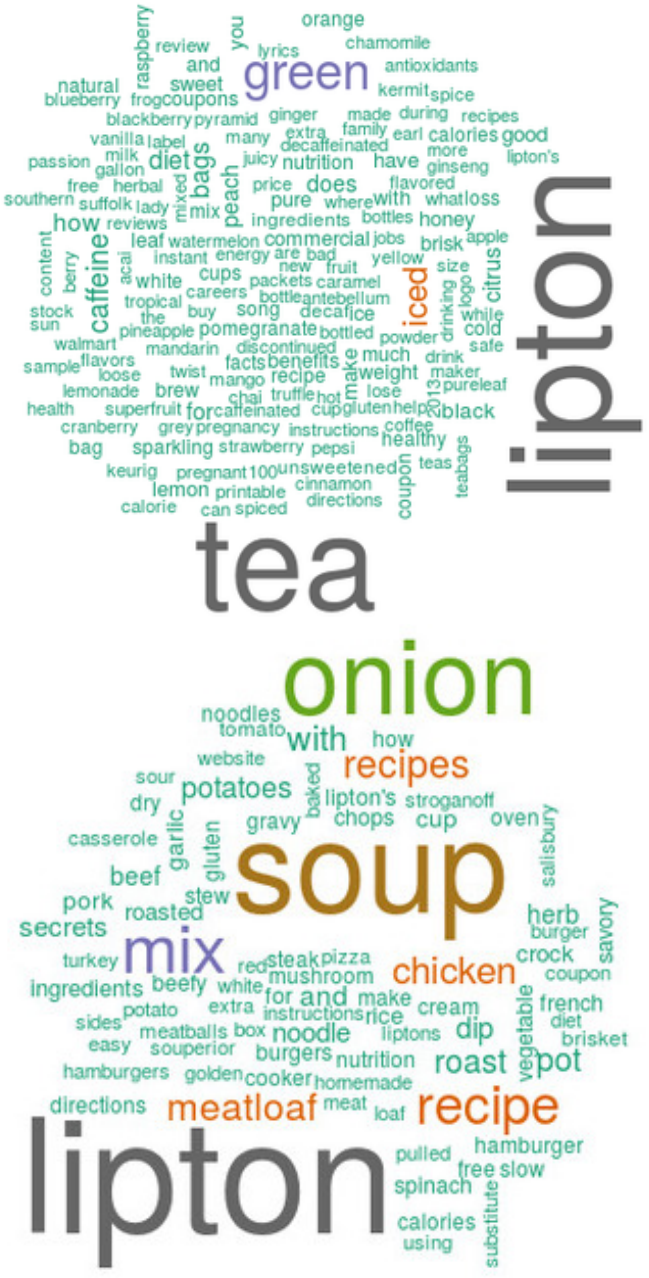

Figure 2: Word co-occurrence clustering results, top for 'tea' cluster and bottom for 'soup' cluster.

search results to induce query level similarity, and then form clusters. It uses Google search engine to understand the semantic meaning of search queries and to deal with low level variations.

The paper is organized as follows. Word co-occurrence clustering is formally introduced in Section 2.1; weighted bigraph clustering in Section 2.2. Case studies for Lipton brand and make-up & cosmetics queries demonstrate the usefulness and performance of each method. Section 3 includes summary and discussion.

2 Methodology: finding common themes in a set of search queries

Before going into details of the methodology we introduce notation and terminology. We can formalize the problem as follows: given a set of L *queries* $\mathbb{Q} = \{q_l \mid 1, \ldots, L\}$ we want to find K *topics*, $\mathbb{T} = \{t_j \mid j = 1, \ldots, K\}$, where each topic t_j represents a meaningful collection of queries. For example, q_1 = "*green tea lipton*" and q_2 = "*is green tea good for you?*" both fall in a semantically meaningful t_1 = 'green tea' topic. We also often refer to unigrams of a query as *words* or *terms*. Let $\mathbb{W} = \{\mathrm{w}_i \mid i = 1, \ldots, N\}$ be the set of unique words of all queries in $\mathbb{Q}$. Furthermore, we denote i) $\mathcal{B}(t_j)$ as the bag of words defining t_j, e.g., $\mathcal{B}(t_1)$ = { 'green', 'tea' }; ii) $\mathcal{B}(q_l)$ as the bag of words contained in query q_l, e.g., $\mathcal{B}(q_1)$ = { 'green', 'tea', 'lipton' }; iii) $\mathcal{Q}(t_j) = \cup_l \{q_l \mid \mathcal{B}(q_l) \cap \mathcal{B}(t_j) \neq \emptyset\}$ the queries that contain words belonging to topic t_j. For example, $\mathcal{Q}(t_1)$ contains q_1 = "*green tea lipton*", q_2 = "*is green tea good for you?*" and q_3 = "*lipton tea*" and q_4 = "*lipton black tea*" as they all share words with t_1.

We propose two methods to estimate $\mathbb{T}$. Word co-ocurrence clustering (Section 2.1) relies on a good specification of a query context. This is often the case when analyzing a particular brand or category, e.g., 'Lipton'. If such a context is not available or the queries are too heterogenous to define a single encompassing context we suggest to use weighted bigraph clustering (Section 2.2).

2.1 Word Co-occurrence Clustering

First, we apply standard text cleaning and processing to queries – including stemming, stopwords and punctuation removal as well as spelling correction (see Jurafsky and Martin (2000) [7] for reference) –, which are then segmented into bags of words $\mathbb{W} = \{\mathrm{w}_i \mid i = 1, \ldots, N\}$. Each w_i represents

a set of variations of a word, e.g., 'lipton', 'liptons', 'lipton's' are all represented by one w_i.

Next, word co-occurrence clustering initializes t_j's explicitly with a number of interesting words w_i given the context of the queries. In order to find topics related to a search query it is useful to look at *lift scores*. The lift score for word w_i given action $\mathbf{a}$ is defined as

$$lift(\mathrm{w}_i; \mathbf{a}) = \frac{P(\mathrm{w}_i \mid \mathbf{a})}{P(\mathrm{w}_i)}, \tag{1}$$

where $\mathbf{a}$ can be any user action, such as visiting a certain website or searching for a specific query. Suppose $lift(\mathrm{w}_i; \mathbf{a}) = 5$, then the chance of w_i being searched given $\mathbf{a}$ is 5 times greater than the chance of w_i being searched in general. A large lift score helps us to construct topics around meaningful rather than uninteresting words. In practice the probabilities in (1) can be estimated using word frequency in Google search history within a recent time window.

For the remainder of this work we usually assume $\mathbf{a}$ represents a user issuing a specific search query, say, a brand or product category name.[3][4] For example, if all queries are related to a brand, say Lipton, the context can be the brand name 'lipton'. If the queries are around a product category, say make-up & cosmetics, the context can be specified as 'beauty'.

The method then expands each topic t_j by introducing words that significantly co-occur with existing words in $\mathcal{B}(t_j)$ among the query set $\mathcal{Q}(t_j)$. $\mathcal{Q}(t_j)$'s provide a non-disjoint partition on the queries. Additional steps below help to achieve clusters with better interpretation. 1. If non-disjoint results are desired, a query is assigned to the topic with the largest intersection, i.e., $\arg\max_j \mid \mathcal{B}(q_l) \cap \mathcal{B}(t_j) \mid$. A query could be assigned to multiple topics if there exists a tie on intersection size. 2. Disjoint clusters can be achieved with hierarchical clustering on pairwise query similarity induced by how much two queries agree on their topic affinity.

The rationale of word co-occurrence clustering leads to the following properties:

1. Clusters usually deliver focused concepts and have good interpretation because the topics are keyed from words highly associated with the context.
2. It has the ability to extrapolate the bag of words to larger topics beyond $\mathbb{W} = \{\mathrm{w}_i \mid i = 1, \ldots, N\}$ due to the topic expansion and hierarchical clustering on induced query similarity.

[3]We estimated the probabilities in lift score using co-searched data for users. They could also be estimated with general document repository, like wikipedia webpages.

[4]One could also use an automated procedure to pick one of the queries as the context. For example, use the query q_i with the largest average lift compared to all other queries in the set. However, when the context is unknown we suggest to use bigraph clustering.

2.1.1 Algorithm

Create hashmap Assuming that text preprocessing is already conducted, the algorithm first segments all queries into words and extracts their variations. Then a hashmap is created with keys being the words $\mathbb{W} = \{\mathrm{w}_i \mid i = 1, \ldots, N\}$ and values being the queries having such words, i.e., the `key:value` pair is $\mathrm{w}_i : \{q_l \mid \mathrm{w}_i \in \mathcal{B}(q_l)\}$. For example, "*lipton chicken soup*" and "*lipton green tea*" are values of the key 'lipton'.

Initialize topics Topics are initialized with a subselection of keys of the hashmap. One could either manually select w_i's, or generate them automatically. Let us elaborate on the automatic route.

Any $\mathrm{w}_i \in \mathbb{W}$ can start a topic. However, not all words are interesting given the background of the queries. For instance, for "*best maybelline mascara*" the word 'best' is not as interesting as 'mascara' in the context of the Maybelline brand. We use lift score (1) to rank the words by importance and then threshold it to obtain a set of words highly associated with the context. Setting a zero threshold includes all words in $\mathbb{W}$ as topics; as it increases, more topic irrelevant and generic words are eliminated. Therefore, one could start with a moderate value, for example 5 or 10, and increase (decrease) the threshold if the clusters are too granular (broad) than expected. This yields the desired topics $\mathbb{T}$.

Up to now, each word in the hashmap is independent from each other and each topic is represented by only one word. In practice, words can co-occur with others to define a concept or to include certain attributes. For example, "*shape eyebrow*" is a valid combination but "*shape lips*" is less common. We often see "*waterproof mascara*" because waterproof is an important attribute of mascara. But "*waterproof nail polish*" is not common since almost all nail polish products are waterproof. Hence, we can expand a topic to other words based on the rationale of word co-occurrence.

Expand topics We first construct a term-document matrix [8], where terms are the w_i's and the documents, indexed by t_j's, are the concatenation of $\mathcal{Q}(t_j)$ which is the collection of queries associated with t_j. In order to pick one (or more) relevant topics for a word w_i, we

need a metric to measure if queries in $\mathcal{Q}(t_j)$ are more likely to include this word. To do this consider the log odds ratio

$$\ell_{jk} = \log_2 \frac{P(t_j \mid \mathrm{w}_i)}{P(t_k \mid \mathrm{w}_i)}, \quad k = 1, \ldots, K \tag{2}$$

which compares the likelihood of t_j to all t_k given word w_i, where $P(t_k \mid \mathrm{w}_i)$ is estimated using the term-document matrix. If $\ell_{jk} > 0$, then t_j is more likely than t_k given w_i.

To compare the association between t_j and w_i across all topics we average ℓ_{jk} over $k = 1, \ldots, K$. Instead of using a simple average we use the conditional probability of each t_k given w_i as the weight of ℓ_{jk} to reflect the importance of topic k given w_i. This yields the association score

$$R(t_j; \mathrm{w}_i) = \log_2 P(t_j \mid \mathrm{w}_i) - \sum_{k=1}^{M} P(t_k \mid \mathrm{w}_i) \log_2 P(t_k \mid \mathrm{w}_i), \tag{3}$$

which can take negative and positive values, but is guaranteed to be non-negative for at least one j. It is noteworthy that (3) also has an information theoretic interpretation as

$$R(t_j; \mathrm{w}_i) = \left\{ -\sum_{k=1}^{M} P(t_k \mid \mathrm{w}_i) \log_2 P(t_k \mid \mathrm{w}_i) \right\} - \{ -\log_2 P(t_j \mid \mathrm{w}_i) \} \tag{4}$$

$$= H(t \mid \mathrm{w}_i) - h(t_j \mid \mathrm{w}_i), \tag{5}$$

where $H(t \mid \mathrm{w}_i) \geq 0$ is the Shannon entropy [9] of topic conditional distribution given w_i, and $h(t_j \mid \mathrm{w}_j) \geq 0$ is the pointwise entropy of topic t_j given w_i. Hence $R(t_j; \mathrm{w}_i)$ measures how much more information is contained in t_j given w_i compared to the expected information of a randomly picked topic. For example, if $R(t_j; \mathrm{w}_i) = 1$, then – conditioned on w_i – topic t_j is 1 bit more informative than a randomly selected topic.

To expand t_j, we threshold on $R(t_j; \mathrm{w}_i)$ and update

$$\mathcal{B}(t_j) \leftarrow \{\mathrm{w}_i \mid R(t_j; \mathrm{w}_i) \geq \tau\} \cup \mathcal{B}(t_j) \tag{6}$$

A larger τ recruits fewer new words to t_j thus leads to a more conservative expansion. One may start with a large τ and decrease it until the words in $\mathcal{B}(t_j)$'s or the output clusters cease to be well focused.

This finishes one iteration of topic expansion. One could update the queries belonging to $\mathcal{Q}(t_j)$, then iterate the above process until no more w_i is added to any t_j.

Form non-disjoint clusters Now we calculate how many words a query intersects with the words in t_j,

$$s_{lj} = \mid \mathcal{B}(q_l) \cap \mathcal{B}(t_j) \mid, j = 1, \ldots, K, \tag{7}$$

then assign the query to the topic with the largest intersection, allowing ties.

If one starts with carefully selected keys, stopping here usually yields good results. If topics are generated automatically, then different t_j's, say initialized by 'eyelash' and 'mascara', represent the same underlying concept and are hence duplicated topics. In this case, additional steps below are helpful.

Induce query similarity Using the size of intersection defined (7), we map each query q_l to a vector of length K – being the number of topics –

$$\mathbf{s}(q_l) = (s_{l1}, \cdots, s_{lK}). \tag{8}$$

Pairwise query similarity between q_l and $q_{l'}$ is defined as

$$sim(q_l, q_{l'}) = \cos\left(\mathbf{s}(q_l), \mathbf{s}(q_{l'})\right) = \frac{\sum_{k=1}^{K} s_{lk} s_{l'k}}{\sqrt{\sum_{k=1}^{K} s_{lk}^2} \cdot \sqrt{\sum_{k=1}^{K} s_{l'k}^2}}, \tag{9}$$

which measures how much two queries agree with their affinities across all topics.

Form disjoint hierarchical clusters Run hierarchical clustering algorithm on the pairwise query similarity matrix. Clusters at different granularity are generated by cutting the hierarchical tree at various levels.

2.1.2 Example: Lipton queries

Here we revisit the Lipton brand queries from the Introduction (see Figure 2). Table 1 lists the largest 5 topics after one iteration of topic expansion with thresholds on lift and $R(t_j; \mathrm{w}_i)$ to be 10 and 4, respectively. Figure 3 shows the hierarchical clustering results on the induced query similarity matrix. The choice of number of clusters K, or equivalently the height to cut the hierarchical tree, is subjective and depends on one's use case. For instance, when broad and general topics are desired for Lipton queries, we

initial $\mathcal{B}(t_j)$	expanded $\mathcal{B}(t_j)$
tea	tea, goji, vert, zero, mango, citron, forest, yellow, lyric, glass, delight, indulge, ginseng, fresh
soup	soup, sausage, homemade
noodle	noodle, sauce, chive, instruction
rice	rice, broccoli, spanish
dip	dip, bowl

Table 1: Example initial topic words and expanded set of words for Lipton queries.

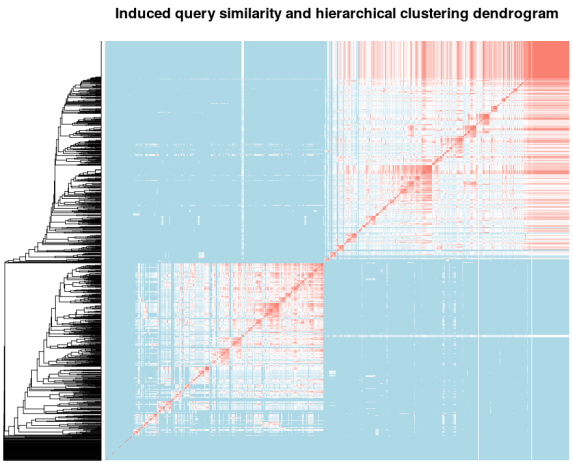

Figure 3: Pairwise similarity matrix for 2,000 Lipton brand queries with associated dendrogram of hierarchical clustering solution.

cut the hierarchical tree at height 0.9, which corresponds to two big clusters in the similarity matrix and Figure 2 reveals that these are a 'tea' and a 'soup/mix/meatloaf' cluster. Granular topics can be obtained by decreasing the cutoff value of hierarchical tree.

Word co-occurrence clustering relies on word preprocessing, i.e. fetching plural forms and variations, and co-occurrence to detect synonyms. However, this is not enough when semantic and contextual synonyms exist in the queries, e.g., 'burn fat' and 'weight loss'. Moreover it relies on a good specification of a context, which might not always be possible. In the following section, we present weighted bigraph co-clustering which augments short queries with URL information to find semantically meaningful clusters.

2.2 Weighted bigraph clustering

Befferman and Berger (2000) [1] proposed a strategy of viewing clickthrough data from search queries to URLs as a bipartite graph, and applied iterative, agglomerative clustering to its vertices. The rationale is that

1. Users may phrase their query differently, including variations and misspellings, but a search engine understands they are close and present the same URL to the users. Hence, URLs can identify queries of similar meaning.

2. URLs that are shown as top search results for a single query are somewhat similar. Hence, queries naturally group similar URLs to together.

In practice we observe that i) click data is usually sparse and we may miss a big portion of queries if focusing on click data alone; ii) organic search results are often quite informative and relevant to the search query itself even when users do not click through. Following these observations we enhance previous work by Befferman and Berger (2000) [1] to introduce *weighted* bigraph clustering that builds on organic Google search results to construct the bipartite graph and uses both *click* and *impression count* data to inform edge weights.

2.2.1 Algorithm

Create bipartite graph This algorithm starts with target queries, $\mathbb{Q} = \{q_l \mid l = 1, \cdots, L\}$, then collects associated URLs, $\mathbb{U} = \{u_m \mid m = 1, \cdots, M\}$, from Google search data, usually restricting on URLs with large impression and small average position (say, top 10 for search results on the first page). With graph theory terminology, the bipartite graph is a triple $\mathbb{G} = (\mathbb{Q}, \mathbb{U}, \mathbb{E})$ where E is the set of edges $\{E_{lm} = weight(q_l, u_m) : q_l \in \mathbb{Q}, u_m \in \mathbb{U}\}$. The edge weights are discussed below.

Calculate edge weights We formulate the problem of calculating E_{lm} from a Bayesian point of view by imposing a $Beta(\alpha, \beta)$ prior on click-through-rate (CTR) for each URL with an edge linked to a query, i.e.

$$CTR = \frac{click}{impression} \sim Beta(\alpha, \beta), \qquad (10)$$

where $\alpha = \gamma$ and $\beta = 1 - \gamma$ are hyperparameters and γ is specified as a typical CTR among organic search results. This choice guarantees that i) the prior average equals a typical CTR γ, and ii) we impose only a

weak prior impression count of $\alpha + \beta = 1$. Given impression and click observations for a (query, URL) pair, the posterior distribution of CTR is again a $Beta(a, b)$ with

$$a = \alpha + click\ count \text{ and} \tag{11}$$
$$b = \beta + nonclick\ count$$
$$= \beta + impression\ count - click\ count. \tag{12}$$

We define E_{lm} the weight measuring the importance of a URL u_m to a query q_l as the inverse posterior coefficient of variation, i.e. the posterior mean divided by the posterior standard deviation

$$E_{lm} = \frac{posterior\ mean}{posterior\ standard\ deviation} \tag{13}$$
$$= \frac{\frac{a}{a+b}}{\sqrt{\frac{ab}{(a+b)^2(a+b+1)}}} = \sqrt{\frac{a}{b}(a+b+1)}. \tag{14}$$

The weight in (13) is a variation adjusted CTR, which puts more importance to a URL with, say, $1,000$ impressions and 100 clicks than a URL with 10 impressions and just 1 click, even though their CTRs are identical. A closer look at (14) shows that the weight is a geometric mean of two interpretable quantities:

- The posterior ratio between click and nonclick counts, a/b, measures how relevant users consider the URL given the query. This quantifies the importance of a URL to a query from a click angle.
- The second term, $a+b+1$, equals the posterior impression count. The Google search engine identifies URLs relevant to the queries so the web pages are displayed on top of the search results. Hence, this term takes into account the contribution from an impression angle.

In applications, we do not impose a prior on non-observed (query, URL) pairs to maintain a sparse graph which is computationally much more efficient.

Figure 4 shows URLs weights on queries for a toy example of Maybelline queries. Obviously, "*maybelline age rewind concealer*" and "*maybelline roll on concealer*" are semantically close and the edges in Figure 4 show that also the large weight to the common URL captures this similarity. Let's see how to mathematically formulate this so that queries clustered together coincide with our semantic intuition.

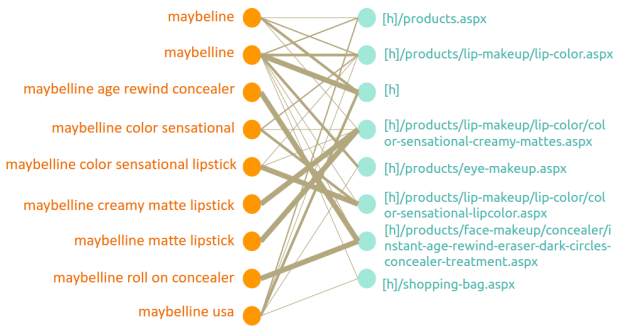

Figure 4: Subset of URL weights on queries for Maybelline brand queries and brand URLs, where *[h]* stands for `http://www.maybelline.com`. Thickness of edges represents edge weights.

Pairwise similarity Each query q_l is represented by a vector of M URL weights

$$\mathbf{w}_u(q_l) = (E_{l1}, \cdots, E_{lM}). \tag{15}$$

We measure pairwise similarity of q_l and $q_{l'}$ using cosine similarity of their edge weight vectors,

$$sim(q_l, q_{l'}) = \cos(\mathbf{w}_u(q_l), \mathbf{w}_u(q_{l'}))$$
$$= \frac{\sum_{m=1}^{M} E_{lm}E_{l'm}}{\sqrt{\sum_{m=1}^{M} E_{lm}^2} \cdot \sqrt{\sum_{m=1}^{M} E_{l'm}^2}}, \tag{16}$$

i.e. using URLs to induce query similarity. Pairwise similarity among URLs is defined in the same way with edge weights on queries.

Successive hierarchical clustering We run a clustering procedure on query and URL nodes successively. Each iteration runs hierarchical clustering on the query and the URL similarity matrix respectively, with a tree cut-off height at h, followed by an update on the weight vectors as well as similarities by treating queries or URLs in a cluster as a whole entity. Iteration stops when maximal similarity in both query and URLs similarity matrix is below a threshold $\in [0, 1]$. A larger threshold leads to more granular clusters.

Here we elaborate on the necessity of an iterative process of forming query and URL clusters. Consider the following scenario: the two queries

- q_l = "*maybelline lipstick [retailer]*"
- $q_{l'}$ = "*maybelline mascara [retailer]*"

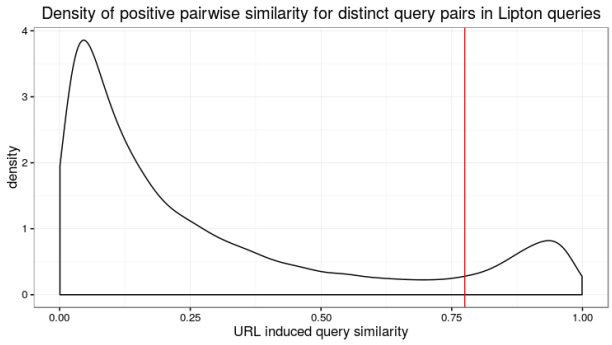

Figure 5: Density plot of URL induced query similarity for distinct query pairs (92.74% exact zero similarity pairs are removed). Red line indicates the similarity between "*does lipton green tea burn fat*" and "*diet lipton green tea weight loss*".

may both have a single link to two different URLs

- u_m = [retailer homepage]/maybelline/lipstick
- $u_{m'}$ = [retailer homepage]/maybelline/mascara.

At iteration 0, the query similarity equals zero. However, both URLs have a second edge to another query "*[retailer] maybelline*" and are therefore grouped together during the URL clustering procedure. Thereafter, u_m and $u_{m'}$ form a single URL entity and the updated similarity between q_l and $q_{l'}$ is no longer 0. In the next iteration, the two queries can be clustered together. The same rationale applies to URL clustering as well by boosting URL similarity using query clusters.

2.2.2 Example: Lipton queries

Impression and clicked URL data for the 2,000 Lipton brand queries were extracted from Google organic search results in January 2016. This gave us 11,358 distinct query and URL pairs. Following the steps above, we computed the weights of URLs (queries) on queries (URLs). To provide a concrete example, Table 2 presents a few common URLs' weights on "*does lipton green tea burn fat*" and "*diet lipton green tea weight loss*". The query similarity between these two queries is 0.78 in the initial iteration, as shown in the red line in Figure 5 which plots the density of URL induced query similarity for distinct pairs.

Weighted bigraph clustering does an excellent job in capturing semantic similarity. For example, Table 3 shows that it can learn that 'nutrition facts' and 'calories' describe a similar concept and that 'burn fat' and 'lose weight' are

cluster	**queries**
nutrition and calories	calories in lipton tea how many calories in lipton tea lipton black tea nutrition facts lipton tea bag calories lipton tea bags nutrition facts lipton tea nutrition
weight loss	can lipton green tea help lose weight does lipton green tea burn fat diet lipton green tea weight loss lipton green tea benefits weight loss
company information	history of lipton tea home country of lipton tea lipton tea origin lipton tea history
k cups (keurig)	keurig lipton iced tea lipton iced tea k cups lipton k cups lipton sweet tea k cup

Table 3: Example semantic clusters for Lipton brand queries using bigraph clustering.

essentially the same. It is noteworthy that in both cases the query pairs do not share a single common word, which demonstrates that this method can indeed achieve a semantic topic clustering, which is very hard to get right using topic modeling techniques that are based purely on a bag of words.

2.2.3 Example: make-up and cosmetics queries

As another example Figure 6 shows 370 generic make-up and cosmetics queries. These queries are challenging for word co-occurrence clustering since they are only loosely related compared to branded queries. Therefore, the expansion step is not very effective and the procedure reduces to creating clusters around each individual word.

In contrast, bigraph clustering is still powerful enough. We collect impression and click data on URLs for these queries from Google organic search data in January 2016, yielding 10,077 distinct (query, URL) pairs. Table 4 lists a subset of URLs normalized weights for "*best foundation*" and "*foundation makeup*" (all URL weights for a query add up to 1). Not suprisingly, queries and URLs tell each others stories in paralell.

Final clusters on queries are obtained by successive hier-

	does lipton green tea burn fat	diet lipton green tea weight loss
weight-loss-tea	0.07	0.08
articles/lipton-green-tea-for-weight-loss/	0.35	0.15
blog/lipton-green-tea-weight-loss-%E2%80%93 -can-lipton-green-tea-help-burn-fat	0.07	0.13

Table 2: Normalized weights on two example queries "*does lipton green tea burn fat*" and "*diet lipton green tea weight loss*" for subset of URLs (removing hostnames).

	best foundation	foundation makeup
best-foundation-makeup	0.06	0.12
best-of/top-10-best-foundations	0.01	0.02
beauty/makeup-skin-care/tips/g9064/ editors-favorite-foundation	0.12	0.03

Table 4: Example URLs (removing hostnames) normalized weights on two example queries "*best foundation*" and "*foundation makeup*"

Figure 6: Wordcloud for words with minimal frequency of 3 among 370 generic make-up and cosmetics queries. Size of word corresponds to term frequency.

archical clustering with maximal similarity threshold of 0.1. Three example clusters are listed in Table 5. Misspellings and variations, e.g. 'smoky' vs. 'smokey' and 'eyebrow' vs. 'brow', are easily handled with the support of Google search engine. Similar queries are very well grouped based on the pairwise similarity induced by URLs.

3 Discussion

In this work we introduce two methods to identify semantically meaningful topics in a collection of short texts such as search queries. *Word co-occurrence* clustering starts with a set of words anchors as initial topics, and generalizes anchors to other words co-appearing with the same queries. Topics are created using hierarchical clustering on pairwise query similarity, which measures to what extent two queries agree on their intersections with the list of words in each topic. This method performs well when the queries are closely related, e.g. brand queries, so that the keywords expansion step can effectively extropolate the scope of words to reach broader topics. For instance, for Lipton brand queries we achieve a big cluster of various tea products and another big clusters of food products covering soup, mix, recipes, etc. Word co-occurrence clustering does not depend on other sources of data thus is applicable to any set of short texts.

Weighted bigraph clustering capitalizes on organic search results to construct a bipartite graph with a set of queries and a set of URLs as nodes. Edge weights of the graph are computed with the impression and click data of (query,

cluster	queries
eye makeup	eye makeup how to do eyemakeup eye makeup tutorial eyeshadow tutorial how to apply eye makeup how to put on eyeshadow how to apply eyeshadow how to do a smokey eye smokey eye makeup smokey eye smokey eyes smoky eye smokey eye tutorial how to do smokey eyes
contouring	contouring contouring kit contouring makeup kit contour kit contouring makeup contour makeup makeup contouring how to contour face contouring how to contour face highlight and contour
eyebrow	best eyebrow pencil brow eyebrow pencil eyebrow eyebrow makeup eyebrows how to do eyebrows how to do your eyebrows how to shape eyebrows how to pluck eyebrow perfect eyebrows eyebrow tutorial

Table 5: Example topic clusters for make-up & cosmetics queries.

URL) pairs from a Bayesian perspective and are used to induce query (URL) pairwise similarities. Successive hierarchical clustering on both the query and URL nodes yields the final clusters. Due to the information embedded in Google search results, this method is superb in grouping semantically close queries together. Therefore, opposed to word co-occurrence clustering, weighted bigraph clustering can still perform very well even if the queries do not share common words, e.g. generic queries in the make-up and cosmetic example.

References

[1] D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 407–416. ACM, 2000.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[3] J. L. Boyd-Graber and D. M. Blei. Syntactic topic models. In *Advances in Neural Information Processing Systems*, pages 185–192, 2009.

[4] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–274. ACM, 2001.

[5] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM, 1999.

[6] L. Hong and B. D. Davison. Empirical study of topic modeling in Twitter. In *Proceedings of the First Workshop on Social Media Analytics*, pages 80–88. ACM, 2010.

[7] J. H. Martin and D. Jurafsky. *Speech and language processing*, volume 710. 2000.

[8] G. Salton, A. Wong, and C.-S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[9] C. E. Shannon and W. Weaver. The mathematical theory of information. 1949.

[10] X. Wang and A. McCallum. Topics over time: a non-Markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 424–433. ACM, 2006.

[11] J. Weng, E.-P. Lim, J. Jiang, and Q. He. Twitterrank: finding topic-sensitive influential Twitterers. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 261–270. ACM, 2010.

[12] Y. Xia, N. Tang, A. Hussain, and E. Cambria. Discriminative Bi-Term Topic Model for Headline-Based Social News Clustering. In *FLAIRS Conference*, pages 311–316, 2015.

[13] X. Yan, J. Guo, Y. Lan, and X. Cheng. A biterm topic model for short texts. In *Proceedings of the 22nd international conference on world Wide Web*, pages 1445–1456. ACM, 2013.