Deciphering Vague Information Searches via Search Result Clustering

Katrina Truebenbach
Data Science Department
Northeastern University
Boston, MA USA
truebenbach.k@husky.neu.edu

1 Introduction

The goal of a search engine is clear: enable users to find their desired results as quickly as possible. This is often not an easy task as the search engine must not only return comprehensive results quickly, but also is ideally able to predict intent from a user's search query. There are two types of searches: known item searches and information searches. Known item searches are usually specific and contain the name of the desired website or document. Information searches are often conveyed via a vague search term, even though the searcher has some more specific question in mind. For example, if a user searches for "jaguar", are they looking for information about the animal or the car?

This report explores a less traditional but heavily researched method of helping users quickly find relevant information despite potentially vague search queries: search result clustering. Result clustering provides the user "clusters" of related documents rather than providing those results as a sorted flat list. In the jaguar example, the engine would likely return one group of documents about jaguar cars and another about jaguar animals. These clusters also need to be labeled as "car" and "animal" so that the user can easrily differentiate.

This technique has the potential to vastly improve how people search. Most commercial search engines personalize results by using data on a user's past searches and clicks, social media history, location etcetera. However, there are several problems with this approach. First, it takes time to accumulate data about a user's preferences, thus creating a "cold start" problem. Second, if the search is unrelated to existing data for that user, it is akin to a cold start. Finally, personalization may over-specialize such that if a car enthusiast wants to know about animals, the top results will only be about jaguar cars. This is especially problematic given that a typical Google user only considers the first few results. In fact, Google data shows that moving a search result one position up on the page results in a 30.8% higher clickthrough rate on average (Dean 2019). Meanwhile, clustering requires no user data and the user only needs to scan a few clusters to understand the range of returned results.

Despite extensive research, search result clustering has not gained popularity as a commercial search technique and thus has a lot of pitfalls and open questions. In this report I implement versions of two popular techniques that have opposing objectives and thereby offer the following contributions:

1. Describe in detail hyperparameter and heuristic choices intended to optimize each algorithm.
2. Directly compare the two algorithms on the same sets of search queries and results in order to understand both how and why their results differ.
3. Recommend a preferred method.
4. Reflect on why, regardless of the algorithm, search result clustering is not a popular search technique given the challenges encountered in implementing both techniques.

2 Related Work

There are many ways of approaching search result clustering. Carpineto et al. (2009) categorizes the research into three categories of algorithms: data-centric, which find good clusters, but struggle to label them; description-aware, which cluster and label based on only one feature; and description-centric, which focus on labeling and allow those labels to guide the clustering. This paper will focus on data-centric and description-centric as they propose opposing clustering objectives. Data-centric algorithms perform traditional clustering where documents are assigned to clusters based on their similarity to each other, while description-centric algorithms assign documents to clusters based on their similarity to a chosen label.

Data-centric algorithms generally use agglomerative hierarchical clustering. This allows the user to view top-level clusters and then navigate to sub-topics. A popular method of labeling clusters is to take the average vector of the cluster and choose the most frequently appearing words in that vector as the label (Carpineto et al 2009).

Description-centric algorithms, as defined by Carpineto et al, are more varied. This report bases its description-centric algorithm off of the Lingo algorithm, which uses Singular Value Decomposition (SVD) to find labels that best describe the latent concept space. Other description-centric algorithms include DisCover (Kummamuru et al. 2004), which iteratively chooses concepts to define clusters, and SRC, which uses supervised techniques to score potential cluster labels (Zeng et al. 2004).

The existing literature focuses on developing better algorithms for search result clustering, and thus there are not enough holistic comparisons of existing algorithm results on the same set of documents. The relatively small set of papers that compare algorithms focus on numeric indicators of cluster cohesion and

separation (Carpineto et al 2009; Leouski and Croft 1996; Mahalaskhmi and Praba 2013). Quantitative comparison is important, but can neglect two important questions: (1) are the clustered results practically useful to a user[1] and (2) why do different algorithms return sometimes significantly different clusters for the same search results? This report provides both quantitative and qualitative comparisons of two algorithms, as well as explores the reasons for the differences in results.

There are also important practical questions that are often omitted from the literature such as choosing hyperparameters and defining heuristics that can be applied to results returned from any search term. This report provides these details and discusses why some of those details make search result clustering difficult across algorithms.

3 Background Information

The classic objective for clustering is to create cohesive and distinct clusters. Specifically for search result clustering, clusters must represent a coherent topic as well as a distinct topic from other clusters. Additionally, clusters must be labeled such that users can quickly understand their contents.

Additional challenges include creating an effective user interface and ensuring that the algorithm can run quickly at query time. The first of these is beyond the scope of this paper and the second is often not an issue because the clustering is only run on results returned for a query, not the entire corpus. This report also seeks to order the clusters appropriately, but will not broach ordering results within a cluster, as that is an information retrieval algorithm question.

There are a large number of clustering techniques that could be applied to this problem. As discussed, this report focuses on just two techniques – agglomerative hierarchical clustering and the Lingo algorithm. These techniques use opposing methods of approaching search result clustering and thus are interesting for comparison. Most other techniques take some middle ground or are less applicable to the specific problem. For example, I do not use K-Means because it is also a data-centric technique but does not provide sub-clusters, which is a significant advantage of hierarchical clustering.

Finally, I define metrics used to evaluate cluster quality. Distortion measures internal cohesion as the sum of squared error between each point and its cluster's centroid. The silhouette coefficient measures both cohesion and separation for each point as one minus the ratio of the average distance to other points within the same cluster and the average distance to points in the closest neighboring cluster. The coefficient will be between zero and one where scores closer to one represent more cohesive and separated clusters.

[1] Some papers have users manually evaluate clustering results for external validation. This paper attempts to provide qualitative comparisons without relying on manual evaluation.

4 Proposed Approach

The proposed approach has two components. First, implement versions of hierarchical clustering and the Lingo algorithm including hyperparameter tuning to optimize each algorithm. Second, develop metrics and techniques to compare the results.

4.1 Algorithm Implementation

The goal of search result clustering is to produce coherent and distinct clusters with representative, human-readable labels. Additionally, these clusters must be easy to use and thus we want a relatively small number of evenly sized clusters (top-level clusters for hierarchical), otherwise users will be overwhelmed by either a large number of overly-specific clusters or one large generic cluster that includes all but a few results. Finally, the mechanism must be able to produce high quality, labeled clusters across a wide variety of search terms and results. The hyperparameter tuning must thus use some universal heuristics that apply across all searches.

The input data is the raw text of documents returned from a search query. This raw text must first be preprocessed (this can be done pre-search and attached as metadata to each document). Preprocessing includes removing symbols, numbers, punctuation, and stop words. The stop words may be augmented if the search engine's ingested corpus is topic specific.

A second round of preprocessing occurs at search time based on the documents returned for a search term. All words are stemmed so that words with the same stem are equivalent. However, stems are often not human-readable, so when creating cluster labels, I record the most frequently occurring word for each stem among the returned documents and substitute it for the stem in labels. Finally, I form a term frequency-inverse document frequency (TF-IDF) vector space model from the processed text of all documents returned by a search, thereby placing each document in Euclidean space.

4.1.1 *Agglomerative Hierarchical Clustering* The first clustering technique uses hierarchical clustering and focuses on cluster rather than label quality. First, I use Principal Component Analysis (PCA) to reduce the dimensionality of the TF-IDF matrix by retaining 80% of variance. The unreduced documents have a large number of words and thus dimensions, making them all relatively far apart in Euclidean space. I choose 80% because it is at the lower end of the typical range of retained variance (80-90%) and I am interested in the overall topics, not the details, of the documents, but also do not want to lose important information.

Next, I perform traditional agglomerative hierarchal clustering where documents are iteratively combined into clusters until all documents are in one cluster. I use the Ward metric to define the distance between clusters, which measures the increase of within-

group sum of squares resulting from joining two clusters. It thus focuses on cluster cohesion rather than separation. Because the returned documents are already within one general topic (returned by the same search term), internal cohesion is a better measure of cluster quality. Other metrics that focus on cluster separation such as Single Link or Group Average generally form a very small number of large clusters.

This clustering process forms a dendrogram that I then cut at some number of clusters k to form the top-level clusters that are presented to users. This is where a universal heuristic for choosing k across all search terms and returned documents is necessary. This is also why I do not cut the dendrogram at some specific distance: some sets of documents will be naturally closer or farther apart on average than others. To select k, I calculate the average distortion of the top-level clusters using various values of k and select the k at the elbow in the negative rate of change of distortion. I only consider k in the range between two and ten clusters. Ten is the number of results per page on Google and thus is an outer bound on the number of results that a user is likely to scan through. Without this restricted range, the selected k is often too large because the documents returned by a search term are already related, so sometimes the best formed clusters represent very small, specific sub-topics. This is also why I find the elbow in the rate of change of distortion rather than in distortion itself, which is a more typical technique: the distortion in this limited k range is often too smooth to find a true elbow, but the rate of change is more varied and better indicates where the decrease in distortion flattens.

I then label the clusters with the three highest scoring words in the mean unreduced TF-IDF vector for each cluster. This can be repeated for sub-clusters within top-level clusters. Finally, I sort the clusters for user presentation by calculating cluster size times the cluster's average silhouette score. This prioritizes larger, well separated clusters, which should represent the dominant sub-topics in the clustering.

4.1.2 Lingo Algorithm: Dimensionality Reduction The second clustering technique is adapted from the Lingo Algorithm and focuses on label rather than cluster quality. First, I perform SVD on the unreduced TF-IDF matrix to find the latent concepts (and thus clusters) within the documents. Similarly to hierarchical, I must use a global heuristic to choose the number of concepts k to retain. Typical heuristics such as preserving 80 to 90 percent of variance returns too many clusters because, as discussed, the documents are already related and have a tendency towards a large number of small clusters. Instead I calculate the percent of retained variance at each possible value of k and choose the k at the elbow in the decreasing rate of change of (increasing) retained variance. Again, retained variance is too smooth at small values of k and thus the rate of change gives better results. This method almost always gives an appropriate number of clusters in the range of two to ten and thus imposing a restricted range for k is unnecessary.

Next, I find labels for each of the k clusters by taking the three largest values in each column (concept) of the reduced V matrix. The V matrix's elements give the strength of the relationship between each word and concept. Therefore, the labels are the words that best describe the latent concepts that each cluster represents.

Documents are then assigned to clusters based on the strength of the document's relationship with the cluster labels. This is calculated by multiplying TF-IDF vectors of the labels by the transpose of the full TF-IDF matrix. I assign a document to a cluster if it has a strength of at least 0.1 with any of the three words in the label. This allows documents to be in multiple clusters because a document can have sufficiently strong relationship with words in multiple labels. Also, a document can be in zero clusters. The strength threshold balances a tradeoff: a high threshold results in many documents being in zero clusters, while a low threshold results in many documents being in multiple clusters. To avoid losing information, I chose a lower threshold. This trade-off is emblematic of a major weakness of the Lingo algorithm, as will be discussed later.

The final step is to combine clusters that have overlapping labels (labels containing the same words) and then relabel the combined clusters. When relabeling clusters, I take the three words with the strongest word-concept relationship among all of the words in the labels of the clusters that are being combined. Lingo's clusters are not distinct in a traditional sense because documents can appear in multiple clusters, but by eliminating label overlap, they are distinct in that they represent truly different sub-topics. Furthermore, it is logical that a single document could belong to two sub-topics.

Finally, clusters are sorted by multiplying the cluster size by the highest label-concept score in each cluster, thereby prioritizing larger clusters with representative labels.

This algorithm is very different from hierarchical clustering. The labels are first determined by the words that best describe the latent concepts in the documents. Then cluster assignments depend solely on how well the documents fit those labels. Documents are never directly compared to each other. Conversely, hierarchical clustering first creates clusters based on document similarity and then forms labels based on those grouped documents.

4.2 Algorithm Comparison

In comparing the algorithms' results, I want to answer two questions: (1) how and why are the results different and (2) which algorithm produces clusters that best allow users to find information faster?

In this comparison, I use both quantitative and qualitative techniques. Quantitatively, I calculate average distortion and the average silhouette coefficient of the overall clusterings to measure cohesion and separation. Additionally, I compare the average number of clusters per search term and the average number of documents per cluster. For Lingo clusters, I also

consider the average percent of documents without a cluster and the average percent of documents in multiple clusters. Finally, I calculate the overlap in labels between the algorithms, which helps understand the degree of difference but not the quality of labeling.

Quantitative metrics are not sufficient to answer either of my questions, but rather guide qualitative investigations: I specifically examine the clusters for searches that have significant differences in the above metrics for the two algorithms. Some of these investigations are discussed in the next section.

5 Experiments

I implement the above algorithms on a small corpus of documents. These experiments show that hierarchical clustering produces a larger number of small, well-formed clusters, but often with confusing, overlapping labels. Lingo clustering produces a smaller number of large clusters with distinct and logical labels, but is often overly aggressive in combining apparent sub-topics in favor of logical labels. I ultimately recommend hierarchical clustering. Finally, I discuss why the search result clustering problem is difficult regardless of the algorithm and has not had success in commercial search engines.

5.1 Data

I use the Reuters dataset built into python's nltk package, which contains the raw text of 10,788 Reuters financial newswire articles along with topic labels (Bird et al. 2019). These topic labels proxy as search terms such that all documents with that label will be the results returned by that search. There are 90 topics and an average of 120 documents per topic. These topics tend to be relatively broad terms such as "coffee" or "yen", and thus are good candidates for which to find latent sub-topics.

5.2 Results Summary

First, I present cluster results from one example search term: "corn". Figure 1 is the dendrogram produced from hierarchical clustering with a cut such that there are four top level clusters with labels ('us', 'grain', 'imports'), ('inspections', 'bushels', 'soybean'), ('sold', 'unknown', 'report'), ('ecus', 'rebate', 'maize'). The largest cluster for hierarchical is ('us', 'grain', 'imports') and has three sub-clusters one level down with labels ('acres', 'acreage', 'program'), ('soviet', 'us', 'sale'), ('us', 'imports', 'grain'). Figure 2 shows a two-dimensional projection of the top-level clusters using t-SNE.

Lingo produces five clusters with labels ('us', 'export', 'soviet'), ('soviet', 'acres', 'agreement'), ('inspections', 'price', 'bushels'), ('contract', 'stocks', 'futures'), ('trades', 'ago', 'gulf'). Out of the documents returned for "corn", 18 percent are not in a cluster and 70 percent of the clustered documents are in at least two clusters. The Lingo results cannot be easily visualized because documents appear in multiple clusters. Additional illustrative examples will be described in the Results Discussion section.

Table 1 presents quantitative comparisons of each algorithm's clusterings. In addition to the table, the percent of label overlap between hierarchical and Lingo is 26.31%.

	Hierarchical	**Lingo**
Silhouette Coefficient	0.42	0.14
Avg. Distortion	0.50	0.74
Avg. # Clusters / Search Term	4.97	4.13
Avg. # Documents / Cluster	21.88	40.32
Avg. % Docs w/o a Cluster	0	23.96
Avg. % Docs in Multiple Clusters	0	30.43

Table 1: Quantitative Comparison of Algorithms

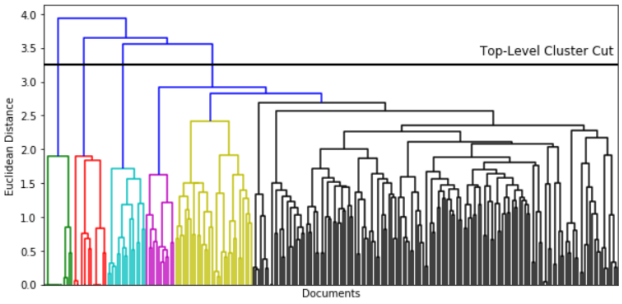

Figure 1: Dendrogram of "corn" hierarchical clusters

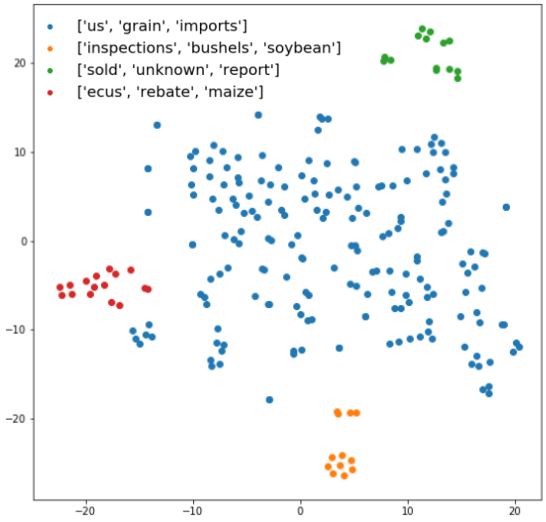

Figure 2: t-SNE visualization of "corn" hierarchical clusters

5.3 Results Discussion

The quantitative comparison shows that hierarchical clusters perform better for the classic clustering evaluation metrics: on average, hierarchical clusters have lower distortion and higher silhouette coefficients than Lingo clusters. This is not surprising because document similarity is not used to calculate Lingo clusters and because the traditional definition of cluster separation does not apply since documents can be in multiple clusters.

In some situations, it is logical that one document belongs in multiple clusters/sub-topics. However, Lingo sometimes returns clusters that are almost identical in terms of the contained documents: for the term "housing", Lingo returns two clusters with 14 documents in the first and 11 in the second. Among these, 9 documents are in both clusters. This does not help a user refine search results.

Lingo's strength threshold determines the algorithm's tendency towards having documents in zero clusters versus documents in multiple clusters. "Housing" represents the downside of the later, but excluding on average 25% of documents from any cluster can be equally problematic as the user loses potentially relevant information. These documents could be presented as a 'miscellaneous' cluster, but will likely be missed by most users.

Quantitative comparison also reveals that hierarchical clustering creates a larger number of small clusters while Lingo creates a smaller number of large clusters. This is mostly because Lingo combines clusters with overlapping labels. This is often a good thing and creates semantically distinct clusters. For example, for the search term "rapeseed", two of the hierarchical clusters are labeled ('crushers', 'canadian', 'japanese') and ('japan', 'canadian', 'undisclosed'). Lingo has just one equivalent cluster labeled ('japan', 'undisclosed', 'price').

However, Lingo can also overly combine clusters in favor of ensuring that their labels are distinct. For example, for "housing", both algorithms produce one cluster about 'completions' and another about 'starts' and 'permits'. Hierarchical however produces a third cluster labeled ('sales', 'homes', 'dollars'). Sales is distinct from completions and starts, but it is folded into those clusters in Lingo. This is especially problematic for Lingo because there is no ability to explore more granular sub-clusters. Meanwhile, in the "corn" example, even though hierarchical clustering produces a very large cluster about US grain, the user can explore the sub-clusters for more detail.

Both algorithms produce just one cluster for some search terms. This is often due to a manually imposed rule that if the search returns fewer than 7 documents, do not form clusters. However, Lingo's cluster combination step also sometimes returns only one cluster after combination. Interestingly, hierarchical creates about 6 clusters on average for these searches, which is higher than its overall average number of clusters. It appears that the documents returned by these searches are very closely related and thus, to form valid clusters, the hierarchical clusters are small and represent very specific sub-topics. In this case, as opposed to the above housing example, Lingo's combination seems reasonable.

The degree of label overlap between the two methods is only about one fourth, so the labels are often very different. Qualitatively, Lingo's labels tend to be more coherent and representative. In the 'corn' example, one hierarchical cluster is labeled ('sold', 'unknown', 'report'). This is not indicative of any sub-topic. Meanwhile, the Lingo labels for this example represent five easily definable sub-topics.

Based on these findings, I recommend hierarchical clustering for search result clustering. The consequences of a bad clustering from hierarchical are less dramatic than Lingo because hierarchical affords the user more agency to explore sub-clusters to understand an overly generic or vaguely labeled cluster. Meanwhile, Lingo too often leaves out documents and combines legitimate sub-topics in favor of distinct labels. Without any functionality to explore sub-clusters, these overly general, large clusters significantly detract from the advantages of search result clustering. Labeling is important, but in this implementation of the Lingo algorithm, the negative impact on cluster quality is too large.

5.4 Takeaways: Search Result Clustering

This is a difficult problem, regardless of the chosen algorithm, as evidenced by the lack of commercial clustering search engines. Two overarching reasons for this difficulty were made clear by my experiments.

First, distinct clusters are difficult to create because the documents are already related since they are returned by the same search term. This is especially exacerbated by my chosen corpus because all documents for all search terms are about financial topics, but this will still be an issue for more general corpuses, especially for more specific search terms.

Second, the need for general heuristics to choose hyperparameters that work across different search terms creates inconsistency in results because hyperparameters cannot be tuned for each individual problem. My corpus mitigates this effect because search terms are all about financial topics, but it is still a problem. For example, in the "corn" hierarchical dendrogram, it appears that there are six natural clusters, but my heuristics cut the dendrogram at four top-level clusters.

6 Conclusions & Future Work

In this report I implemented two opposing search result clustering algorithms and detailed my hyperparameter and heuristic choices such that they are applicable to all search terms. I compared the algorithms on the same set of queries and detailed both how and why the results are different. Ultimately, I recommended hierarchical clustering over the Lingo algorithm. This report also clarified several general weaknesses with search result clustering.

It is important to understand the limitations of this work. My hyperparameter choices are largely dependent on the specific Reuters financial newswire corpus. A more diverse corpus and set

of search terms may require different choices. Additionally, a major drawback of hierarchical clustering not mentioned above is its large computational complexity, especially when iteratively finding and labeling sub clusters. For the Reuters corpus, the number of documents returned by a "search" is never more than a couple hundred, but for a Google-sized corpus, this could become a significant issue.

Despite these limitations, this report offers important contributions to the field. It illustrates the importance of understanding *why* various algorithms produce different results: choosing between hierarchical and Lingo is not as simple as comparing their cohesion and separation statistics. It requires understanding that Lingo often combines sub-topics too aggressively and seeing the advantages of hierarchical's ability to explore sub-clusters. This report also clarifies specific cross-cutting challenges that hinder true success in this field and require further research.

There are many additional opportunities for continued research beyond the challenges I have already described. Labeling remains a challenge. The original Lingo paper uses Suffix-Tree Stemming to surface human-readable phrases as labels in addition to individual words. Another alternative is to use word2vec and doc2vec algorithms to semantically embed the documents and potential labels to both better define similarity and to eliminate semantic overlap in labels. This could be used instead of stemming to preprocess documents before forming the TF-IDF matrix. It could also be used to tune Lingo's cluster combination step so that we can impose a threshold for the degree of label semantic similarity rather than simply checking for presence of the same words. Finally, the computational complexity of hierarchical clustering must be mitigated if it is to be used on a Google-sized scale where thousands of documents may be returned from a search.

In sum, search result clustering is a promising field that could revolutionize the way we search if some practical limitations are overcome.

REFERENCES

[1] Anton Leouski and Bruce Croft. 1996. An Evaluation of Techniques for Clustering Search Results. *Computer Science Department Faculty Presentation Series.* https://scholarworks.umass.edu/cs_faculty_pubs/36/

[2] Stevent Bird, Ewan Klein, and Edward Loper. 2019. Accessing Text Corpora and Lexical Resources. *Natural Language Processing with Python.* https://www.nltk.org/book/ch02.html

[3] Brain Dean. 2019. "We Analyzed 5 Million Google Search Results. https://backlinko.com/GOOGLE-CTR-STATS

[4] Claudio Carpineto et al. 2009. A Survey of Web Clustering Engines. *ACM Computing Surveys* 41, 3 No. 17 (July 2009). https://doi.org/10.1145/1541880.1541884

[5] H. Zeng et al. 2004. Learning to Cluster Web Search Results. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA.* https://doi.org/10.1145/1008992.1009030

[6] K. Kummamuru et al. 2004. A Hierarchical Monothetic Document Clustering Algorithm for Summarization and Browsing Search Results. In *Proceedings of the 13th International Conference on World Wide Web, New York, NY, USA.* https://doi.org/10.1145/988672.988762

[7] R. Mahalakshmi and V. Laskshmi Praba. 2013. A Relative Study on Search Result Clustering Algorithms – K-Means, Suffix Tree and LINGO. *International Journal of Engineering and Advanced Technology* 2, 6 (August 2013). https://pdfs.semanticscholar.org/7010/32c71a2a4a2e54aeeb6cba787801d0c042ca.pdf

[8] Stanislaw Osinski, Jerzy Stefanowski, and Dawid Weiss. 2004. Lingo: Search Result Clustering Algorithm Based on Singular Value Decomposition. In *Intelligence Information Processing and Web Mining, Proceedings of the International IIS: IIPWM'03 Conference, Zakopane, Poland.*