From Key Points to Key Point Hierarchy: Structured and Expressive Opinion Summarization

Arie Cattan[1]* **Lilach Eden**[2]* **Yoav Kantor**[2] **Roy Bar-Haim**[2]
[1]Computer Science Department, Bar Ilan University
[2]IBM Research
arie.cattan@gmail.com {lilache, yoavka, roybar}@il.ibm.com

Abstract

Key Point Analysis (KPA) has been recently proposed for deriving fine-grained insights from collections of textual comments. KPA extracts the main points in the data as a list of concise sentences or phrases, termed *key points*, and quantifies their prevalence. While key points are more expressive than word clouds and key phrases, making sense of a long, flat list of key points, which often express related ideas in varying levels of granularity, may still be challenging. To address this limitation of KPA, we introduce the task of organizing a given set of key points into a hierarchy, according to their specificity. Such hierarchies may be viewed as a novel type of *Textual Entailment Graph*. We develop THINKP, a high quality benchmark dataset of key point hierarchies for business and product reviews, obtained by consolidating multiple annotations. We compare different methods for predicting pairwise relations between key points, and for inferring a hierarchy from these pairwise predictions. In particular, for the task of computing pairwise key point relations, we achieve significant gains over existing strong baselines by applying directional distributional similarity methods to a novel distributional representation of key points, and further boost performance via weak supervision.

https://github.com/IBM/kpa-hierarchy

1 Introduction

Many organizations face the challenge of extracting insights from large collections of textual comments, such as user reviews, survey responses, and feedback from customers or employees. Current text analytics tools summarize such datasets via word clouds (Heimerl et al., 2014) or key phrases (Hasan and Ng, 2014; Merrouni et al., 2019), which are often too crude to capture fine-grained insights. Multi-document summarization methods, on the other hand (Chu and Liu, 2019; Bražinskas et al., 2020a,b; Angelidis et al., 2021; Louis and Maynez, 2022), do not quantify the prevalence of each point in the summary, and are not well-suited for representing conflicting views (Bar-Haim et al., 2021).

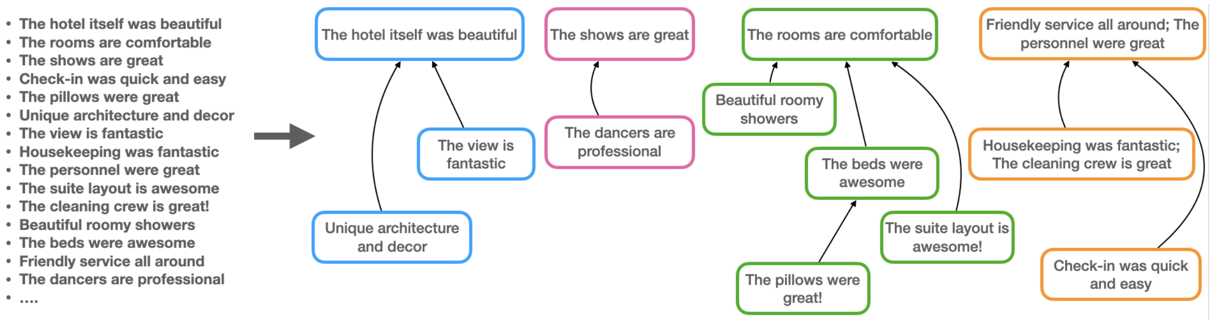
Key Point Analysis (KPA) is a recent opinion summarization framework that aims to address the above limitations (Bar-Haim et al., 2020b). KPA extracts concise sentences and phrases termed *Key Points (KPs)*, which represent the most salient points in the data, and quantifies the prevalence of each KP as the number of its matching input sentences. One remaining shortcoming of KPA, however, is that it generates a flat list, which does not capture the relations between the key points. For example, consider the sample set of key points in Figure 1 (left), which was automatically extracted from reviews of one of the hotels in the Yelp Open Dataset[1]. The results do not provide a high level view of the main themes expressed in the reviews. It is hard to tell which key points convey similar ideas, and which key points support and elaborate on a more general key point. As the number of key points in the summary increases, such output becomes even harder to consume.

In this work we introduce *Key Point Hierarchies (KPH)* as a novel structured representation of opinion summaries. Organizing the key points in a hierarchy, as shown in Figure 1 (right), allows the user to quickly grasp the high-level themes in the summary (*the hotel is beautiful*, *the shows are great*, *comfortable rooms*, *great service*), and drill down on each theme to get more fine-grained insights, e.g., from *"The personnel were great"* to *"check-in was quick and easy"*. Furthermore, key points that (nearly) convey the same meaning (e.g., *"Housekeeping was fantastic"*, and *"The cleaning crew is great"*) are clustered together and represented as a

[*] Equal contribution. Work done while the first author was an intern at IBM Research.

[1] https://www.yelp.com/dataset

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 912–928
July 9-14, 2023 ©2023 Association for Computational Linguistics



• The hotel itself was beautiful
• The rooms are comfortable
• The shows are great
• Check-in was quick and easy
• The pillows were great
• Unique architecture and decor
• The view is fantastic
• Housekeeping was fantastic
• The personnel were great
• The suite layout is awesome
• The cleaning crew is great!
• Beautiful roomy showers
• The beds were awesome
• Friendly service all around
• The dancers are professional
• ....
The hotel itself was beautiful
The view is fantastic
Unique architecture and decor
The shows are great
The dancers are professional
The rooms are comfortable
Beautiful roomy showers
The beds were awesome
The suite layout is awesome!
The pillows were great!
Friendly service all around; The personnel were great
Housekeeping was fantastic; The cleaning crew is great
Check-in was quick and easy

Figure 1: From a flat list of key points to a key point hierarchy (KPH). Nodes group together key points that roughly express the same idea and directed edges connect specific key points to more general ones. The number of matches for each key point is omitted.

single node in the hierarchy. This structured output makes KPA results more consumable, informative, and easier to navigate. KPH can be viewed as a new type of textual entailment graph (§2).

We develop THINKP (***T**ree **HI**erarchy of **N**aturally-occuring **K**ey **P**oints*), the first benchmark dataset for Key Point Hierarchies, created from KPA summaries of user reviews in multiple domains (§4). Due to the complexity of KPH annotation, THINKP was created by consolidating multiple annotations, to ensure its high quality.

We explore different methods for automatic KPH construction from a given set of key points (§5). Following previous work on entailment graphs (§2), this is formulated as a two-step approach. We first compute local scores predicting the directional relation between each pair of key points. We then construct a hierarchy guided by these local pairwise predictions.

We present novel methods and algorithmic improvements for each of the above subtasks. In particular, for the task of predicting pairwise key point relations, we achieve significant gains over existing strong baselines by applying directional distributional similarity methods to a novel distributional representation of key points, and further boost performance via weak supervision. We release the THINKP dataset to encourage further research on this challenging task.

Overall, our work contributes to several lines of research, including key point analysis, opinion summarization, entailment graphs, and distributional methods for natural language inference. Furthermore, as we demonstrate in §4.3, our novel THINKP dataset captures diverse types of inferences between pairs of naturally-occurring texts, making it an interesting resource for NLI research in general.

2 Background

Key Point Analysis. Bar-Haim et al. (2020a,b) proposed *Key Point Analysis (KPA)* as a summarization framework that provides both textual and quantitative summary of the main points in a collection of comments. KPA extracts a set of concise, high-quality sentences or phrases, termed *Key Points*, and maps each of the input sentences to its corresponding key points. The prevalence of each key point is quantified as the number of its matching sentences. KPA summaries are more expressive than the commonly-used word clouds and key phrases, while adding an important quantitative dimension that is missing from plain text summaries.

The KPA algorithm aims to extract a set of key points that provide high coverage of the data, while removing redundancies. It employs two supervised models: one for assessing the quality of key point candidates, and another one for computing a match score between a sentence and a candidate key point. Bar-Haim et al. (2021) adapted KPA to business reviews, by introducing several extensions to the original algorithm. In particular, they integrated sentiment analysis into KPA, creating separate summaries for positive and negative sentences. They also developed a specialized key point quality model for the business reviews domain.

Entailment Graphs. Most of the prior work on entailment graphs has focused on learning entailment relations between predicates, while satisfying some global constraints such as transitivity (Berant et al., 2010), soft transitivity (Chen et al., 2022), and other types of soft constraints (Hosseini et al., 2018). Levy et al. (2014) extended the notion of entailment graphs to instantiated predicates.

Most similar to our Key Point Hierarchies are entailment graphs over text fragments, introduced by Kotlerman et al. (2015). Their motivating scenario was summarizing customer feedback, for which they developed a benchmark dataset. However, the text fragments in this dataset were extracted manually. The approach proposed in the current work, which first finds the most salient points in the data using KPA, and then constructs a hierarchy from the extracted key points, allows fully-automatic generation of structured summaries for large collections of opinions, views or arguments. Constructing hierarchies over automatically-extracted key points, which are often noisy and imperfect, represents a more realistic scenario, and makes both manual annotation of KPHs and their automatic construction more challenging.

3 Key Point Hierarchies

Figure 1 illustrates the transformation of a flat key point list into a Key Point Hierarchy (KPH). Formally, given a list of key points $\mathcal{K} = \{k_1, k_2, ..., k_n\}$, we define a KPH $H = (\mathcal{V}, \mathcal{E})$ as a directed forest, that is, H is a Directed Acyclic Graph (DAG) where each node has no more than one parent. The vertices $\mathcal{V}$ are clusters of key points $\{C_1, ..., C_m\}$ that convey similar ideas, and the directed edges $\epsilon_{ij} \in \mathcal{E}$ represent hierarchical relations between clusters C_i and C_j. Similar to Kotlerman et al. (2015), a directed edge $C_i \rightarrow C_j$ indicates that the key points in C_i provide elaboration and support for the key points in C_j. By transitivity, this relation extends to any two clusters C_i and C_k such that there is a directed path in H from C_i to C_k, which we denote as $C_i \rightsquigarrow C_k$. Accordingly, we define $\mathcal{R}(H)$ as the set of directional relations between pairs of *key points* (x, y) that can be derived from H as:

$$\mathcal{R}(H) = \{(x, y)) \mid C_x = C_y \vee C_x \rightsquigarrow C_y\} \quad (1)$$

where $C_x, C_y \in \mathcal{V}$ are the clusters of x and y respectively. Considering the example in Figure 1, $\mathcal{R}(H)$ includes the relations *"Housekeeping was fantastic"* → *"The personnel were great"*, *"Housekeeping was fantastic"* → *"Friendly service all around"*, *"Housekeeping was fantastic"* → *"The cleaning crew is great"*, and so on.

We chose a hierarchical representation over a more general graph structure since it results in a simpler output that is easier to consume. In addition, this greatly simplified the annotation process. We found that hierarchical representation works well in practice, as the vast majority of the nodes in our dataset did not have more than one potential parent. This is in line with previous work, which suggested that entailment graphs tend to have a tree-like structure (Berant et al., 2012).

4 THINKP: A Dataset for Key Point Hierarchies

In this section we present THINKP, a benchmark dataset of key point hierarchies. To build THINKP, we first apply Key Point Analysis to reviews of businesses and products from multiple domains (§4.1). A KPH is then constructed manually from the set of key points extracted for each business or product (§4.2). We provide statistics on the resulting dataset, as well as qualitative analysis of the types of inferences it includes (§4.3).

4.1 Key Point Set Generation

The first step in creating the dataset was to run KPA on the reviews of selected businesses and products. Our implementation follows (Bar-Haim et al., 2021), who suggested several extensions of KPA for analyzing business reviews.[2] For each business, two separate summaries of positive and negative key points are created.

To obtain a diverse dataset, we considered three different domains, from two data sources:

Yelp. This dataset includes 7M written business reviews, where each business may be classified into multiple categories, in varying levels of granularity. We apply KPA to a sample of businesses that include at least one of the following categories: RESTAURANTS, HOTELS, and ART & ENTERTAINMENT, and had at least 1,000 reviews. For the KPH annotation, we selected four restaurants (which we refer to as the RESTAURANTS domain), and four businesses categorized as ART & ENTERTAINMENT, out of which three were hotels (hereafter, the *Hotels & Entertainment* domain, or HOTELS for brevity). Each domain includes two positive and two negative KPA summaries.

Amazon[3]. This dataset includes over 130M customer reviews for a huge collection of products in `Amazon.com` across a wide variety of domains.

[2] Specifically, our implementation follows their *RKPA-FT* configuration, except that we extract the key points for each business independently, and allow each sentence to match multiple key points.

[3] `https://s3.amazonaws.com/amazon-reviews-pds/tsv/index.txt`

Here, we focused on laptops and tablets from the PC domain, for which we could expect a rich and diverse set of key points discussing various aspects such as size, ease of use, design etc. Eventually, we annotated a KPH for three positive and one negative KPA summaries.

4.2 KPH Annotation

Annotating complex structures such as KPHs is a challenging task, since it involves global, interdependent decisions. Furthermore, the annotator needs to consider different types of hierarchical relations that may hold between the key points, as we further discuss in Section 4.3. Finally, user reviews make extensive use of informal and figurative language. For example, *"The food is outrageous!"* should be interpreted as great food; *"Elevators should go up and down, not diagonal"* means that the elevators were scary and *"Internet was a joke to get to work"* indicates a poor WiFi signal.

To overcome these challenges and obtain a high-quality dataset, three annotators individually constructed a KPH for each KPA summary (§4.2.1); The annotators then met to resolve their disagreements and reach a consolidated KPH (§4.2.2).

4.2.1 Creating an Initial KPH

To construct an initial KPH, annotators were shown the key points one by one in a descending order according to the number of their matched sentences. For each key point, they first decided whether it conveys the same idea as any previously seen key point, in which case it was added to an existing cluster. If not, a new node was added to the KPH, and the annotator dragged it to its right position in the hierarchy. Since key points with many matches tend to be more general, the key point ordering facilitated top-down construction of the KPH. At any point in the annotation process, annotators had a complete view of the KPH constructed so far, and could adjust it by modifying previous decisions, including both clustering and hierarchical relations. Each KPH was annotated separately by three of the authors and took about one hour to complete per annotator. Our annotation guidelines are detailed in Appendix A.1.

Since the key points were extracted automatically, some of them did not satisfy the desired properties of a key point - a concise and self contained sentence or phrase that discusses a single point with a certain polarity (Bar-Haim et al., 2021). To avoid noise in THINKP, annotators could mark such bad key points as candidates for removal from the final KPH.

As our annotation tool, we used CoRefi (Bornstein et al., 2020), an interface for cross-document coreference annotation with Cattan et al. (2021)'s extension for annotating a forest of clusters, which we adapted to handle key points (see Appendix A.2).

4.2.2 KPH Consolidation

To obtain the final KPHs, the three annotators met to discuss and resolve the differences in their individual KPHs annotations. This is a complex process because both clusters and the relations between them can differ. We therefore separated the consolidation process into two subsequent stages: clustering and hierarchy.

In the first phase, following the reviewer mode in CoRefi (Bornstein et al., 2020), annotators were shown one key point at a time with their original clustering decisions. In case of disagreement, the annotators discussed and reached a joint decision, which automatically modified their original KPH accordingly. At the end of this stage, the initial KPH of each of the annotators was modified to include the exact same nodes. In the second phase, since each key point has a single parent, we could easily identify the remaining disagreements by comparing the parent of each node across the different annotators. To support this consolidation phase, we enhanced CoRefi with the ability to identify and highlight both clustering and hierarchy disagreements between any number of annotators (see Appendix A.3 for more details).

Consolidating multiple annotations was also efficient due to the hierarchical structure of the KPH and took about an hour per KPH.

4.2.3 Dataset Quality Assessment

To verify the quality of the resulting dataset, we asked two additional annotators to annotate and consolidate a portion of THINKP (3 RESTAURANTS, 2 HOTEL and 2 PC).[4] We then evaluated their individual and consolidated KPHs against our consolidated annotation, as follows. In each domain, we compared the two sets of annotated KPHs by taking the union of the KP relations induced by the KPHs in each set (Eq. 1), and computing the F1 score over the two resulting sets of relations. The

[4] See Appendix A.4 for more details about annotators training.

	REST	HOTEL	PC	Total
#KPHs	4	4	4	12
#Key points	181	208	128	517
#Filtered KPs	21	17	48	86
# $\mathcal{R}(H)$	850	302	266	1,418

Table 1: Statistics of THINKP. $\mathcal{R}(H)$ is the set of key point relations that can be derived from a KPH H (§3).

final F1 was obtained by macro-averaging over the three domains.

The annotators' performance after consolidation reached an F1 of 0.756, indicating substantial agreement.[5] Furthermore, consolidation was shown to increase individual performance by 5-6 points.

4.3 Dataset Properties

Table 1 shows some statistics for the THINKP dataset. Overall, THINKP includes 12 KPHs, 517 key points, and 1,418 key points relations ($\mathcal{R}(H)$) out of the total 24,430 key point pairs. Due to its size, we did not split THINKP into development and test sets, but rather used the entire dataset for evaluation. As described in Section 4.2.1, during the annotation, we filter a relatively small number of key points (14%), mostly from the PC domain. This is mainly because the key point quality model that we used was not trained on this domain.

From a qualitative perspective, THINKP has several appealing properties that make it a valuable benchmark for NLI. First, recall that the KPA algorithm aims to remove similar key points to avoid redundancy in the summary (Bar-Haim et al., 2020b). Hence, remaining equivalent key points in THINKP are mostly non-trivial paraphrases that are challenging to detect (e.g., *"Took forever to get our room"* $\leftrightarrow$ *"Lines to check in are ridiculous"*). In addition, hierarchical relations between key points represent diverse types of inferences. Table 2 shows a few examples of common relations we observed by analyzing a sample from the dataset. Finally, THINKP comprises naturally-occurring texts and relations, coming from real-world data.

5 Automatic KPH Construction

We use a two-step approach to automatically build a KPH from a set of key points. In the first step, we predict directional scores between all pairs of key points (§5.1). In the second step, we construct a hierarchy based on the local scores (§5.2).

5.1 Scoring Pairwise Key Point Relations

Given a pair of key points (i, j), we aim to predict whether a directional relation $i \rightarrow j$ holds between i and j, by computing a likelihood score $s(i, j) \in [0, 1]$. We experimented with both existing baselines and new methods we developed for this task. Due to the size of THINKP, it was not used to fine-tune the scoring models (§4.3).

Baselines. Identifying directional relations between two key points is closely related to two existing tasks: Textual Entailment, also known as Natural Language Inference (NLI) (Dagan et al., 2007) and matching arguments to key points (Bar-Haim et al., 2020a). Accordingly, we implemented two baselines: (1) ***NLI***, a RoBERTa model (Liu et al., 2019) fine-tuned on the MNLI dataset (Williams et al., 2018) to predict whether i *entails* j[6] and (2) ***KPA-Match***, a RoBERTa model trained on the *ArgKP* dataset (Bar-Haim et al., 2020a) to predict whether i *matches* j, following (Bar-Haim et al., 2021)'s implementation.

Directional Distributional Similarity. Geffet and Dagan (2005) introduced the distributional inclusion hypothesis for lexical entailment (Geffet and Dagan, 2004), which suggests that the context surrounding an entailing word w_1 is naturally expected to occur also with the entailed word w_2. Specifically, for each word w, they built a sparse feature vector where the value of the i-th entry is the PMI of the i-th word in the dictionary with w. Many distributional similarity metrics have been proposed to predict directional relations such as hyponymy between a pair of words, based on their distributional feature vectors. Among these methods are WeedsPrec (Weeds and Weir, 2003), BInc (Szpektor and Dagan, 2008), ClarkeDE (Clarke, 2009) and APinc (Kotlerman et al., 2009).

In this work, we argue that this distributional inclusion hypothesis may be extended to identify directional relations between two key points. Indeed, if $i \rightarrow j$, it is likely that an input sentence that matches the key point i will also match j. For example, the sentence *"The beds were really comfortable, I literally knocked out as soon as my head touched the pillow."* matches both *"The beds*

[5] We do not report Kappa because decisions are mutually dependent.

[6] https://huggingface.co/roberta-large-mnli

Relation Type	Examples
Support / Elaboration	Housekeeping needs worked on ← The beds weren't even made right The room was poorly maintained ← The air conditioning was not functioning right. The device itself is so difficult to use ← Transferring data was a nightmare! Customer service is a joke ← No help moving rooms
Part-of	The *hardware* is fantastic ← *Sound* is surprisingly good The *theatre* is great ← The *entrance* is absolutely beautiful.
IS-A	The *toiletries* they offer are the worst ← not even good *shampoo* in room *Food* varieties was very limited ← *Desert* selection was below average as well

Table 2: Examples of relations between key points in THINKP.

were awesome" and *"The rooms are comfortable"*. Therefore, we construct a feature vector for each key point k, whose length is equal to the number of input sentences. The value at the i-th position in this vector is the likelihood that the i-th sentence matches k, as predicted by the KPA matching model (§4.1). Then, we apply the aforementioned distributional similarity metrics to predict a directional score $s(i, j)$. We only report the performance of ***APinc*** as it slightly outperformed other metrics. Additionally, we implemented a simple variant of WeedsPrec, in which the entries in the feature vectors are binary (match/no match). This metric, termed *Binary Inclusion* (***BinInc***), computes the ratio between the number of sentences matched by KPA to both i and j and the number sentences matched to i. Intuitively, when most of the sentences that were mapped to i were also mapped to j, it is a strong indication that $i \rightarrow j$.

Combining NLI with Distributional Methods. As further discussed in Section 6, we empirically found that the NLI model and the distributional methods have complementary strengths. The NLI model performs better on RESTAURANTS, whereas the distributional methods perform better on the HOTEL and PC domains. Furthermore, even within each domain, those two methods produce very different rankings, as indicated by a low Spearman correlation between their output scores (see Appendix C for more details).

To take advantage of the strengths of both approaches, we explored two alternatives for combining BinInc, the best-performing distributional method (as shown in Section 6), with NLI:

1. Averaging the output scores of NLI and BinInc (denoted ***NLI+BinInc-Avg***).

2. Fine-tuning the NLI model on weak labels created by the BinInc model (denoted ***NLI+BinInc-WL***). Specifically, we first apply the *BinInc* method to a large number of unlabeled KPA summaries and obtain local scores between all pairs of key points. We then convert these pairwise scores to the NLI format, where we consider all pairs above some threshold as entailment and the others as neutral. Finally, we fine-tune the NLI model on this automatically-generated training data and use the resulting model to predict the local scores $s(i, j)$ on THINKP. Implementation details and statistics on the silver data are detailed in Appendix B.

5.2 Hierarchy Construction

We proceed to construct a KPH by determining its semantic clusters and the hierarchical relations between them. Intuitively, we would like to generate a KPH such that the set of pairwise key point relations induced by its structure are consistent with the local directional scores: high-scoring relations should be included, and low-scoring relations should be excluded. We explored several alternatives for constructing a KPH, described below. Each of these methods employs a decision threshold τ over the local scores, which needs to be tuned over some development data.

Reduced Forest. Berant et al. (2012) described a simple transformation of a directed graph G into a forest of clusters. In our case, we start with a graph that includes the key points as nodes, and the directional edges $e(i, j)$ for pairs with local score $s(i, j) > \tau$.

The reduced forest is constructed as follows: (a) the condensation of G is computed by contracting each strongly connected component into a single vertex that represents a cluster of nodes in G. The resulting DAG is transformed into a forest by (b) taking its transitive reduction, and (c) heuristically selecting a single parent for each node with multiple parents. We select the larger cluster as a parent, and as a tie breaker, we use the mean over all the pairwise scores $s(i, j)$ such that i is in the child cluster and j is in the parent cluster.

As defined by Berant et al., G is a *Forest Reducible Graph (FRG)* if after applying step b above, none of the nodes has multiple parents.

Tree Node and Component Fix (TNCF). Given a directed graph with local edge weights that are either positive (predicting pairwise entailment between connected nodes) or negative (predicting non-entailment), the optimal entailment graph may be defined as the transitive subgraph in which the sum of the edge weights is maximized (Berant et al., 2012). Berant et al. showed that this problem is NP-Hard, even when further constraining the resulting graph to be forest-reducible.

To address the computational complexity of finding an exact solution, Berant et al. presented an efficient approximation algorithm, termed *Tree-node-fix (TNF)* that generates forest-reducible entailment graphs, and showed empirically that the quality of the resulting graphs is close to the exact solution found via Integer Linear Programming (ILP). Starting from some initial FRG, their algorithm iteratively improves the graph objective function by removing and reattaching one node at a time, while keeping the graph forest-reducible.

Berant et al. (2015) proposed an extension for this algorithm, termed *Tree-Node-and-Component-Fix (TNCF)*, where in each iteration a whole cluster may be re-attached, in addition to individual nodes. We found this extension beneficial.

Since a KPH is also a forest of clusters, the TNF and TNCF algorithms are directly applicable to our setting. Following Berant et al. (2012) we defined the edge weights as $w_{i,j} = s(i, j) - \tau$ so that local scores below the threshold τ are considered negative.

One difference between the original TNF implementation and ours is the initialization: while they used (Berant et al., 2011)'s exact solution, computed via ILP for a sparse configuration, we take a simpler approach and start with the reduced forest described above, constructed with the same threshold τ.

Greedy. As an alternative to the TNF/TNCF algorithms, we also adapted the greedy algorithm proposed by Cattan et al. (2021) for the task of hierarchical cross-document coreference resolution, which also generates a forest of clusters. First, key point clusters are obtained by agglomerative clustering with average linkage and distance threshold of $1-\tau$, where the distance metric between two key points i and j is defined as $1-min(s(i, j), s(j, i))$.

Second, we define the score of the directional edge between two clusters $(\mathcal{C}_1, \mathcal{C}_2)$ as the average of the $s(i, j)$ scores between the key points in the two clusters:

$$S(\mathcal{C}_1, \mathcal{C}_2) = \frac{1}{|\mathcal{C}_1| \cdot |\mathcal{C}_2|} \sum_{i \in \mathcal{C}_1} \sum_{j \in \mathcal{C}_2} s(i, j) \quad (2)$$

The KPH is constructed by repeatedly adding the highest-scoring edge (if the score is above the τ threshold), skipping edges that would violate the definition of the KPH as a directed forest. The process is terminated when no more edges can be added.

Note that unlike the TNF/TNCF algorithms, the Greedy algorithm does not modify existing clusters and edges in each iteration, but only adds new edges.

Greedy with Global Score (Greedy GS). One limitation of the Greedy algorithm is that the edge scoring function is *local* and hence ignores indirect relations between clusters that would result from adding the edge. For example, consider a KPH with three clusters $\{A, B, C\}$ such that $B \rightarrow A$. The criterion to add the edge $C \rightarrow B$ will consider only $S(C, B)$ but not $S(C, A)$, which corresponds to the indirect relation $C \rightsquigarrow A$. To address this issue, we modified the algorithm to consider the relations between each cluster and all its ancestors in the resulting KPH, as follows:

$$E_{k+1} = E_k \cup \operatorname*{argmax}_{\epsilon \in E^* \setminus E_k} O(\mathcal{V}, E_k \cup \epsilon) \quad (3)$$

$$O(\mathcal{V}, \mathcal{E}) = \sum_{\mathcal{C}_i \in \mathcal{V}} \sum_{\mathcal{C}_j \in A_{\mathcal{V},\mathcal{E}}(\mathcal{C}_i)} S(\mathcal{C}_i, \mathcal{C}_j) \quad (4)$$

where E_k is the set of edges in the resulting KPH after k iterations, E^* is the set of all edges scoring above τ and $A_{\mathcal{V},\mathcal{E}}(C)$ denotes the set of ancestors of C in $H(\mathcal{V}, \mathcal{E})$.

6 Evaluation

Predicting Local Pairwise Relations. Figure 2 compares the performance of the different local scoring methods (§5.1). For each domain, we consider all the key point pairs in the dataset, and show

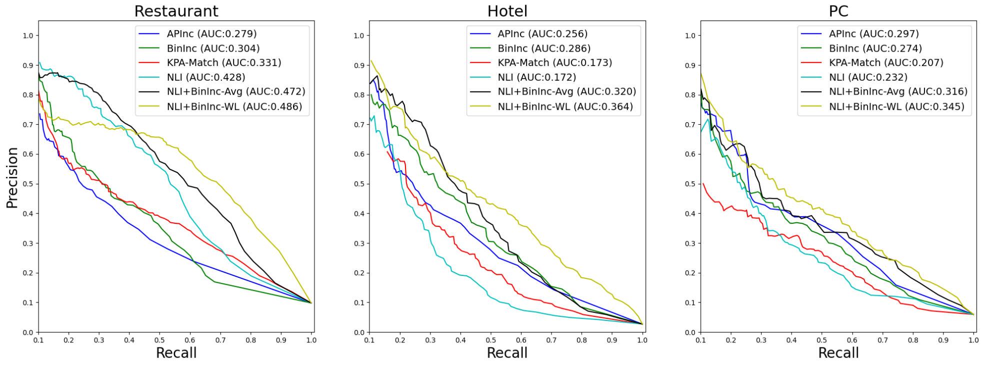


Figure 2: Precision-Recall curves of local scoring methods on RESTAURANT, HOTEL and PC.

the Precision/Recall curve and the Area Under the Curve (AUC) for each method. AUC results are also summarized in Table 3.

We first observe that applying the *KPA-match* model indirectly via the distributional methods (*APinc* and *BinInc*) outperforms its direct application in two out of the three domains, and increases the average AUC from 0.237 to 0.277/0.288, respectively. The *NLI* model has a clear advantage over the distributional methods in the RESTAURANTS domain, but is much worse for HOTEL and PC. Both *NLI+BinInc-Avg* and *NLI+BinInc-WL* models are able to combine the complementary strengths of *NLI* and *BinInc* and outperform all the stand-alone models. Model combination via weak labeling (*NLI+BinInc-WL*) achieves the best performance in all three domains by a large margin (+0.11 average AUC improvement over the best stand-alone method).

To further assess the contribution of model combination in the weak labeling setting, we also tested a configuration in which the silver data is labeled by the NLI model (denoted ***NLI-WL***). The results are shown on the last row of Table 3. While the performance is better than NLI alone (demonstrating the value of weak labeling), it is still far below *NLI+BinInc-WL*. Overalll, the results affirm the importance of both model combination and the weakly-labeled data for local scoring performance.

Hierarchy Construction. Next, we compare different methods for constructing a KPH from the set of local pairwise scores (§5.2). We use the scores from the best performing local method, *NLI+BinInc-WL*, as found in the previous experiment.

We use the F1 measure as defined in Section 4.2.3 as our evaluation metric, similar to Kotlerman et al. (2015). Since THINKP has no development set (§4.3), we employ a leave-one-out scheme to tune the threshold τ. Specifically, for each KPA summary S, we find the threshold that maximizes the F1 score of the three other KPHs in the same domain and predict a KPH for S using this threshold. We then compute the F1 score for the predicted KPHs in each domain.

The results are summarized in Table 4. *TNCF* achieves the best overall performance on THINKP with an average F1 of 0.526, substantially improving the *Reduced Forest* baseline. The *Greedy GS* algorithm is the top performer in the Restaurants domain (F1=0.641). Adding a global scoring function to the greedy algorithm improves the performance by 0.059 (from 0.45 to 0.509).

We also evaluated the quality of the predicted relations using only the local scores, with a threshold determined via leave-one-out, as before (last row in Table 4). While the resulting set of relations may not represent a valid hierarchy, it still provides an interesting reference point for comparison with the various KPH construction algorithms. We can see that both *Greedy GS* and *TNCF* improve the local results by a substantial margin (+0.028 and +0.045, resp.). These two global methods not only satisfy the constraints of generating a valid KPH, but also improve the pairwise relation prediction of the local scorer.

7 Conclusion

We introduced Key Point Hierarchies as a novel representation for structured, expressive opinion summaries. We explored several approaches for automatic hierarchy construction from a given set of key points, which were evaluated on a new benchmark dataset we developed for this task. We also

	REST	HOTEL	PC	Avg.
NLI	0.428	0.172	0.232	0.277
KPA-Match	0.331	0.173	0.207	0.237
APinc	0.279	0.256	0.297	0.277
BinInc	0.304	0.286	0.274	0.288
NLI+BinInc-Avg	0.472	0.320	0.316	0.369
NLI+BinInc-WL	**0.486**	**0.364**	**0.345**	**0.398**
NLI-WL	0.466	0.243	0.233	0.314

Table 3: Evaluation of local scoring methods (AUC for Recall ≥ 0.1)

	REST	HOTEL	PC	Avg.
Reduced Forest	0.597	0.335	0.396	0.443
TNCF	0.614	**0.460**	**0.505**	**0.526**
Greedy	0.512	0.424	0.416	0.450
Greedy GS	**0.641**	0.433	0.451	0.509
Local (no tree)	0.568	0.437	0.439	0.481

Table 4: Evaluation of hierarchy construction algorithms (F1 scores). All methods use the *NLI+BinInc-WL* local scores.

proposed a novel distributional representation for key points, which we leveraged via weak supervision to achieve substantial improvement on the subtask of predicting pairwise key point relations. While our initial results are promising, there is still much room for improvement, and we hope that releasing our dataset would encourage the community to further promote this line of research.

Limitations

Key Point Hierarchies may be valuable for summarizing opinions and views in multiple domains, including reviews, survey responses, customer feedback, political debates etc. However, in this work, we only demonstrated their value for business and product reviews, leaving other types of data to future work. Also, we only attempted to create KPHs for English reviews, for which an abundance of resources is available, including a huge number of written reviews and high-quality trained models, e.g. for NLI and key point matching. Applying these methods to low-resource languages is expected to be far more challenging. Finally, the quality of the resulting KPHs depends on the quality of the extracted key points provided as input, which may vary across different domains. To alleviate this problem in THINKP, we manually filtered out problematic key points from the dataset (§4.2).

Acknowledgments

The first author is partially supported by the PBC fellowship for outstanding PhD candidates in data science.

References

Stefanos Angelidis, Reinald Kim Amplayo, Yoshihiko Suhara, Xiaolan Wang, and Mirella Lapata. 2021. Extractive opinion summarization in quantized transformer spaces. *Transactions of the Association for Computational Linguistics*, 9:277–293.

Roy Bar-Haim, Lilach Eden, Roni Friedman, Yoav Kantor, Dan Lahav, and Noam Slonim. 2020a. From arguments to key points: Towards automatic argument summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4029–4039, Online. Association for Computational Linguistics.

Roy Bar-Haim, Lilach Eden, Yoav Kantor, Roni Friedman, and Noam Slonim. 2021. Every bite is an experience: Key Point Analysis of business reviews. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3376–3386, Online. Association for Computational Linguistics.

Roy Bar-Haim, Yoav Kantor, Lilach Eden, Roni Friedman, Dan Lahav, and Noam Slonim. 2020b. Quantitative argument summarization and beyond: Cross-domain key point analysis. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 39–49, Online. Association for Computational Linguistics.

Jonathan Berant, Noga Alon, Ido Dagan, and Jacob Goldberger. 2015. Efficient global learning of entailment graphs. *Computational Linguistics*, 41(2):249–291.

Jonathan Berant, Ido Dagan, Meni Adler, and Jacob Goldberger. 2012. Efficient tree-based approximation for entailment graph learning. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 117–125, Jeju Island, Korea. Association for Computational Linguistics.

Jonathan Berant, Ido Dagan, and Jacob Goldberger. 2010. Global learning of focused entailment graphs. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1220–1229, Uppsala, Sweden. Association for Computational Linguistics.

Jonathan Berant, Ido Dagan, and Jacob Goldberger. 2011. Global learning of typed entailment rules. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human*

Language Technologies, pages 610–619, Portland, Oregon, USA. Association for Computational Linguistics.

Ari Bornstein, Arie Cattan, and Ido Dagan. 2020. CoRefi: A crowd sourcing suite for coreference annotation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 205–215, Online. Association for Computational Linguistics.

Arthur Bražinskas, Mirella Lapata, and Ivan Titov. 2020a. Few-shot learning for opinion summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4119–4135, Online. Association for Computational Linguistics.

Arthur Bražinskas, Mirella Lapata, and Ivan Titov. 2020b. Unsupervised opinion summarization as copycat-review generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5151–5169, Online. Association for Computational Linguistics.

Arie Cattan, Sophie Johnson, Daniel S Weld, Ido Dagan, Iz Beltagy, Doug Downey, and Tom Hope. 2021. Scico: Hierarchical cross-document coreference for scientific concepts. In *3rd Conference on Automated Knowledge Base Construction*.

Zhibin Chen, Yansong Feng, and Dongyan Zhao. 2022. Entailment graph learning with textual entailment and soft transitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5899–5910, Dublin, Ireland. Association for Computational Linguistics.

Eric Chu and Peter Liu. 2019. MeanSum: A neural model for unsupervised multi-document abstractive summarization. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1223–1232. PMLR.

Daoud Clarke. 2009. Context-theoretic semantics for natural language: an overview. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, pages 112–119, Athens, Greece. Association for Computational Linguistics.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2007. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*.

William Falcon et al. 2019. Pytorch lightning. *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, 3.

Maayan Geffet and Ido Dagan. 2004. Feature vector quality and distributional similarity. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 247–253, Geneva, Switzerland. COLING.

Maayan Geffet and Ido Dagan. 2005. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 107–114, Ann Arbor, Michigan. Association for Computational Linguistics.

Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1262–1273, Baltimore, Maryland. Association for Computational Linguistics.

Florian Heimerl, Steffen Lohmann, Simon Lange, and Thomas Ertl. 2014. Word cloud explorer: Text analytics based on word clouds. In *2014 47th Hawaii International Conference on System Sciences*, pages 1833–1842.

Mohammad Javad Hosseini, Nathanael Chambers, Siva Reddy, Xavier R. Holt, Shay B. Cohen, Mark Johnson, and Mark Steedman. 2018. Learning typed entailment graphs with global soft constraints. *Transactions of the Association for Computational Linguistics*, 6:703–717.

Lili Kotlerman, Ido Dagan, Bernardo Magnini, and Luisa Bentivogli. 2015. Textual entailment graphs. *Natural Language Engineering*, 21:699 – 724.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. 2009. Directional distributional similarity for lexical expansion. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 69–72, Suntec, Singapore. Association for Computational Linguistics.

Omer Levy, Ido Dagan, and Jacob Goldberger. 2014. Focused entailment graphs for open IE propositions. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 87–97, Ann Arbor, Michigan. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Annie Louis and Joshua Maynez. 2022. Opinesum: Entailment-based self-training for abstractive opinion summarization. *ArXiv*, abs/2212.10791.

Zakariae Alami Merrouni, Bouchra Frikh, and Brahim Ouhbi. 2019. Automatic keyphrase extraction: a survey and trends. *Journal of Intelligent Information Systems*, 54:391 – 424.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Idan Szpektor and Ido Dagan. 2008. Learning entailment rules for unary templates. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 849–856, Manchester, UK. Coling 2008 Organizing Committee.

Julie Weeds and David Weir. 2003. A general framework for distributional similarity. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 81–88.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

A Data Collection

A.1 Annotation Guidelines

We began the annotation process of THINKP by drafting guidelines in which we describe the KPH structure (§3) and define the annotation task as follows. *"Given two key points A and B, (1) if A and B roughly convey the same idea or opinion, they should be clustered together in the same node (e.g. Friendly service all around vs. Staff was nice and helpful) and (2) if B elaborates on A and supports it, then B should be placed under A in the hierarchy (e.g., the rooms are comfortable ← The bed was very comfy)"*. Importantly, as key points are automatically extracted from human reviews written by different people in their own vocabulary, we advise to ignore subtle differences because they do not reflect different opinions. For example, *"Not much choice of fruits and desserts"* and *"Dessert selection was below average as well"* should be considered equivalent because *"Dessert"* usually includes fruits.

A.2 Annotation

Figure 3 shows the COREFI interface that we use to annotate THINKP. For each key point, annotators decide whether to add it to an existing cluster or to create a new node in the hierarchy.

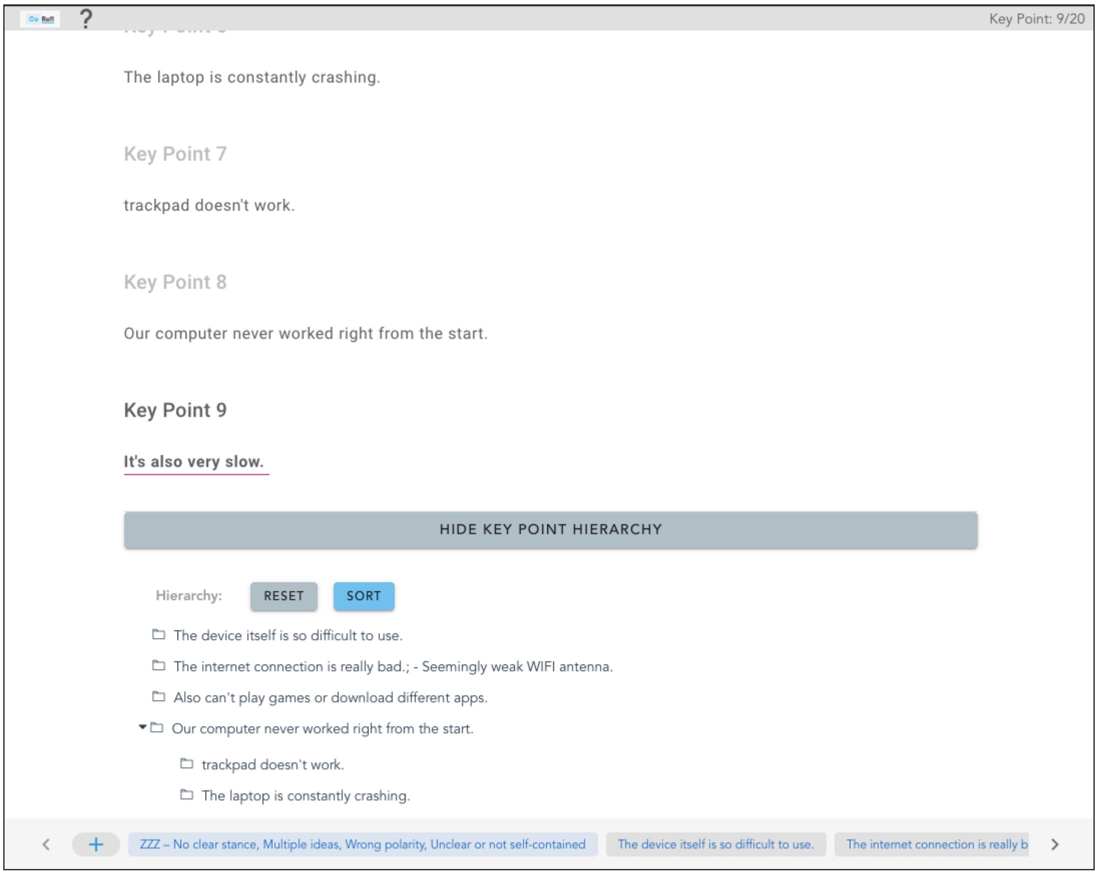
A.3 Consolidation

As described in the paper (§4.2.2), we split the consolidation stage into two subsequent steps: clustering and hierarchy, illustrated in Figures 4 and 5.

For the clustering step (Figure 4), we extend the reviewer algorithm in COREFI (Bornstein et al., 2020) with the ability to review multiple annotations for the same input. In case of disagreement, we display a red thumb-down at the bottom left of the annotation interface and the annotators discuss to reach a joint decision.

Each clustering decision automatically modifies their original KPHs. Considering the example in Figure 4 with a clustering disagreement for the key point *"The directions also leave a lot to be desired (KP1)"*: annotator A1 grouped it together with *"The device itself is so difficult to use (KP2)"* whereas annotator A2 left it as a standalone node in the KPH (indicated by the + button in purple). Now, if A1 and A2 decide to follow A1's decision, A2's original KPH will be automatically modified to include a grouped node {*The device itself is so difficult to use, The directions also leave a lot to be desired*} (instead of two separated nodes) whose children will be the concatenation of the initial children of *KP1* and *KP2*. On the other hand, if A1 and A2 decide to follow A2's decision, a new node *"The directions also leave a lot to be desired"* will be added in A1's KPH. In this case, the children of the initial grouped node will stay under *"The device itself is so difficult to use"*. This automatic process ensures that the original KPHs will include the exact same nodes.

In the second step, as shown in Figure 5, as the nodes in the two KPHs are identical, a disagreement will occur when a cluster $C \in \mathcal{V}$ has a different direct parent in each KPH. To identify the next disagreement, annotators can click on the "Go To Next Disagreement" button to highlight the key point in blue and its direct parent in violet on both KPHs. Once all hierarchical disagreements have been resolved, the structure of both KPHs will be



The laptop is constantly crashing.

Key Point 7

trackpad doesn't work.

Key Point 8

Our computer never worked right from the start.

Key Point 9

It's also very slow.

- □ The device itself is so difficult to use.
- □ The internet connection is really bad.; - Seemingly weak WIFI antenna.
- □ Also can't play games or download different apps.
- ▾□ Our computer never worked right from the start.
  - □ trackpad doesn't work.
  - □ The laptop is constantly crashing.

ZZZ – No clear stance, Multiple ideas, Wrong polarity, Unclear or not self-contained | The device itself is so difficult to use. | The internet connection is really b

Figure 3: COREFI annotation interface adapted to annotate THINKP.

identical and the annotators can submit their consolidated KPH.

A.4 Annotators Training

To assess the quality of THINKP (§4.2.3), we provided a team of in-house annotators with the same annotation guidelines (§A.1), while explicitly mentioning the purpose of the data collection. Following (Bornstein et al., 2020), we also provided them an automated walk-through tutorial to get familiar with the tool functionalities (§A.2). As part of the training, we asked the annotators to construct a KPH for 2 different businesses and gave them detailed feedback. Finally, we gave them a test and proceeded with the annotators who passed the test.

B Implementation Details

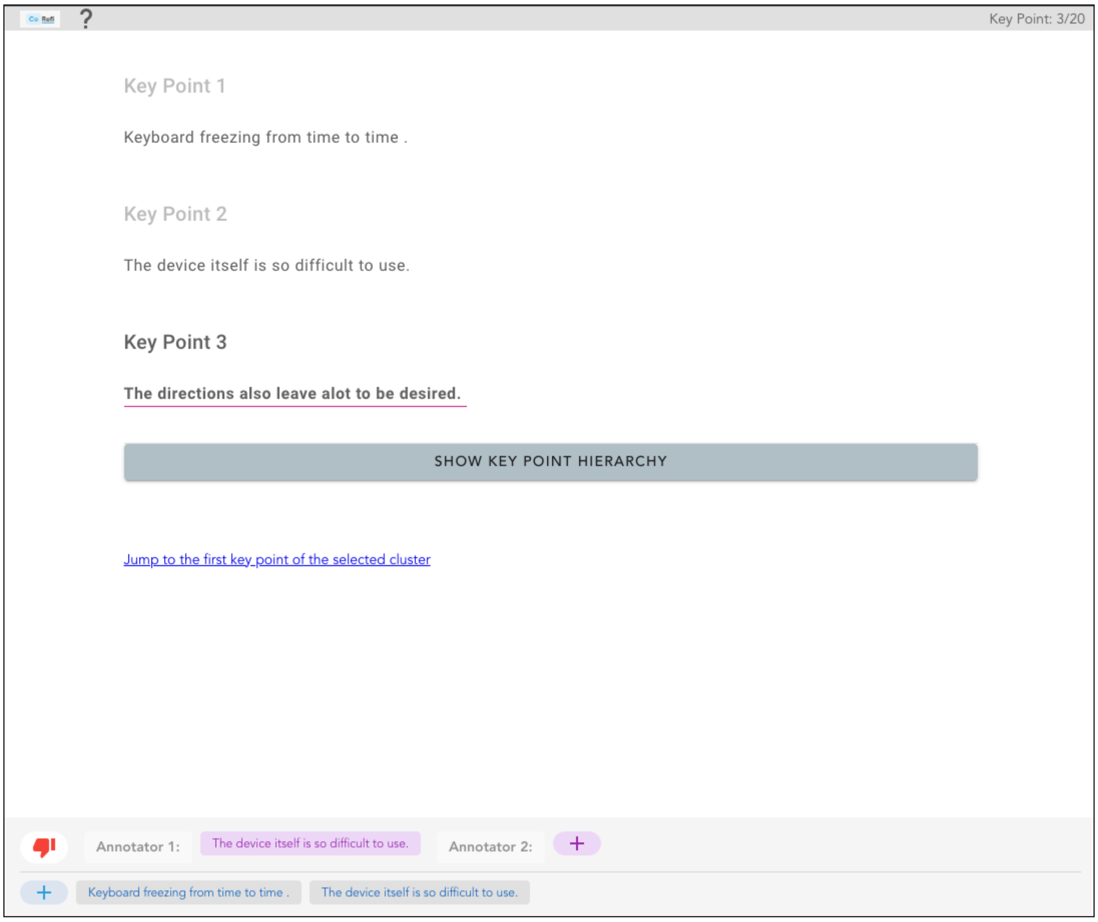
As described in Section 5.1, our best local scorer is obtained by fine-tuning an NLI model on weakly-labeled data, automatically collected as follows. We first applied KPA to reviews from 152 YELP businesses. The resulting KPA summaries included 38 key points on average. We then ran the *BinInc* method on all possible key point pairs in each KPA summary. After fixing the decision threshold to 0.5, we obtained 5,379 positive pairs and 295K negative pairs. In the final dataset that was used to train the model, we downsampled the negative examples so that the ratio between positive and negative examples was 1:5.[7]

We train our model using PyTorch (Paszke et al., 2019), PytorchLightning (Falcon et al., 2019) and the Transformers library (Wolf et al., 2020) for 5 epochs with a batch size of 64 and a learning rate of 1e-7.

C Analysis

Figure 6 shows the Spearman correlation coefficients between the output scores of the different local methods that we define in Section 5.1. NLI has a low correlation with the distributional methods (*APinc* and *BinInc*) in each of the three domains. This indicates that NLI and the distributional methods rank the key point pairs quite differently.

[7] We experimented with multiple ratios (1:1, 1:2, 1:3, 1:5, 1:10) as well as considering all the pairs and found that the 1:5 ratio achieves the best performance.



Key Point 1

Keyboard freezing from time to time .

Key Point 2

The device itself is so difficult to use.

Key Point 3

The directions also leave alot to be desired.

Annotator 1: The device itself is so difficult to use.

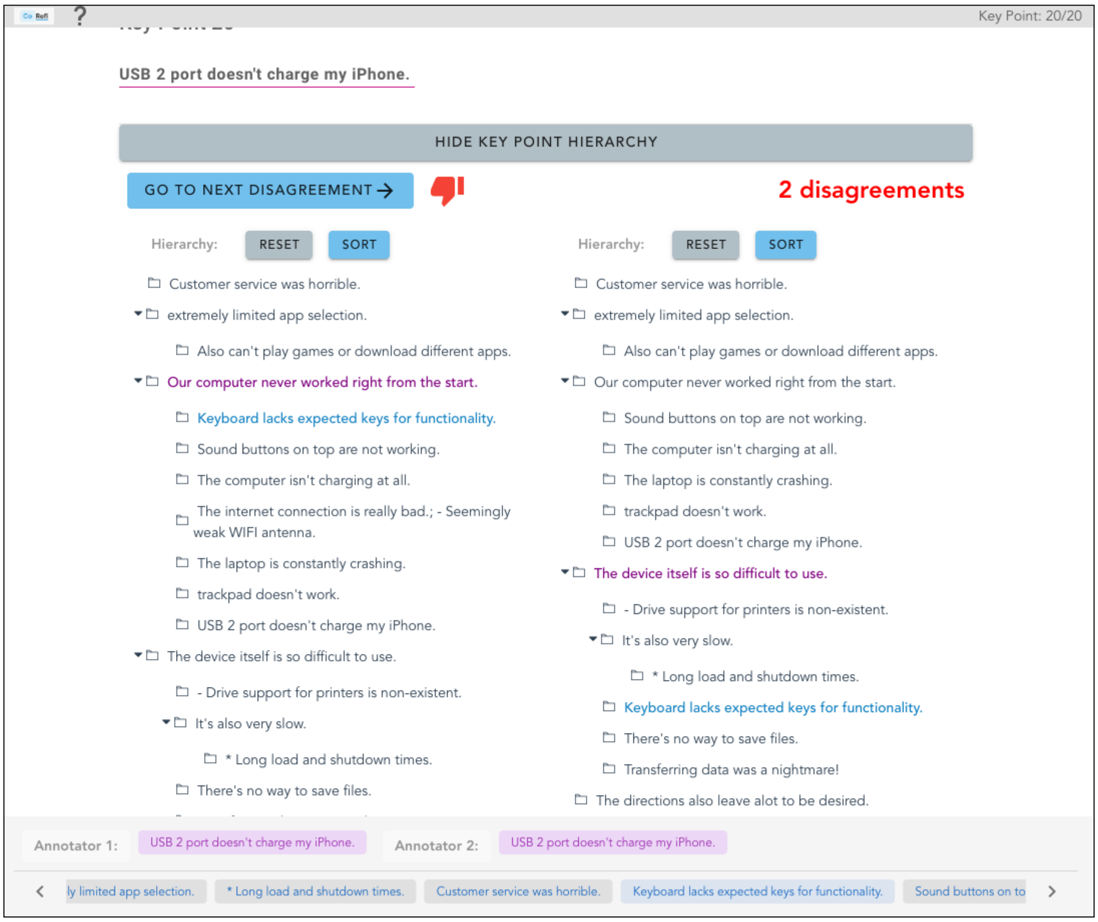
Annotator 2:

Keyboard freezing from time to time . The device itself is so difficult to use.

Figure 4: Clustering step. The thumb-down at the bottom left of the screen indicates a clustering disagreement between the annotators for Key Point 3: *“The directions also leave a lot to be desired”*. Annotator A1 assigned it to *“The device itself is so difficult to use”* while annotator A2 created a new cluster, as indicated in purple.

D Datasets

- The Yelp and Amazon datasets used in this work have been released for academic use, and accordingly, we have only used them for academic research.
- The authors have reviewed the THINKP dataset and verified that it does not contain any personal information or offensive content.



Key Point: 20/20

USB 2 port doesn't charge my iPhone.

HIDE KEY POINT HIERARCHY

GO TO NEXT DISAGREEMENT →

2 disagreements

Hierarchy: RESET SORT

- Customer service was horrible.
- extremely limited app selection.
  - Also can't play games or download different apps.
- Our computer never worked right from the start.
  - Keyboard lacks expected keys for functionality.
  - Sound buttons on top are not working.
  - The computer isn't charging at all.
  - The internet connection is really bad.; - Seemingly weak WIFI antenna.
  - The laptop is constantly crashing.
  - trackpad doesn't work.
  - USB 2 port doesn't charge my iPhone.
- The device itself is so difficult to use.
  - - Drive support for printers is non-existent.
  - It's also very slow.
    - * Long load and shutdown times.
  - There's no way to save files.

Hierarchy: RESET SORT

- Customer service was horrible.
- extremely limited app selection.
  - Also can't play games or download different apps.
- Our computer never worked right from the start.
  - Sound buttons on top are not working.
  - The computer isn't charging at all.
  - The laptop is constantly crashing.
  - trackpad doesn't work.
  - USB 2 port doesn't charge my iPhone.
- The device itself is so difficult to use.
  - - Drive support for printers is non-existent.
  - It's also very slow.
    - * Long load and shutdown times.
  - Keyboard lacks expected keys for functionality.
  - There's no way to save files.
  - Transferring data was a nightmare!
- The directions also leave alot to be desired.

Annotator 1: USB 2 port doesn't charge my iPhone. Annotator 2: USB 2 port doesn't charge my iPhone.

< ly limited app selection. | * Long load and shutdown times. | Customer service was horrible. | Keyboard lacks expected keys for functionality. | Sound buttons on to >

Figure 5: Consolidation of hierarchical relations. The cluster *“Keyboard lacks expected keys for functionality.”* is highlighted in blue in both KPHs because the two annotators placed it under different parents (colored in violet in both KPHs).

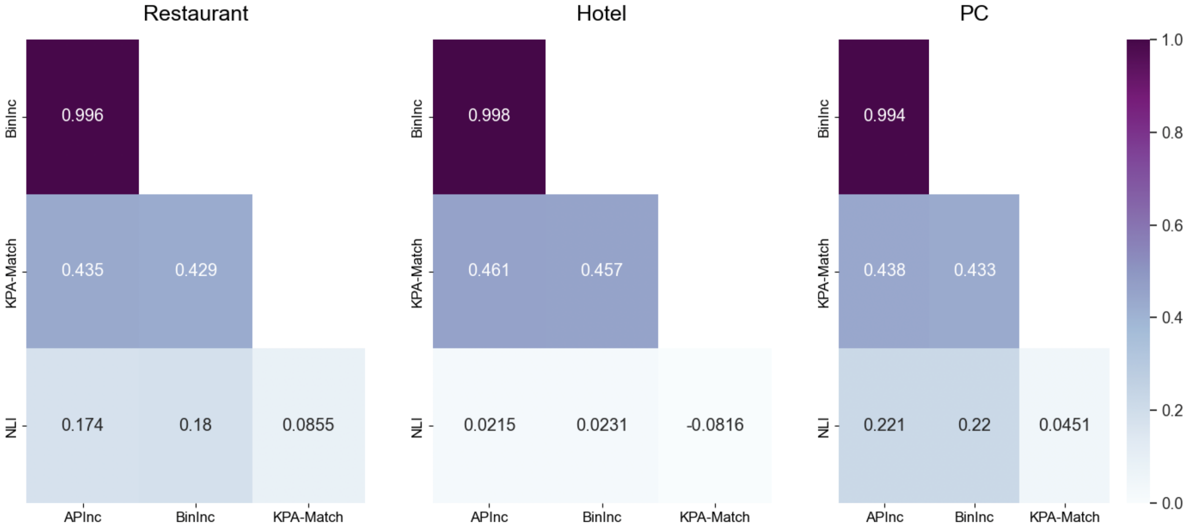


Figure 6: Spearman correlations between the scores of the local methods

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
The last section (unnumbered), immediately following the conclusion

☒ A2. Did you discuss any potential risks of your work?
We carefully reviewed the guidelines and could not think of potential risks worth mentioning in the paper.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
See abstract and the first section (Introduction).

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 4

☑ B1. Did you cite the creators of artifacts you used?
Section 4.1

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
See Appendix D. The exact terms of use and licensing information for the dataset we release will be provided upon its release.

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
See Appendix D. The exact terms of use and licensing information for the dataset we intend to release will be provided upon its release.

□ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
No response.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Section 4

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Section 4.3

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

C ☑ Did you run computational experiments?

Section 6

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 5 (we specified the model we used, RoBERTa-large)

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 6 and Appendix B

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 6

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Appendix B

D ☑ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Section 4.2 and Appendix A

☑ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Appendix A

□ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☑ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Appendix A, in particular A.4

☒ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not relevant for this annotation task

☒ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Not relevant, only two annotators