Learning Meta Word Embeddings by Unsupervised Weighted Concatenation of Source Embeddings

Anonymous ACL submission

Abstract

We propose a method to protect the privacy of search engine users by decomposing the queries using semantically *related* and unrelated *distractor* terms. Instead of a single query, the search engine receives multiple decomposed query terms. Next, we reconstruct the search results relevant to the original query term by aggregating the search results retrieved for the decomposed query terms. We show that the word embeddings learnt using a distributed representation learning method can be used to find semantically related and distractor query terms. We derive the relationship between the *obfuscity* achieved through the proposed query anonymisation method and the *reconstructability* of the original search results using the decomposed queries. We analytically study the risk of discovering the search engine users' information intents under the proposed query obfuscation method, and empirically evaluate its robustness against clustering-based attacks. Our experimental results show that the proposed method can accurately reconstruct the search results for user queries, without compromising the privacy of the search engine users.

1 Introduction

As web search engine users, we are left with two options regarding our privacy. First, we can trust the search engine not to disclose the keywords that we use in a search session to third parties, or even to use for any other purpose other than providing search results to the users who issued the queries. However, the user agreements in most web search engines do not allow such user rights. Although search engines pledge to protect the privacy of their users by encrypting queries and search results[1], the encryption is between the user and the search engine – the original non-encrypted queries are still available to the search engine. The keywords issued by the users are a vital source of information for improving the relevancy of the search engine and displaying relevant adverts to the users. For example, in learning to rank (He et al., 2008), keywords issued by a user and the documents clicked by that user are recorded by the search engine to learn the optimal dynamic ranking of the search results. user interests and extract attributes related to frequently searched entities (Pasca, 2014; Sadikov et al., 2010; Santos et al., 2010; Richardson, 2008; Pasca, 2007). Considering the fact that placing advertisements for the highly bid keywords is one of the main revenue sources for search engines, there are obvious commercial motivations for the search engines to exploit the user queries beyond simply providing relevant search results to their users. For example, it has been reported that advertisements contribute to 96% of Google's revenue[2]. Therefore, it would be unrealistic to assume that the user queries will not be exploited in a manner unintended by the users.

As an alternative approach that does not rely on the goodwill of the search engine companies, we propose a method (shown in Figure 1), where we disguise the queries that are sent to a search engine such that it is difficult for the search engine to guess the real information need of the user by looking at the keywords, yet it is somehow possible for the users to *reconstruct* the search results relevant for them from what is returned by the search engine. The proposed method does not require any encryption or blindly trusting the search engine companies or any third-party mediators. However, this is a non-trivial task because a search engine must be able to recognise the information need of a user in order to provide relevant results in the first place. Therefore, query obfuscation and relevance of search results are at a direct trade-off.

Specifically, given a user query A, our proposed

[1] https://goo.gl/JSBvpK

[2] https://www.wordstream.com/articles/google-earnings

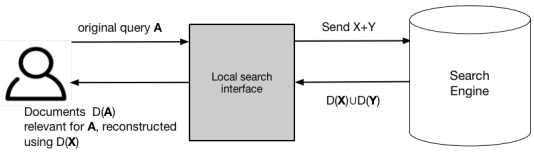

Figure 1: Overview of the proposed method. The original query A is decomposed into a set of relevant (X) and distractor (Y) terms at the user-end. The search engine returns documents relevant for both X and Y, denoted by $D(X) \cup D(Y)$. We will ignore $D(Y)$ and reconstruct the search results for A using $D(X)$.

method first finds a set of n noisy relevant terms $X_1, X_2, \ldots, X_n$ (denoted by the set $\{X_i\}_{i=1}^n$) and m distractor terms $Y_1, Y_2, \ldots, Y_m$ (denoted by the set $\{Y_j\}_{j=1}^n$) for A to obfuscate the user query. We use pre-trained word embeddings for identifying the noisy-relevant and distractor terms. We add Gaussian noise to the relevant terms such that it becomes difficult for the search engine to discover A using $\{X_i\}_{i=1}^n$. However, $\{X_i\}_{i=1}^n$ is derived using A, so there is a risk that the search engine will perform some form of de-noising to unveil A from $\{X_i\}_{i=1}^n$. Therefore, using $\{X_i\}_{i=1}^n$ alone as the keywords does not guarantee obfuscity. To mitigate this risk, we generate a set of distractor terms $\{Y_j\}_{j=1}^n$ separately for each user query. We then issue $X_1, X_2, \ldots, X_n, Y_1, Y_2, \ldots, Y_m$ in random order to the search engine to retrieve the corresponding search results. We then reconstruct the search results for A using the search results we retrieve from the noisy-relevant terms and discard the search results retrieved from the distractor terms. It is noteworthy that during any stage of the proposed method, we *do not* issue A as a standalone query nor in conjunction with any other terms to the search engine. Moreover, we do not require access to the search index, which is typically not shared by the search engine companies with the outside world.

We do not collect, process or release any personal data with ethical considerations in this work. Our contributions in this paper can be summarised as follows:

- We propose a method to obfuscate user queries sent to a search engine by semantic decomposition to protect the privacy of the search engine users. Our proposed method uses pre-trained word embeddings.

- We introduce the concepts of *obfuscity* (i.e., how difficult it is to guess the original user query by looking at the auxiliary queries sent to the search engine?), and *reconstructability* (i.e. how easy it is to reconstruct the search results for the original query from the search results for the auxiliary queries?), and propose methods to estimate their values.

- We theoretically derive the relationship between obfuscity and reconstructability using known properties of distributed word representations.

- We evaluate the robustness of the proposed query obfuscation method against clustering-based attacks, where a search engine would cluster the keywords it receives within a single session to filter out distractors and predict the original query from the induced clusters. Our experimental results show that by selecting appropriate distractor terms, it is possible to guarantee query obfuscity, while reconstructing the relevant search results.

2 Query Obfuscation via Semantic Decomposition

2.1 Finding Noisy-Related Terms

Expanding a user query using related terms (Carpineto and Romano, 2012) is a popular technique in information retrieval to address sparse results. Although query expansion is motivated as a technique for improving recall, we take a different perspective in this paper – we consider query expansion as a method for obfuscating (Gervais et al., 2014) the search intent of a user. Numerous methods have been proposed in prior work on query expansion to find good candidate terms for expanding a given user query such as using pre-compiled thesauri containing related terms and query logs (Carpineto and Romano, 2012). We note that any method that can find related terms for a given user query A can be used for our purpose given that the following requirements are satisfied:

1. The user query A must never be sent to the search engine when retrieving related terms for A because this would obviously compromise the obfuscation goal.

2. Repeated queries to the search engine must be minimised in order to reduce the burden on the search engine. We assume that the query

obfuscation process to take place outside of the search engine using a publicly available search API. Although modern Web search engines would gracefully scale with the number of users/queries, obfuscation methods that send excessively large numbers of queries are likely to be banned by the search engines because of the processing overhead. Therefore, it is important that we limit the search queries that we issue to the search engine when computing the related terms.

3. No information regarding the distribution of documents nor the search index must be required by the related term identification method. If we had access to the index of the search engine, then we could easily find the terms that are co-occurring with the user query, thereby identifying related terms. However, we assume that the query obfuscation process happens outside of the search engine. None of the major commercial web search engines such as Google, Bing or Baidu provide direct access to their search indices due to security concerns. Therefore, it is realistic to assume that we will not have access to the search index during anytime of the obfuscation process, including the step where we find related terms to a given user query.

4. The related terms must not be too similar to the original user query A because that would enable the search engine to guess A via the related terms it receives. For this purpose, we would add noise to the user query A and find *noisy related neighbours* that are less similar to A.

We propose a method that uses pre-trained word embeddings to find related terms for a user query that satisfy all of the above-mentioned requirements. Context-independent word embedding methods such as word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014) can represent the meanings of words using low dimensional dense vectors. Using word embeddings is also computationally attractive because they are low dimensional (typically $100-600$ dimensions are sufficient), consuming less memory and faster when computing similarity scores. Although we focus on single word queries for the ease of discussion, we note that by using context-sensitive phrase embeddings such as Elmo (Peters et al., 2018) and BERT (Devlin et al., 2019) we can obtain vectors representing multi-word queries, which we defer to future work.

We denote the pretrained word embedding of a term A by $v(A)$. To perturbate word embeddings, we add a vector, $\boldsymbol{\theta} \in \mathbb{R}^d$, sampled independently for each A from the d-dimensional Gaussian with a zero mean and a unit variance, and measure the cosine similarity between $v(A)+\boldsymbol{\theta}$ and each of the words $X_i \in \mathcal{V}$ in a predefined and fixed vocabulary $\mathcal{V}$, using their word embeddings $v(X_i)$. We then select the top most similar words $\{X_i\}_{i=1}^n$ as the noisy related terms of A.

Let us denote the set of documents retrieved using a query A by $\mathcal{D}(A)$. If we use a sufficiently large number of related terms X_i to A, we will be able to retrieve $\mathcal{D}(A)$ exactly using

$$\mathcal{D}'(A) = \bigcup_{i=1}^{n} \mathcal{D}(X_i). \tag{1}$$

However, in practice we are limited to using a truncated list of n related terms because of computational efficiency and to limit the number of queries sent to the search engine. Therefore, in practice $\mathcal{D}'(A)$ will not be exactly equal to $\mathcal{D}(A)$. Nonetheless, we assume the equality to hold in (1), and later in the theoretical proofs given in the supplementary material discuss the approximation error. To model the effect of ranking, we consider only the top-ζ ranked documents as $\mathcal{D}(X_i)$ and set $\zeta = 100$ in our experiments.

2.2 Obfuscation via Distractor Terms

Searching using noisy related terms X_i alone of a user query A, does not guarantee the obfuscity. The probability of predicting the original user query increases with the number of related terms used. Therefore, we require further mechanisms to ensure that it will be difficult for the search engine to predict A from the queries it has seen. For this purpose, we select a set of unrelated terms $\{Y_j\}_{j=1}^n$, which we refer to as the *distractor* terms.

Several techniques can be used to find the distractor terms for a given query A. For example, we can randomly select terms from the vocabulary $\mathcal{V}$ as the distractor terms. However, such randomly selected distractor terms are unlikely to be coherent, and could be easily singled-out from the related terms by the search engine. If we know the semantic category of A (e.g. A is a *person* or a *location* etc.), then we can limit the distractor terms to the

same semantic category as A. This will guarantee that both related terms as well as distractor terms are semantically related in the sense that they both represent the same category. Therefore, it will be difficult for the search engine to discriminate between related terms and distractor terms. Information about the semantic categories of terms can be obtained through different ways such as Wikipedia category pages, taxonomies such as the WordNet (Miller, 1995) or by named entity recognition (NER) tools. Moreover, we consider distractor terms Y_j that have similar average frequency as the original query A and the noisy related terms X_i so that it will be difficult to differentiate between distractor terms and noisy related terms based on frequency information.

We propose a method to find distractor terms Y_j for each query A using pre-trained word embeddings as illustrated in Figure 2. Let us consider a set of candidate terms $\mathcal{C}$ from which we must select the distractor terms. For example, $\mathcal{C}$ could be a randomly selected subset from the vocabulary of the corpus used to train word embeddings. First, we select a random hyperplane (represented by the normal vector $\boldsymbol{h} \in \mathbb{R}^d$ to the hyperplane) in the embedding space that passes through the point corresponding to A. Next, we split $\mathcal{C}$ into two mutually exclusive sets $\mathcal{C}_+ = \{x : x \in \mathcal{C}, \boldsymbol{x}^\top \boldsymbol{h} \geq 0\}$ and $\mathcal{C}_- = \{x : x \in \mathcal{C}, \boldsymbol{x}^\top \boldsymbol{h} < 0\}$ depending on which side of the hyperplane the word is located. Let us define $\mathcal{C}_{\max}$ and $\mathcal{C}_{\min}$ to be respectively the larger and smaller of the two sets $\mathcal{C}_+$ and $\mathcal{C}_-$ (i.e. $\mathcal{C}_{\max} = \operatorname{argmax}_{\mathcal{S} \in \{\mathcal{C}_+, \mathcal{C}_-\}} |\mathcal{C}|$ and $\mathcal{C}_{\min} = \operatorname{argmin}_{\mathcal{S} \in \{\mathcal{C}_+, \mathcal{C}_-\}} |\mathcal{C}|$) Next, we remove the top 10% of the similar words in $\mathcal{C}_{\max}$ to the original query A. We then use this reduced $\mathcal{C}_{\max}$ as $\mathcal{C}$ (i.e. $\mathcal{C} \leftarrow \mathcal{C}_{\max}$) and repeat this process until we are left with the desired number of distractor terms in $\mathcal{C}$. Intuitively, we are partitioning the candidate set into two groups in each iteration considering some attribute (dimension) of the word embedding of the query (possibly representing some latent meaning of the query), and removing similar terms in that subspace.

2.3 Reconstructing Search Results

Once we have identified a set of noisy related terms, $\{X_i\}_{i=1}^n$, and a set of distractor terms, $\{Y_j\}_{j=1}^n$, we issue those terms as queries to the search engine and retrieve the relevant search results for each individual term. We issue related terms and distractor terms in a random sequence, and ignore the results returned by the search engine for the distractor terms. Finally, we can reconstruct the search results for A using (1).

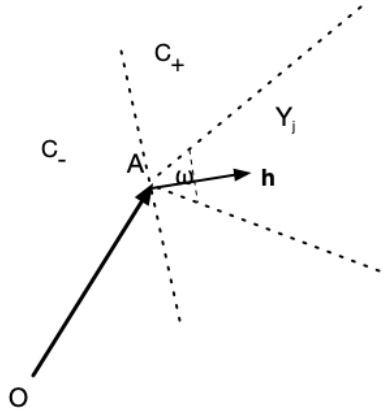

Figure 2: Selecting distractor terms for a given query A. We first compute the noise ($\boldsymbol{\theta}$) added vector $\boldsymbol{A'}$ for $\boldsymbol{A}$, and then search for terms Y_j that are located inside a cone that forms an angle ω with $\boldsymbol{A'}$. This would ensure that distractor terms are sufficiently similar to the noise component, therefore difficult to distinguish from A.

3 Obfuscity vs. Reconstructability

Our proposed query decomposition method strikes a fine balance between two factors (a) the difficulty for the search engine to guess the original user query A, from the set of terms that it receives $\mathcal{Q}(A) = \{X_1, X_2, \ldots, X_n, Y_1, Y_2, \ldots, Y_m\}$, and (b) the difficulty to reconstruct the search results, $\mathcal{D}(A)$, for the original user query, A, using the search results for the noisy related terms following (1). We refer to (a) as the *obfuscity*, and (b) as the *reconstructability* of the proposed query decomposition process.

3.1 Obfuscity

We define *obfuscity*, α, as the ease to guess the user query A, from the terms issued to the search engine and compute it as follows:

$$\alpha = 1 - \frac{1}{|\mathcal{Q}(A)|} \sum_{q \in \mathcal{Q}(A)} \operatorname{sim}(v(A), v(q)) \quad (2)$$

Specifically, we measure the average cosine similarity between the word embedding, $v(A)$, for the original user query A, and the word embeddings $v(q)$ for each of $q \in \mathcal{Q}(A)$ search terms. If the similarity is higher, then it becomes easier for the search engine to guess A from the search terms. The difference between this average similarity and

1 (i.e. the maximum value for the average similarity) is considered as a measure of obfuscity we can guarantee through the proposed query decomposition process. Even if we are not exactly sending A to the search engine as a keyword, the search engine will be able to figure out A from $\mathcal{Q}(A)$. By using word embeddings to measure the similarity between the original query A and the keywords $\mathcal{Q}(A)$ sent to the search engine, we are able to consider not only exact matches but semantically similar keywords, which can be seen as a soft match between words. The definition of obfuscity given by (2) is based on this intuition.

3.2 Reconstructability

We reconstruct the search results for A using the search results for the queries $\{X_i\}_{i=1}^{n}$ following (1). We define *reconstructability*, ρ as a measure of the accuracy of this reconstruction process and is defined as follows:

$$\rho = \frac{|\mathcal{D}(A) \cap \mathcal{D}'(A)|}{|\mathcal{D}(A)|} \tag{3}$$

A document retrieved and ranked at top-ζ by only a single noisy related term might not be relevant to the original user query A. A more robust reconstruction procedure would be to consider a document as relevant if it has been retrieved by at least l different noisy related terms. If a user query A can be represented by a set of documents where, each document is retrieved by at least $l < n$ different noisy related terms, then we say A to be *l-reconstructable*. In fact, the reconstruction process defined in (1) corresponds to the special case where $l = 1$. Increasing the value of l would decrease the number of relevant documents retrieved for the original user query A, but it is likely to increase the relevance of the retrieval process. In the supplementary material, we prove that the trade-off relationship (18) holds between ρ and α.

Theorem 1. *Given a query A, represented by d-dimensional embedding, $v(A)$, let us obfuscate it with n distractor terms and use all (i.e. $n = l$) distractor terms to reconstruct the search results for A. The obfuscity α and the reconstructability ρ is in the inverse (trade-off) relationship given by (18), where c and Z are query independent constants.*

$$\log \rho = \frac{cl}{2d}\left(c + 2(1-\alpha)\,||v(A)||_2\right) - \log Z \tag{4}$$

3.3 Extension to Multi-word Expressions

The anonymisation method and its theoretical analysis described in the paper so far can be easily generalised to handle multi-word queries. Specifically, in the case of multi-word queries we must embed not only unigrams but phrasal n-grams. Directly modelling n-gram co-occurrences is challenging for higher-order n-grams because of data sparseness issues (Turney and Pantel, 2010). Compositional approaches (Cordeiro et al., 2016; Hashimoto and Tsuruoka, 2016; Poliak et al., 2017; Yu and Dredze, 2015) have been proposed to overcome this problem, where unigram, subword, or character level embeddings are iteratively combined to create representations for longer phrasal queries. These methods can compute length-invariant vector representations for n-grams, which can then be used in the same manner as described in Section 2.1 for finding noisy-related terms and in Section 2.3 for finding distractor terms.

3.4 Effect of Ranking

If the number of documents containing q, $|\mathcal{D}(q)|$, is less than ζ for all $q \in \mathcal{Q}(A)$, we will be able to retrieve all documents containing the related and distractor terms. However, when this condition does not hold for one or more terms in $\mathcal{Q}(A)$, the reconstruction process is not guaranteed to perfectly reconstruct $\mathcal{D}(A)$, depending on the accuracy of the ranking method used in the search engine. Note that due to the relatedness between the terms $\{X_i\}_{i=1}^{n}$, even though a particular relevant document $d \in \mathcal{D}(A)$ is not retrieved by a term X_i due to the truncation by ranking, it could still be retrieved by a different X_j ($j \neq i$) term. Moreover, in practice, the number of relevant documents for a query is significantly smaller than ζ and modern search engines have accurate ranking models that return relevant results among top-ζ, thus mitigating this risk of truncation.

4 Experiments

4.1 Effect of Noise and Distractor Terms

To evaluate the proposed method we create a dataset where we select 50 popular queries from Wikipedia query logs and associate them with the relevant Wikipedia articles. We use the December 2015 dump of English Wikipedia for this purpose and build a keyword-based inverted search index. We use 300 dimensional pretrained GloVe (Pennington et al., 2014) embeddings trained from a

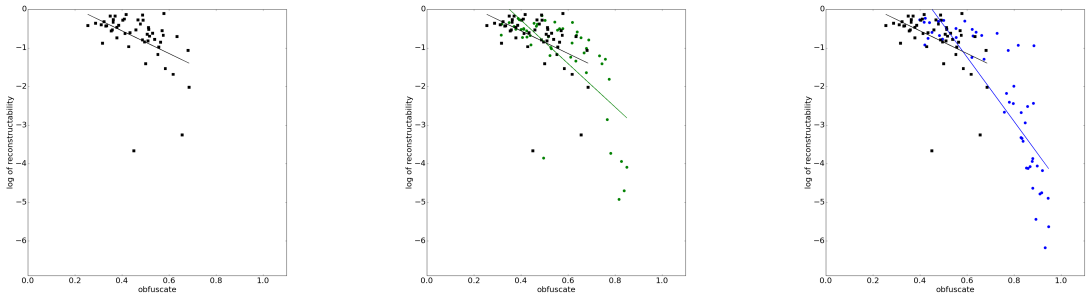

Figure 3: Relationship between obfuscity and reconstructability under different levels of added noise and no distractor terms (left: no-noise, middle: medium-level of noise, and right: high-level of noise). Base reconstructability scores for no noise and no distractor terms are super imposed in black boxes.

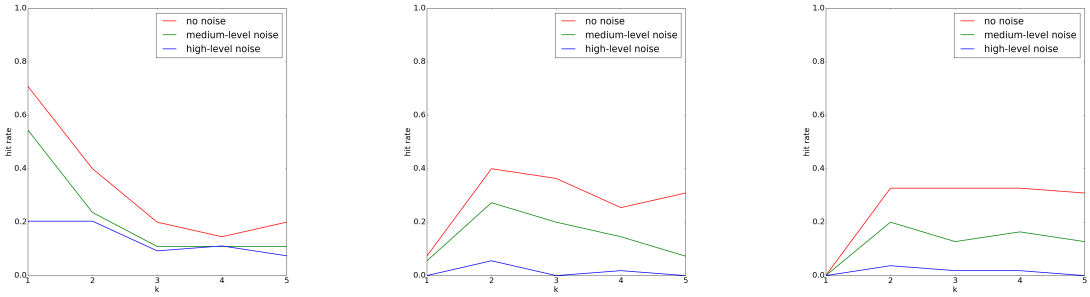

Figure 4: Hit rates for the k-means clustering attacks for increasing number of clusters (k) and distractor terms. (left: no distractors, middle: 20 distractors, and right: 40 distractors). In each figure, we show results for three levels of added noise.

42 billion token Web crawled corpus[3]. Figure 3 and show the obfuscity and the natural (base e) log-reconstructability to values for the 50 queries in our dataset at different levels of noise. Specifically, we add Gaussian noise with zero-mean and standard deviations of 0.6 and 1.0 respectively to stimulate medium and high levels of noise, whereas the no-noise case corresponds to not perturbing the word embeddings. (trend with distractor terms are shown in Appendix Figure 6) Distractor terms that have similar average frequency to the original query and the noisy relevant terms are randomly selected from Wikipedia articles that belong to the same Wikipedia category tag as the article for the original query.

We see a negative correlation between obfuscity and reconstructability in all plots as predicted by (18). Addition of noise affects the selection of related terms but not the selection of distractor terms. However, related terms influence both obfuscity as well as reconstructability. Because Gaussian noise is added to the word embedding of the original query, and the nearest neighbours to this noise added embedding are selected as the related terms, this process would help us to increase obfuscity. On the other hand, the search results obtained using noisy related terms will be less relevant to the original user query. Therefore, reconstructing the search results for the original user query using the search results for the noisy related terms will become more difficult, resulting in decreasing the reconstructability. The overall effect of increasing obfuscity and decreasing reconstructability is shown by the increased negative gradient of the line of best fit in the figures.

4.2 Robustness against Attacks

An important aspect of a query obfuscation method is its robustness against attacks. Given that the proposed method sends two groups of terms (relevant and distractor) to a search engine, a natural line of attack is to cluster the received terms to filter out distractor terms and then guess the user query from the relevant terms. We call such attacks as *clustering attacks*. As a concrete example, we simulate a clustering attacker who applies k-means clustering

[3] https://nlp.stanford.edu/projects/glove/

to the received terms. The similarity between terms for the purpose of clustering is computed using the cosine similarity between the corresponding word embeddings. Any clustering algorithm can be used for this purpose. We use k-means clustering because of its simplicity. Next, the attacker must identify a single cluster that is likely to contain the relevant terms. For this purpose, we measure the *coherence*, $\mu(\mathcal{C})$, of a cluster $\mathcal{C}$ given by (5).

$$\mu(\mathcal{C}) = \frac{2}{|\mathcal{C}|(|\mathcal{C}|-1)} \sum_{u,v \in \mathcal{C}, u \neq v} \text{sim}(u, v) \quad (5)$$

Here, $u, v \in \mathcal{C}$ are two distinct terms in $\mathcal{C}$. Because a cluster containing relevant terms will be more coherent than a cluster containing distractor terms, the attacker selects the cluster with the highest coherence as the relevant cluster. Finally, we find the term from the entire vocabulary that is closest to the centroid of the cluster as the guess $\hat{A}$ of the original user query A. We define *hit rate* to be the proportion of the queries that we disclose via the clustering attack. Figure 4 shows the hit rates for the clustering attacks under different numbers of distractor terms.

From Figure 4 left we see that the hit rate is high when we do not use any distractor terms. In this case, the set of candidate terms consists purely of related terms X_i. We see that if we cluster all the related terms into one cluster ($k = 1$) we can easily pick the original query A by measuring the similarity to the centroid of the cluster. The hit rate drops when we add noise to the word embeddings, but even with the highest level of noise, we see that it is possible to discover the original query in 19% of the time. However, the hit rate drops significantly for all levels of noise when we add distractor terms as shown in the middle and right plots in Figure 4.

Hit rate is maximum when we set $k = 2$, which is an ideal choice for the number of clusters considering the fact that we have two groups of terms (related terms and distractors) among the candidates. Increasing k also increases the possibility of further splitting the related terms into multiple clusters thereby decreasing the probability of discovering the original query from a single cluster. We see that hit rates under no or medium levels of noise drops when we increase the number of distractor terms from 20 to 40, but the effect on high-level noise added candidates is less prominent. This result suggests that we could increase the number of distractor terms while keeping the level of noise to a minimum. Examples of terms discovered by clustering attacks are shown in Appendix B, and the trade-off between reconstructability and hit rate is investigated in Appendix C.

5 Human Evaluation

To empirically evaluate the difficulty, not only for a search engine as done in previous sections, but even also for human attackers to predict the original query given the related and distractor terms, we devise a *query prediction game*, where a group of human attackers are required to predict the original query from the related and distractor terms suggested by the proposed method. A group of 63 graduate students (all native English speakers) participated in this experiment. The query prediction game is conducted in two stages. In the first stage, we randomly shuffle the related and distractor term sets extracted by the proposed method for a hidden query. Human attackers are unaware as to which of the terms are related to the original user-query and which are distractors. A human attacker has a single guess to predict the user-query and wins only if the original query is correctly predicted. If the human attacker fails at this first step, then we remove all distractor terms and display only the related terms to the human attacker. This is expected to significantly simplify the prediction task because now the candidate set is smaller, does not contain distractor terms and the human attacker has already had a shot at the prediction. The human attacker then has a second chance to predict the original query from the related set of terms. If the human attacker correctly predicts the original query in the second stage, we consider it to be a winning case. Otherwise, the current round of the game is terminated and the next set of terms are shown to the human attacker. Winning rate is defined as the number of games won by the human attackers, where the original user query was correctly predicted.

Figure 5 shows the winning rates for the first and second stages of the query prediction game against the obfuscity of the queries. All queries selected for the prediction game have reconstructability scores greater than 0.3, which indicates that the search results for the original query can be accurately reconstructed from the related terms shown to the human attackers. From Figure 5, we see that the winning rate for the first stage is lower than that for the second stage, indicating that it is easier for

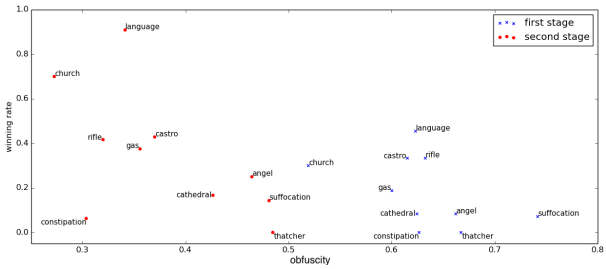

Figure 5: Winning rate vs. obfuscity for the first and second stages of the query prediction game

humans to guess the original query when the distractor terms are removed. Moreover, we see a gradual negative correlation between hit rate and obfuscity. This shows that more obfuscatory the terms are, it becomes difficult even for the human attackers to predict the original query, which is a desirable property for a query obfuscation method.

6 Related Work

One of the early incidents of query logs leaking private information in the public domain is the AOL's release of query log data in 2006.[4] Following this incident various methods have been proposed to obfuscate user queries such as token-based hashing (Kumar et al., 2007) and query-log bundling (Jones et al., 2008). However, in these approaches obfuscation happens only at the Web search engine's side without any intervention by the users, and the users must trust the good intentions of the search engine with respect to the user privacy. Moreover, (Kumar et al., 2007) showed that hashing alone *does not* guarantee user privacy.

Accessing Web search engines via an anonymised proxy server such as the onion routing (Goldschlang et al., 1999), TOR (Dingledine et al., 2004), Dissent (Corrigan-Gibbs and Ford, 2010) or RAC (Mokhtar et al., 2013) is a popular strategy employed by common users. The goal is to prevent the search engine link the queries issues by a user to his or her user profile. Anonymised search engines such as duckduckgo, Qwant, Swisscows provide privacy-oriented alternatives to Web users where the IP addresses, search profiles, location information etc. related to the users are kept anonymised. On the other hand, the query obfuscation method we propose in this paper can be used in conjunction with other anonymisation techniques in existing private-browsing browser plug-ins/add-ons and search portals to further increase the level of privacy.

Obfuscation-based private web search (Balsa et al., 2012) includes dummy keywords to prevent search engines from guessing users' query intent. Several browser add-ons that automatically append unrelated fake terms have been developed such as TrackMetNot (Howe and Nissenbaum, 2009), OptimiseGoogle, Google Privacy, Private Web Search tool (Saint-Jean et al., 2007) and GooPIR (Domingo-Ferrer et al., 2009). Although this approach is similar to our proposal to append user queries with distractor terms, those prior proposals have relied on pre-compiled ontologies (Petit et al., 2014) such as the WordNet or queries issued by other users shared via a peer network. Such approaches have scalability issues because most named entities that appear in search queries do not appear in the WordNet and it is unlikely that users would openly share their keywords to be used by their peers. Recently, outside IR, obfuscation has been applied successfully for anonymising users in social media platforms (Masood et al., 2018; Papadopoulos et al., 2013).

The goal in Private Information Retrieval (Yekhanin, 2010) is to retrieve data from a database without revealing the query but only some encrypted or obfuscated version of it (Ostrovsky and Skeith, 2007; Chor et al., 1997). For example, in hompmophic encryption-based methods the user (client) submits encrypted keywords and the search engine (server) performs a blinded lookup and returns the results again in an encrypted form, which can then be decrypted by the user. Embellishing queries with decoy terms further protects the privacy of the users. PIR has been applied in recommender systems (Gupta et al., 2016) and public data (Wang et al., 2017). However, unlike our proposed method, PIR methods assume search engines to accommodate the client side encryption methods, which is a critical limitation because modern commercial Web search engines do not allow this.

7 Conclusion

We proposed a method to obfuscate queries sent to a Web search engine by decomposing the query into a set of related terms and a set of distractor terms. We empirically showed that the proposed query obfuscation method is robust against a k-means clustering attack.

[4] https://tinyurl.com/y9qx9ufz

Appendix

A Proof of Theorem 1

In this section, we derive the relationship between obfuscity and reconstructability. Because obfuscity can be increased arbitrarily by increasing the distractor terms, in this analysis, we ignore distractor terms. This can be seen as a lower-bound for the obfuscity that can be obtained, without using any distractor terms. We first discuss the case where we have only one related term (i.e. $n = l = 1$) and then consider $l > 1$ reconstructability case.

A.1 $n = l = 1$ case

Let us consider the case where $n = 1$. Here, for a given query A, we have only a single related term $X = X_1$. In this case, $l = 1$, and we consider all documents retrieved using X as relevant for A. We first note that reconstructability, ρ, can be written as,

$$\rho = \frac{|\mathcal{D}(A) \cap \mathcal{D}'(A)|}{|\mathcal{D}(A)|} \tag{6}$$

from the definition of reconstructability.

Because we have a single noisy related term X, we have $\mathcal{D}'(A) = \mathcal{D}(X)$. By substituting this in (6), we get

$$\rho = \frac{|\mathcal{D}(A) \cap \mathcal{D}(X)|}{|\mathcal{D}(A)|}. \tag{7}$$

If we consider the co-occurrence context of two terms to be the document in which they co-occur, and divide the numerator and denominator in (7) by the total number of documents indexed by the search engine, then (7) can be written as a conditional probability as in (8).

$$\rho = \frac{p(A, X)}{p(A)} = p(X|A) \tag{8}$$

Theorem 2.2 in (Arora et al., 2016) provides a useful connection between the probability of a word (or the joint probability of two words) and their word representations, which we summarise below.

$$\log p(A, X) = \frac{||v(A) + v(X)||_2^2}{2d} - 2\log Z \pm \epsilon \tag{9}$$

$$\log p(A) = \frac{||v(A)||_2^2}{2d} - \log Z \pm \epsilon \tag{10}$$

Here, Z is the partition function and ϵ is the approximation error. (10) shows the relationship between the norm of the embedding of a word and the frequency of that word in a corpus, whereas (9) shows the relationship between the norm of the addition of the embeddings of two words and the co-occurrence frequency of those two words in a corpus. Both these relations are proved by Arora et al. (2016) and we would like to direct the interested readers to the original paper for the detailed proofs.

Next, by taking the logarithm of both sides in (8) we obtain,

$$\begin{aligned}\log \rho &= \log p(A, X) - \log p(A) \\ &= \frac{||v(X)||_2^2 + 2v(X)^\top v(A)}{2d} - \log Z \end{aligned} \tag{11}$$

Obfuscity for a single query term X can be computed using the cosine similarity as follows:

$$\alpha = 1 - \frac{v(A)^\top v(X)}{||v(A)||_2 \, ||v(X)||_2} \tag{12}$$

By substituting (12) in (11) we get,

$$\log \rho = \frac{||v(X)||_2^2}{2d} + \frac{(1-\alpha)\,||v(A)||_2\,||v(X)||_2}{d} - \log Z. \tag{13}$$

Because A is a given query, $v(A)$ is a constant. Moreover, if we assume that different related terms X_i have similar norms, (from (10) it follows that such related terms must have similar frequencies of occurrence in the corpus), then from (13) we see that there exists a linear inverse relationship between $\log \rho$ and α. Because logarithm function is monotonically increasing, (13) implies an inverse relationship between ρ and α.

A.2 $n = l > 1$ case

Let us now extend the relationship given by (13) to the case where we consider a document to be relevant if it can be retrieved from all of the n related terms. In other words, we have $l = n$ reconstructability in this case. Because each search result is retrieved by all l terms, we have

$$\mathcal{D}'(A) = \cap_{i=1}^{l} \mathcal{D}(X_i). \tag{14}$$

Reconstructability can be computed in this case

as follows:

$$\rho = \frac{p(A, X_1, X_2, \ldots, X_k)}{p(A)}$$
$$= p(X_1, X_2, \ldots, X_l | A)$$
$$\approx \prod_{i=1}^{l} p(X_i | A) \tag{15}$$

In (15) we have assumed that the related terms are mutually independent given the query A.

Let us take the logarithm on both sides of (15), and use (9) and (10) in the same manner as we did in Section A.1 to derive the relationship given by (16).

$$\log \rho = \frac{1}{2d} \sum_{i=1}^{l} ||v(X_i)||_2^2 + \frac{1}{d} \sum_{i=1}^{l} v(A)^\top v(X_i) - \log Z \tag{16}$$

In the $n = l$ case, obfuscity can be computed as follows:

$$\alpha = 1 - \frac{1}{l} \sum_{i=1}^{l} \frac{v(A)^\top v(X_i)}{||v(A)||_2 \, ||v(X_i)||_2} \tag{17}$$

Let us further assume that all related terms $X_1, X_2, \ldots, X_l$ occur approximately the same number of times in the corpus. From (10) it then follows that $||v(X_i)||_2 = c$ for $i = 1, 2, \ldots, l$ for some $c \in \mathbb{R}$. By plugging (17) in (16), and using the approximation $||v(X_i)||_2 = c$ we arrive at the relationship between ρ, α, and l given by (18).

$$\log \rho = \frac{cl}{2d} \left(c + 2(1 - \alpha) \, ||v(A)||_2\right) - \log Z \tag{18}$$

□

A.3 General Case

In the general case of l-reconstructability, we will have a subset of $l \leq n$ related terms retrieving each document. Exact analysis of this case is hard, and the reconstructability given by (18) must be considered as a lower-bound for this general case because we will still be able to reconstruct the search results using $\binom{n}{l}$ subsets of l related terms selected from a set of n related terms. Obfuscity can be arbitrarily increased without affecting the reconstructability by simply increasing the number of distractor terms. However, doing so would increase the burden on the search engine and is not recommended. In our experiments, we find that 20-40 distractor terms to be adequate to provide a good balance between obfuscity and efficiency.

The theoretical analysis presented in Section A depends on the relationships given by (9) and (10) for joint and marginal probabilities of unigram co-occurrences, originally proved by Arora et al. (2016). However, these relationships were later extended to cover co-occurrences of higher-order n-grams by Bollegala et al. (2018), who showed that the squared sum of embeddings of constituent unigrams in an n-gram phrase is proportional to the logarithm of the joint probability of those unigrams. On the other hand, Arora et al. (2017) showed that the inverse frequency-weighted average of unigram embeddings to be a competitive alternative to word-order sensitive supervised recurrent models for the purpose of creating phrase embeddings. Therefore, the relationship given in (18) still holds both theoretically and empirically in the case of multi-word queries, enabling us to extend the proposed method to multi-word phrasal queries.

B Terms discovered in clustering attacks

We show the terms discovered by clustering attacks for two example queries, *Hitler* (Table 1) and *mass murder* (Table 2) using a relatively small (< 10) distractor terms. We see terms that are related to the original queries can be accurately identified from the word embeddings. Moreover, by adding a high-level of noise to the embeddings, we can generate distractor terms that are sufficiently further from the original queries. Consequently, we see that both obfuscity and reconstructability are relatively high for these examples. Interestingly, the clustering attack is unable to discover the original queries, irrespective of the number of clusters produced.

C Trade-off between Reconstructability and the Hit Rate in Clustering Attacks

If the terms sent to the search engine are related to the original query, we will be able to accurately reconstruct the search results. However, this increases the risk of an adversary correctly guessing the query. Hit rate was defined as the fraction of the user queries correctly predicted by the clustering attack and is a measure of the robustness of the proposed method. Therefore, a natural question is *what is the relationship between the reconstructability and the hit rate*.

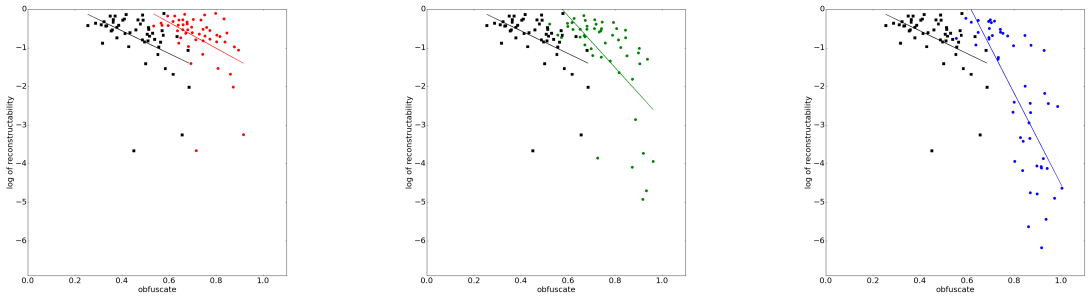

Figure 6: Relationship between obfuscity and reconstructability under different levels of added noise and with 20 distractor terms (left: no-noise, middle: medium-level of noise, and right: high-level of noise). Base reconstructability scores for no noise and no distractor terms are super imposed in black boxes.

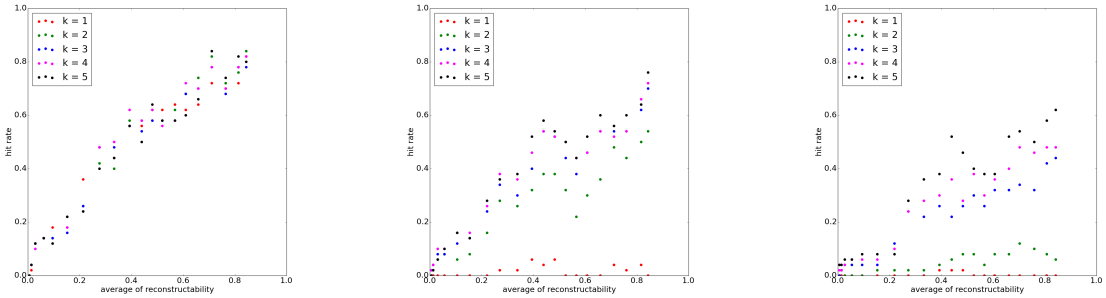

Figure 7: Hit-rate shown against reconstructability for k-means attacks with 0 (left), 60 (middle) and 120 (right) distractor terms.

Query	Hitler
noise	high-level
related terms	nazi, führer, gun, wehrmacht, guns, nra, pistol, bullets
obfuscity	0.867
reconstructability	0.831
Clustering Attack	Revealed Query
k=1	motagomery
k=2	albany, george
k=3	smith, albany
k=4	smith, fresno
k=5	rifle, albany

Table 1: Terms revealed by the clustering attacks for the query *Hitler*. Clustering attack with different number of clusters (k) *does not* reveal the original query.

Query	mass murder
noise	high-level
related terms	terrorism, killed, wrath, full-grown
obfuscity	0.789
reconstructability	0.747
Clustering Attack	Revealed Query
k=1	richmond
k=2	fremont, death
k=4	pasadena, words
k=4	pasadena, words
k=5	pasadena, anderson

Table 2: Terms revealed by the clustering attacks for the query *mass murder*. We see that the query nor its two tokens are revealed by the clustering attacks with different k values.

To empirically study this relationship, we randomly select 109 user-queries and add Gaussian noise with zero-mean and standard deviations 0 (no noise), 0.6, 1.0, 1.4 and 1.8. In each case, we vary the number of distractor terms 0-120 and apply k-means clustering attacks with $k = 1, 2, 3, 4$ and 5.[5] To conduct a conservative evaluation, we consider the terms in the vocabulary closest to the respective centroids in all clusters and not only the most coherent one as in Section 4.2. If the original query matches any of those k terms, we consider it to be a hit (e.g. to be revealing the original query). We randomly sample data points from even intervals of reconstructability values and plot in Figure 7.

We see a positive relationship between the reconstructability and the hit rate in all figures. This indicates a trade-off between the reconstructabil-

[5] In total, for a fixed k-value and the number of distractor terms, we have 545 clustering attacks.

ity and the hit rate, which shows that if we try to increase the reconstructability by selecting more relevant keywords to the original user-query, then it simultaneously increases the risk of the search engine discovering the query via a clustering attack. Moreover, we see that when we increase the number of distractor terms the hit rate drops for the same value of reconstructability. This result shows that in order to overcome the trade off between the reconstructability and the hit rate we can simply increase the number of distractor terms, thereby making the query obfuscation method more robust against clustering attacks. Moreover, the drop due to distractor terms is more prominent for the $k = 1$ attacks when we have distractor terms compared to that when we do not have distractor terms. This is because both related and distractor terms will be contained in this single cluster from which it is difficult to guess the original user-query. Therefore, multiple clusters are required for a successful k-means clustering attack.

Overall, the hit rate drops in the order $k = 5$, $k = 3$ and $k = 2$ when we increase the number of distractor terms. This result suggests that if one wants to increase the hit rate, then an effective strategy is to increase the number of clusters because we consider it to be a hit if the user-query is found via any of the clusters. Intuitively, if we form more clusters and pick all terms from the vocabulary closest to any one of the centroids, then the likelihood of predicting the original user-query increases with the number of clusters formed. However, in practice, we will need to further select one term from all the clusters. Nevertheless, we can consider the hit rate obtained in this manner to be a more conservative estimate, whereas in reality it will be less and therefore be more robust against attacks.

References

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2016. A latent variable model approach to pmi-based word embeddings. *Transactions of the Association for Computational Linguistics*, 4:385–399.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *Proc. of ICLR*.

Ero Balsa, Carmela Troncoso, and Claudia Diaz. 2012. Ob-pws: Obfuscation-based private web search. *2012 IEEE Symposium on Security and Privacy*.

Danushka Bollegala, Yuichi Yoshida, and Ken-ichi Kawarabayashi. 2018. Using k-way Co-occurrences for Learning Word Embeddings. In *Proc. of AAAI*, pages 5037–5044.

Claudio Carpineto and Giovanni Romano. 2012. A survey of automatic query expansion in information retrieval. *Journal of ACL Computing Surveys*, 44(1):1 – 50.

B Chor, N. Gilboa, and M. Naor. 1997. Private information retrieval by keywords. Technical report, Department of Computer Science, Technion, Israel Institute of Technology.

Silvio Cordeiro, Carlos Ramisch, Marco Idiart, and Aline Villavicencio. 2016. Predicting the compositionality of nominal compounds: Giving word embeddings a hard time. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1986–1997, Berlin, Germany. Association for Computational Linguistics.

H. Corrigan-Gibbs and B. Ford. 2010. Dissent: accountable anonymous group messaging. In *Proc. of CCS*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proc. of NAACL-HLT*.

R. Dingledine, N. Mathewson, and P. Syverson. 2004. Tor: The second generation onion router. In *Proc. of the Usenix Security Symposium*.

Josep Domingo-Ferrer, Agusti Solanas, and Jordi Castella-Roca. 2009. $h(k)$-private information retrieval from privacy uncooperative queryable databases. In *Proc. of Online Information Review*, volume 33, pages 720–744.

Arthur Gervais, Reza Shokri, Adish Singla, Srdjan Capkun, and Vincent Lenders. 2014. Quantifying web-search privacy. *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security - CCS '14*.

D. Goldschlang, M. Reed, and P. Syverson. 1999. Onion routing. *Communications of the ACM*, 42(2).

Trinabh Gupta, Natacha Crooks, Whitney Mulhern, Srinath Setty, Lorenzo Alvisi, and Michael Walfish. 2016. Scalable and private media consumption with popcorn. In *13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16)*, pages 91–107.

Kazuma Hashimoto and Yoshimasa Tsuruoka. 2016. Adaptive joint learning of compositional and non-compositional phrase embeddings. In *Proc. of ACL*, pages 205–215.

Chuan He, Cong Wang, Yi-Xin Zhong, and Rui-fan Li. 2008. A survey on learning to rank. In *Proc. of the 7th Intl. Conf. on Machine Learning and Cybernetics*, pages 1734 – 1739.

D. C. Howe and H. Nissenbaum. 2009. Trackmenot: Resisting surveillance in web search. Lessons from the Identity Train: Anonymity, Privacy and Identity in a Networked Society.

Rosie Jones, Ravi Kumar, Bo Pang, and Andrew Tomkins. 2008. Vanity fair: Privacy in querylog bundles. In *Proc. of CIKM*, pages 853–862.

Ravi Kumar, Jasmine Novak, Bo Pang, and Andrew Tomkins. 2007. On anonymizing query logs via token-based hashing. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 629–638, New York, NY, USA. ACM.

Rahat Masood, Dinusha Vatsalan, Muhammad Ikram, and Mohamed Ali Kaafar. 2018. Incognito: A method for obfuscating web data. In *Proceedings of the 2018 World Wide Web Conference*, pages 267–276. International World Wide Web Conferences Steering Committee.

Tomas Mikolov, Kai Chen, and Jeffrey Dean. 2013. Efficient estimation of word representation in vector space. In *Proc. of International Conference on Learning Representations*.

George A. Miller. 1995. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39 – 41.

S. Ben Mokhtar, G. Berthou, A. Diarra, V. Quéma, and A. Shoker. 2013. Rac: A freerider-resilient scalable, anonymous communication protocol. In *Proc. of ICDCS*.

R Ostrovsky and W. I. Skeith. 2007. A survey of single-database pir: techniques and applications. In *Proc. of Public Key Cryptography (PKC)*, volume 4450, pages 393–411.

Panagiotis Papadopoulos, Antonis Papadogiannakis, Michalis Polychronakis, Apostolis Zarras, Thorsten Holz, and Evangelos P Markatos. 2013. K-subscription: Privacy-preserving microblogging browsing through obfuscation. In *Proceedings of the 29th Annual Computer Security Applications Conference*, pages 49–58. ACM.

Marius Pasca. 2007. Organizing and searching the world wide web of facts-step two: Harnessing the wisdom of the crowds. In *WWW 2007*, pages 101–110.

Marius Pasca. 2014. Queries as a source of lexicalized commonsense knowledge. In *Proc. of EMNLP*, pages 1081–1091.

Jeffery Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: global vectors for word representation. In *Proc. of EMNLP*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL-HLT*.

Albin Petit, Sonia Ben Mokhtar, Lionel Brunie, and Harald Kosch. 2014. Towards efficient and accurate privacy preserving web search. *Proceedings of the 9th Workshop on Middleware for Next Generation Internet Computing - MW4NG '14*.

Adam Poliak, Pushpendre Rastogi, M. Patrick Martin, and Benjamin Van Durme. 2017. Efficient, compositional, order-sensitive n-gram embeddings. In *Proc. of EACL*.

Matthew Richardson. 2008. Learning about the world through long term query logs. *ACM Transactions on the Web*, 2(4).

Eldar Sadikov, Jayant Madhavan, Lu Wang, and Alon Halevy. 2010. Clustering query refinements by user intent. In *WWW 2010*, pages 841–850.

F. Saint-Jean, A. Johnson, D. Boneh, and J. Feigenbaum. 2007. Private web search. In *Proc. of ACM Workshop on Privacy in Electronic Society*, pages 84–90.

Rodrygo L. T. Santos, Craig Macdonald, and Idah Ounis. 2010. Exploiting query reformulations for web search result diversification. In *WWW 2010*, pages 881–890.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Aritificial Intelligence Research*, 37:141 – 188.

Frank Wang, Catherine Yun, Shafi Goldwasser, Vinod Vaikuntanathan, and Matei Zaharia. 2017. Splinter: Practical private queries on public data. In *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, pages 299–313.

Sergey Yekhanin. 2010. Private information retrieval. *Commun. ACM*, 53(4):68–73.

Mo Yu and Mark Dredze. 2015. Learning composition models for phrase embeddings. *Transactions of the Association for Computational Linguistics*, 3:227–242.