KwikBucks: Correlation Clustering with Cheap-Weak and Expensive-Strong Signals

Sandeep Silwal[†1], **Sara Ahmadian**[2], **Andrew Nystrom**[2], **Andrew McCallum**[2], **Deepak Ramachandran**[*2], **Mehran Kazemi**[*2]
[1] MIT, [2] Google Research
silwal@mit.edu, {sahmadian, nystrom, mccallum, ramachandrand, mehrankazemi}@google.com

Abstract

For text clustering, there is often a dilemma: one can either first embed each examples independently and then compute pair-wise similarities based on the embeddings, or use a cross-attention model that takes a pair of examples as input and produces a similarity. The former is more scalable but the similarities often have lower quality, whereas the latter does not scale well but produces higher quality similarities. We address this dilemma by developing a clustering algorithm that leverages the best of both worlds: the scalability of former and the quality of the latter. We formulate the problem of text clustering with embedding-based and cross-attention models as a novel version of the Budgeted Correlation Clustering problem (BCC) where along with a limited number of queries to an expensive oracle (a cross-attention model in our case), we have unlimited access to a cheaper but less accurate second oracle (embedding similarities in our case). We develop a theoretically motivated algorithm that leverages the cheap oracle to judiciously query the strong oracle while maintaining high clustering quality. We empirically demonstrate gains in query minimization and clustering metrics on a variety of datasets with diverse strong and cheap oracles.

1 Introduction

Modern ML techniques have made incredible advances at the cost of needing resource-intensive models (Sharir et al., 2020). Many recent approaches are so resource-intensive that despite amazing accuracy, they are infeasible to be scaled as-is in practical usage. The total effect of all such deployments on energy usage is also a major sustainability concern (Wu et al., 2022).

With the increased cost in querying ML models, the cost of obtaining similarities between objects of different types (texts, images, etc.) has also substantially increased. In this paper, we aim to answer a challenging question when working with such costly similarity measure models: how can we group similar objects together when similarities of objects are obtained via expensive queries? This problem can be naturally cast as a popular and versatile clustering framework, named *Correlation Clustering (CC)*, which has been extensively studied over the past 15+ years (Bonchi et al., 2022): given similarities between arbitrary objects represented as a graph, CC minimizes a natural objective that attempts to cluster together similar vertices while simultaneously separating dissimilar ones. The high cost of querying large ML models motivates the use of the Budgeted CC (BCC) setting studied in (Bressan et al., 2019; García-Soriano et al., 2020a) where relationships between nodes are determined by making a limited number of queries to an oracle, e.g. a large ML model.

We posit that in many practical settings, coarse but efficient approximations of an expensive model can be obtained through substantially cheaper but weaker models. These weaker models can be used as a guide to spend the query budget for the expensive model more carefully. A motivating example, which heavily inspires our work, is in text clustering where one wishes to obtain similarity signals from the latest highly-accurate *cross-attention (CA)* language models (e.g., (Brown et al., 2020; Thoppilan et al., 2022)), but may be hindered by the computational burden as obtaining each pair-wise similarity between data points requires an inference call to the model, giving rise to a worse case $O(n^2)$ inference calls, where n is the number of data points. *Embedding based models* (e.g., (Mikolov et al., 2013; Devlin et al., 2018) can come to the rescue as they require only $O(n)$ inference calls to obtain embedding vectors for each data point that can then be used for fast similarity computation. While embedding models typically produce substantially lower quality similarity signals than CA

[*]Co-advised. [†]Work done while interning at Google.

Proceedings of The Fourth Workshop on Simple and Efficient Natural Language Processing (SustaiNLP), pages 1–31
July 13, 2023 ©2023 Association for Computational Linguistics

models (see, e.g., (Menon et al., 2022)), they can still provide a good approximation to guide where the budget for the CA model should be spent.

Inspired by the above, we introduce a variant of BCC where, along with a limited number of queries to an expensive oracle, we also have unlimited access to a cheaper but less accurate second oracle. We develop an algorithm dubbed **`KwikBucks`** that extends the well-known **Kwik**Cluster algorithm to **bu**dgeted CC with **c**heap-wea**k** and expensive-**s**trong signals. `KwikBucks` uses the weak signal as a guide to minimize the number of calls to the strong signal. Under the assumption that the weak signal returns a strict superset of the strong signal edges, our algorithm approximately matches the performance of `KwikCluster`, i.e., a 3-approximation, using a small number of queries to the expensive model. In our experiments, we strengthen our theoretical modelling with several well-motivated optimizations and demonstrate that `KwikBucks` manages to produce high quality clusterings with only a small number of queries to the expensive oracle even when there is only a weak correlation between the weak and strong signal.

We conduct extensive experiments with multiple datasets to evaluate the performance of `KwikBucks` over natural extensions of previous algorithms for closely-related problems. `KwikBucks` recovers the best clustering solution with a much smaller strong signal budget than the alternatives, and it finds asymptotically better solutions in many cases. `KwikBucks` is also robust to the choice of weak signal oracle across different dataset settings and obtains significant improvements over five baselines — **64%** relative improvement in clustering quality (measured in terms of F1 score) when averaging over 9 datasets, and over $>$ **3.5x** reduction in query complexity compared to the best baseline.

1.1 Related Work

Our paper spans correlation clustering, clustering with budget constraints, and learning from multiple annotators. For brevity, we focus on the closely related key works in these areas.

Correlation clustering is one of the most well studied graph clustering problems and has been actively researched over the past 15+ years (see the book (Bonchi et al., 2022)). It has numerous applications in ML and beyond, including spam detection (Ramachandran et al., 2007; Bonchi et al., 2014), social network analysis (Bonchi et al., 2015; Tang et al., 2016), entity resolution (Getoor and Machanavajjhala, 2012), and many others (Gionis et al., 2005; Hassanzadeh et al., 2009; Cohen and Richman, 2002; Kim et al., 2011). (Bansal et al., 2004) introduced and gave the first constant factor approximation for complete graphs (see Def. 1). Variants include incomplete signed graph (Bansal et al., 2004; Ailon et al., 2008), where the problem is APX-Hard (Demaine et al., 2006), and weighted graphs (Charikar et al., 2005), where it is Unique-Games hard (Chawla et al., 2006).

Clustering under budget constraints studies the problem of a limited number of pairwise similarity queries. In this setting, a line of work looked at spectral clustering on partially sampled matrices: Fetaya et al. (2015) in general setting, and Shamir and Tishby (2011) and Wauthier et al. (2012) for bi-partitioning. The most relevant works to our paper are those of García-Soriano et al. (2020a) and Bressan et al. (2021) who devised algorithms for correlation clustering that given a budget of Q queries attain a solution whose expected number of disagreements is at most $3\cdot$ OPT $+O(\frac{n^3}{Q})$, where OPT is the optimal cost for the instance. Another closely related line of work studies "same-cluster" queries for various clustering problems including CC (Ailon et al., 2018; Saha and Subramanian, 2019). The differences between these works and ours are (1) they assume *all* $\binom{n}{2}$ similarity queries are already known in advance whereas we must query the strong signal to obtain similarities, (2) their queries give access to the *optimal* clustering, whereas we only query for edge signs.

Learning from multiple annotators considers cost-effective learning from multiple annotators where the cost of a labeler is proportional to its overall quality. The most relevant work to our setting is (Guha et al., 2015) as it considers hierarchical clustering which uses lightweight similarity scores to identify candidate pairs with high similarity (detailed comparison in Section A). Ensemble approaches (Dietterich, 2000) are also relevant, but they require knowing the similarities in advance and do not apply to our budgeted setting. Lastly we survey additional related works on learning from multiple annotators as well as algorithms with predictions in Section A.

1.2 Preliminaries and Notation

The input of correlation clustering is a complete undirected graph $G = (V, E^+ \cup E^-)$ on $|V| = n$

vertices. E^+ and E^- represent the partitions of all possible $\binom{n}{2}$ edges where an edge $e = (u, v) \in E^+$ indicates that u and v are similar and $e = (u, v) \in E^-$ indicates that u and v are dissimilar. We simplify the notation to $G = (V, E = E^+)$ so any present edge is a positive edge and any missing edge is a negative edge. Additionally, we use m to denote the size of $|E|$ and $\Gamma(v) = \{u \mid (v, u) \in E\}$ to denote the neighborhood of vertex v.

A *clustering* is a partitioning $C = \{C_1, C_2, \cdots\}$ of V into disjoint subsets. Let $C_{v,u}$ denote the indicator variable if the vertices v and u are assigned to the same cluster. We study the min-disagreement formulation of the correlation clustering problem defined as follows (Bansal et al., 2004).

Definition 1 (Correlation Clustering (CC))**.** *Given a graph $G = (V, E)$, the objective of correlation clustering (CC) is to output a clustering C that minimizes:*

$$\sum_{e=(v,u)\notin E} C_{v,u} + \sum_{e=(v,u)\in E} (1 - C_{v,u}). \qquad (1)$$

`KwikCluster` (Ailon et al., 2008) is a well-known CC algorithm which proceeds by successively picking a vertex p, called a pivot, uniformly at random from the graph and forming a cluster $\{p\} \cup \Gamma(p)$. The algorithm removes this cluster and recurses on the remaining graph until all vertices are assigned to clusters. Based on the fact that the set of pivots is a maximal *independent set* constructed from a random order of vertices, Bonchi et al. (2013) suggests an equivalent algorithm that first constructs the independent set and then assigns any non-pivot to its first neighbor in the independent set. Both algorithms yield 3-approximation in expectation, however the second algorithm is more efficient as the assignment of non-pivots can be performed in parallel.

Despite practicality and simplicity of `KwikCluster` (and its variants), the algorithm assumes access to the full similarity graph G and is not feasible when similarity measures are expensive to acquire. We consider budget CC studied before by (García-Soriano et al., 2020b; Bressan et al., 2019) where there is a limit (budget) for the number of queries that can be made.

Definition 2 (Expensive / Strong Oracle)**.** *Given an edge e, the query $\mathcal{O}_S(e)$ outputs whether $e \in E$, i.e., e is a positive edge.*

Following the motivations provided in Section 1, we also introduce a second weaker oracle which is cheaper to query.

Definition 3 (Cheap / Weak Oracle)**.** *Given any vertex v, the query $\mathcal{O}_W(v)$ outputs a similarity score in $\mathbb{R}$ between v and every other vertex in V, where higher values indicate higher similarity*

We frequently refer to G as the *strong signal graph* and likewise a strong signal edge refers to an edge in E. We interchangeably use the terms signal or oracle, the terms strong and expensive signal, and also the terms weak and cheap signal.

2 Theoretical Modelling

We introduce an algorithm that leverages the cheap signal for strong signal query efficiency. Our goals are twofold: **(1)** Design a flexible algorithm paradigm which can adapt to incorporate constraints necessitated by practice, i.e., limited access to expensive queries, **(2)** Analyze the quality of the produced solution with respect to the CC objective (see equation 1). We first introduce a modelling assumption for the weak oracle for the purpose of theoretical analysis. While this results in a *different* but related weak oracle formulation compared to Definition 3, it lets us derive a robust algorithm design which we subsequently adapt to the more realistic setting of Definition 3.

First, we introduce a noise factor γ that determines the usefulness of a weak signal. $\gamma = 0$ corresponds to a perfect weak signal that exactly matches the strong signal and $\gamma = n$ corresponds to a completely uninformative weak signal.

Assumption 1. *For a fixed noise parameter $\gamma > 0$, the query $\mathcal{O}^{\gamma}_W(v)$ outputs a subset of V such that $\Gamma(v) \subseteq \mathcal{O}^{\gamma}_W(v)$ and $|\mathcal{O}^{\gamma}_W(v)| \leq (1 + \gamma)|\Gamma(v)|$.*

The existence of such a signal with a small γ, say $O(1/n)$ or $\gamma < 1$ might seem like a strong assumption for most applications. However, our experiments show that weak signal can actually provide predictive hints about the true underlying strong signal graph. More precisely, given vertex v, we order V with respect to the weak signal, and observe that *true* strong signal neighbors of v are often ranked higher. Thus, returning the most similar vertices for an input node captures many of the true strong signal neighbors of v and mimics the clean abstraction of Assumption 1 (See Appendix F.6 for further empirical justification).

Using the above characterization, we next explain the high level ideas of our algorithm `KwikBucks` (Algorithm 1). It is inspired by a variant of `KwikCluster` (Bonchi et al., 2013) adapted

Algorithm 1 KwikBucks (Our Algorithm)

Require: A bound on sampled vertices, t, the strong signal budget, Q.

$P \leftarrow \texttt{GetPivots}(t, Q)$
return $\texttt{AssignToClusters}(P, V \setminus P, Q)$

Algorithm 2 $\texttt{AssignToClusters}(P, U, Q)$

Require: List of pivots, P, a vertex set, U, remaining strong signal budget, Q.

$A \leftarrow \emptyset$ {the set of singletons}
$C_p \leftarrow \{p\}$ {cluster for any pivot $p \in P$}
while $Q > 0$ and $U \neq \emptyset$ **do**
  $v \leftarrow$ extract first vertex of U
  $N_v \leftarrow \texttt{WeakFilter}(v, P)$
  $p \leftarrow \texttt{FirstNeighbor}(v, N_v, Q)$
  if $p \neq \emptyset$ **then**
    $C_p = C_p \cup \{v\}$
  else
    $A \leftarrow A \cup \{v\}$
$A \leftarrow A \cup U$
return $\cup_{p \in P} C_p \cup_{v \in A} \{v\}$

Algorithm 3 $\texttt{GetPivots}(t, Q)$

Require: A bound on the number of sampled vertices, t, the remaining strong signal budget, Q.

$\{v_1, \ldots, v_t\} \leftarrow t$ sampled vertices
$P \leftarrow \{v_1\}$
for $i \geq 2$ **do**
  $N_i \leftarrow \texttt{WeakFilter}(v_i, P)$
  if $\texttt{FirstNeighbor}(v_i, N_i, Q) = \emptyset$ **then**
    $P \leftarrow P \cup \{v_i\}$
return P

Algorithm 4 $\texttt{FirstNeighbor}(v, N, Q)$

Require: Input vertex, v, an ordered list of vertices, N, the remaining strong signal budget, Q.

while $Q > 0$ and $N \neq \emptyset$ **do**
  $u \leftarrow$ extract first vertex from N
  $Q \leftarrow Q - 1$
  if $\mathcal{O}_S(v, u) = 1$ **then**
    return $\{u\}$
return $\emptyset$

to our two oracle setting. In this variant, we first pick pivots by forming a maximal independent set from a random ordering of vertices, and then assign non-pivots to their first neighbor (which must exist by maximality of the independent set). A naive extension of this algorithm can result in $\Omega(n^{1.5})$ queries to the strong signal (see 1). However, one may argue that by using the weak signal, we can prune the possible neighborhood of vertices which results in fewer strong signal queries.

While the weak signal can help us make smarter queries to the strong signal, we can still show that even for a weak signal with a small error rate, i.e., $\gamma = O(1)$, we still need $\Omega(n^2)$ queries to the strong signal when forming a maximal independent set (see 2). To circumvent this difficulty, we consider another modification of KwikCluster by Bonchi et al. (2013) where instead of picking a maximal independent set, we pick t vertices uniformly at random and then pick an independent set from them. The caveat of this approach is that some non-pivots may not have any neighbor in the chosen pivot set, and so these non-pivots are returned as singleton clusters. This results in an algorithm which returns a solution with cost at most $3OPT + O(n^2/t)$. We additionally modify this algorithm by incorporating the weak signal to further prune possible strong signal queries. While this algorithm has an additive error for correlation clustering cost, it helps us direct our queries to "impactful" portions of graph. We now have all the ingredients for describing our algorithm, KwikBucks.

KwikBucks first picks *all* pivots via random sampling as shown in GetPivots: each sampled vertex is added to the pivot set if it is not connected to the current subset of pivots that is trimmed down by WeakFilter (which uses the weak signal). Then, it continues to assign non-pivots to clusters, through AssignToClusters, which finds the first vertex (FirstNeighbor) in the subset of ordered pivots trimmed down by the weak signal via WeakFilter. If no such vertex exists, then the vertex is assigned to its own cluster, i.e., a singleton cluster. Note that having a small t (the number of sampled vertices) helps query efficiency by functionally reducing the set of vertices that the weak signal is applied to (both when selecting pivots and when assigning to pivots) and then further queried by the strong signal. This comes at the cost of a small additive error. Our next theorem formally bounds the number of queries and the effect of t (proof in Appendix B).

Theorem 1. *Under Assumption 1,* KwikBucks *uses* $n + t + 2\gamma tm/n + 2\gamma t^2 m/n^2$ *queries to* $\mathcal{O}_S$ *to achieve approximation* 3OPT + $O(n^2/t)$.

Our next corollary, considers the interesting case

Algorithm 5 `WeakFilter`(v, S)

Return $\mathcal{O}_W^{\gamma}(v) \cap S$

of a constant-size pivot set, i.e., $t = 1/\epsilon$, which will incur an additive error of εn^2. This can be thought as the 'right scale' as we make a mistake on only an ε fraction of all edges. We complement our corollary by presenting a matching lower bound in the appendix (Lemma 3) showing that $\Omega(n + d\gamma/\varepsilon)$ strong signal queries are necessary to obtain the guarantees of Corollary 1.

Corollary 1. *Let d be the average degree of the strong signal graph and suppose n is sufficiently large ($n > 1/\varepsilon$). We can achieve approximation $3\cdot$OPT $+\varepsilon n^2$ with $n + O(d\gamma/\varepsilon)$ queries to $\mathcal{O}_S$.*

3 The Final Empirical Algorithm

We now extend the algorithm for the idealized setting of Section 2 into a practical version of `KwikBucks` for general weak signals, i.e. Definition 3. While this version does not satisfy Assumption 1 and hence does not have similar approximability guarantees, it still retains some theoretical motivation (sketched below), and is empirically very successful (see Section 4).

The modifications we make for our practical algorithm are based on the following natural inductive bias: *'similar' edges according to the strong signal are likely to have a high weak signal similarity score*. At a high level, we incorporate this assumption throughout our algorithm design by ranking potential queries to the strong signal according to weak signal similarity values.

Weak Filter by Ranking The most noticeable change occurs for `WeakFilter`: In our theoretical modelling, it returns a subset of S which intersects with the noisy neighborhood returned by the cheap oracle. For the general weak signal version (Definition 3), we update the `WeakFilter` function to instead rank the vertices in S with respect to the weak signal similarity to v and then output the top k elements in S with the highest similarities (Algorithm 6)). Intuitively, for a suitable parameter k, the top k candidates capture many of the strong signal neighbors of v in S. Indeed, we empirically verify this in our experiments and show that predictive weak signals usually rank true strong signal neighbors much higher compared to a random ordering. For our experiments we fix $k = 100$ and perform ablation studies on this parameter.

Algorithm 6 `WeakFilterByRanking`(v, S, k)

Require: Input vertex v; set $S \subseteq V$

$w_i \leftarrow$ similarity of (v, u_i) for all $u_i \in S$ as computed by weak signal

Sort elements of S in decreasing w_i values

Return First k elements of new sorted order

To better understand the effect of this modification, consider `AssignClusters` where non-pivot vertices attempt to connect to a pivot. In our theoretical modeling and in the classical `KwikCluster` algorithm, each non-pivot vertex checks for a strong signal edge among the list of pivots in an ordering which is fixed for all vertices. This ordering can be thought of as the ordering inherited from `GetPivots`. In contrast, `WeakFilterByRanking` introduces a data adaptive ordering step where each non-pivot vertex can re-rank pivots based on weak signal similarities. As shown in Section 4, this has a sizeable impact on the empirical performance of our algorithm. In Section D.1, we explain these gains by introducing a natural data model that makes some well-motivated assumptions about the relationship between the strong and weak signal, as well as the inherent clusterability of the underlying graph. Under this model, we prove that the quality of the clustering after re-ranking is strictly better than for the unranked filter.

Further optimizations. We make three additional enhancements to `KwikBucks`. The first one simply sorts the non-pivots based on the maximum weak signal similarity to the pivots so that 'easier' non-pivots are assigned clusters first which improves query efficiency. The second one modifies the `WeakFilterByRanking` function slightly by increasing the weak signal similarity value between a non-pivot v and a pivot p if p has 'many' nearest neighbors (in weak signal similarity) of v already in its cluster. Finally, the last enhancement introduces a post-processing step where we potentially merge some clusters after our algorithm terminates. As shown in Section D.2, this optimization is motivated by a theoretical worst-case example for `KwikCluster`. The merging step proceeds by first curating a list of clusters to consider for merging based on the average weak signal value between the two clusters and we sample a small number of strong signal edges between potential clusters to merge to determine if the pair is suitable for merging. Each of these optimizations is described in detail in Section C.

4 Experiments

Datasets. We use 9 datasets, 8 publicly available and 1 proprietary internal. Each dataset exhibits different properties such as varying strong signal graph densities and diverse strong and weak signals to demonstrate the versatility of our method. We provide high-level descriptions here and refer to Section E for more details.

Four public datasets are comprised of text inputs: Stackoverflow (SOF) (Xu et al., 2017), SearchSnippets (Phan et al., 2008), Tweet (Yin and Wang, 2016) and AgNews (Rakib et al., 2020). For Stackoverflow and SearchSnippets, we use word2vec embedding similarities (Mikolov et al., 2013) as the cheap signal and a large cross-attention based language model as the strong signal. For Tweet and AgNews, BERT embedding similarities (Devlin et al., 2018) are the cheap signal; the strong signal of an input pair is the indicator variable of the two examples belonging to the same class plus a small amount of i.i.d. noise to prevent the formation of disconnected connected components, which is the 'easy' case for `KwikCluster` [1].

The other four public datasets are comprised of attributed graphs: Cora (Sen et al., 2008), Amazon Photos (Shchur et al., 2018), Citeseer (Sen et al., 2008), and Microsoft Medicine (Shchur and Günnemann, 2019). For Cora and Amazon photos, node embedding (learned using deep graph infomax (Velickovic et al., 2019)) similarities are the cheap signal; the strong signal is generated similarly to those of Tweet and AgNews. For Citeseer and Microsoft Medicine, node attribute similarities are the cheap signal and the existence/absence of edges in the graph is the strong signal.

Moreover, we report results on a large proprietary dataset based on the shopping reviews of a commercial website. We use internally developed (and finetuned) embedding based and cross-attention based language models for the cheap and expensive signals respectively; both models are based on the publicly available language models such as BERT and T5 (Raffel et al., 2020).

Baselines. Since our work is the first BCC algorithm which utilizes both strong and weak signals, we adapt algorithms from prior work, e.g. some which only use a strong signal, to our setting. We also propose several new algorithms as baselines.

Baseline 1: A variant of `KwikBucks` where we do not use the weak signal ordering computed in Algorithm 6 when checking for a strong signal edge between a node and a set of pivots. Rather we use the the order the pivots were picked.

Baseline 2: Algorithm presented in (García-Soriano et al., 2020b; Bressan et al., 2019). It follows the `KwikCluster` algorithm and uses the strong signal to query edges. If the query budget is depleted, the algorithm is terminated and any remaining vertices are returned as singletons.

Baseline 3 / 4: We compute a k-NN graph based on the weak signal to narrow down the set of all possible queries to a small set of relevant pairs. Each edge of the k-NN graph is re-weighted (either 0 or 1) based on the strong signal. Baseline 3 runs the classic spectral clustering algorithm and baseline 4 runs the vanilla `KwikCluster` algorithm to completion on this graph.

Baseline 5: This baseline is inspired by the baseline used in (García-Soriano et al., 2020b). We pick k random vertices and query their complete neighborhood using the strong signal. k is again chosen as high as possible within the allotted query budget. Instead of running an affinity propagation algorithm, which was already shown in (García-Soriano et al., 2020b) to be inferior to Baseline 2, we run the vanilla `KwikCluster` algorithm.

Evaluation metrics. We evaluate our algorithm and baselines based on the correlation clustering objective (equation 1). For the purpose of evaluating metrics, we use all edges of the strong signal graph in contrast to the duration of algorithm execution which we limit the access. In addition, we compute the **precision** and **recall** of edges of the strong signal graph. Given a clustering $\mathcal{C}$, its precision is defined as the ratio between the number of strong signal edges whose endpoints are together in the same cluster and the total number of pairs of vertices clustered together in $\mathcal{C}$. The recall is defined as the fraction of all strong signal edges whose vertices are clustered together in some cluster of $\mathcal{C}$; see equation 2 and 3. We combine the precision and recall into a single metric via the standard F_1 score

Parameter configurations. Our algorithm has two main parameters to select: t in Algorithm 3 corresponding to the number of vertices we select uniformly at random which is then pruned to form the set of pivots, and k in Algorithm 6 corresponding to the number of top vertices we select based on the weak-signal similarity for the strong signal

[1] One can easily show that in such a case the classical `KwikCluster` algorithm is able to recover OPT.

Table 1: F_1 values for a fixed budget of $3n$ to the expensive-strong signal, where n indicates the dataset size. For Citeseer and Medicine, we use a budget of $50n$ as they have substantially sparse graphs. Winners are in bold and second winners are underlined.

	SOF	**Search**	**Tweet**	**AgNews**	**Cora**	**Photos**	**Citeseer**	**Medicine**	**Internal**	**Avg.**
B1	$.13_{\pm.02}$	$.73_{\pm.15}$	$.02_{\pm.00}$	$\underline{.74_{\pm.01}}$	$.57_{\pm.08}$	$.44_{\pm.01}$	$.07_{\pm.01}$	$.02_{\pm.00}$	$.00_{\pm.00}$	.30
B2	$.28_{\pm.10}$	$\underline{.81_{\pm.12}}$	$.15_{\pm.07}$	$\underline{.74_{\pm.01}}$	$\underline{.58_{\pm.13}}$	$.53_{\pm.13}$	$.09_{\pm.01}$	$.03_{\pm.01}$	$\underline{.05_{\pm.04}}$	$\underline{.36}$
B3	$\underline{.33_{\pm.07}}$	$.70_{\pm.04}$	$\underline{.21_{\pm.03}}$	$.66_{\pm.04}$	$.54_{\pm.02}$	$\underline{.66_{\pm.05}}$	$.00_{\pm.00}$	$.00_{\pm.00}$	-	-
B4	$.01_{\pm.00}$	$.01_{\pm.00}$	$.03_{\pm.00}$	$.00_{\pm.00}$	$.01_{\pm.00}$	$.00_{\pm.00}$	$\mathbf{.46_{\pm.01}}$	$\underline{.25_{\pm.00}}$	$.00_{\pm.00}$	.08
B5	$.00_{\pm.00}$	$.00_{\pm.00}$	$.00_{\pm.00}$	$.00_{\pm.00}$	$.00_{\pm.00}$	$.00_{\pm.00}$	$.04_{\pm.01}$	$.00_{\pm.00}$	$.00_{\pm.00}$	.00
KwikBucks	$\mathbf{.72_{\pm.05}}$	$\mathbf{.92_{\pm.05}}$	$\mathbf{.28_{\pm.04}}$	$\mathbf{.87_{\pm.00}}$	$\mathbf{.82_{\pm.02}}$	$\mathbf{.83_{\pm.00}}$	$\underline{.41_{\pm.01}}$	$\mathbf{.29_{\pm.00}}$	$\mathbf{.14_{\pm.01}}$	**.59**

to query. We pick both these parameters in a data-driven manner. Thorough motivation and trade-offs associated with both parameters are presented in Section F.2; ablation studies of these parameters are provided in our empirical results. Lastly, we always reserve 10% of the query budget for performing the merge post processing step. If the main algorithm terminates with remaining budget, we correspondingly increase the merge post processing budget to incorporate this.

Results. We highlight key experimental themes and defer additional details to Appendix F.

Superior performance over baselines: Table 1 shows that our algorithm outperforms the baselines in terms of the F_1 metric: it consistently has the highest F_1 value for the fixed query budget result displayed in Table 1. For example for the SOF dataset, the best baseline has a **2.2x** factor smaller F_1 value. Figures 1(a),(b) show the CC objective and F_1 score as a function of the query budget for the SOF dataset. It shows that our algorithm achieves a higher F_1 score and a lower correlation clustering objective value with only $\approx 7 \cdot 10^3$ queries whereas the baselines require at least $25 \cdot 10^3$ queries to match KwikBucks with $7 \cdot 10^3$ queries, showing the efficacy of our algorithm with a **3.6x** reduction in query complexity. Intuitively, the weak signal allows us to make clustering progress much faster by directing the query budget to impactful strong signal queries after filtering using the weak signal. The results for other datasets are deferred to Figures 3 and 4 in the appendix which display qualitatively similar behaviour. The strong signal graphs of Citeseer and Medicine are quite sparse. Therefore for these datasets, the trivial clustering of all singletons already achieves a very low CC objective score. As argued above, in these cases the F_1 score is a much more meaningful measure of cluster quality. As shown in Figures 4, our algorithm achieves superior F_1 values compared to the baselines. Lastly we note that the performance of our algorithm stabilizes once it has exploited sufficiently many strong signal queries. We note that B3 is omitted from the CC objective value plots for clarity as it always had much higher objective value than other algorithms.

Relative performance of baselines is dataset dependent: As shown in Table 1, for many datasets such as Cora, Search, and AgNews, B2 is the best among our five baselines. However this does not generalize across all datasets. As shown in Figures 4, B3 is the best baseline (with respect to the F_1 score) for the Tweet and Photos datasets while B4 is the best baseline for the Citeseer and Medicine datasets. B4 can be a competitive baseline in the case where the strong signal graph is extremely sparse, such as in Citeseer (see Figure 4). This is because the weak signal k-NN graph is able to recover many relevant edges of the (sparse) graph if the weak signal is informative.

Varying weak signal performance: We perform addition weak signal ablation studies with the SOF and Search datasets. We replace the Word2Vec (W2V) embeddings used in our cheap oracle with tf-idf embeddings and fix all other components of the algorithm. Figure 1(c) and 7 show the performance of our algorithm on these datasets and in both cases, the algorithm's performance noticeably worsens. The intuitive answer for *why* this is the case is because the alternative weak similarities computed from tf-idf embeddings are worse than W2V embeddings at ranking strong signal neighbors. We empirically verify this claim. For every vertex v in the SOF dataset, we rank all other vertices in decreasing weak signal similarities to v. The average rank of the true strong signal neighbors of v is then computed and this value is plotted in a histogram for all vertices v in Figure 1(d). A

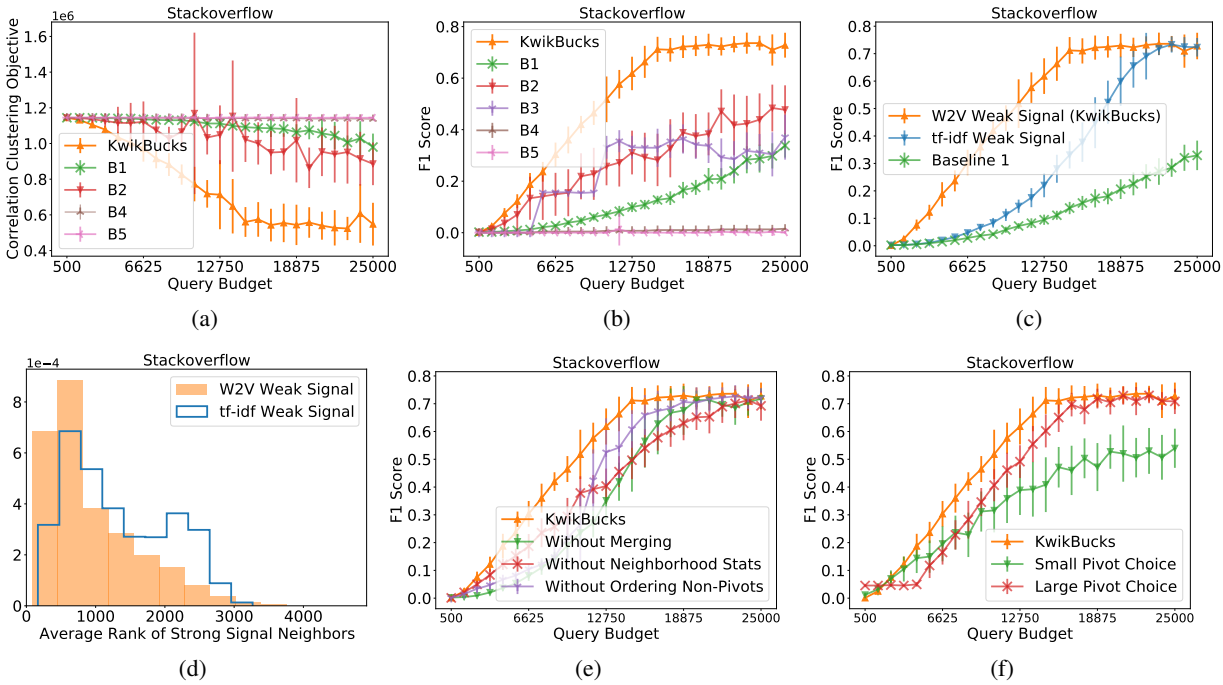


Figure 1: (a) and (b) represent the CC objective and the F1-scores for the Stackoverflow dataset across various query budgets. (c) compares performance across weak signals of various strength (Baseline 1 corresponds to a random weak signal). (d) represents a (normalized) histogram showing average rank assigned to actual strong signal neighbors by two different weak signals (lower is better). (e) represents an ablation study on some of the main components of the algorithm. (f) represents a sensitivity analysis to the parameter corresponding to the number of pivots selected.

'good' weak signal should rank actual strong signal neighbors much higher than non strong signal neighbors. Indeed we observe this to be the case for the W2V embeddings and this fact is qualitatively captured the aforementioned figures which show that W2V has superior F1 score plots. We also observe that even the weaker tf-idf embeddings still provide significant gains over not using a weak signal. Overall, these experiments along with Baseline B1 empirically verify that (1) the quality of the weak signal correlates with the performance of the algorithm, and (2) the two-oracle framework we introduced is superior than the previously studied single-oracle setting even when the cheap signal is considerably weak.

Ablation studies. We perform ablation studies on all tune-able parameters of our algorithm. A sample of the ablation studies for SOF is shown in Figure 1(e) and details of other results are presented in Section F.4. We observe that removing any of the main components of the algorithm (merging, ordering with respect to weak signal, and ordering with respect to the statistics of the neighboring nodes) deteriorates the performance of the algorithm, thus all the introduced components are paramount in `KwikBucks`. We also verify the role of the parameter t corresponding to the number of pivots we select for our algorithm in Figure 1(f). We observe that both large and small choices for this parameters can be harmful, but choosing larger values is a safer option compared to smaller values as it asymptotically offers a similar performance as the optimal value.

5 Conclusion

We introduced and studied a novel variant of the (budgeted) correlation clustering algorithm where besides having a limited query budget to an expensive-strong oracle, one also has access to a readily available cheap-weak oracle. We developed an algorithm for this setting with strong theoretical motivations and demonstrated its strong practical performance for text clustering. We anticipate the proposed framework could become a standard building block, especially for text clustering strategies.

References

Nir Ailon, Anup Bhattacharya, and Ragesh Jaiswal. 2018. Approximate correlation clustering using same-cluster queries. In *LATIN 2018: Theoretical Informatics - 13th Latin American Symposium, Buenos Aires, Argentina, April 16-19, 2018, Proceedings*, volume 10807 of *Lecture Notes in Computer Science*, pages 14–27. Springer.

Nir Ailon, Moses Charikar, and Alantha Newman. 2008. Aggregating inconsistent information: Ranking and clustering. *J. ACM*, 55(5):23:1–23:27.

Keerti Anand, Rong Ge, Amit Kumar, and Debmalya Panigrahi. 2022. Online algorithms with multiple predictions.

Alexandr Andoni, Piotr Indyk, and Ilya Razenshteyn. 2018. Approximate nearest neighbor search in high dimensions. In *Proceedings of the International Congress of Mathematicians: Rio de Janeiro 2018*, pages 3287–3318. World Scientific.

Antonios Antoniadis, Christian Coester, Marek Elias, Adam Polak, and Bertrand Simon. 2020a. Online metric algorithms with untrusted predictions. In *International Conference on Machine Learning*, pages 345–355. PMLR.

Antonios Antoniadis, Themis Gouleakis, Pieter Kleer, and Pavel Kolev. 2020b. Secretary and online matching problems with machine learned advice. *Advances in Neural Information Processing Systems*, 33:7933–7944.

Hassan Ashtiani, Shrinu Kushagra, and Shai Ben-David. 2016. Clustering with same-cluster queries. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3216–3224.

Pranjal Awasthi, Avrim Blum, and Or Sheffet. 2012. Center-based clustering under perturbation stability. *Inform. Process. Lett.*, 112(1-2):49–54.

Maria Florina Balcan and Yingyu Liang. 2012. Clustering under perturbation resilience. In *International Colloquium on Automata, Languages and Programming*.

Étienne Bamas, Andreas Maggiori, Lars Rohwedder, and Ola Svensson. 2020a. Learning augmented energy minimization via speed scaling. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems, NeurIPS*.

Étienne Bamas, Andreas Maggiori, and Ola Svensson. 2020b. The primal-dual method for learning augmented algorithms. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems, NeurIPS*.

Nikhil Bansal, Avrim Blum, and Shuchi Chawla. 2004. Correlation clustering. *Machine learning*, 56(1):89–113.

Michael Barz and Daniel Sonntag. 2021. Incremental improvement of a question answering system by re-ranking answer candidates using machine learning. In *Increasing Naturalness and Flexibility in Spoken Dialogue Interaction*, pages 367–379. Springer.

Jo Bergum. 2022. Pretrained transformer language models for search.

Francesco Bonchi, David García-Soriano, and Francesco Gullo. 2022. Conclusions and open problems. *Correlation Clustering*, pages 113–114.

Francesco Bonchi, David García-Soriano, and Konstantin Kutzkov. 2013. Local correlation clustering. *CoRR*, abs/1312.5105.

Francesco Bonchi, David Garcia-Soriano, and Edo Liberty. 2014. Correlation clustering: From theory to practice. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, page 1972, New York, NY, USA. Association for Computing Machinery.

Francesco Bonchi, A. Gionis, Francesco Gullo, Charalampos E. Tsourakakis, and Antti Ukkonen. 2015. Chromatic correlation clustering. *ACM Trans. Knowl. Discov. Data*, 9:34:1–34:24.

Marco Bressan, Nicolò Cesa-Bianchi, Silvio Lattanzi, and Andrea Paudice. 2021. On margin-based cluster recovery with oracle queries. *Advances in Neural Information Processing Systems*, 34:25231–25243.

Marco Bressan, Nicolò Cesa-Bianchi, Andrea Paudice, and Fabio Vitale. 2019. *Correlation Clustering with Adaptive Similarity Queries*. Curran Associates Inc., Red Hook, NY, USA.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Moses Charikar, Venkatesan Guruswami, and Anthony Wirth. 2005. Clustering with qualitative information. *Journal of Computer and System Sciences*, 71(3):360–383.

Moses Charikar, Samir Khuller, David M. Mount, and Giri Narasimhan. 2001. Algorithms for facility location problems with outliers. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '01, page 642–651, USA. Society for Industrial and Applied Mathematics.

Shuchi Chawla, Robert Krauthgamer, Ravi Kumar, Yuval Rabani, and D Sivakumar. 2006. On the hardness of approximating multicut and sparsest-cut. *computational complexity*, 15(2):94–114.

Justin Y. Chen, Talya Eden, Piotr Indyk, Honghao Lin, Shyam Narayanan, Ronitt Rubinfeld, Sandeep Silwal, Tal Wagner, David P. Woodruff, and Michael Zhang. 2022. Triangle and four cycle counting with predictions in graph streams. In *The Tenth International Conference on Learning Representations, ICLR*.

William W Cohen and Jacob Richman. 2002. Learning to match and cluster large high-dimensional data sets for data integration. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 475–480.

Ofer Dekel, Claudio Gentile, and Karthik Sridharan. 2012. Selective sampling and active learning from single and multiple teachers. *The Journal of Machine Learning Research*, 13(1):2655–2697.

Erik D Demaine, Dotan Emanuel, Amos Fiat, and Nicole Immorlica. 2006. Correlation clustering in general weighted graphs. *Theoretical Computer Science*, 361(2-3):172–187.

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

Ilias Diakonikolas, Vasilis Kontonis, Christos Tzamos, Ali Vakilian, and Nikos Zarifis. 2021. Learning online algorithms with distributional advice. In *Proceedings of the 38th International Conference on Machine Learning, ICML*, pages 2687–2696.

Thomas G Dietterich. 2000. Ensemble methods in machine learning. multiple classifier systems. *Lecture Notes in Computer Science*, 1857:1–15.

Pinar Donmez. 2008. Proactive learning: Towards cost-sensitive active learning with multiple imperfect oracles.

Talya Eden, Piotr Indyk, Shyam Narayanan, Ronitt Rubinfeld, Sandeep Silwal, and Tal Wagner. 2021. Learning-based support estimation in sublinear time. In *9th International Conference on Learning Representations, ICLR*.

Derek Eder. 2022. [link].

Meng Fang, Xingquan Zhu, Bin Li, Wei Ding, and Xindong Wu. 2012. Self-taught active learning from crowds. In *2012 IEEE 12th international conference on data mining*, pages 858–863. IEEE.

Ethan Fetaya, Ohad Shamir, and Shimon Ullman. 2015. Graph approximation and clustering on a budget. In *Artificial Intelligence and Statistics*, pages 241–249. PMLR.

David García-Soriano, Konstantin Kutzkov, Francesco Bonchi, and Charalampos Tsourakakis. 2020a. Query-efficient correlation clustering. In *Proceedings of The Web Conference 2020*, pages 1468–1478.

David García-Soriano, Konstantin Kutzkov, Francesco Bonchi, and Charalampos Tsourakakis. 2020b. Query-efficient correlation clustering. In *Proceedings of The Web Conference 2020*, pages 1468–1478.

Lise Getoor and Ashwin Machanavajjhala. 2012. Entity resolution: Theory, practice & open challenges. *Proc. VLDB Endow.*, 5:2018–2019.

A. Gionis, Heikki Mannila, and Panayiotis Tsaparas. 2005. Clustering aggregation. *21st International Conference on Data Engineering (ICDE'05)*, pages 341–352.

Sreenivas Gollapudi and Debmalya Panigrahi. 2019. Online algorithms for rent-or-buy with expert advice. In *Proceedings of the 36th International Conference on Machine Learning, ICML*, pages 2319–2327.

Ramanathan Guha, Vineet Gupta, Vivek Raghunathan, and Ramakrishnan Srikant. 2015. User modeling for a personal assistant. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 275–284.

Oktie Hassanzadeh, Fei Chiang, Renée J. Miller, and Hyun Chul Lee. 2009. Framework for evaluating clustering algorithms in duplicate detection. *Proc. VLDB Endow.*, 2(1):1282–1293.

Chen-Yu Hsu, Piotr Indyk, Dina Katabi, and Ali Vakilian. 2019. Learning-based frequency estimation algorithms. In *7th International Conference on Learning Representations, ICLR*.

Sheng-Jun Huang, Jia-Lve Chen, Xin Mu, and Zhi-Hua Zhou. 2017. Cost-effective active learning from diverse labelers. In *IJCAI*, pages 1879–1885.

Panagiotis G Ipeirotis, Foster Provost, Victor S Sheng, and Jing Wang. 2014. Repeated labeling using multiple noisy labelers. *Data Mining and Knowledge Discovery*, 28(2):402–441.

Tanqiu Jiang, Yi Li, Honghao Lin, Yisong Ruan, and David P. Woodruff. 2020. Learning-augmented data stream algorithms. In *8th International Conference on Learning Representations, ICLR*.

Sungwoong Kim, Sebastian Nowozin, Pushmeet Kohli, and Chang Yoo. 2011. Higher-order correlation clustering for image segmentation. *Advances in neural information processing systems*, 24:1530–1538.

Christopher H Lin, M Mausam, and Daniel S Weld. 2015. Reactive learning: Actively trading off larger noisier training sets against smaller cleaner ones. In *Proceedings of the 32nd International Conference on Machine Learning, Lille, France (ICML)*.

Christopher H Lin, Daniel S Weld, et al. 2014. To re (label), or not to re (label). In *Second AAAI conference on human computation and crowdsourcing*.

Thodoris Lykouris and Sergei Vassilvitskii. 2018. Competitive caching with machine learned advice. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, pages 3302–3311.

Luigi Malago, Nicolo Cesa-Bianchi, and J Renders. 2014. Online active learning with strong and weak annotators. In *NIPS Workshop on Learning from the Wisdom of Crowds*.

Aditya Menon, Sadeep Jayasumana, Ankit Singh Rawat, Seungyeon Kim, Sashank Reddi, and Sanjiv Kumar. 2022. In defense of dual-encoders for neural ranking. In *International Conference on Machine Learning*, pages 15376–15400. PMLR.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Michael Mitzenmacher. 2018. A model for learned bloom filters and optimizing by sandwiching. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems, NeurIPS*, pages 462–471.

Kim Thang Nguyen and Christoph Dürr. 2021. Online primal-dual algorithms with predictions for packing problems. *CoRR*, abs/2110.00391.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

Xuan-Hieu Phan, Le-Minh Nguyen, and Susumu Horiguchi. 2008. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *Proceedings of the 17th international conference on World Wide Web*, pages 91–100.

Manish Purohit, Zoya Svitkina, and Ravi Kumar. 2018. Improving online algorithms via ML predictions. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems, NeurIPS*, pages 9684–9693.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

Md Rashadul Hasan Rakib, Norbert Zeh, Magdalena Jankowska, and Evangelos Milios. 2020. Enhancement of short text clustering by iterative classification. In *International Conference on Applications of Natural Language to Information Systems*, pages 105–117. Springer.

Anirudh Ramachandran, Nick Feamster, and Santosh S. Vempala. 2007. Filtering spam with behavioral blacklisting. In *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*, pages 342–351. ACM.

Radim Rehurek and Petr Sojka. 2011. Gensim–python framework for vector space modelling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, 3(2).

Barna Saha and Sanjay Subramanian. 2019. Correlation clustering with same-cluster queries bounded by optimal cost. In *27th Annual European Symposium on Algorithms, ESA 2019, September 9-11, 2019, Munich/Garching, Germany*, volume 144 of *LIPIcs*, pages 81:1–81:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine*, 29(3):93–93.

Ohad Shamir and Naftali Tishby. 2011. Spectral clustering on a budget. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 661–669. JMLR Workshop and Conference Proceedings.

Or Sharir, Barak Peleg, and Yoav Shoham. 2020. The cost of training nlp models: A concise overview. *arXiv preprint arXiv:2004.08900*.

Oleksandr Shchur and Stephan Günnemann. 2019. Overlapping community detection with graph neural networks. *arXiv preprint arXiv:1909.12201*.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*.

Jiliang Tang, Yi Chang, Charu C. Aggarwal, and Huan Liu. 2016. A survey of signed network mining in social media. *ACM Computing Surveys (CSUR)*, 49:1 – 37.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. 2022. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*.

Ruth Urner, Shai Ben David, and Ohad Shamir. 2012. Learning from weak teachers. In *Artificial intelligence and statistics*, pages 1252–1260. PMLR.

Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2019. Deep graph infomax. *ICLR*, 2(3):4.

Fabian L Wauthier, Nebojsa Jojic, and Michael I Jordan. 2012. Active spectral clustering via iterative uncertainty reduction. In *Proceedings of the 18th*

ACM SIGKDD international conference on Knowledge discovery and data mining, pages 1339–1347.

Alexander Wei and Fred Zhang. 2020. Optimal robustness-consistency trade-offs for learning-augmented online algorithms. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS*.

Jing Wong. 2022. Model-based candidate generation for account recommendations.

Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga, Jinshi Huang, Charles Bai, et al. 2022. Sustainable ai: Environmental implications, challenges and opportunities. *Proceedings of Machine Learning and Systems*, 4:795–813.

Jiaming Xu, Bo Xu, Peng Wang, Suncong Zheng, Guanhua Tian, and Jun Zhao. 2017. Self-taught convolutional neural networks for short text clustering. *Neural Networks*, 88:22–31.

Yan Yan, Romer Rosales, Glenn Fung, and Jennifer G Dy. 2011. Active learning from crowds. In *ICML*.

Yan Yan, Rómer Rosales, Glenn Fung, Faisal Farooq, Bharat Rao, and Jennifer Dy. 2012. Active learning from multiple knowledge sources. In *Artificial Intelligence and Statistics*, pages 1350–1357. PMLR.

Jianhua Yin and Jianyong Wang. 2016. A model-based approach for text clustering with outlier detection. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 625–636. IEEE.

Taraneh Younesian, Dick Epema, and Lydia Y Chen. 2020. Active learning for noisy data streams using weak and strong labelers. *arXiv preprint arXiv:2010.14149*.

Chicheng Zhang and Kamalika Chaudhuri. 2015. Active learning from weak and strong labelers. *Advances in Neural Information Processing Systems*, 28.

Xiang Zhang and Yann LeCun. 2015. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*.

Zhun Zhong, Liang Zheng, Donglin Cao, and Shaozi Li. 2017. Re-ranking person re-identification with k-reciprocal encoding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1318–1327.

A Additional Related Works

In the realm of learning from multiple annotators, there is a long line of work studying these both empirically and theoretically. Empirical work on this can be divided into two main streams: (1) each labeler is coming from a different generative model, (2) each labeler is an expert over an unknown subset of categories, (3) different labelers with quality proportional to their cost. In the first case, the learning algorithm focuses on learning parameters of each labeler and then for each example decides which labeler to query (Yan et al., 2011, 2012; Lin et al., 2015, 2014; Fang et al., 2012). In the second case, it uses data to measure the class-wise expertise in order to optimally place label queries (Ipeirotis et al., 2014; Donmez, 2008). In the last case, empirical results comparing designed algorithms to baselines are developed: active learning from noisy data streams (Younesian et al., 2020), active learning using diverse labelers (Huang et al., 2017), and content segmentation for personal assistants (Guha et al., 2015). Theoretical work looked at the setting where the weak labeler made mistakes mostly in heterogeneous regions of space, i.e., correct in label-homogeneous regions but may deteriorate near classification boundaries. Different formulations were considered in this setting: non-parametric setting (Urner et al., 2012), fitting classifiers in a hypothesis class (Zhang and Chaudhuri, 2015), online selective sampling with applications in linear classifiers and robust regression (Malago et al., 2014; Dekel et al., 2012).

The idea of judiciously utilizing an expensive but accurate strong model with the help of cheaper but noisier methods have already been successfully used in many practical and important domains. In nearest neighbor search and information retrieval, the dominant algorithmic paradigm is to return multiple possible nearest neighbors using scalable methods and then re-rank the returned points using exact distance calculations (which is prohibitive to perform over the entire input)[2]. In recommendation systems, the "standard practice for machine learning-driven recommendations in industry" (Wong, 2022) is driven by the two-step procedure of cheaply retrieving a set of possible candidates and iterating over them using a more powerful but costlier ML models (Wong, 2022; Bergum, 2022; Eder, 2022). Similar ideas are also used in question answering and vision applications (Zhong et al., 2017; Barz and Sonntag, 2021).

There has also been extensive work in incorporating additional predictions in algorithmic design for online algorithms (Bamas et al., 2020b; Purohit et al., 2018; Lykouris and Vassilvitskii,

[2]We refer to `http://ann-benchmarks.com/` for a large collection of practical nearest neighbor search algorithms and (Andoni et al., 2018) for a overview of theoretical works.

2018; Purohit et al., 2018; Gollapudi and Panigrahi, 2019), sublinear space and time algorithms (Chen et al., 2022; Hsu et al., 2019; Eden et al., 2021), and other algorithmic and data structural problems (Mitzenmacher, 2018; Bamas et al., 2020b,a; Wei and Zhang, 2020; Jiang et al., 2020; Diakonikolas et al., 2021; Charikar et al., 2001; Antoniadis et al., 2020a,b; Anand et al., 2022; Nguyen and Dürr, 2021). We refer to the Algorithms-with-Predictions website[3] for comprehensive references. The high level motivations of these works is to apply predictions to aid in beyond worst-case analysis of algorithmic problems. The prototypical examples of predictions used in these works include algorithm parameter settings (for example 'warm starts' or 'seeds' which can be constructed from past inputs). Thus a common underlying assumption is that many similar inputs are given so that predictions are meaningful and feasible. Furthermore in many of these works, the predictions are modeled after particular problem settings in mind and the inputs are always fully specified. In contrast, our predictions are inspired by a particular application domain, e.g. text clustering, which we connect to CC, rather than motivating the predictions from a purely algorithmic problem perspective. Furthermore, our predictions (e.g. queries from the weak signal) help us learn about the true underlying input (e.g. the strong signal graph).

We also give a detailed comparision to the work of (Guha et al., 2015). While at a high level both (Guha et al., 2015) and our work aggregate information across various signals, the two works differ in terms of the generality of oracles considered, the formal guarantees given, and the problems studied. The oracles used in (Guha et al., 2015) are highly specialized to the datasets at hand; for example, the cheap oracle used in (Guha et al., 2015) is an inverted index model which heavily relies on the specifics of the datasets used. In contrast, we take a broader view of weak and strong oracles and present theoretically founded algorithms which only assume query access to the weak and strong model and not any particular model idiosyncrasies. Therefore, our algorithm has provable guarantees on both the approximation quality and the query complexity, making it broadly applicable across different oracles. In terms of problems, we study correlation clustering while the focus of (Guha et al., 2015) is not on a clustering problem. Rather, they use hierarchical clustering as an intermediate problem to perform user modeling and do not consider any specific clustering objective functions. The strong signal queries made by our algorithm are guided through formal reasoning and they exploit the structure of the clustering problem we are studying. In (Guha et al., 2015) the weak signal is used at a more intuitive level and serves the informal role of filtering possible strong signal queries with no formal reasoning.

[3] https://algorithms-with-predictions.github.io/

B Omitted Proofs of Section 2

Proposition 1. *There exists a strong signal graph G such that the* `KwikCluster` *algorithm makes $\Omega(n^{1.5})$ strong signal queries.*

Proof. Consider the case where the strong signal graph consists of $\sqrt{n}$ cliques, all of size $\sqrt{n}$. In this case, every time `KwikCluster` picks a pivot, it has to examine an existence of an edge from this pivot to all the unassigned vertices. So for at least the first $\sqrt{n}/2$ times `KwikCluster` picks a pivot, it has to make at least $n/2$ calls to O_S , resulting in $\Omega(n^{1.5})$ calls to O_S. □

Proposition 2. *Consider a variation of* `KwikCluster`*, called* $\texttt{KwikCluster}^\gamma$*, which for a chosen pivot p from uncovered vertices V', only queries $V' \cap \mathcal{O}_W^\gamma(p)$ from strong signal. There exists a graph G such that* $\texttt{KwikCluster}^\gamma$ *still makes $\Omega(n^2)$ strong signal queries in expectation even when $\gamma = O(1)$.*

Proof. Consider the following strong signal graph: the graph G is comprised a fully connected clique on $0.9n$ nodes. The graph also has $0.1n$ nodes, called 'outside vertices' which all connect to the same $0.1n$ vertices in the fully connected clique but have no edges between them. Suppose that $\mathcal{O}_W^\gamma$ returns the correct strong signal neighborhood for vertices in the clique but for the outside vertices, it returns an additional $\Omega(n)$ arbitrary vertices among the outside vertices.

Now consider the simulation of $\texttt{KwikCluster}^\gamma$ on G. With constant probability, the first time a pivot is picked, it comes from the clique vertices which have no neighbors among the outside vertices. Condition on this event. Now the algorithm still needs to run until the outside vertices have been selected in a cluster. However, every time each such vertex is picked as a pivot, we need to check over $\Omega(n)$ erroneous vertices. Furthermore,

removing an outside vertex v and its neighborhood $\Gamma(v)$ does not remove any of the other outside vertices. Thus it follows that the expected number of queries made to O_S, while still utilizing $\mathcal{O}_W^\gamma$ in this natural variant of `KwikCluster`, is $\Omega(n^2)$ with constant probability since there are $\Omega(n)$ outside vertices, each which requires $\Omega(n)$ queries to O_S. Altogether, this natural algorithm can incur $\Omega(n^2)$ queries, a super linear amount. □

Lemma 1. *Let t be the parameter of* `GetPivots` *(Algorithm 3). Algorithm 3 uses $t + 2\gamma t^2 m/n^2$ queries to O_S.*

Proof. Let $T = \{v_1, \ldots, v_t\}$ denote the set of t sampled vertices in line 1 of Algorithm 3. Fix a vertex $v \in T$, we show that there are $O(\gamma t|E|/n + 1)$ queries in expectation for such vertex. Let P_i denote the set of pivots right before we start scanning vertex v_i. When we check for the neighbors of vertex v_i, we immediately stop if a strong signal neighbor in P_i is found. Let $A(v_i) = \mathcal{O}_W^\gamma(v_i) \setminus \Gamma(v_i)$ be the vertices which $\mathcal{O}_W^\gamma$ errs on for query v_i and can result in needless calls to O_S. The number of expected calls to the strong signal is exactly the expected number of unnecessary calls $\mathbb{E}[|A(v_i) \cap P_i|]$ plus one call that may result in the early stoppage in line 4 of Algorithm 4. So for each v_i, the expected number of calls to O_S can be bounded by

$$\begin{aligned}
1 + \mathbb{E}[|A(v_i) \cap P_i|] &\leq 1 + \mathbb{E}[|A(v_i) \cap T|] \\
&= 1 + \mathbb{E}_{v_i}\mathbb{E}[|A(v_i) \cap T| \mid v_i] \\
&\leq 1 + 2\mathbb{E}_{v_i}\mathbb{E}\left[|A| \cdot \frac{t}{n} \mid v_i\right] \\
&\leq 1 + \frac{2t\gamma}{n}\,\mathbb{E}_{v_i}[|\Gamma(v_i)|] \\
&= 1 + \frac{2t\gamma m}{n^2}.
\end{aligned}$$

Summing over all t vertices in T results in the final bound. □

Lemma 2. *Let P be the output of* `GetPivots(t,Q)`*, then* `AssignToClusters`$(P, V \setminus P, Q)$ *makes $n + 2\gamma tm/n$ queries to O_S.*

Proof. Consider a fixed vertex $u \in V \setminus P$. We perform a similar analysis as in Lemma 1: ideally $S = \mathcal{O}_W^\gamma(u) \cap P$ informs us the pivot which u should connect to. However since the cheap oracle can be noisy, we can have many vertices in $(\mathcal{O}_W^\gamma(u) \setminus \Gamma(u)) \cap P$. The number of queries to O_S is at most $|(\mathcal{O}_W^\gamma(u) \setminus \Gamma(u)) \cap P| + 1$. It remains to calculate the following expected value:

$$\mathbb{E}[|(\mathcal{O}_W^\gamma(u) \setminus \Gamma(u)) \cap P|] \leq \gamma\,|\Gamma(u)| \cdot \frac{t}{n}.$$

Thus the total expected number of queries for non-pivot vertices is

$$\sum_{u \in V \setminus P} \left(1 + \gamma\,|\Gamma(u)| \cdot \frac{t}{n}\right) = |V \setminus P| + \frac{\gamma t}{n} \sum_{u\ in V \setminus P} |\Gamma(u)| \leq n + \frac{2\gamma tm}{n}.$$

□

Theorem 2 ((Bonchi et al., 2013))**.** *Let T' be the maximal independent set formed by scanning randomly sampled t vertices of a graph G. Then the expected number of edges of G not incident with an element of T' union the neighborhood of T' in G is at most $n^2/2t$.*

Proof. The bound on the number of queries to O_S. There are two tasks which require calls to O_S: forming the set of pivots in `GetPivots` and assigning non-pivot vertices to a pivot in `AssignClusters`. The expected number of queries for`GetPivots` is handled by Lemma 1 and the expected number of queries for `AssignClusters` is handled by Lemma 2.

We now need to bound the approximation guarantee. Consider the subgraph G' of the strong signal graph which is the union of the pivots returned by `GetPivots` and their neighborhood. Theorem 2 gives us that the number of edges not part of this subgraph is at most $O(n^2/t)$ which can be charged to the additive error incurred by our algorithm (all vertices which do not have a strong signal edge to any of the pivots are clustered as singletons). Now on this subgraph note that we are exactly mimicking the `KwikCluster` algorithm on G'. This is because the pivots of Get-Pivots are chosen from the same distribution as the `KwikCluster` algorithm since we ensure that all pivots chosen are not in the neighborhood of previously chosen pivots. Thus we obtain a $3\cdot$ OPT guarantee on G'. To obtain the final guarantee on the original strong signal graph, note that the OPT clustering of G restricted to G' cannot be better than the OPT correlation clustering of G'. The result follows from considering our additive error as well. □

We now show a lower bound on the query complexity of our algorithm. First we recast the lower bound result of (García-Soriano et al., 2020b) in the language of strong and weak oracles. They show that *any* algorithm which only has access to the strong signal must make $\Omega(n^3/(\Delta c^2))$ queries to obtain a $c\cdot$ OPT $+\Delta$ correlation clustering objective guarantee. We can translate their lower bound into our setting of strong and weak oracles by essentially making the weak oracle useless through a suitable choice of γ. The lower bound shows that for constant ε and n large enough, Corollary 1 is optimal. First we formally state the guarantees given by (García-Soriano et al., 2020b) in our language of strong and weak signals.

Theorem 3 ((García-Soriano et al., 2020b)). *For any $c \geq 1$ and Δ such that $8n < \Delta \leq \frac{n^2}{2048c^2}$, any algorithm finding a clustering with expected cost at most $c\cdot$ OPT $+\Delta$ must make at least $\Omega(n^3/(\Delta c^2))$ adaptive strong signal queries.*

Lemma 3. *Let $\varepsilon \geq \Omega(1/n)$ be sufficiently small. In the worst case input, any algorithm must use at least $\Omega(n + d\gamma)$ strong signal queries to obtain a $3\cdot$ OPT $+O(\varepsilon n^2)$ approximation to the correlation clustering objective.*

Proof. We recall the lower bound example of (García-Soriano et al., 2020b) (which is proved in Theorem 4.1 in (García-Soriano et al., 2020b)). Let $k = n^2/(32c\Delta)$ (note that $k < n$ by design). Their worst case strong signal graph example consists of k equal sized cliques and all vertices have degree $\Theta(n/k)$. Now we consider the case where the weak oracle is completely useless and always returns the entire set of vertices on any query. This corresponds to the case where $\gamma = \Theta(k)$ (for γ defined in Assumption 1). Now directly applying Theorem 4.1 of (García-Soriano et al., 2020b) gives us that any algorithm which only has access to the strong signal must make at least $\Omega(n^3/(\Delta c^2))$ queries to obtain a $c\cdot$ OPT $+\Delta$ correlation clustering objective guarantee. The theorem follows by noting that if $\Delta = \varepsilon n^2$ then any algorithm must make $\Omega(n/\varepsilon) = \Omega(n + k\cdot n/k) = \Omega(n + d\gamma)$ queries in this worst case example, as desired. Note that the valid range of ε here follows from the restriction on Δ so $\varepsilon \geq 8/n$ and cannot be larger than some fixed constant. □

C Additional Algorithmic Details for Empirical Algorithm.

We provide additional details on the algorithm design of Section 3.

Algorithm 7 `SortNonPivots`$(T, V \setminus T)$

Require: T, the set of pivots; $V \setminus T$, vertices which are not pivots

1: **for** vertices $v \in V \setminus T$ **do**
2: $\quad w_v \leftarrow$ Maximum weak signal similarity between v and any vertex in T
3: **Return** $V \setminus T$ sorted in decreasing w_v values

Optimizing non-pivot order. Continuing on the theme of ranking, once we curate the pivots, we need to assign the non-pivots to a pivot. To do so, we sort the non-pivots based on 'easiness' to assign to a pivot. Hence we sort the non-pivots by the maximum weak similarity to some pivot. This has the effect of utilizing our query budget as efficiently as possible as 'easier' non-pivots are checked first. We re-rank the vertices $V \setminus P$ in `SortNonPivots` before calling `AssignClusters` in `KwikBucks`.

Utilizing Weak Signal Neighborhood. We can use strong signal queries that we have already made for vertices in the weak signal neighborhood to further optimize the sorting of the pivots in `WeakFilterByRanking`. The inductive bias we are using is that if many vertices in the immediate weak signal neighborhood of a vertex v connect to the same pivot p, then the likelihood v having a strong signal edge to p is high. Thus to better utilize our expensive query budget, we should query (v, p) earlier than later. To make the intuition more precise, we simply update the similarities to pivots computed by v in `WeakFilterByRanking` to account for the inductive bias. The new similarity score $w'p$ of a pivot p is equal to w_p, the similarity score between v and p computed by the weak signal, plus a term for the number of vertices in the k-weak signal neighborhood of v that are already connected to p:

$$w'_p = w_p + \lambda(\#\text{ of vertices in } k\text{-weak signal neighborhood of } v \text{ that are already in } p\text{'s cluster})$$

This has the affect of 'boosting' some pivots to a higher ranking. See Section F.2 for further details.

C.1 Details on Post-processing Merging

In this section we provide the details of our post-processing merging strategy outlined in Section 3.

Let $C_1, \ldots, C_r$ be the clusters outputted by our algorithm. First we curate a list of cluster pairs to consider for merging. Then we rank the pairs in terms of suitability for merging. Finally we enumerate over the pairs in the order computed (until we run out of any query budget) and determine if the pair should be merged. Each of the three steps is described in detail below.

1. **Curating pairs of clusters.** It is prohibitive to consider all pairs of clusters (which might be super linear if there are many clusters). We again appeal to the weak signal and construct a k-nn weak signal similarity graph on the vertices for some small k, such as $k = 20$. Then we only consider pairs of clusters which are edges in the graph. More precisely, we consider the pair (C_i, C_j) for merging if there is some $v \in C_i$ and $u \in C_j$ such that (v, u) is an edge in the k-nn graph. This narrows down the number of pairs considerably.

2. **Ranking pairs by suitability.** For each pair (C_i, C_j) of clusters from the prior step, we compute the average weak signal value between vertices in C_i and C_j respectively. We then rank the pairs in decreasing order based on this value.

3. **Determining if a pair should be merged.** Finally, we enumerate over the pairs in the order computed previously. Suppose we are deciding if we want to merge the pair (C_i, C_j). We must ensure the pair has a high number of strong signal edges (more than 0.5 fraction). To do so we simply sample a small number of random pairs of vertices (say 20), one vertex from each cluster, and estimate the fraction of these random edges which are strong signal edges.

D Theoretically Motivating Practical Modifications of the Algorithm

In this section we provide theoretical justifications for the practical modifications of our algorithm.

D.1 Theoretically motivating raking pivots by weak signal

In the classical `KwikCluster` algorithm and our query efficient variant in the two oracle model of Section 2, it is imperative that the pivots are selected in a random order to provide theoretical guarantees on the quality of the computed clustering. Specifically, the worst case theoretical guarantees dictate that vertices must connect to the *first* pivot in the random order which they have a strong signal edge to.

Nevertheless, in our data driven optimization of the algorithm, we choose an adaptive ordering of the pivots for each vertex where the order is based on the weak-signal similarity scores. We empirically observed that this ordering is superior to the random ordering and achieves a higher clustering quality while utilizing a $>$ **3.5x** factor or more less strong signal queries. The explanation behind this improvement is two fold:

1. **Increased query efficiency:** fewer strong signal queries are used when a vertex attempts to connect to a pivot.

2. **Maintaining cluster quality:** connecting to pivots with larger weak signal similarities are high quality pivots.

The positive effects of the first point are straightforward to explain. Indeed, making the natural assumption that higher weak signal similarities are more indicative of a strong signal edge, checking pivots in the weak signal ordering leads to less queries wasted when a vertex attempts to connect to a pivot. In addition to the empirical results of Section 4, this point is further expanded upon in Figure 7 and Section F.6.

Thus the main goal of this section is to provide an intuitive and theoretically motivated understanding of the second point. While it may not be true that re-ranking pivots according to the weak signal similarities maintains the worst case guarantees proved in Section 2, we study a natural data set model where such a re-ranking provably helps. We wish to capture our data driven observations that pivots with larger weak signal similarities are of high quality and larger weak signal values indicate better cluster relationships.

In our experiments, the weak signal scores are mostly computed using distances between embedding vectors. If a weak signal is useful, then it

must have be predicative of the strong signal values, even if the weak signal is noisy. To mimic this, we consider the following general family of data sets:

- Each vertex v has an associated vector $p_v \in \mathbb{R}^d$, representing it's 'true' embedding representation.
- The weak signal values are computed according to an appropriate distance measure d on the embedding vectors (for example cosine or Euclidean distances) plus a random noise term ξ (expanded upon shortly). This models the setting where the weak signals are helpful but noisy signals as they only have noisy access to the 'true' representations.
- There exists a function $f : \mathbb{R}^{\geq 0} \to [0, 1]$ which gives the probability of a strong signal edge. More precisely, let p_v and p_u denote the embedding of vertices v and u. Then the probability of having the strong signal edge (v, u) is given by $f(d(v, u))$. This is quite a general formulation as it includes a wide array of geometric or kernel similarity graphs for appropriate choices of f and d.

For example, if $f = \exp(-x/\sigma)$ and d is the Euclidean distance, then the true strong signal graph is the Gaussian kernel similarity graph where σ is the scale of the kernel. Intuitively, the closer u and v are under the metric d, the higher probability f assigns to the edge between u and v.

We additionally impose the following clusterability assumptions on the data set. Our goal is to capture a natural underlying cluster structure which can be accessed via strong and weak signal queries. Vertices which are part of the same underlying cluster should have higher weak signal similarity scores, even if the scores are noisy, and the strong signal edges should be highly accurate. Our model defined below satisfies these intuitive criteria. Furthermore under our natural model, there is a 'true' pivot for a vertex v, even though v may have strong signal edges to other pivots.

Our cluster assumptions on the data set is the following.

1. The 'true' embedding vectors $p_v \in \mathbb{R}^d$ can be partitioned into k clusters such that all vectors in a cluster are within distance R of each other.
2. All embedding vectors in different clusters are distance at least $2R$ from each other.
3. The probability of a strong signal edge is at least $1 - p$ for distances at most $2R$ and at most p for distances at least $2R$. We think of $p < 1/2$ as a small parameter close to 0.
4. Given inputs u, v, the weak signal outputs $d(p_u, p_v) + \xi$ where ξ is uniformly random in $[-R, R]$. Thus smaller values are interpreted as having higher weak signal similarity.

Note that we only have access to the strong and weak signal values via queries and do not know the true underlying embedding vectors p_v. We now argue that the above assumptions are motivated and natural.

- The assumption (1) gives a cluster structure to the data and allows us to compare the classical `KwikCluster` algorithm and our re-ranking modification under natural clusterability assumptions. The exact formulation we are employing is inspired by the works of (Awasthi et al., 2012; Balcan and Liang, 2012; Ashtiani et al., 2016) which study clustering under similar proximity assumptions. For example, it can be easily checked that the 'margin' property assumption of (Ashtiani et al., 2016) directly implies our assumptions (1) and (2).
- Our assumption (3) is a natural and necessary assumption on the function f as it ensures the true strong signal graph captures the underlying cluster structure of the inputs. This also corresponds to picking an appropriate *scale* parameter if f is a kernel function, for example picking σ in the Gaussian kernel. A judicious choice of σ ensures that the underlying kernel similarity graph, which corresponds to the strong signal, is able to capture the cluster structure of the data set. Thus our assumption that $p \ll 1$ ensures that the similarity graph has strong inter-cluster connectivity while having sparse connectivity across different clusters. Indeed in practice, the kernel scale parameter is often picked using the 'median' rule and thus $\sigma = \Theta(R)$ is a natural choice which ensures our choice of p.
- Our data set construction ensures that the strong signal is 'more powerful' than the weak

signal. Indeed, the weak signal only has access to the distances between the true embedding vectors up to some additive noise as stated in assumption (4). While the exact form of the random noise is not very consequential, we stick to the uniform noise model as it as several desirable properties:

1. Given vertices v, u, w where u is in v's true cluster (according to the true embedding vectors) and w is not, the weak signal can potentially output a smaller value on query (v, w) compared to (v, u). Thus the weak signal incorrectly states w is more similar to v than u due to the additive noise. For example if $d(v, u) = R$ and $d(v, w) = 2R$, this happens with probability $1/8$. Therefore the weak signal accurately reflects our desired goal of an indicative but noisy signal.
2. The weak signal can be modeled by fast nearest neighbor search algorithms which return noisy nearest neighbor estimates. On the other hand, we imagine the strong signal as being expensive since it needs the true distances among the embedding vectors without any additive noise.

We believe that this natural graph model we examined for our algorithm modification helps explain and predict the strong empirical performance of our method. Thus our goal is to show that under the above data set modelling, re-ranking pivots based on weak signal similarity values provably helps. Assume that we have picked a pivot u from each of the k clusters of Assumption (1). We permute them randomly to form an ordering $u_1, \ldots, u_k$. This corresponds to the random ordering used by the `KwikCluster` algorithm and our theoretical algorithm of Section 2. Each non-pivot vertex v re-ranks the pivots forming the ordering $u_{\pi_v(1)}, \ldots, u_{\pi_v(k)}$ where π_v is a permutation depending on the weak signal similarities from the pivots to v. The weak signal similarities are calculated as detailed above: the weak signal outputs 'noisy' distances based on the true embedding vectors and smaller distances correspond to higher similarities. Note that each non-pivot vertex v has a 'true' pivot u corresponding to the pivot chosen from the cluster that the true embedding vector p_v is part of. We say that a non-pivot vertex v is correctly assigned by a clustering algorithm $\mathcal{C}$ if $\mathcal{C}$ assigns v to its 'true' pivot. The following lemma shows that assigning vertices to pivots based on weak signal similarities strictly outperforms using a random order.

Intuitively, if another pivot u' is ranked higher than u in the random ordering, our proposed medication of re-ranking asked on weak signal similarities is likely to *correct* the ordering re-ranking u to ahead. The lemma below provides theoretical justification of why this is sound and complements our experimental evaluation which demonstrates the empirical advantage of our re-ranking procedure.

Lemma 4. *Consider the setting above. Let $\mathcal{C}$ be the clustering where every non pivot vertex picks the* first *pivot in this ordering that it has a strong signal edge to. Let $\mathcal{C}'$ be the clustering where each non pivot vertex re-ranks the pivots using weak signal similarities then picks the first pivot that it has a strong signal edge to. Let A be the number of non-pivot vertices that $\mathcal{C}$ correctly assigns and similarly define B. We have $\mathbb{E}[A] < \mathbb{E}[B]$.*

Proof. Fix a non-pivot vertex v. Let X denote the indicator variable for $\mathcal{C}$ correctly assigning v and define Y similarly for $\mathcal{C}'$. It suffices to show that $\mathbb{E}[X] < \mathbb{E}[Y]$. The lemma then follows by linearity of expectations and summing across all non-pivot vertices v. Note here that the expectation of each variable is with respect to the randomness used by the respective algorithms.

Let u denote the true pivot of v. If v does not have a strong signal edge to u (according to the strong signal) then both algorithms will fail. Similarly, if v only has a strong signal edge to u and to no other pivots, then the performance of either clustering is the same. Now consider the case that v has at least two strong signal edges to pivots, one to u and rest arbitrary. Then the probability that v is correctly classified by the random ordering is at most $1/2$. This is because if there is at least 1 other pivot that v as a strong signal edge to, then the probability that the random ordering places u ahead of it is at most $1/2$.

One the other hand, the probability that v is correctly classified by the weak signal ordering is strictly larger than $1/2$. To see this, we calculate the probability that u has the highest weak signal similarity. Identically, it suffices to calculate the probability that the weak signal outputs the smallest noisy distance value for u. Recall the modelling assumptions of the data set: we know that

$d(v,u) \leq R$ whereas $d(v,u') \geq 2R$ for all other pivots u' that are not equal to u. The weak signal outputs $d(v,u')$ plus a uniformly random value in $[-R,R]$. Let ξ_u be the random value added for $d(v,u)$. With probability $1/2$, this value is negative so the noisy distance computed by the weak signal is strictly smaller for u than all other pivots u' (since in the best case, their distance is at least $2R - R = R$). Furthermore, conditioning on the additive noise being positive for u, there is a non-zero probability that u has the smallest additive noise (in absolute value) among all pivots. u is again ranked the highest in this case. Altogether, the probability that u is ranked the highest in terms of the weak signal similarity is strictly larger than $1/2$. It follows that $\mathbb{E}[X] < \mathbb{E}[Y]$, as desired. □

D.2 Explaining why merging helps

We consider a particular worst case example for the `KwikCluster` algorithm which motivates why a post processing merging step helps. At a high level, it is possible to pick pivots which do not have a strong signal edge but nevertheless 'should' belong to the same cluster. Then when we a clustering algorithm is run, these two pivots can possibly lead to two disjoint clusters whereas that merging them lowers the correlation clustering objective and improves the overall clustering quality.

Concretely, consider the following example: we have a complete graph on n vertices where every edge is a strong signal edge except a single edge (u,v) which is not. In the classical `KwikCluster` algorithm, if u is picked as a pivot then we will form two clusters, one consisting of all vertices besides u and the the other cluster being the singleton $\{u\}$. The same is true if we pick v to be the pivot. Thus the expected correlation clustering objective of the algorithm is

$$\frac{2}{n} \cdot (n-1) + \left(1 - \frac{1}{n}\right) \cdot 1 \to 3.$$

On the other hand, clustering every vertex to be one cluster has correlation clustering objective value 1. Thus in the case where there are two clusters in the above example, a merging post-processing improves the overall cluster quality. This crisply captures our motivation.

While the above situation may not be representative, our merging post processing verifies that a possible merge is sound (after ranking possible cluster candidates to merge using the weak signal) by querying a (small) number of strong single values and merging only if the average strong single similarity is sufficiently high. Thus our post processing merge routine can only help the overall clustering.

D.3 Inherent Trade-offs Between Precision and Recall

The goal of this section is to show that there is an inherent tradeoff between precision and recall of *any* clustering algorithm on graphs.We first restate the definitions of precision and recall as defined in our experimental section. Let G be an unweighted (not necessarily complete) graph and let $\mathcal{C}$ be a clustering of its vertices. The edges of G correspond to the edges in the strong signal graph (i.e., the edges are pairs of vertices the strong signal labels as 'similar.'). Correspondingly, the non-edges of G represent the negative edges of the strong signal.

We first restate the definitions of precision and recall as defined in our experimental section. The recall of $\mathcal{C}$ is defined as the fraction of edges of G which are together in some cluster given by $\mathcal{C}$. The precision of $\mathcal{C}$ is defined as the fraction whose numerator is the number of edges of G which are together in some cluster and the denominator is the total number of pairs of vertices that are clustered together.

We state a natural (random) graph dataset such that with high probability, *any* clustering $\mathcal{C}$ has either recall or precision bounded away from 1 by a fixed constant. In particular, G will be sampled from the standard $G(n,1/2)$ Erdos-Renyi graph distribution. Note the order of the quantifiers: we first generate a random graph. There is an event E which G satisfies with high probability. Condition on this event, *any* clustering of the vertices of G will either have its precision or recall bounded away from 1, including the OPT correlation clustering.

Note that we have not made an attempt to optimize the constants in the following lemma for clarity. It is likely that one can optimize our proof and obtain a smaller constant than 0.75.

Lemma 5. *Let G be sampled from the $G(n,1/2)$ distribution. With probability at least $1 - 1/poly(n)$, **all** clusterings of G have recall or precision at most 0.75.*

Proof. Let $C > 1$ be a fixed constant. We first consider the following event E : for any subset S of vertices of size at least $C \log n$, there are at

most $1.01|S|^2/4$ and at least $0.99|S|^2/4$ edges of G within S.

We now show that E holds with probability at least $1 - 1/poly(n)$. For a fixed subset S of k vertices for $k \geq C \log n$, the expected number of edges within S is $\binom{k}{2}/2$. Thus the probability that there are more than $1.01k^2/4$ and less than $0.99k^2/4$ edges within S is at most $\exp(-ck^2)$ by a standard Chernoff bound for a fixed constant $c > 0$. There are $\binom{n}{k}$ such choices of S and thus union bounding over all S and all $k \geq C \log n$, we have that the probability there exists some set S with $|S| \geq C \log n$ vertices violating the required number of edges is at most

$$\begin{aligned}
&\sum_{k \geq C \log n} \binom{n}{k} \exp(-ck^2) \\
&\leq \sum_{k \geq C \log n} 2 \left(\frac{ne}{k}\right)^k \exp(-\varepsilon^2 k^2/6) \\
&= \sum_{k \geq C \log n} \exp(\log 2 + k \log(ne) - k \log k - ck^2) \\
&\leq n \cdot \exp(-\Omega(k^2)) \\
&\leq \frac{1}{poly(n)}
\end{aligned}$$

for $k \geq C \log n$ for a sufficiently large constant C and n large enough. Thus $\mathbb{P}(E) \geq 1 - 1/poly(n)$ where we can make the polynomial arbitrarily large by increasing C. We also condition on the fact that G itself has at least $0.999n^2/4$ edges with also happens with inverse polynomial failure probability.

Now consider an arbitrary clustering $\mathcal{C}$. If the recall of $\mathcal{C}$ is at most 0.75 then we are done so suppose the recall is at least 0.75. Given this, we claim that there exist a cluster within $\mathcal{C}$ of size at least $0.74n$.

To see this, let $C_1, \cdots, C_j$ be the clusters of $\mathcal{C}$. The clusters of size at most $C \log n$ have at most $n \cdot (C \log n)^2/2$ edges of G inside them. All other clusters C_i have at most $1.01|C_i|^2/4$ edges of G inside them. Altogether, the number of edges of G inside some cluster is at most

$$n \cdot (C \log n)^2/2 + \sum_{|C_i| \geq C \log n} \frac{1.01|C_i|^2}{4}$$

subject to the constraint that $|C_i| \leq 0.74n$ and $\sum_i |C_i| \leq n$. This is a convex function which is maximized at its boundary, meaning the number of edges of G inside some cluster of $\mathcal{C}$ is at most

$$\begin{aligned}
n \cdot (C \log n)^2/2 + \frac{1}{0.74} \cdot \frac{1.01 \cdot 0.74^2 n^2}{4} \\
\ll 0.75 \cdot \frac{0.999n^2}{2}
\end{aligned}$$

which contradicts the fact that the recall of $\mathcal{C}$ is at least 0.75. Thus there exists a cluster of $\mathcal{C}$ of size at least $0.74n$. Now given this, we show that the precision must be at most 0.75.

Towards this end, let C_i be the cluster of $\mathcal{C}$ of size at least $0.74n$. It has at most $1.01|C_i|^2/4$ edges of G inside it and $\binom{|C_i|}{2}$ pairs of vertices. Let A be the other edges of G not inside C_i and let B be the total pairs of vertices where both of the vertices in the pair lie outside of C_i. Then the precision of C_i is bounded by

$$\frac{1.01|C_i|^2/4 + A}{\binom{|C_i|}{2} + B}.$$

Since $|C_i| \geq 0.74n$, one can easily verify that

$$B \leq \frac{(0.26n)^2}{2} < \frac{1}{8} \cdot \binom{|C_i|}{2}$$

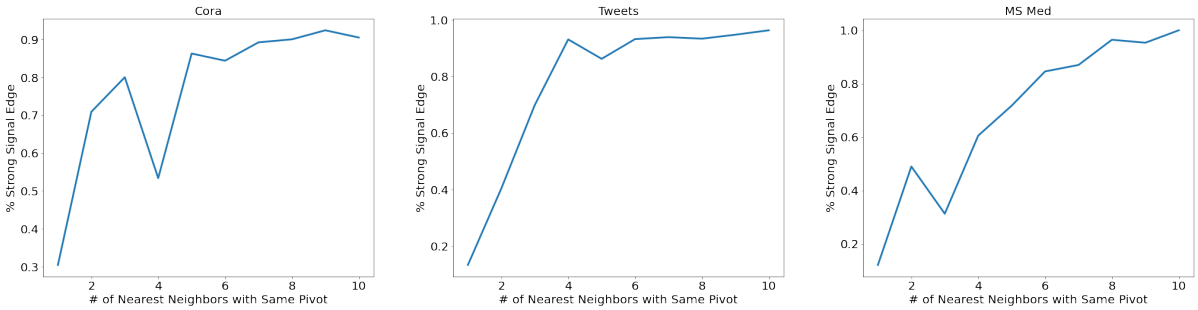
and thus

$$\begin{aligned}
\frac{1.01|C_i|^2/4 + A}{\binom{|C_i|}{2} + B} &\leq \frac{1.01|C_i|^2/4 + B}{\binom{|C_i|}{2} + B} \\
&\leq \frac{1.01|C_i|^2/4}{\binom{|C_i|}{2}} + \frac{B}{\binom{|C_i|}{2} + B} \\
&\leq 0.51 + \frac{B}{9B} \\
&< 0.75,
\end{aligned}$$

as desired. □

D.4 Motivating Using Weak Signal Neighborhood Statistics.

In Figure 2, we plot the the fraction of times a vertex v connects to a pivot p in the `KwikBucks` algorithm as a function of the number of nearest neighbors of v (in terms of the weak signal similarity) which have already connected to the same pivot p.. We see that the probability increases as a function of the number of nearest neighbors, empirically justifying our algorithmic design optimization of 'Utilizing Weak Signal Neighborhood' in Section C. Note that this optimization has the affect of slightly boosting such pivots p (if they exist) to a higher similarity (and thus a better ranking).



(a) (b) (c)

Figure 2: The propensity for a vertex v to connect to a pivot p given that k of v's neighbors have already connected to p.

E Details for Dataset & Weak/Strong Signals

We provide a detailed description of the datasets used in the paper as well as the weak and strong signals used for each of the datasets. Table 2 provides a summary.

Stackoverflow (SOF) and SearchSnippets: Stackoverflow and SearchSnippets are commonly used for short-text clustering/classification. For stackoverflow, we used a subset collected by (Xu et al., 2017) consisting of of 20,000 question titles associated with 20 different categories obtained from a dataset released as part of a Kaggle challenge. For SearchSnippets, we used the dataset from (Phan et al., 2008) which consists of 12,340 snippets (extracted from web search snippets) associated with 8 groups. For these two datasets, we experimented with two different types of cheap signals: word2vec embeddings (Mikolov et al., 2013) and tf-idf embeddings. In both cases, we trained/finetuned on the training set of the datasets. We used the Gensim package (Rehurek and Sojka, 2011) for word2vec and sklearn (Pedregosa et al., 2011) for tf-idf. Word2vec provides a vector representation for each English word; to compute the embedding for a sentence/document, we average the embeddings of each of its words. For the strong signal, for each dataset we finetuned a T5-1.1 XXL model (11B parameters) (Raffel et al., 2020) on the training data where given two examples, the model was finetuned to predict if they belong to the same cluster or not. In both cases, we sampled 10K positive pairs and 50K negative pairs and finetuned the model for 10 epochs on a 4x4 DragonFish TPU architecture.

Twitter and AgNews: Twitter and News data are commonly used for short-text clustering/classification. From Twitter, we use the dataset created by (Yin and Wang, 2016) consisting of 2,472 tweets with 89 categories. From News, we use the data from (Rakib et al., 2020) which is a subset of the dataset from (Zhang and LeCun, 2015) containing 4 topics. For the cheap signal, we use pretrained BERT embeddings (Devlin et al., 2018) where we feed each example into the BERT model, obtain contextual token embeddings, and then average them (ignoring the [CLS] and [SEP] tokens) to obtain the embedding for each example. We use the 12-layer uncased BERT-Base model for this experiment. For the strong signal, we first created a graph by connecting two nodes if they belong to the same category, then added noise to the graph by flipping the existence/non-existence of an edge for 5% of node pairs selected uniformly at random (note that without adding noise, the problem becomes much easier as graph of the strong signal becomes composed of multiple connected components).

Internal: This is a vertical of a large, internal, proprietary text dataset. The weak signal is embedding similarity, and the strong is an indicator variable from a cross-attention model.

Citeseer and Microsoft Medicine: Citeseer (Sen et al., 2008) and Microsoft Medicine (Shchur and Günnemann, 2019) are attributed graph datasets. Citeseer is a citation network in which nodes represent papers, edges represent citations, and features are bag-of-word abstracts. Microsoft Medicine is a subset of the Microsoft Academic graph where the nodes represent authors, edges represent co-authorship, and node features are a collection of paper keywords from author's papers. For both datasets, we used the cosine similarity between the node features as the weak signal and

we assume the edges of the graph correspond to the strong signal.

Cora and Amazon Electronics Photos: Similar to Citeseer and Microsoft Medicine, Cora and Amazon Electronics Photos are also attributed graph datasets. They are typically used for node classification but here we adapt them to our problem. Cora (Sen et al., 2008) is a citation network similar to the Citeseer dataset with the node labels corresponding to paper topics. Amazon Electronics Photos (Shchur et al., 2018) is a subgraphs of the Amazon copurchase graph where the nodes represent goods, an edge between two nodes represents that they have been frequently purchased together, node features are bag-of-word reviews, and class labels are product categories. For these two datasets, we used the deep graph infomax (DGI) model (Velickovic et al., 2019) to learn unsupervised node representations and used these representations as the cheap signal. We also used noisy labels as the strong signal similar to the Twitter dataset.

Total cost analysis: Our work is mostly based on the applications where the weak oracle values are computed via distances based on embeddings and the strong signal values are the output of a large cross-attention transformer model. In this case, there are three different factors that comprise the total cost of the clustering algorithm: 1- the cost of the queries to the strong signal, 2- the cost of computing embeddings from the cheap signal, and 3- the cost of geometric operations on the embeddings. So the total cost can be summarized as follows:

$$Total\ Cost = \eta_S\zeta_S + \eta_E\zeta_E + \eta_G\zeta_G$$

where η_S represents the number of calls to the strong signal, ζ_S represents the cost of making a call to the strong signal, η_E represents the number of calls needed to compute embeddings, ζ_E represents the cost of obtaining one embedding, η_G represents the number of geometric operations (cosine similarity in our case) we perform on the embeddings, and ζ_G represents the cost of a single geometric operation.

The number of calls η_E required to obtain embeddings is n (i.e. the number of data points) which is smaller than η_S (which, in our case, is typically a linear factor of n) and the cost ζ_E of obtaining one embedding is significantly smaller than the cost of obtaining one strong signal similarity ζ_S. Therefore, $\eta_E\zeta_E$ can be subsumed in $\eta_S\zeta_S$.

When using 32 TPU v3 chips for the strong signal and a CPU for the geometric operations, each call to the strong signal was approximately 10^4 times slower (i.e. $\zeta_S \approx 10^4\zeta_G$). This gap becomes even more stark if we use fast geometric algorithms such as nearest neighbor search or use TPUs for geometric operations. It follows from the analysis of our algorithm that $\eta_G \in O(nk)$ where k is the parameter defined in Algorithm 6. This is comparable to η_S. Therefore, $\eta_G\zeta_G$ is negligible compared to $\eta_S\zeta_S$ in our experiments.

Following the above justifications, as well as for theoretical simplicity, in this paper we ignored the cost of querying the weak signal in our analysis (i.e. assume $\eta_E\zeta_E + \eta_G\zeta_G \approx 0$). However, if future work considers costlier operations for the cheap signal, these extra terms should also be considered in determining the total clustering cost.

F Additional Experimental Results

F.1 Precision and Recall

The precision and recall (with respect to a clustering C) definitions used in Section 4 are defined as follows:

$$\text{Precision}(C, O_S) = \frac{\sum_{e=(i,j)} C_{i,j}O_S(e)}{\sum_{e=(i,j)} C_{i,j}} \quad (2)$$

where $C_{i,j}$ is the indicator for if vertices i, j are in the same cluster.

$$\text{Recall}(C, O_S) = \frac{\sum_{e=(i,j)} C_{i,j}O_S(e)}{\sum_{e=(i,j)} O_S(e)}. \quad (3)$$

As stated in (García-Soriano et al., 2020b), while our algorithm and baselines have been designed to minimize the total correlation clustering objective, it is important to consider precision and recall as they are problem independent measures of cluster quality. Furthermore in cases where the underlying strong signal graph is extremely sparse, the correlation cost objective might not be meaningful. For example in such a case, returning all vertices as singleton clusters already has low objective value (equation 1). We use the entire strong signal graph for the purposes of evaluating the experimental metrics, such as CC objective, precision, and recall.

F.2 Parameter Selection Details

We first describe how to select the value t in Algorithm 3 and k in Algorithm 6, which selects the top

Table 2: Properties of datasets used in our experiments. n denotes the number of vertices and Non-zero entries denotes the number of non-zero entries in the adjacency matrix of the strong signal graph (i.e. twice the number of edges), both rounded to two significant digits.

Name	Type	Weak Signal	Strong Signal	n	Non-zero entries
SOF	Text	W2V / tf-idf	Cross-attention model	$4.9 \cdot 10^3$	$2.3 \cdot 10^6$
Search	Text	W2V / tf-idf	Cross-attention model	$3.3 \cdot 10^3$	$2.0 \cdot 10^6$
Twitter	Text	BERT Embeddings	Noisy label indicator	$2.4 \cdot 10^3$	$4.7 \cdot 10^5$
AgNews	Text	BERT Embeddings	Noisy label indicator	$8.0 \cdot 10^3$	$1.8 \cdot 10^7$
Internal	Text	Embeddings	Cross-attention model	$1.0 \cdot 10^5$	$9.5 \cdot 10^7$
Cora	Attributed Graph	DGI Embeddings	Noisy label indicator	$2.7 \cdot 10^3$	$1.5 \cdot 10^6$
Photos	Attributed Graph	DGI Embeddings	Noisy label indicator	$7.7 \cdot 10^3$	$1.2 \cdot 10^7$
Citeseer	Attributed Graph	Node Features	Adjacency matrix	$3.3 \cdot 10^3$	10^4
Med.	Attributed Graph	Node Features	Adjacency matrix	$6.3 \cdot 10^4$	$1.6 \cdot 10^6$

k vertices in weak-signal similarity for the strong signal to query.

The intuition in picking t is that it must be sufficiently large so that only few vertices do not have a pivot in their neighborhood (and thus contribute to the additive error of Theorem 1). This parameter naturally depends on the density of the underlying strong signal graph: for sparser graphs, one must pick a larger value of t since each vertex on average has a small degree and is thus less likely to have a pivot chosen in its neighborhood than a vertex with a larger degree. We use the above intuition to design the following data-dependent method to select t: we first sample a sublinear number of random strong signal edges ($\sqrt{n}$ strong signal edges to be exact). This returns an estimate of the density of the graph up to small additive error (for example via standard Chernoff bounds). We then set t to be 10 times the inverse of the density. If the density is extremely sparse, i.e. less than $1/1000$ fraction of possible edges exist, we simply set t to be equal to $n/2$.

The second parameter we set is k in `WeakFilterByRanking`. We can pick a value of $k \ll n$ because intuitively, a meaningful weak signal assigns a high similarity score to relevant pivots relative to all other pivots and thus such pivots have higher ranking. To understand the trade offs in selecting k, consider the most prominent place where it is used in our algorithm: when a vertex v attempts to find a strong signal edge to one of the pivots by iterating through them in the weak signal ordering. The trade offs are the following: a smaller value of k leads to better query efficiency as v is guaranteed to only make k strong signal queries in this step. However the clustering quality can suffer because the first k pivots, for a small k, in the weak signal order might not have a strong signal edge to v. Conversely a larger value of k leads to increased exploration from v as it attempts to connect to a pivot. However in the case that v is truly a singleton cluster, i.e. it has no strong signal edges to any pivot, we potentially waste many strong signal queries. To balance these trade offs, we pick an 'intermediate' value of $k = 100$ *for all our experiments*. Ablation studies for both parameters are given in Section F.

We also always set $k = 10$ when we use the "Utilizing Weak Signal Neighborhood" optimization of Section 3. We also always fix $\lambda = 1/10$ which appropriately normalizes the second term to be between 0 and 1 (note the weak signal similarity w_p is between -1 and 1). The parameter 10 here is fairly robust and can likely be replaced by any (small) reasonable value and we also perform ablation studies on this optimization.

For spectral clustering, we always use $k = 25$ for the number of clusters. Higher values were computationally prohibitive to use.

F.3 Results

We present additional experimental results in Figure 3 and 4 which show similar qualitative results as Figure 1: our algorithm `KwikBucks` has superior query complexity over the baselines as it achieves a higher F_1 value (and lower CC objective values) while utilizing fewer strong signal queries than baselines.

F.4 Additional Ablation Results

In our ablation experiments, we fix all parameter settings except the component we are altering. We perform ablation studies on 4 representative

Table 3: CC objective values are shown for a fixed budget of $3n$. See Table 1 for the corresponding F_1 values. We normalize the smallest CC value to 1.0 so smaller quantities are desirable. See Figures 3 and 4 for results as a function of query budget. For the sparser graph datasets of Citeseer, Med., and Internal we use the budget of $50n$. Due to their sparsity, the CC objective value is less meaningful than F_1 values for these two datasets.

	SOF	**Search**	**Tweet**	**AgNews**	**Cora**	**Photos**	**Citeseer**	**Medicine**	**Internal**
B1	1.9	2.5	1.2	2.0	2.0	2.5	1.3	1.1	1.01
B2	1.8	2.0	1.2	2.0	2.0	2.4	1.3	1.1	1.04
B3	6.4	4.0	6.3	2.5	2.5	2.2	745.1	2550.8	-
B4	2.0	6.0	1.1	4.1	3.0	3.2	1.0	1.0	1.01
B5	2.0	6.0	1.1	4.1	3.0	3.2	1.3	1.3	1.01
KwikBucks	1.0	1.0	1.0	1.0	1.0	1.0	1.1	1.0	1.0

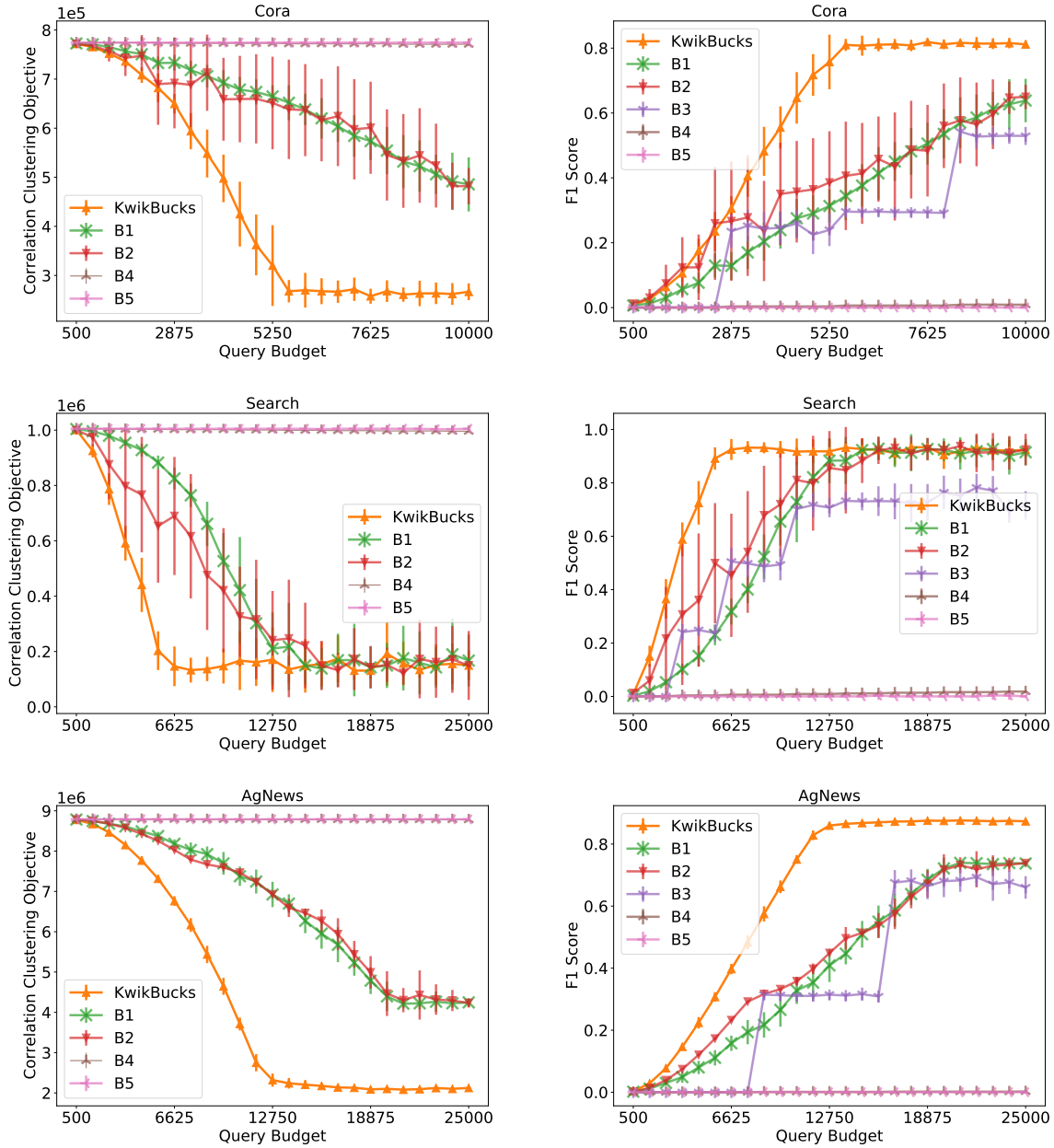


Figure 3: Empirical results for the Cora, Stackoverflow, Search, and News datasets.

datasets: Cora, Citeseer, Stackoverflow (SOF), and Search. Our first observation is that the merge post processing procedure can help return a higher quality clustering, for example for the Cora, SOF, and Citeseer datasets; see Figures 5 and 6 for details and Section D.2 for theoretical intuition of why post processing merging helps.

Next we consider removing the `SortNonPivots` step and replacing it with an using an arbitrary ordering of non pivot vertices. We see that the positive benefits of removing this component are more subdued compared to the merge post processing. However, this change never hurts the quality of the clustering. Overall, we view the different data-driven components introduced in Section 3 and C as having complementary benefits as each optimize a different part of the algorithm.

We observe that one must choose a sufficiently large value of t in `GetPivots` which is the initial number of random vertices sampled which are later processed to be pivots. As argued in Section F.2, it is important to select a sufficiently large value of t to limit the number of vertices which do not have a pivot in their strong signal neighborhood (as captured by the additive error term in Theorem 1). For our ablation studies, we consider two other settings of t, one which is a factor of 10 smaller than the choice used in our main experimental results and one which is a factor of 10 larger. They represent the 'Small' and 'Large' pivot choices respectively. We see in Figures 5 and 6 for the Cora and Citeseer datasets, that a smaller choice of t can lead to a decrease in the performance of our algorithm. Nevertheless, our data driven density based approach outlined in F.2 hits the 'sweet spot' and performs comparable to the best choice of pivots in all cases as shown in Figures 5 and 6.

We also perform ablation experiments on the choice of k in `WeakSignalFilterPractice` by considering $k = 10$ and $k = 1000$ (a 'small' and 'large' choice respectively as before). Our ablation experiments also show that a large choice of k in `WeakSignalFilterPractice` can lead to many queries wasted as argued in Section F.2. Indeed, we see in the above figures that for the Citeseer dataset, a large value of k leads to worse performance initially as we waste many strong signal queries on vertices which have no strong signal edge to any of the pivots. This is due to the sparse nature of the Citeseer dataset. However as the query budget is increased, the quality of the clustering improves. The choice of k seems to have negligible impact on the other datasets we tested on and our choice of $k = 100$ (which we fixed in the main experimental results) was always competitive.

F.5 Measuring the Quality of Weak Signals

We design a simple and informative experiment to measure the quality of weak signals. For the Stackoverflow (SO) dataset, we run `KwikBucks` where we replace the weak signal with a linear *interpolation* of the strong signal and a random matrix will all entries i.i.d. from the uniform distribution in $[-1, 1]$. The purpose of this experiment is to show a higher quality weak signal gives better clustering results than using a lower quality weak signal. Indeed, Figure 8 shows that `KwikBucks` performs the best if we replace the weak signal completely with the strong signal, as naturally expected. As we vary the amount of randomness in the weak signal, the performance degrades and the case where the weak signal is a fully random matrix performs the worst as a function of query budget. It is also interesting to consider the cases where the weak signal are given by the (stronger) W2V model versus the comparatively weaker tf-idf model: the performance of using the W2V embeddings for the weak signal lies between the 'half-random' and '$2/3$ random' case whereas the tf-idf plot lies between the '$2/3$ random' and 'fully random' cases. The random interpolated weak signal cases, while artificial, help us qualitatively access the usefulness of a particular real world weak signal instance.

F.6 Average Rankings of Strong Signal Neighbors

In this Section we present additional experiments in the similar spirit as the right figure of the second row of Figure 1 for the Tweet, Med., and Cora datasets. For every vertex v in these datasets, we rank all the other vertices in decreasing weak signal similarities to v. The average rank of the true strong signal neighbors of v is computed and plotted as a histogram (normalized to be a distribution). Intuitively, a good weak signal should have the property that *true* strong signal neighbors have much higher weak signal similarity scores (and thus better rankings) than the an arbitrary vertex. Indeed, we see that to be the case of the datasets in Figure 9 where the distributions are much more left shifted and has a much smaller mean compared to the case if the weak signal was fully random. This validates the connection between our empirical weak signal Def-

inition 3 and the theoretical assumption we made for the weak oracle in Assumption 1. Indeed, Figure 9 gives empirical validation to the claim that returning a top k most similar vertices to a vertex v in terms of weak signal similarity captures many actual *true* strong signal neighbors.

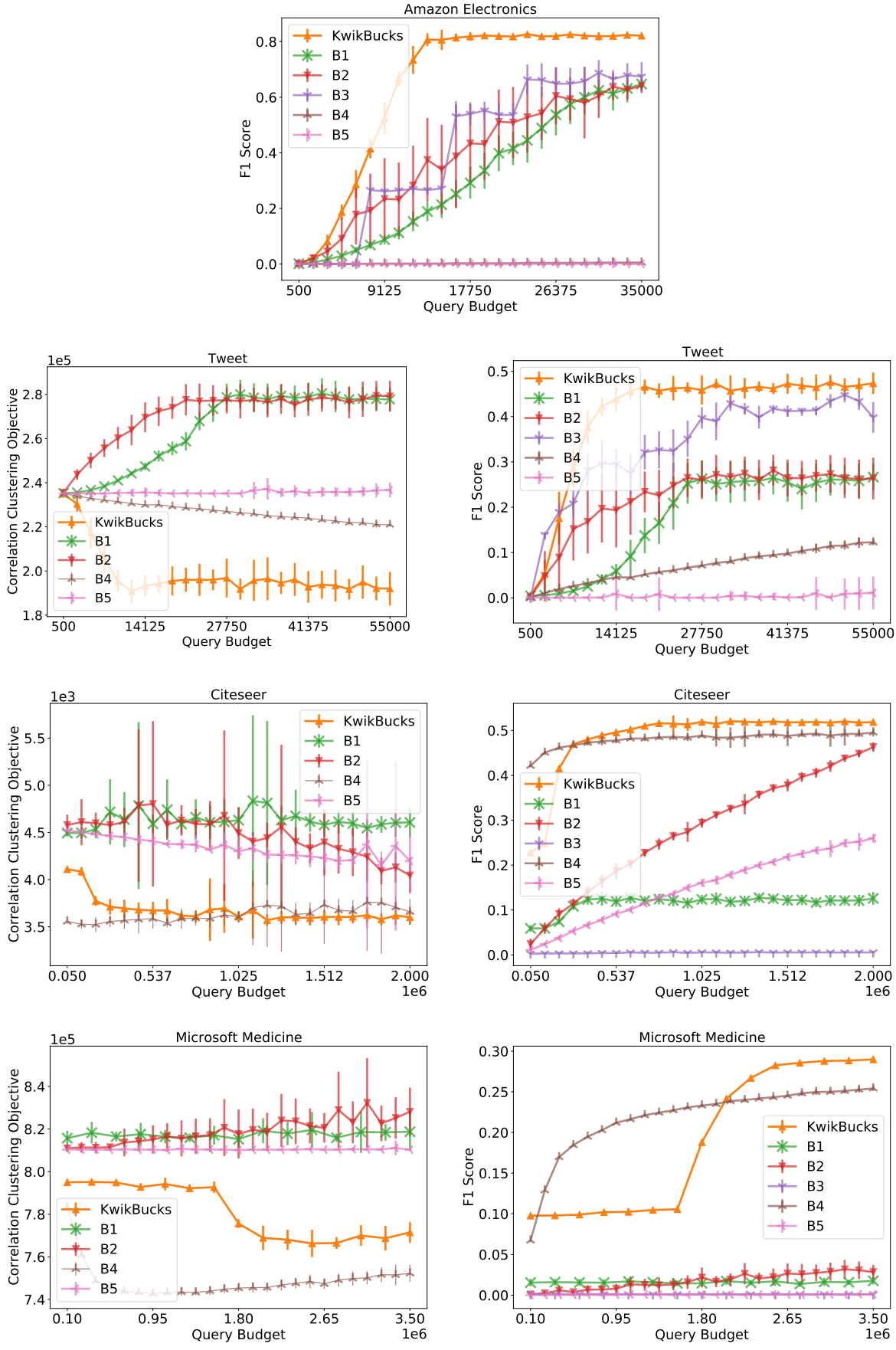


Figure 4: Empirical results on the datasets omitted from Figure 3. The results are qualitatively similar to that of Figure 3.

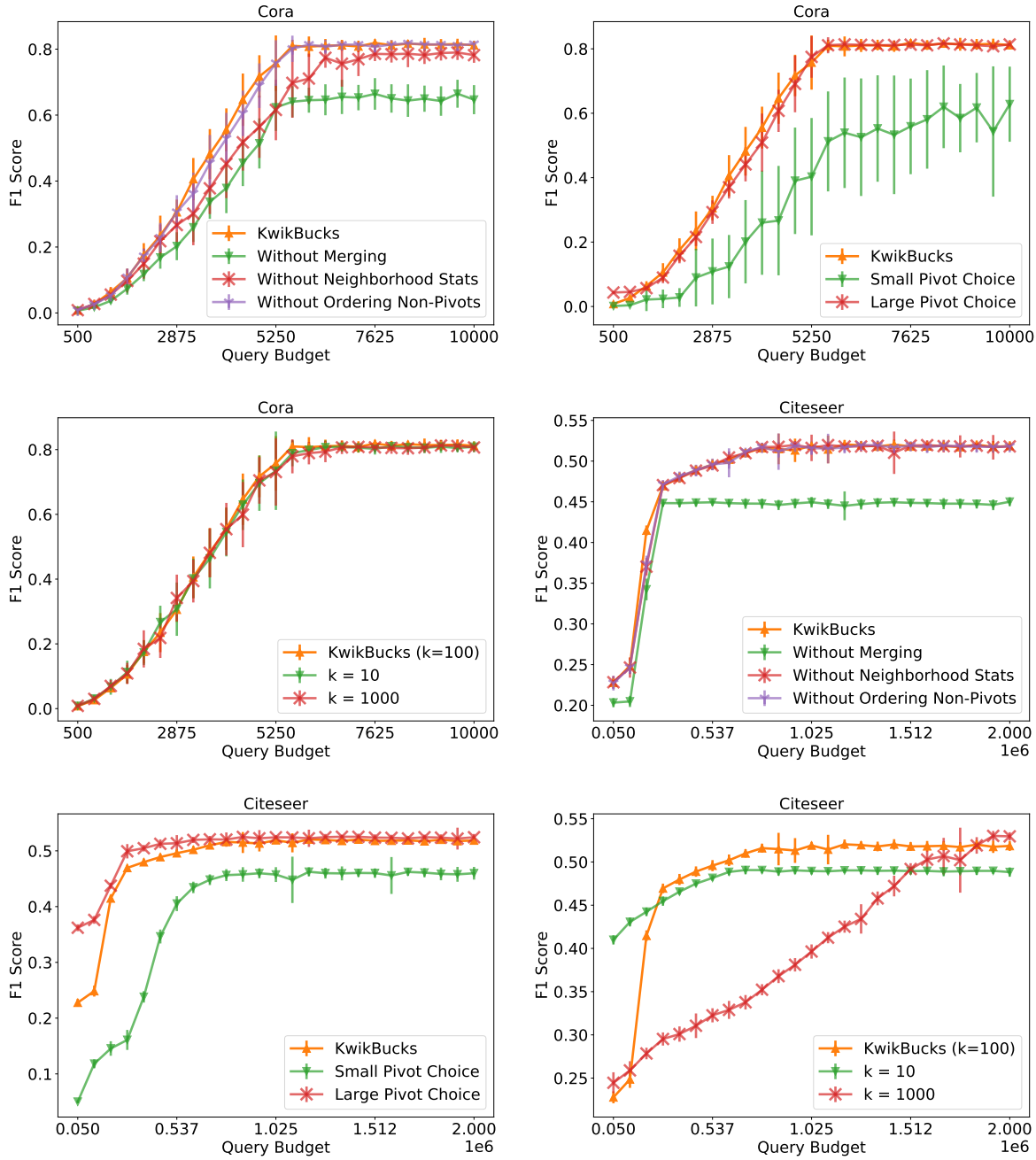


Figure 5: Figures for ablation studies for Cora and Citeseer datasets.

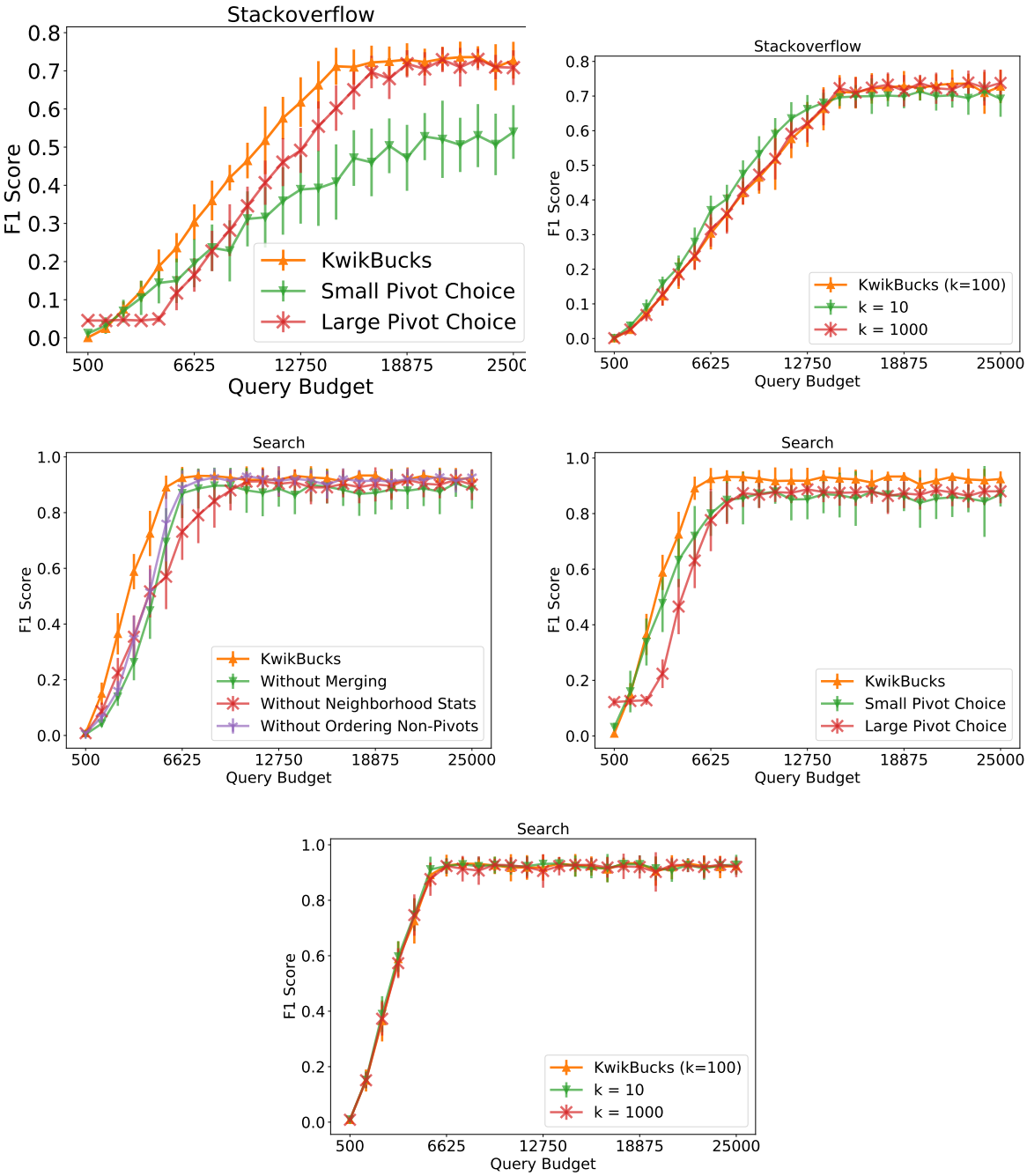


Figure 6: Figures for ablation studies for SOF and Search datasets.

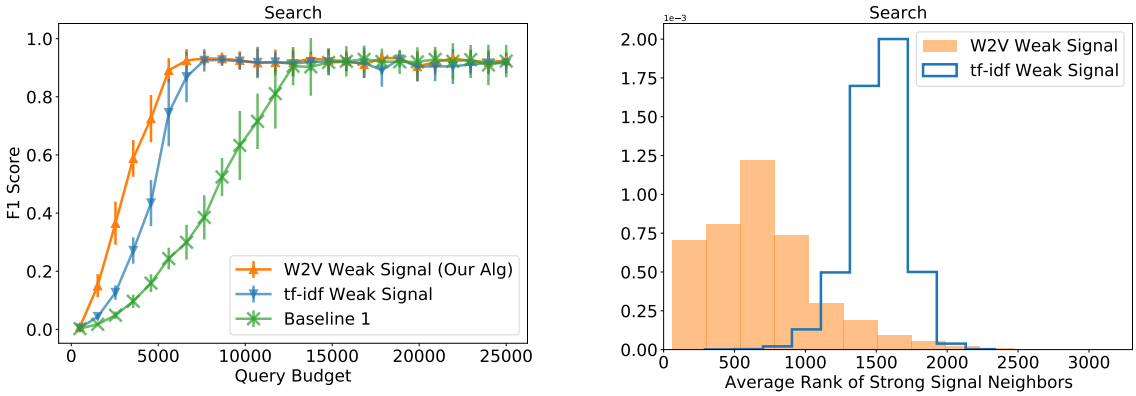


Figure 7: The figure shows a qualitatively similar result as the SOF results shown in Figure 1.

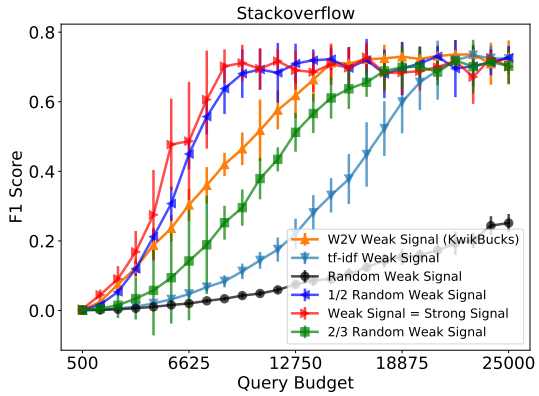


Figure 8: Interpolating the weak signal between uniformly random values and the strong signal.

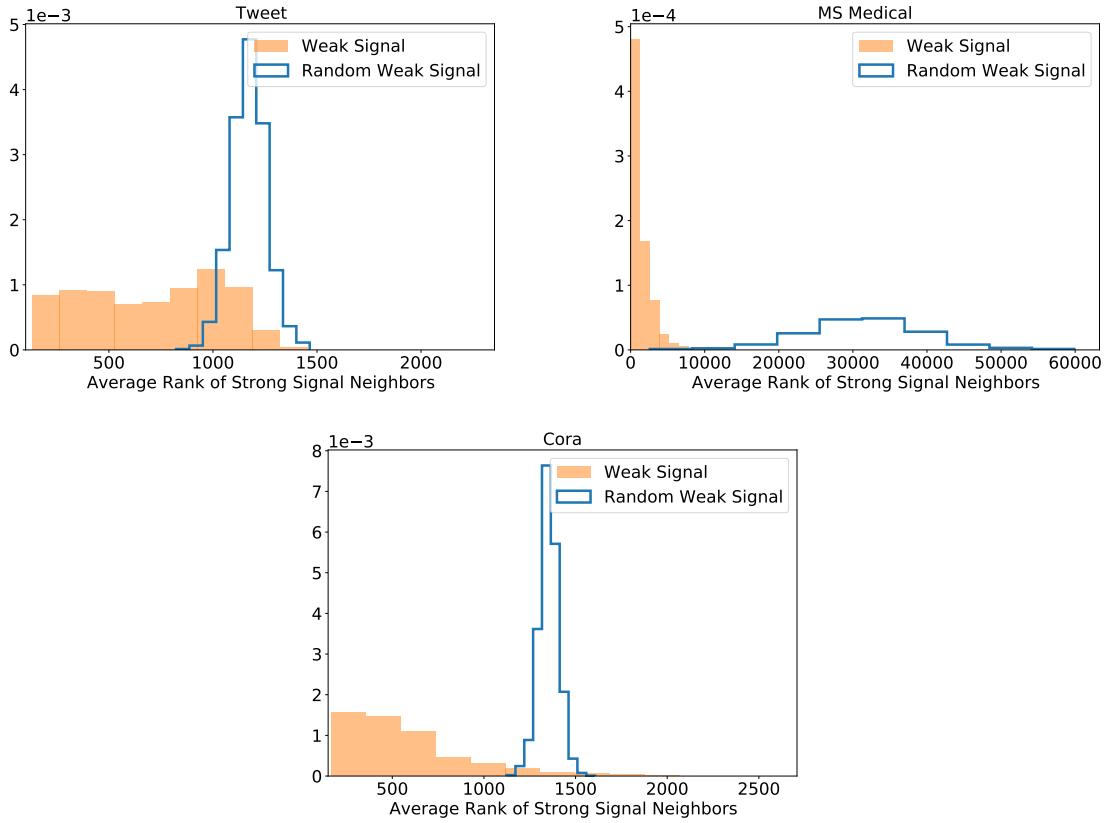


Figure 9: The average weak signal rank of actual strong signal neighbors is shown in orange. The blue curve shows the average rank if the weak signal was fully random.