Data-driven Coreference-based Ontology Building

Shir Ashury-Tahan[12], **Amir David Nissan Cohen**[1], **Nadav Cohen**[1], **Yoram Louzoun**[1] and **Yoav Goldberg**[13]
[1]Bar-Ilan University, [2]IBM Research, [3]Allen Institute for AI
shirashury@gmail.com

Abstract

While coreference resolution is traditionally used as a component in individual document understanding, in this work we take a more global view and explore what can we learn about a domain from the set of all document-level coreference relations that are present in a large corpus. We derive coreference chains from a corpus of 30 million biomedical abstracts and construct a graph based on the string phrases within these chains, establishing connections between phrases if they co-occur within the same coreference chain. We then use the graph structure and the betweeness centrality measure to distinguish between edges denoting hierarchy, identity and noise, assign directionality to edges denoting hierarchy, and split nodes (strings) that correspond to multiple distinct concepts. The result is a rich, data-driven ontology over concepts in the biomedical domain, parts of which overlaps significantly with human-authored ontologies. We release the coreference chains and resulting ontology [1] under a creative-commons license, along with the code [2].

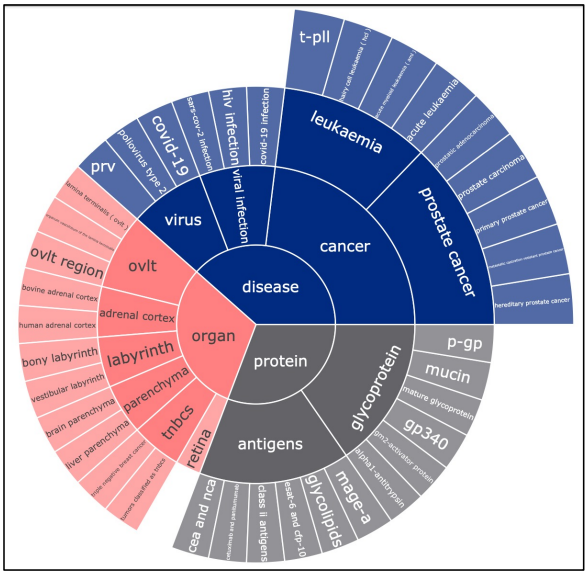

Figure 1: **Resulting Ontology Example** that may reflect the type of structure achievable using our method.

1 Introduction

Ontologies categorize concepts into groups and arranges them in a hierarchy and are essential for researchers in the biomedical domains (Bodenreider and Burgun, 2005; Rubin et al., 2008; Matentzoglu et al., 2022), as evidenced by the vast number of ontologies available in repositories such as BioPortal[3]. These ontologies are predominately human curated, they each contains a collection of concepts arranged in a hierarchy, and for each concept a list of aliases, which are different equivalent names for this concept. While useful, such ontologies have deficiencies: being manually curated they are both expensive to create and maintain and also non-comprehensive; they do not cover all areas of interest a researcher may be interested in, especially for long-tail interests (for example, BioPortal does not contain an ontology containing a comprehensive list of peptides); and the concept names and their aliases may not be aligned with how the concepts appear in text, reducing their utility for text mining applications (Blair et al., 2014) (for example, the UMLS ontology entry for "Alzheimer's disease" does not contain the string "alzheimer", although it is a very common way to refer to this condition in text). Thus, a data-driven, text-based ontology derived directly from the scientific literature can be of immense value: (a) it will provide coverage of all (or most) the concepts that appear in the text, including long-tail ones, arranged in hierarchies based on their actual use in scientific texts; (b) concepts names and aliases will be naturally aligned with their text appearances; and (c) they can aid

[1]https://huggingface.co/spaces/biu-nlp/Data-driven_Coreference-based_Ontology
[2]https://github.com/ShirApp/Coreference-based-Ontology-Building
[3]https://bioportal.bioontology.org/

Findings of the Association for Computational Linguistics: EMNLP 2024, pages 14290–14300
November 12-16, 2024

manual creation, extension and maintenance of existing ontologies by surfacing areas of deficiencies in coverage, and suggesting alternative hierarchies and potentially missing entries.

In this work we propose to create such a data-driven ontology from text, using a novel signal: the topology of a graph created by running in-document coreference resolution over scientific documents, and creating a graph where nodes are textual strings and edges represent that the two strings participated in a coreference chain. We show that his graph's topology contains rich information which allows to identify concepts, aliases and hierarchies like the ones in the Figure 1.

We exploit the dynamics of phrase co-occurrence within the graph, observing a correlation between a phrase's contribution to information flow and its level of generality. Therefore, our approach centers on a single centrality measure, specifically betweenness centrality (Freeman, 1977), aimed at understanding information flow. This measure guides the transformation of the graph into a directed structure, establishing the framework for ontology construction.

2 Coreference-based Ontology Construction

2.1 Coreference Graph Construction

We run a coreference resolution algorithm (Otmazgin et al., 2023)[4] on each of 30M PubMed abstracts to extract coreference chains from each abstract. Each coreference chain is a list of phrases that occur in the same document, and were determined by the coreference algorithm to co-refer to the same concept.

We filter phrases that correspond to pronouns and stop words (as determined by SciSpacy (Neumann et al., 2019) and NLTK (Bird et al., 2009)), remove stop-words, pronouns, determiners and quantifiers from the beginning of phrases, and unified singular and plural versions of phrases. We further remove phrases which we determine to contain only verbs, as these stem mostly from coreference mistakes, and do not correspond to entities. We then designate each of the unique remaining phrases as nodes, and connect two nodes if their phrases co-occur in the same coreference chain, weighing the edge by the number of chains in which this pair co-occurs. (The same phrase may appear in different documents, hence participating in multiple chains)

The resulting graph G has over 3 million nodes and approximately 7 million weighted edges.

2.2 Ontology Extraction

Our aim is to take the corefence graph G and extract an ontology: a directed acyclic graph where each node corresponds to a concept, and an edge from node B to node A indicates that A is more specific than B ("A is a B"). Each node is associated with one or more strings which are aliases for this concept. To extract an ontology from G, we:

1. Identify equivalence relations between nodes, which will form the aliases. We do this by marking some edges in G as indicating *identity*.
2. Mark the remaining edges in G as indicating a hierarchy, and assign them a direction.
3. Split some nodes where the same string corresponds to multiple distinct concepts.
4. Tag some edges in G as noisy or irrelevant.

At a high level, we utilize estimated betweenness centrality values of the nodes to determine the kind and direction of each edge in the graph, thereby transforming the graph into a Directed Acyclic Graph (DAG), from which we will derive the ontology.

Betweenness Centrality as a Main Measure The coreference graph is undirected, and we wish to assign edges with direction that indicate IS-A relations. A major observation is that phrases that denote concepts that are higher-up in the IS-A hierarchy (are more general) co-occur in many different coreference clusters, and with many different phrases, while phrases that are more specific belong in only few clusters, with a more restricted set of phrases (e.g., concepts like "disease" will appear in many clusters denoting specific diseases, "lung diseases" will appear with "disease" as well as with many specific lung diseases, while "asthma" may share a cluster with "diseases" and with "lung disease", but likely not with other lung diseases). Consequently, if we choose two random nodes in the graph, if there is a path between them it either goes directly from the less specific to the more specific one, or it goes to a common shared parent (which is more general than both) and then back. Thus,

[4]We selected this algorithm for its highly efficient runtime, with minimal impact on performance, which is crucial for handling large corpora like those used in our experiments.

we expect the more general concepts to be on more paths connecting pairs of nodes in the graph. This is precisely the notion that is captured by the *betweeness centrality* measure.

Therefore, the first step in establishing the edge direction is to compute the betweeness centrality score of each node, and assign the direction of an edge to be from the the node with higher centrality (more general nodes) to one with lower centrality. When both nodes havea centrality score of zero (these nodes don't connect different concepts) we denote them as identity edges.

As exact centrality computation is expensive—$O(V^2 + V \times E)$ when using the fast algorithm of Brandes (2001)—and our graph is large, we opted for an approximate solution that relies on performing a restricted number of shortest-path computations above a small set of randomly chosen pivots (Brandes and Pich, 2007). Using this approximation with 500 pivots works well for our purpose without reducing accuracy, as we use centrality scores only for computing relative ordering, rather than needing the exact values (as also detailed in Appendix A).

From Graph to DAG Roughly 70% of the nodes have a betweenness value of 0, suggesting they function as leaves in the DAG hierarchy. This is while 240,000 of the edges in the graph connect such nodes. Consulting a random sample of edges reveal that they indeed connect aliases of the same concept. We subsequently mark these edges as (unordered) identity edges, indicating alises. The rest of the edges (6.7 million edges) are marked temporarily as indicating a hierarchy, and we assign direction from the node with higher betweeness score to the one with the lower score. (In the next steps, we will reveal that some of these "hierarchy edges" are actually identity edges or noise, and we relabel them as such.)

Betweenness Centrality Challenges Using betweenness centrality for determining edge direction is sensitive to common nodes that represent specific entities. These nodes often exhibit higher betweenness than their neighbors, leading to misleading hierarchical relations. We identified three instances of this:

1. A node that co-occurs with other phrases in coreference chains more frequently than its more general neighbor (e.g., "COVID-19" vs. "epidemic"), as illustrated in Figure 4.

2. A specific entity name appears more frequently than its aliases (e.g., "gys2" vs. "glycogen synthase 2"), as depicted in Figure 5.

3. A node shares a common name with multiple entities (e.g., "IL" for "Illinois," "IL-6," "IL-8"), as shown in Figure 6.

The first case leads to incorrect hierarchy direction, while the second leads to incorrect tags.

For the first case, we establish that names are more specific than general nouns, so we change the direction of edges from nouns to names, correcting around 200,000 edges[5].

We failed to tag edges as identity in cases 2 or 3 due to their substructure in the graph. All edges to neighbors were wrongly marked as hierarchical instead of identity, and case 3 also involves string ambiguity, causing incorrect paths. Our goal is to fix these tags and paths. Identifying these cases requires finding all names with children. Case 2 involves one concept, while case 3 has at least two distinct senses. We group nodes into concepts using semantic representations, ensuring nodes of the same entity are closer. For case 3, we split the ambiguous node into separate concepts.

We designed Algorithm 1 to resolve incorrect substructures by grouping strings into distinct entities. We embed the node n and its children $C = c_1, c_2, \ldots, c_l$ using a language model[6], capturing their semantic similarities. Next, we create a KNN subgraph, where n and $c_i \in C$ are the nodes, and each is connected by an edge to its k closest neighbors. The resulting subgraph captures relations between phrases based on their semantic similarity, meaning that strongly connected phrases are more likely to refer to the same entity. The Louvain algorithm (Blondel et al., 2008) is then applied to the subgraph to detect communities (dense areas) representing distinct senses, allowing us to split nodes accordingly, if necessary. Finally, each community is treated as a concept. If one community is detected (case 2), we merge n and $c_i \in C$ into a single concept. Otherwise (case 3), each detected community is treated as a separate concept, and n is split into different nodes (with the same string)

[5]Names are identified based on capitalization; consistently capitalized terms are names, while others are nouns.

[6]We used the S-BERT model (Reimers and Gurevych, 2019) from (Deka et al., 2022), trained on health data. The only input to the model was the phrase itself.

for each concept. During this process we tagged about 230,000 edges from hierarchical to identity.

Cleaning noisy edges We observe unwanted noisy edges in the graph that connect very common phrases (e.g. "group" and "variant") that are not supposed to be connected. These edges arise from pairs of phrases that are mistakenly assigned to the same coreference cluster.

In these cases, the erroneous relations are due to mistakes made by the coreference annotator, and their edges have much lower weight compared to other relations in which their respective nodes participate. We therefore used the PMI measure (Fano and Hawkins, 1961) to calculate the association between each pair of connected phrases. Edges whose association was less than expected by chance, i.e., those with negative PMI values, were filtered out.

Let the probability $P(\text{phrase}_i)$ be defined as the ratio of the count of co-occurrences of phrase phrase_i with other phrases to the total number of co-occurrences of all phrases in the corpus: $P(\text{phrase}_i) = \frac{\text{count}(\text{phrase}_i)}{\sum_{k=1}^{N}\text{count}(\text{phrase}_k)}$ where N is the total number of distinct phrases in the corpus. Let us also denote the joint probability $P(\text{phrase}_i, \text{phrase}_j)$, which is defined as the number of co-occurrences of phrases phrase_i and phrase_j divided by the total number of co-occurrences in the corpus: $P(\text{phrase}_i, \text{phrase}_j) = \frac{\text{count}(\text{phrase}_i, \text{phrase}_j)}{\sum_{k=1}^{N}\text{count}(\text{phrase}_k)}$ We calculated the PMI for each pair of phrases connected by an edge in the graph as follows: $PMI(\text{phrase}_i, \text{phrase}_j) = \log\left(\frac{P(\text{phrase}_i, \text{phrase}_j)}{P(\text{phrase}_i)\cdot P(\text{phrase}_j)}\right)$. We identified approximately 350,000 such edges and labeled them as noise.

The Final Graph Overall, we were able to label over 6 million graph edges. We marked most of them as indicating an identity or hierarchy relation, and the rest as noise. We found the hierarchical relation to be much more common in our graph. We marked approximately 5.3 million edges as directed edges indicating a hierarchy, and about 440,000 as identity edges. The rest of 350,000 edges are tagged as noise.

3 Evaluation and Results

Evaluating the quality of the resulting graph is challenging, as there is no ground-truth to compare to (McCrae, 2009). Still, we compare our results to existing human curated ontologies in the biomedical domain (UMLS (Bodenreider, 2004) and SnomedCT (Bodenreider et al., 2018)), and assess how well we manage to capture concepts from them. UMLS provides aliases for identity nodes, while SnomedCT provides hierarchical relations and directions between concepts. If these resources were perfect, we wouldn't need to create the data-driven one to begin with. We thus combine automatic metrics with human evaluations.

Evaluating Hierarchy Assignments We compare ourselves to SnomedCT, an ontology with 1.4M medical phrases and 1.7M corresponding "is a" relation tuples. We consider only edges between the strings that are available in both SnomedCT and our data resulting in 226,278 edges for evaluation. Let *correct* denote the number of predicted hierarchy edges that participate in the same hierarchy in SnomedCT (there is a directed path between them in SnomedCT). We compute *precision* as *correct / all predicted hierarchy edges*, and *recall* as *correct / all edges that are marked as hierarchy in SnomedCT*. We achieve a high **recall of 84.3%**, with a **lower precision, at only 40.1%**. However, examining the precision error reveals that many cases stem from valid disagreements between the resources. For example, our graph places "*defibrillation*" under "*procedure*", which is not reflected in SnomedCT. We thus sample 100 random hierarchy edges and annotate them manually (not compared to SnomedCT), revealing a **substantially higher precision of 75%**.

Hierarchy Edge Direction Evaluation For hierarchy edges whose end-points are reachable also in SnomedCT, we find the edge direction is **consistent with SnommedCT in 92.1% of the cases**.

Identity-edge Evaluation Finally, we evaluate the accuracy of the identity edges, which represent aliases. Here, we value precision over recall: it is better to miss an alias than to introduce an incorrect one: mistaking an alias relation for a hierarchical one is not as bad as erroneously equating two concepts. Here, we compare to UMLS aliases, focusing on the 29,798 strings that are shared between our ontology and UMLS. We treat identity edges as inducing clusters, evaluate the clustering using two metrics: entropy, to measure the homogeneity of the predicted clusters compared to a gold standard (lower means more homogeneous) and Adjusted Rand Index (ARI) to measures similarity between our clustering and UMLS's. We obtain an **entropy**

Algorithm 1 Split and Classify Names with Hierarchical Behavior to Senses Algorithm

1. Identify all nodes that are names (starting with a capital letter) and are hierarchical in our DAG (have children).
2. For each node n, and its children $C = \{c_1, c_2, \ldots, c_l\}$:
   (a) Embed each string in $V = \{n\} \cup C$ using an LLM.
   (b) Create a nearest neighbour graph over V, where each node has an edge to its k closest nodes.
   (c) Use the Louvain algorithm to find the communities, representing the senses.
   (d) Split n into $|communities|$ nodes and unify each split with a community. The split node would inherit only the common parents with the concept it is connected to.

Ontology	Task	Measure	Score
SnomedCT	Hierarchy edges	Precision	0.401
		Recall	0.842
		F1	0.541
	Hierarchy Directions	Precision	0.921
UMLS	Identity edges	Entropy	0.406
		ARI	0.387

Table 1: **Our results compared to other ontologies** show that we successfully identified the majority of hierarchy edges, with a small number of errors in concepts.

of 0.406 for the predicted clusters, suggesting the clusters are reasonably pure (do not contain many erroneous entries). Moreover, the moderate **ARI score of 0.387** indicates that our clusters are also split well.

The evaluation suggests our approach aligns well with established ontologies. Human assessments of hierarchical edges and entropy metrics indicate near-precision, while recall and ARI measures suggest the extracted ontology is close to complete.

4 Related Work

Ontology learning, essential for many applications, has traditionally relied on linguistic approaches, using pattern extraction and syntactic analysis combined with statistical methods. While syntactic patterns can be effective, they often require human intervention. A well-known example is (Fellbaum and Miller, 1998), which designed a process to extract patterns for specific relations from phrase pairs. Some works apply syntactic analysis by parsing dependency trees to extract relations and build ontologies automatically, as seen in (Ciaramita et al., 2005; Gamallo et al., 2002). Statistical methods like (Drymonas et al., 2010) use hierarchical clustering for taxonomy building and conditional probabilities for non-taxonomic relations, while (Faure and Nédellec, 1998) applies conceptual clustering to automatically acquire and organize subcategorization frames.

These techniques differ from ours by focusing on local relations, while we make use of global signals—cross-document-level coreference relations—to identify relationships more globally within the corpus.

Recent studies have explored using large language models (LLMs) for ontology building due to their ability to capture contextual information (Brown et al., 2020; Devlin et al., 2019; Achiam et al., 2023). For example, (Giglou et al., 2023) applied a zero-shot prompting approach to term typing, taxonomy discovery, and relation extraction, while (Funk et al., 2023) used ordered prompts to construct concept hierarchies. Both studies show that while LLMs can aid ontology learning, they cannot yet construct ontologies independently.

5 Conclusions

We demonstrated that a text-based, data-driven biomedical ontology[7] can be created by considering the topology of a coreference graph obtained from a large corpus. Furthermore, we achieved this primarily through the use of a single centrality measure. A significant contribution of this approach is its generality, allowing for easy adaptation to other fields. Additionally, our method is scalable and can be implemented for networks of varying sizes. Compared to existing ontologies, we obtain very accurate directionality and high recall of hierarchical structure. We also find accurate hierarchical relations that are not reflected in the human-curated ontologies. In future work, our automatically constructed ontology could be applied to downstream tasks.

[7] https://huggingface.co/spaces/biu-nlp/Data-driven_Coreference-based_Ontology

6 Limitations

Evaluation difficulties. Assessing our unsupervised approach poses challenges in achieving a comprehensive and scalable evaluation. Direct comparisons to established ontologies are complicated, as these may not fully capture diverse language usages present in extensive corpora. Manual evaluations, limited by scalability, may not be wholly representative of our graph's overall quality.

Error Propagation. The propagation of errors or inconsistencies from the corpus into the ontology might compromise its quality and accuracy.

7 Ethics Statement

We do not identify ethical concerns with this work. The resulting ontology is useful but not perfectly accurate, and must be used with care and using human oversight.

References

OpenAI Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haim ing Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Benjamin Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Sim'on Posada Fishman, Juston Forte, Is abella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Raphael Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Lukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Hendrik Kirchner, Jamie Ryan Kiros, Matthew Knight, Daniel Kokotajlo, Lukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Ma teusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel P. Mossing, Tong Mu, Mira Murati, Oleg Murk, David M'ely, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Ouyang Long, Cullen O'Keefe, Jakub W. Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alexandre Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Pondé de Oliveira Pinto, Michael Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack W. Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario D. Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin D. Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas A. Tezak, Madeleine Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cer'on Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll L. Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qim ing Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2023. Gpt-4 technical report.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc.

David R. Blair, Kanix Wang, Svetlozar Nestorov, James A. Evans, and A. Rzhetsky. 2014. Quantifying the impact and extent of undocumented biomedical synonymy. *PLoS Computational Biology*, 10.

Vincent D. Blondel, Jean-Loup Guillaume, Renaud

Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008.

O Bodenreider. 2004. The unified medical language system (umls): integrating biomedical terminology. *Nucleic Acids Res*, 32(Database issue):D267–70.

Oliver Bodenreider, Ronald Cornet, and Daniel J Vreeman. 2018. Recent developments in clinical terminologies - snomed ct, loinc, and rxnorm. *Yearbook of medical informatics*, 27:129–139.

Olivier Bodenreider and Anita Burgun. 2005. Biomedical ontologies. *Medical Informatics: Knowledge Management and Data Mining in Biomedicine*, pages 211–236.

Ulrik Brandes. 2001. A faster algorithm for betweenness centrality. *Journal of mathematical sociology*, 25(2):163–177.

Ulrik Brandes and Christian Pich. 2007. Centrality estimation in large networks. *International Journal of Bifurcation and Chaos*, 17(07):2303–2318.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Ma teusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *ArXiv*, abs/2005.14165.

Massimiliano Ciaramita, Aldo Gangemi, Esther Ratsch, Jasmin Saric, and Isabel Rojas. 2005. Unsupervised learning of semantic relations between concepts of a molecular biology ontology. pages 659–664.

Pritam Deka, Anna Jurek-Loughrey, and P Deepak. 2022. Improved methods to aid unsupervised evidence-based fact checking for online health news. *Journal of Data Intelligence*, 3(4):474–504.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*.

Efthymios Drymonas, Kalliopi Zervanou, and Euripides G. M. Petrakis. 2010. Unsupervised ontology acquisition from plain texts: The ontogain system. In *International Conference on Applications of Natural Language to Data Bases*.

Robert M Fano and David Hawkins. 1961. Transmission of information: A statistical theory of communications. *American Journal of Physics*, 29(11):793–794.

David Faure and Claire Nédellec. 1998. A corpus-based conceptual clustering method for verb frames and ontology.

Christiane Fellbaum and George Miller. 1998. *Automated Discovery of WordNet Relations*, pages 131–151.

Linton C Freeman. 1977. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41.

Maurice Funk, Simon Hosemann, Jean Christoph Jung, and Carsten Lutz. 2023. Towards ontology construction with language models. *ArXiv*, abs/2309.09898.

Pablo Gamallo, Marco González, Alexandre Agustini, Gabriel Lopes, and Vera Lúcia Strube de Lima. 2002. Mapping syntactic dependencies onto semantic relations.

Hamed Babaei Giglou, Jennifer D'Souza, and S. Auer. 2023. Llms4ol: Large language models for ontology learning. *ArXiv*, abs/2307.16648.

Nicolas Matentzoglu, Damien Goutte-Gattat, Shawn Zheng Kai Tan, James P. Balhoff, Seth Carbon, Anita R. Caron, William D. Duncan, Joe E. Flack, Melissa Haendel, Nomi L. Harris, et al. 2022. Ontology development kit: a toolkit for building, maintaining and standardizing biomedical ontologies. *Database*, 2022:baac087.

John P. McCrae. 2009. Automatic extraction of logically consistent ontologies from text corpora.

Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. 2019. ScispaCy: Fast and robust models for biomedical natural language processing. In *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 319–327, Florence, Italy. Association for Computational Linguistics.

Shon Otmazgin, Arie Cattan, and Yoav Goldberg. 2023. LingMess: Linguistically informed multi expert scorers for coreference resolution. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2752–2760, Dubrovnik, Croatia. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Conference on Empirical Methods in Natural Language Processing*.

Daniel L. Rubin, Nigam H. Shah, and Natalya F. Noy. 2008. Biomedical ontologies: a functional perspective. *Briefings in bioinformatics*, 9(1):75–90.

A Centrality Approximation for Betweenness in Large Networks

A.1 Overview of Betweenness Centrality

Betweenness centrality (Freeman, 1977) is a centrality measure that aids, among other things, in the analysis and characterization of graphs. It is particularly valuable in networks where intermediaries play a crucial role in facilitating information flow or identifying connectivity. At its core, betweenness centrality quantifies the significance of a node within a network by assessing how frequently it lies on the shortest paths between other nodes. Mathematically, for a given node v, its betweenness centrality $BC(v)$ is defined as:

$$BC(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

where σ_{st} denotes the total number of shortest paths from node s to node t, and $\sigma_{st}(v)$ represents the number of those paths that pass through node v.

However, computing the betweenness centrality for each node in the graph requires calculating all shortest paths between pairs of nodes. The fastest known algorithm for this task (Brandes, 2001) has a time complexity of $O(V^2 + V \times E)$.

A.2 Approximation Algorithm

As networks scale in size, such as in our case, the number of shortest paths that must be computed is huge, making exact calculations infeasible. A practical solution is to employ an approximate betweenness centrality calculation. This approach is particularly valid when only relative importance is required, as is the case in our work where we aim to establish an order among the nodes.

To address the computational cost, we sought an appropriate approximation that could dramatically reduce the number of computations while still allowing us to create the desired order. The only approximation that is adapted to large network as ours is (Brandes and Pich, 2007), which proposes a solution that relies on performing a limited number of shortest-path computations over a small set of randomly chosen pivots. By focusing on this smaller subset, the number of path computations is significantly reduced, yielding a complexity of $O(k(V + E))$, where k represents the number of sampled nodes. Increasing k enhances accuracy but may also extend computation time. The main constraints in their framework indicate that the number of pivots should be greater than $\log(V)$ and that the graph's diameter should be constant, conditions that our case satisfies.

A.3 Experimental Results

Accuracy of Results. Since the algorithm assists in creating a DAG, we evaluated its performance based on two criteria. The first criterion is its accuracy in determining the direction of edges in the graph that represent hierarchical relations within the ontologies we compared. The second criterion assesses its accuracy in identifying leaves that are connected to the same concept. We found both metrics to have high scores for $k = 500$: 91.3 for the direction of edges and 86.7 for the accuracy of the connected leaves.

Consistency of Results. To measure approximation consistency and variability, we executed the algorithm 5 times on our graph, using different k values: $100, 500, 1{,}000, 2{,}000, 2{,}500$. We calculated the order for each edge in each run and discovered that only 6% of all edges were conflicted (at least one run disagreed with the others regarding the edge direction). Furthermore, only 0.01% of the cases lacked consensus, with no majority of more than $\frac{2}{3}$ of the runs agreeing on the direction.

B Figures

Here are some figures that may clarify our intuitions and solutions presented in the paper.

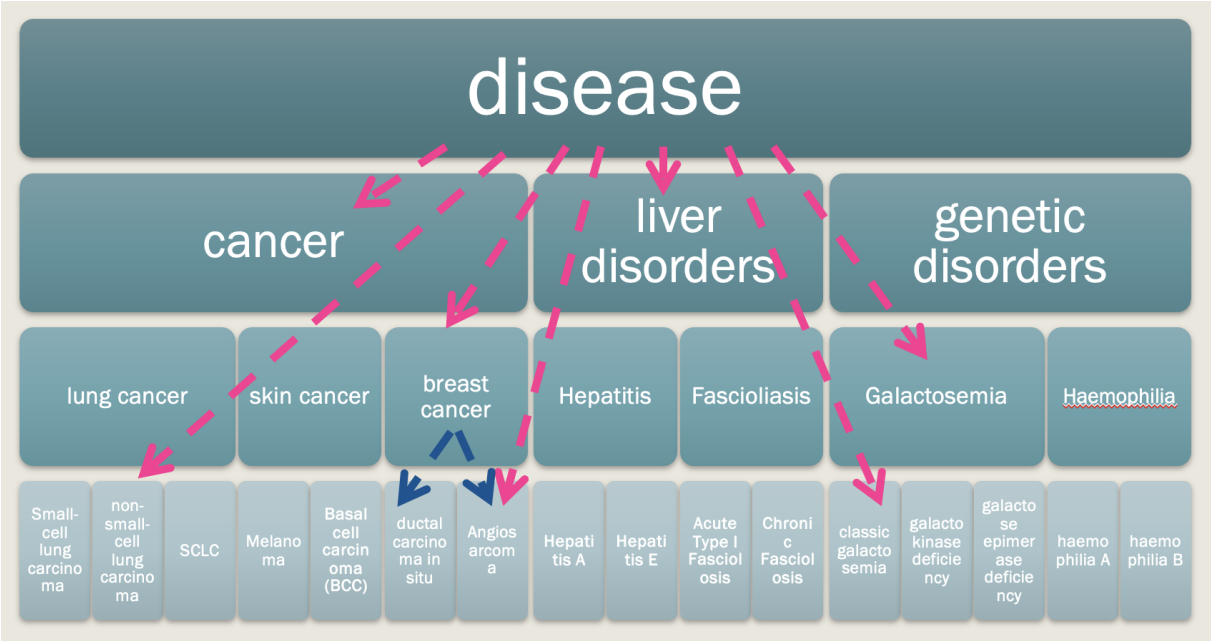

Figure 2: **Co-occurrence behavior example** demonstrating why the more general the phrase, the more central the phrase. Each phrase can appear with any of the phrases that are more specific than it, making a phrase like "disease" a bridge between communities that is much more central than "breast cancer" in our graph.

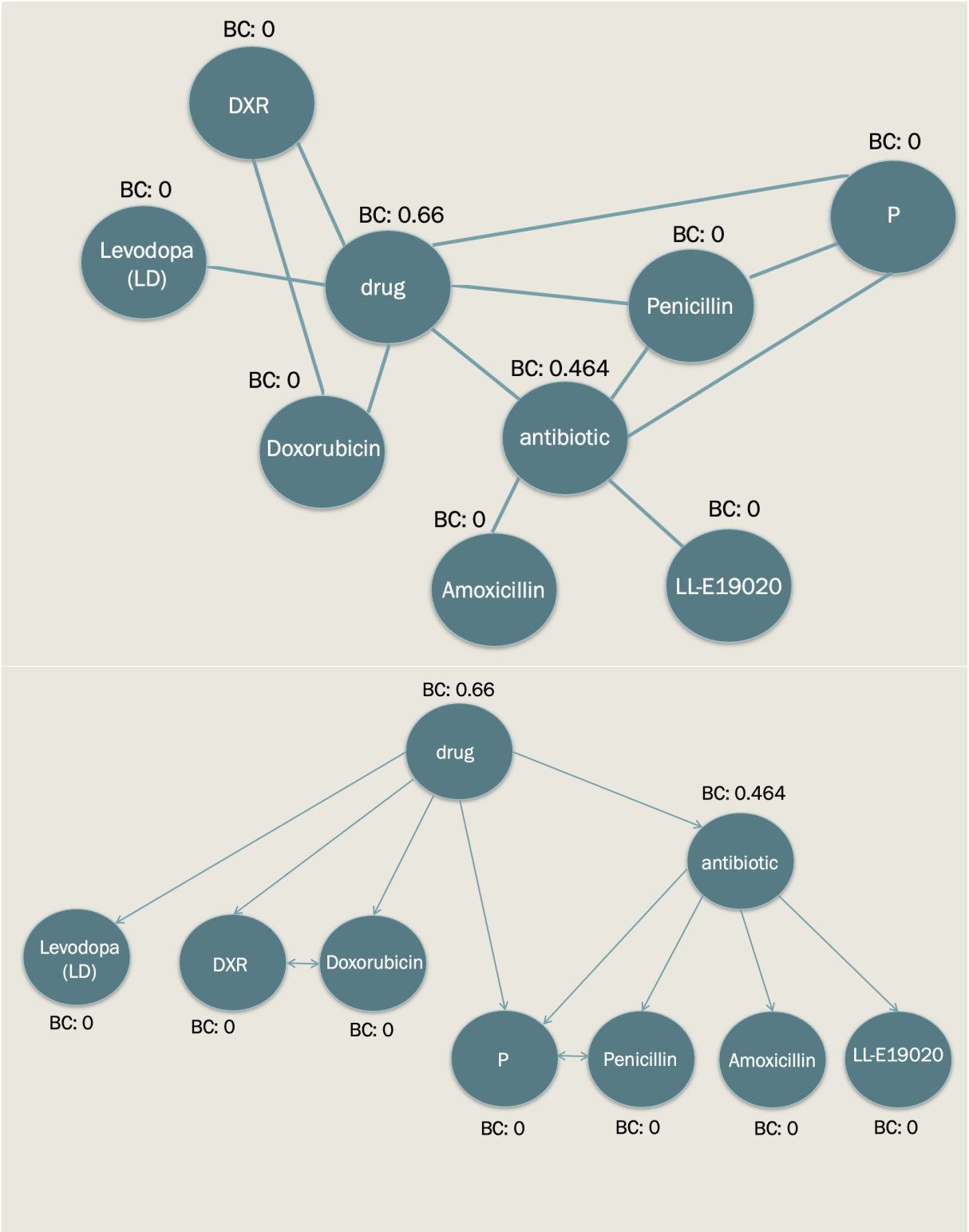

Figure 3: **Example of directions assignment** to the edges in the graph. The upper graph demonstrates the connections between phrases that appeared in our corpus, and their betweeness centrality (BC) values in this graph. The one below shows the result of a directed graph using them.

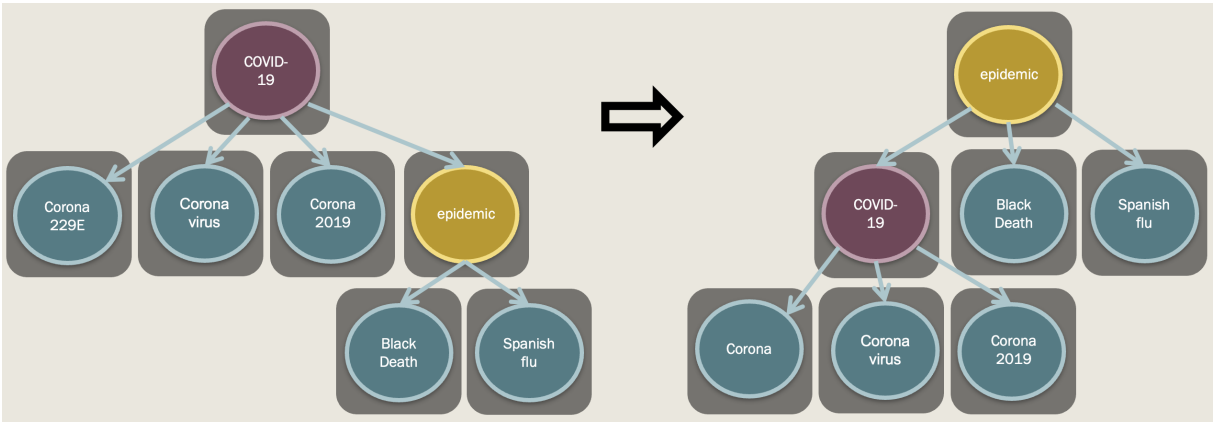

Figure 4: **Fixing Edge Direction** in cases where a name (e.g., "COVID-19") co-occurs with others in coreference chains more frequently than its general phrase neighbor (e.g., "epidemic"). Our solution (on the right) for correcting the directionality in these cases helps make the paths more accurate (The gray background represents a concept)

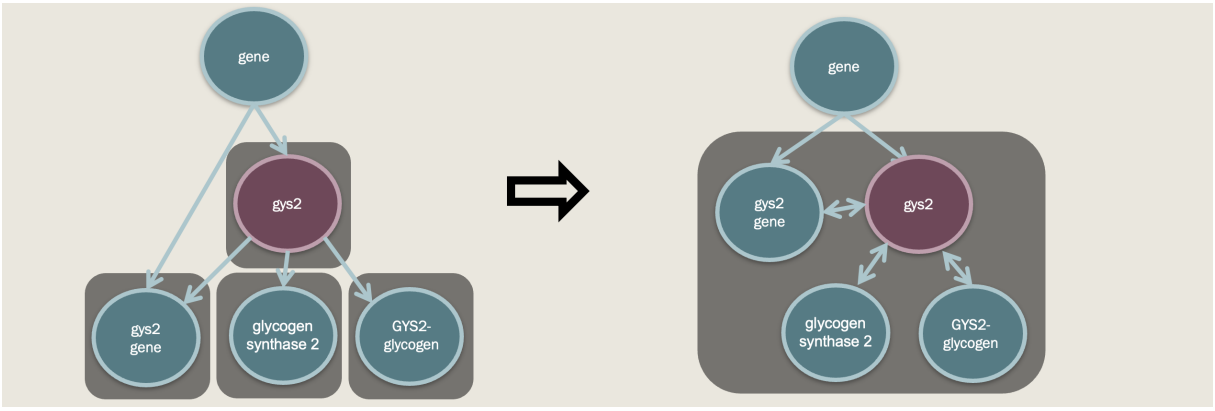

Figure 5: **Union Nodes to a Concept** when a common name (e.g., "gys2") is incorrectly identified as being more hierarchical than its alias neighbors. Our solution (on the right) that is based on semantic similarity representations helps in solving such cases. (The gray background represents a concept)

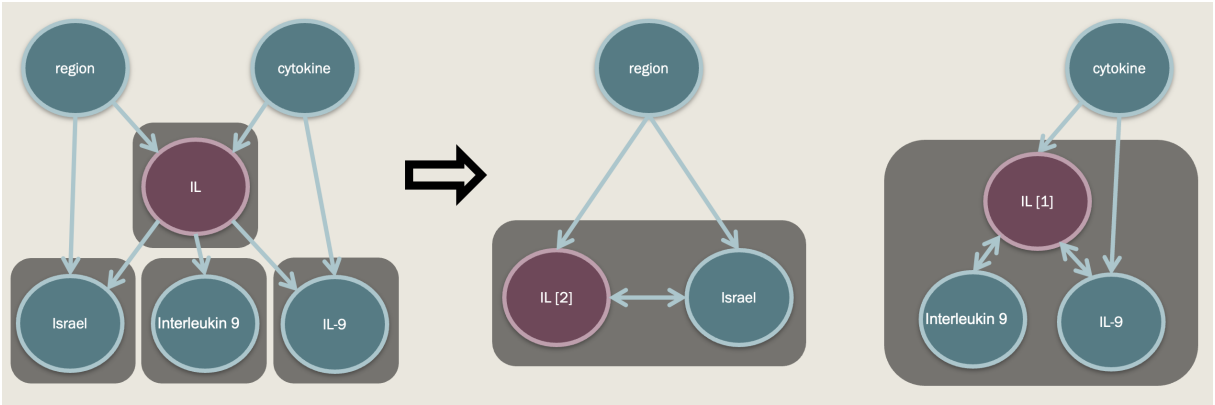

Figure 6: **Splitting Nodes with Multiple Senses** when an abbreviation (e.g., "IL") needs to be split into separate alias nodes for its different meanings (e.g., "Israel" and "IL-9"). Our solution is depicted on the right, showing the rearrangement of the subgraph into concepts, using the semantic similarity representations (The gray background represents a concept.)