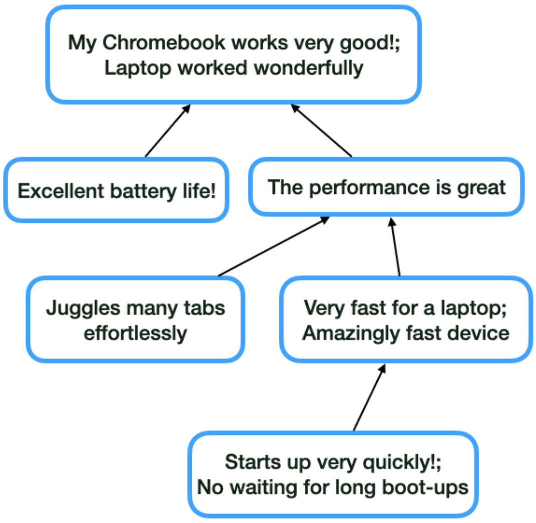
CHAMP: Efficient Annotation and Consolidation of Cluster Hierarchies

Arie Cattan[1] Tom Hope[2,3] Doug Downey[2] Roy Bar-Haim[4]
Lilach Eden[4] Yoav Kantor[4] Ido Dagan[1]
[1]Computer Science Department, Bar Ilan University
[2]Allen Institute for Artificial Intelligence
[3]School of Computer Science, The Hebrew University of Jerusalem
[4]IBM Research
arie.cattan@gmail.com

Abstract

Various NLP tasks require a complex hierarchical structure over nodes, where each node is a cluster of items. Examples include generating entailment graphs, hierarchical cross-document coreference resolution, annotating event and subevent relations, etc. To enable efficient annotation of such hierarchical structures, we release CHAMP, an open source tool allowing to incrementally construct both clusters and hierarchy simultaneously over any type of texts. This incremental approach significantly reduces annotation time compared to the common pairwise annotation approach and also guarantees maintaining transitivity at the cluster and hierarchy levels. Furthermore, CHAMP includes a consolidation mode, where an adjudicator can easily compare multiple cluster hierarchy annotations and resolve disagreements.

https://github.com/ariecattan/champ



My Chromebook works very good!; Laptop worked wonderfully
Excellent battery life!
The performance is great
Juggles many tabs effortlessly
Very fast for a laptop; Amazingly fast device
Starts up very quickly!; No waiting for long boot-ups

Figure 1: Example of hierarchy of clusters from THINKP (Cattan et al., 2023). Nodes group similar statements together and arrows represent child-parent relations, relating specific statements to more general ones.

1 Introduction

In numerous annotation tasks, the annotator needs to perform individual and independent decisions. Such tasks include Named Entity Recognition (NER), text categorization and part-of-speech tagging, among others (Stenetorp et al., 2012; Yimam et al., 2013; Samih et al., 2016; Yang et al., 2018; Tratz and Phan, 2018; Mayhew and Roth, 2018). However, certain annotation tasks are more demanding because they involve the construction of a complex structure that must satisfy global constraints. One such complex structure is clustering, where annotated clusters must respect the equivalence relation. Specifically, if items A and B belong to the same cluster, and items B and C also belong to the same cluster, then A and C must belong to the same cluster as well. Another prominent example of a global structure is hierarchy, where typically, if A is an ancestor of B and B is an ancestor of C, then A must also be an ancestor of C.

In this work, we focus on annotating a *hierarchy of clusters*, a global structure that combines the constraints of both clustering and hierarchy, thereby posing further challenges. In this hierarchy, nodes are clusters of (text) items, where each node can have at most a single parent, as illustrated in Figure 1. Annotating a hierarchy of clusters is relevant for a multitude of tasks, such as hierarchical cross-document coreference resolution (Cattan et al., 2021), structured summarization as a hierarchy of key points (Cattan et al., 2023), entailment graph construction (Berant et al., 2012) and event-subevent relations detection (O'Gorman et al., 2016; Wang et al., 2022). While there are some annotation tools for annotating either clustering or a hierarchy (§2.1), to the best of our knowledge there is no available tool allowing to annotate a hierarchy of clusters simultaneously within the same tool.

Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 403–412
December 6-10, 2023 ©2023 Association for Computational Linguistics

To address this need, we introduce CHAMP (**C**luster **H**ierarchy **A**nnotation for **M**ultiple **P**articipants), an intuitive and efficient tool for annotating a hierarchy of clusters in a globally consistent manner, supporting multiple annotators (§3). Specifically, annotators are presented with input text spans one by one and form *incrementally* and *simultaneously* the clusters and their hierarchy (§3.1).

Additionally to the annotation process, we develop an adjudication mode for easily comparing multiple annotated hierarchies of clusters (§3.2). This mode can be used either by an adjudicator, which is typically a more reliable annotator, or by the original annotators during discussions to resolve conflicts. Indeed, adjudication is crucial to ensure quality in general (Roit et al., 2020; Klein et al., 2020), and particularly important for our structure, requiring a more challenging global annotation.

We demonstrate the use of CHAMP in two notably different use-cases, both involving annotating hierarchies of clusters: hierarchical cross-document coreference resolution (Cattan et al., 2021) and key point hierarchy (Cattan et al., 2023). In both settings, CHAMP is significantly more efficient than a pairwise annotation approach, in which the relation between each pair of items is annotated independently. Moreover, our consolidation phase enhances the annotation quality, yielding an improvement of 5-6 F1 points (Cattan et al., 2023).

CHAMP was implemented on top of COREFI (Bornstein et al., 2020), which was initially designed for coreference, and allowed only standard (non-hierarchical) annotation. CHAMP includes a WebComponent, which can easily be embedded into any HTML page, including popular crowdsourcing platforms such as Amazon Mechanical Turk. We also develop an annotation portal (the link appears in our github repository), allowing users to perform online the annotation task and dataset developers to effortlessly compute inter-annotator agreement.

Overall, CHAMP is an intuitive tool for efficiently annotating and adjudicating hierarchies of clusters. We believe that CHAMP will remove barriers when annotating such challenging global tasks and will facilitate future dataset creation.

2 Background

2.1 Tools for Annotating Global Structures

Certain NLP tasks involve a structure that should be annotated in a global manner due to mutually dependent labels. In this work, we focus on two specific structures: clustering and hierarchy.

A prominent clustering task is coreference resolution, where the goal is to group mention spans into clusters. This implies that if A and B are coreferent and B and C are coreferent, then A and C should also be coreferent. However, early tools for coreference annotation relied on a series of local binary decisions over all possible mention pairs (Stenetorp et al., 2012; Widlöcher and Mathet, 2012; Landragin et al., 2012; Kopeć, 2014; Chamberlain et al., 2016). In contrast, cluster-based tools aim for global annotation by directly assigning mentions to clusters (Ogren, 2006; Girardi et al., 2014; Reiter, 2018; Oberle, 2018; Aralikatte and Søgaard, 2020; Bornstein et al., 2020; Gupta et al., 2023). Among these cluster-based tools, COREFI (Bornstein et al., 2020) stands out for its beneficial features that enable cost-effective and efficient annotation. These features include quick keyboard operations (instead of slow drag-and-drop), an onboarding mode for training annotators on the task, and a reviewing mode that facilitates systematic review and quality improvement of a given annotation (as described in §2.2).

Some other tasks such as taxonomy induction and entailment graph construction also involve structures (e.g., graphs, DAG, hierarchy) that impose global transitivity constraints. For example, if a taxonomy includes the relationships “A is a kind of B” and “B is a kind of C”, then it follows that A must also be a kind of C. Yet, for example, Berant et al. (2011) annotated an entailment graph dataset by annotating all possible edges between predicates, resulting in a complexity of $\mathcal{O}(n^2)$. Subsequent works follow the pairwise approach but apply some heuristics for reducing the number of annotations (Levy et al., 2014; Kotlerman et al., 2015). Closely related to taxonomy, the Redcoat annotation tool (Stewart et al., 2019) allows to annotate hierarchical entity typing, while allowing to modify the hierarchy during annotation.

To the best of our knowledge, there is no available tool that supports joint annotation of a hierarchy of clusters, as proposed in CHAMP.

2.2 Consolidation of Multiple Annotations

To promote quality, datasets often rely on multiple annotators per instance, especially when the annotation is obtained via crowdsourcing. Then, the annotations can be combined either *automatically*, using simple majority vote or more sophisticated aggregation techniques (Dawid and Skene, 1979; Raykar et al., 2010; Hovy et al., 2013; Passonneau and Carpenter, 2014; Paun et al., 2018), or *manually*, by asking the annotators themselves or a more reliable annotator to adjudicate and resolve annotation disagreements (Pradhan et al., 2012; Roit et al., 2020; Pyatkin et al., 2020; Klein et al., 2020). However, those aggregation methods were mostly investigated for classification tasks where each instance can be annotated independently, but not for global tasks, like those discussed above (§2.1).

To the best of our knowledge, COREFI (Bornstein et al., 2020) is the only annotation tool that supports *manual* reviewing of a global structure annotation, specifically for coreference annotation. In this interface, the reviewer is shown the annotated mentions one by one along with the original annotator's cluster assignment. The reviewer can then decide whether to retain the original annotation or to make a different clustering assignment. However, showing the original cluster assignment of each mention in turn is not straightforward, because earlier reviewer decisions may have deviated from the original clustering annotation. For instance, consider a scenario where the original annotator creates a cluster with the mentions x, y, z. Subsequently, the reviewer decides that y should not be linked to x but should instead form a new cluster. At this point, when the reviewer encounters the mention z, it becomes uncertain whether it should be considered by the original annotation as linked with x or y. To address this issue, when the reviewer is shown a mention m, the candidate clusters implied by the original annotation becomes the *set* of clusters in the current reviewer's clustering configuration that include at least one of the previously annotated antecedents of m according to the original annotation.

While the reviewing mode in COREFI is effective, an important limitation is that it enables reviewing only a single annotation, not supporting the consolidation of multiple annotations, as common in NLP annotation setups. We address this need in CHAMP by supporting consolidation of multiple annotations (§3.2).

3 CHAMP

We present CHAMP, a new tool for annotating a *hierarchy of clusters*. To annotate such a structure, the annotators are provided with a list of input spans, denoted as $S = \{s_1, ..., s_n\}$, that they need to group into disjoint clusters of semantically equivalent spans $\mathcal{C} = \{C_1, ..., C_k\}$. In addition, annotators need to form a *directed* forest $G = (\mathcal{C}, E)$, constituting a Directed Acyclic Graph (DAG) in which every node—representing the cluster C_i—has no more than one parent. Within this structure, each edge e_{ij} represents a hierarchical relation between clusters $C_i \rightarrow C_j$, signifying that C_i is a child of C_j. Considering the example in Figure 1, the cluster *{Starts up very quickly, No waiting for long boot-ups}* is more specific than the cluster *{Very fast for a laptop, Amazingly fast device}*. Importantly, input spans can be standalone spans (as in Figure 1) or appear within a surrounding context. For the remainder of this section, we will focus on demonstrating CHAMP using standalone spans, while an example featuring spans within context is provided in Appendix A.

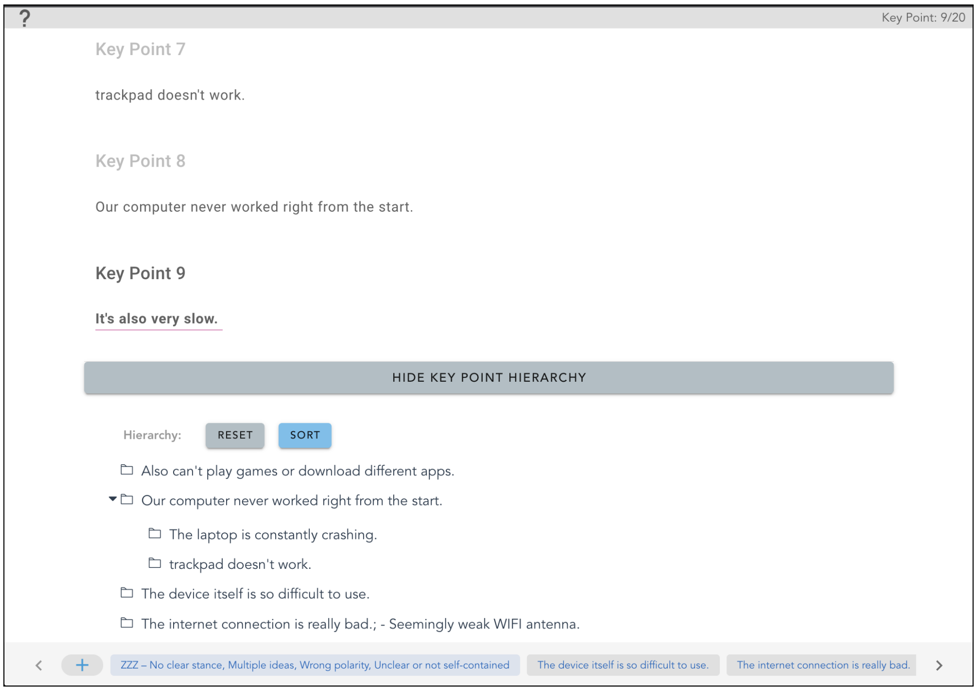
We next describe the core annotation interface (§3.1), and then present the *adjudication* mode, which allows to effectively compare multiple annotations and build a consolidated hierarchy of clusters (§3.2).

3.1 Cluster Hierarchy Annotation

Figure 2 shows the annotation interface in CHAMP.

A naive approach for supporting the annotation of a hierarchy of clusters would involve two separate steps: (1) cluster input spans and (2) construct a hierarchy over the fixed annotated clusters. Although straightforward, this method lacks the flexibility for annotators to modify the clustering annotation while simultaneously working on the hierarchy. This inflexibility is problematic since typically many annotation decisions fall at the intersection of clustering, which reflects semantic equivalences, and hierarchy, which denotes the relationships between more general and specific clusters (e.g., *Takes a long time for check in* vs. *The absolute worst check in process anywhere*). Moreover, employing two separate annotation steps would burden annotators with the additional challenge of remembering the context of each cluster during hierarchy annotation.

Therefore, we propose an *incremental* approach for annotating both the clustering and the cluster hi-



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.

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

‹
+
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 bad.
›

Figure 2: User interface for annotating both clustering and hierarchical relations between clusters. The current statement to assign is underlined in purple: "It's also very slow". The annotator can decide whether to add it to an existing cluster, in which case it will be concatenated in the display of the corresponding node in the hierarchy, separated by ";", or to open a new cluster, in which case a new node will be automatically added to the hierarchy, initiated under the root.

erarchy together as a single annotation task, which we develop upon COREFI (Bornstein et al., 2020). At initialization, the first span is automatically assigned to the first cluster C_1 and to a corresponding node in the hierarchy. Then, for each subsequent span s, the annotator first decides its cluster assignment, by choosing whether to assign s to an existing or a new cluster. In the latter case, a new node is automatically created in the hierarchy under the root and the annotator can drag it to its right position in the current hierarchy. Considering the example in Figure 2, the current span to annotate s is "It's also very slow" (underlined in purple), the current clusters C are shown in the cluster bank (in the footer of the screen), and the current hierarchy is shown in the lower portion of the window.

Importantly, when the annotator re-assigns a previously assigned span to another cluster, CHAMP will automatically update nodes and relations in the hierarchy. Keeping in sync cluster assignments and hierarchy is not trivial because different clustering modifications will have different effects on the resulting hierarchy. In particular, we consider the following cases of re-assigning the span s:

1. From a *singleton* cluster C_i to a cluster C_j: s will be added to C_j and C_i's children will move under C_j.

2. From a *non-singleton* cluster C_i to a cluster C_j: s will be added to C_j but C_i's children will stay under C_i.

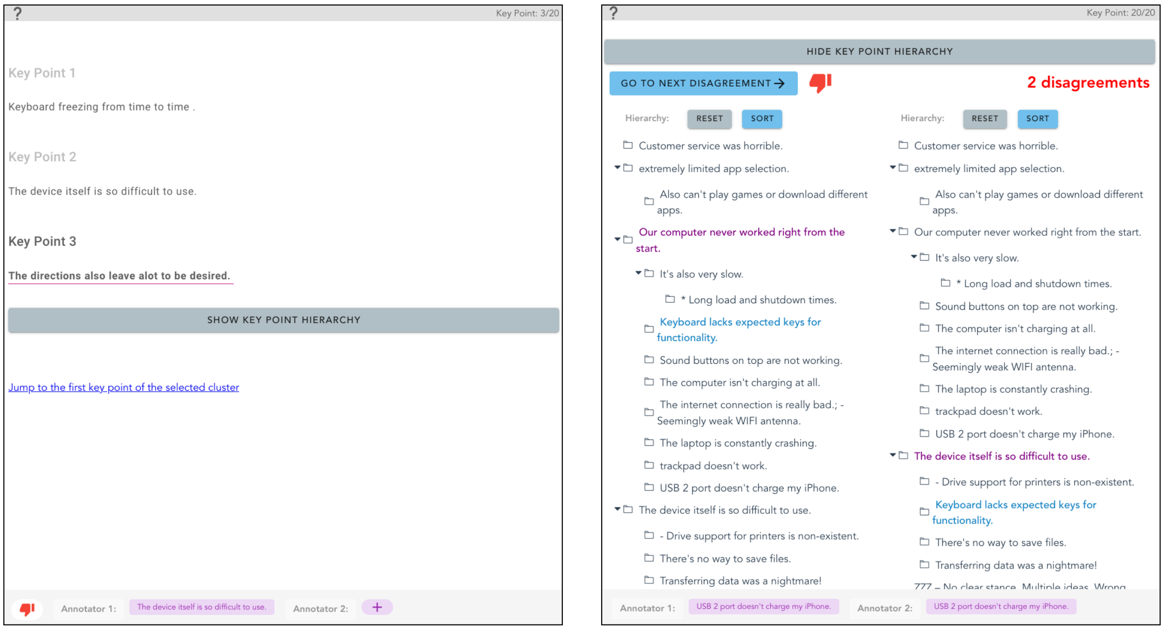
3. From a cluster C_i to a new singleton cluster: a new node C_j will be created in the hierarchy and will be initially situated as a sibling of C_i.[1] Annotators can then drag it to its desired place.

This hierarchy update procedure is a key ingredient for enabling the annotation of hierarchy of clusters as a single task.

3.2 Adjudication

In order to facilitate the manual adjudication of multiple hierarchy annotations by different annotators, we added an adjudication mode within CHAMP that supports easily identification and resolution of disagreements between any number of annotations. This mode can be used by an adjudicator, which is usually a more reliable annotator, or by the original annotators during discussions to resolve conflicts.

[1] We take this approach because, when annotators re-assign s to a standalone cluster, their intention is not to eliminate the hierarchical relationship between s and its parent cluster.



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.

Jump to the first key point of the selected cluster

Annotator 1: The device itself is so difficult to use. Annotator 2: +

- 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.
  - It's also very slow.
    - * Long load and shutdown times.
  - 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.
  - There's no way to save files.
  - Transferring data was a nightmare!

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

- 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.
  - It's also very slow.
    - * Long load and shutdown times.
  - 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.
  - Keyboard lacks expected keys for functionality.
  - There's no way to save files.
  - Transferring data was a nightmare!

??? – No clear stance. Multiple ideas. Wrong

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

(a) Clustering consolidation. The thumb-down at the bottom left of the screen indicates a clustering disagreement between the annotators for the span *"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.

(b) Hierarchy consolidation. The red thumb-down near the "Go to next disagreement" button indicates a hierarchy disagreement for the node *"Keyboard lacks expected keys for functionality"*. Annotator A1 placed it under *"Our computer never worked right from the start"*, while A2 placed it under *"The device itself is so difficult to use."*

Figure 3: Adjudication of multiple annotations of hierarchy of clusters.

Comparing multiple annotations of a hierarchy of clusters can be challenging due to variations in annotators' clustering assignments, leading to different sets of nodes in the respective hierarchies. To illustrate this issue, consider a scenario where annotator A1 annotates the relation $\{s_1, s_2, s_4\} \rightarrow \{s_3, s_5\}$, while A2 annotates $\{s_1, s_2, s_6\} \rightarrow \{s_3\}$ and $\{s_4\} \rightarrow \{s_5\}$. The two hierarchies have similarities (e.g. both cluster s_1 and s_2 together and have s_5 as a parent of s_4) but differ in other ways, making their adjudication process non-trivial.

To tackle this problem, we decoupled the adjudication process into two consecutive stages, adjudicating separately clustering and hierarchy decisions, as illustrated in Figure 3.

In the first step, the adjudicator is shown the annotated spans in a sequential manner, along with the cluster assignments of each of the original annotations. To achieve this, we leverage the reviewing procedure that COREFI applies for reviewing a single clustering annotation (§2.2), implement it separately to each original annotation. We then present to the adjudicator a set of candidate clusters per original annotation. These sets of candidates are displayed in purple at the bottom of the screen, as illustrated in Figure 3a.

It should be pointed out here that resolving a cluster assignment disagreement means that the adjudicator alters the assignment for at least one of the annotators. Therefore, we apply the hierarchy update procedure (§3.1) to the modified annotations, in order to update accordingly the involved cluster nodes and their hierarchical relations. Considering the example in Figure 3a with a clustering disagreement for the span *"The directions also leave a lot to be desired (s_1)"*. In this instance, annotator A1 has merged it with *"The device itself is so difficult to use (s_2)"*, while annotator A2 has designated it as a singleton cluster in the hierarchy, as highlighted by the purple '+' button. If the adjudicator follows A1's decision, A2's hierarchy will be restructured to combine spans $\{s_1, s_2\}$ into the same cluster. Conversely, siding with A2's decision will separate s_2 from s_1 in A1's hierarchy. This automatic process ensures that the modified hierarchies will include the exact same set of nodes (clusters) $\mathcal{C}$ at the end of the clustering consolidation step.

In the second step of hierarchy adjudication, as the sets of nodes $\mathcal{C}$ in the hierarchies of all annotators are identical, a disagreement arises when a node $C_i \in \mathcal{C}$ has a different direct parent in different hierarchies. To efficiently identify such discrepancies, the adjudicator can click on the "Go To Next Disagreement" button, which highlights the node C_i in blue along with its direct parent in violet on all input hierarchies. As shown in Figure 3b, for instance, the node *"Keyboard lacks expected keys for functionality"* was placed under *"Our computer never worked right from the start"* by A1, and under *"The device itself is so difficult to use"* by A2. The adjudicator then decides the correct hierarchical relation, manually updates the other hierarchies accordingly, and moves on to the next disagreement. Once all hierarchical disagreements have been resolved, the adjudicator can confidently submit the obtained consolidated hierarchy.

4 Applications

We used CHAMP for annotating datasets for two different tasks that require annotating of hierarchy of clusters:

1. SciCo (Cattan et al., 2021), a dataset for the task of hierarchical cross-document coreference resolution (H-CDCR). In this dataset, the inputs are paragraphs from computer science papers with highlighted mentions of scientific concepts, specifically mentions of tasks and methods. The goal is to first cluster all mentions that refer to the same concept (e.g., *categorical image generation* ↔ *class-conditional image synthesis*) and then infer the referential hierarchy between the clusters (e.g., *categorical image generation* → *image synthesis*).

2. THINKP (Cattan et al., 2023), a recent benchmark of key point hierarchies, where each key point is a concise statement relating to a particular topic (Bar-Haim et al., 2020). Key point hierarchies were proposed as a novel structured representation for large scale opinion summarization. The nodes in these graphs group statements conveying the same opinion (e.g., *the cleaning crew is great!* ↔ *housekeeping is fantastic*) while the edges indicate hierarchical specification-generalization relationships between nodes (e.g., *housekeeping is fantastic* → *the personnel is great*). The entailment graphs in THINKP are designed in a hierarchical form, where each node has at most a single parent.

Despite the different nature of these tasks and their unit of annotation (i.e., standalone state,emts vs. concept spans in context), we seamlessly leveraged CHAMP for both with minimal effort (using a simple JSON configuration schema), as both tasks involve annotating a hierarchy of clusters.

In our experiments, we observed that annotating or consolidating a hierarchy of clusters for fifty statements takes approximately one hour (Cattan et al., 2023). In contrast, collecting annotations for all possible pairs, as commonly done in prior datasets for entailment graphs (Berant et al., 2011), would have been much more expensive since it would require at least 1225 decisions on average for our data, which would obviously take much more than one hour. Furthermore, unlike the pairwise annotation approach, our incremental method for constructing a hierarchy of clusters guarantees that the resulting annotation will respect the global constraint of transitivity. Finally, our experiments also revealed that the consolidation mode significantly enhances human performance, yielding a gain of 5-6 F1 points (Cattan et al., 2023).

5 Implementation Details and Release

We implement CHAMP on top of COREFI (Bornstein et al., 2020), using the `Vue.js` framework, that we open source under the permissive MIT License. Following COREFI, we release CHAMP as a WebComponent, which can easily be embedded into any HTML page, including popular crowdsourcing platforms such as Amazon Mechanical Turk. Both the annotation and consolidation processes share the same interface and are easily configurable using a straightforward JSON schema. We also develop an annotation portal where users can upload a configuration file (either for annotation or adjudication), perform the annotation task and download it upon completion. This portal also provides the capability to upload multiple annotation files from various annotators and to compute the inter-annotator agreement. As such, CHAMP is not only easy-to-use for annotators, but it is also easy to setup and manage for dataset developers.

6 Conclusion

This paper aims to foster research on global annotation tasks by introducing CHAMP, an efficient tool designed for annotating a *hierarchy of clusters*. This annotation tool also incorporates an adjudication mode that conveniently supports identification and consolidation of annotators' disagreements. As CHAMP enables efficient and high-quality annotation, we believe that it will facilitate the creation of datasets for various tasks involving this complex structure, and will inspire tool development for other global annotation tasks.

Acknowledgements

This work was supported by the Israel Science Foundation (grant no. 2827/21). Arie Cattan is partially supported by the PBC fellowship for outstanding PhD candidates in data science.

References

Rahul Aralikatte and Anders Søgaard. 2020. Model-based annotation of coreference. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 74–79, Marseille, France. European Language Resources Association.

Roy Bar-Haim, Lilach Eden, Roni Friedman, Yoav Kantor, Dan Lahav, and Noam Slonim. 2020. 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.

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. 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.

Arie Cattan, Lilach Eden, Yoav Kantor, and Roy Bar-Haim. 2023. From key points to key point hierarchy: Structured and expressive opinion summarization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 912–928, Toronto, Canada. 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*.

Jon Chamberlain, Massimo Poesio, and Udo Kruschwitz. 2016. Phrase detectives corpus 1.0 crowdsourced anaphoric coreference. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 2039–2046, Portorož, Slovenia. European Language Resources Association (ELRA).

A. Philip Dawid and Allan Skene. 1979. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of The Royal Statistical Society Series C-applied Statistics*, 28:20–28.

Christian Girardi, Manuela Speranza, Rachele Sprugnoli, and Sara Tonelli. 2014. CROMER: a tool for cross-document event and entity coreference. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 3204–3208, Reykjavik, Iceland. European Language Resources Association (ELRA).

Ankita Gupta, Marzena Karpinska, Wenlong Zhao, Kalpesh Krishna, Jack Merullo, Luke Yeh, Mohit Iyyer, and Brendan O'Connor. 2023. ezCoref: Towards unifying annotation guidelines for coreference resolution. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 312–330, Dubrovnik, Croatia. Association for Computational Linguistics.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with MACE. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1120–1130, Atlanta, Georgia. Association for Computational Linguistics.

Ayal Klein, Jonathan Mamou, Valentina Pyatkin, Daniela Stepanov, Hangfeng He, Dan Roth, Luke Zettlemoyer, and Ido Dagan. 2020. QANom: Question-answer driven SRL for nominalizations. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3069–3083, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Mateusz Kopeć. 2014. MMAX2 for coreference annotation. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 93–96, Gothenburg, Sweden. Association for Computational Linguistics.

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

Frédéric Landragin, Thierry Poibeau, and Bernard Victorri. 2012. ANALEC: a new tool for the dynamic annotation of textual data. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 357–362, Istanbul, Turkey. European Language Resources Association (ELRA).

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.

Stephen Mayhew and Dan Roth. 2018. TALEN: Tool for annotation of low-resource ENtities. In *Proceedings of ACL 2018, System Demonstrations*, pages 80–86, Melbourne, Australia. Association for Computational Linguistics.

Bruno Oberle. 2018. SACR: A drag-and-drop based tool for coreference annotation. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Tim O'Gorman, Kristin Wright-Bettner, and Martha Palmer. 2016. Richer event description: Integrating event coreference with temporal, causal and bridging annotation. In *Proceedings of the 2nd Workshop on Computing News Storylines (CNS 2016)*, pages 47–56, Austin, Texas. Association for Computational Linguistics.

Philip V. Ogren. 2006. Knowtator: A protégé plug-in for annotated corpus construction. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Demonstrations*, pages 273–275, New York City, USA. Association for Computational Linguistics.

Rebecca J. Passonneau and Bob Carpenter. 2014. The benefits of a model of annotation. *Transactions of the Association for Computational Linguistics*, 2:311–326.

Silviu Paun, Bob Carpenter, Jon Chamberlain, Dirk Hovy, Udo Kruschwitz, and Massimo Poesio. 2018. Comparing Bayesian models of annotation. *Transactions of the Association for Computational Linguistics*, 6:571–585.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 1–40, Jeju Island, Korea. Association for Computational Linguistics.

Valentina Pyatkin, Ayal Klein, Reut Tsarfaty, and Ido Dagan. 2020. QADiscourse - Discourse Relations as QA Pairs: Representation, Crowdsourcing and Baselines. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2804–2819, Online. Association for Computational Linguistics.

Vikas Chandrakant Raykar, Shipeng Yu, Linda H. Zhao, Gerardo Hermosillo, Charles Florin, Luca Bogoni, and Linda Moy. 2010. Learning from crowds. *J. Mach. Learn. Res.*, 11:1297–1322.

Nils Reiter. 2018. CorefAnnotator - A New Annotation Tool for Entity References. In *Abstracts of EADH: Data in the Digital Humanities*.

Paul Roit, Ayal Klein, Daniela Stepanov, Jonathan Mamou, Julian Michael, Gabriel Stanovsky, Luke Zettlemoyer, and Ido Dagan. 2020. Controlled crowdsourcing for high-quality QA-SRL annotation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7008–7013, Online. Association for Computational Linguistics.

Younes Samih, Wolfgang Maier, and Laura Kallmeyer. 2016. SAWT: Sequence annotation web tool. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*, pages 65–70, Austin, Texas. Association for Computational Linguistics.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. brat: a web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107, Avignon, France. Association for Computational Linguistics.

Michael Stewart, Wei Liu, and Rachel Cardell-Oliver. 2019. Redcoat: A collaborative annotation tool for hierarchical entity typing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pages 193–198, Hong Kong, China. Association for Computational Linguistics.

Stephen Tratz and Nhien Phan. 2018. A web-based system for crowd-in-the-loop dependency treebanking. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Xiaozhi Wang, Yulin Chen, Ning Ding, Hao Peng, Zimu Wang, Yankai Lin, Xu Han, Lei Hou, Juanzi Li, Zhiyuan Liu, Peng Li, and Jie Zhou. 2022. MAVEN-ERE: A unified large-scale dataset for event coreference, temporal, causal, and subevent relation extraction. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 926–941, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Antoine Widlöcher and Yann Mathet. 2012. The glozz platform: A corpus annotation and mining tool. In *Proceedings of the 2012 ACM Symposium on Document Engineering*, DocEng '12, page 171–180, New York, NY, USA. Association for Computing Machinery.

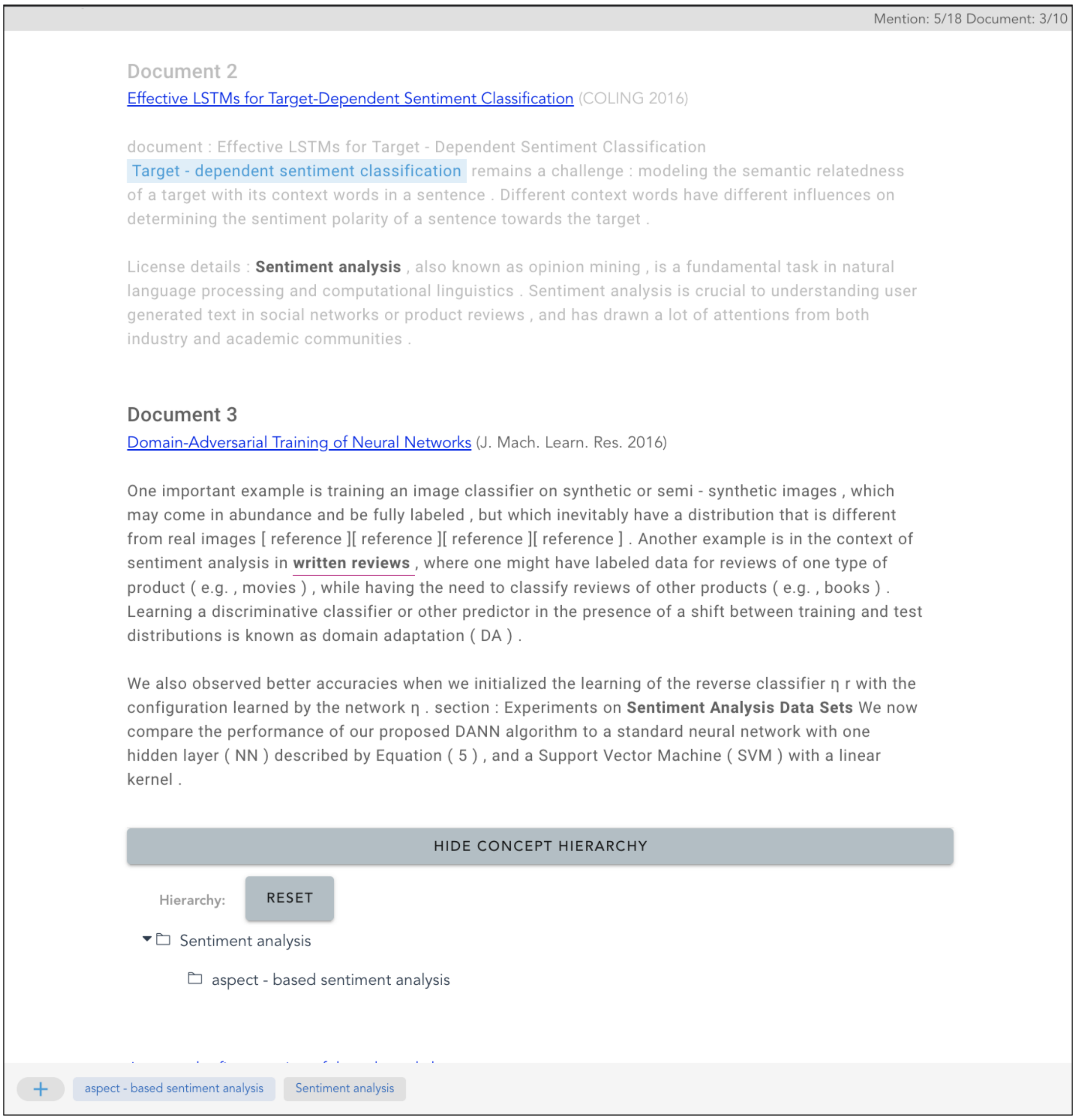
Jie Yang, Yue Zhang, Linwei Li, and Xingxuan Li. 2018. YEDDA: A lightweight collaborative text span annotation tool. In *Proceedings of ACL 2018, System*

Demonstrations, pages 31–36, Melbourne, Australia. Association for Computational Linguistics.

Seid Muhie Yimam, Iryna Gurevych, Richard Eckart de Castilho, and Chris Biemann. 2013. WebAnno: A flexible, web-based and visually supported system for distributed annotations. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 1–6, Sofia, Bulgaria. Association for Computational Linguistics.

A Appendix

Figure 4 shows the interface of CHAMP for annotating a hierarchy of clusters over text spans appearing in their context. This example was taken from SCICO.



Document 2

Effective LSTMs for Target-Dependent Sentiment Classification (COLING 2016)

document : Effective LSTMs for Target - Dependent Sentiment Classification Target - dependent sentiment classification remains a challenge : modeling the semantic relatedness of a target with its context words in a sentence . Different context words have different influences on determining the sentiment polarity of a sentence towards the target .

License details : **Sentiment analysis** , also known as opinion mining , is a fundamental task in natural language processing and computational linguistics . Sentiment analysis is crucial to understanding user generated text in social networks or product reviews , and has drawn a lot of attentions from both industry and academic communities .

Document 3

Domain-Adversarial Training of Neural Networks (J. Mach. Learn. Res. 2016)

One important example is training an image classifier on synthetic or semi - synthetic images , which may come in abundance and be fully labeled , but which inevitably have a distribution that is different from real images [reference][reference][reference][reference] . Another example is in the context of sentiment analysis in **written reviews** , where one might have labeled data for reviews of one type of product (e.g. , movies) , while having the need to classify reviews of other products (e.g. , books) . Learning a discriminative classifier or other predictor in the presence of a shift between training and test distributions is known as domain adaptation (DA) .

We also observed better accuracies when we initialized the learning of the reverse classifier η r with the configuration learned by the network η . section : Experiments on **Sentiment Analysis Data Sets** We now compare the performance of our proposed DANN algorithm to a standard neural network with one hidden layer (NN) described by Equation (5) , and a Support Vector Machine (SVM) with a linear kernel .

Figure 4: User interface for annotating hierarchy of clusters over textual spans that appear within surrounding context.