Improving Community Detection in Wikipedia Articles Using Semantic Features

Ambika Acharya (aacharya), Edwin Park (edpark), Emily Tang (emjtang)

1 Introduction

Understanding the organization of knowledge is important for search and information retrieval online. Being able to categorize knowledge, and infer a knowledge hierarchy, could help bring insight on how information is transferred or learned. Wikipedia is an example of a large knowledge base that is organized into categories, and perhaps has some inherent hierarchical structure of knowledge.

We consider the problem of finding communities of topics within the Wikipedia knowledge base. Given articles and their outlinks can we predict how topics cluster together to form communities? Furthermore, how can we exploit the semantic attributes of Wikipedia to improve performance of current community detection algorithms? In this paper, we compare current community detection algorithms, and introduce a new method of augmenting the link clustering community detection algorithm with natural language features. We will first examine literature covering different ways of detecting communities, in particular overlapping communities with both dense and non-dense overlaps. Then, we will go into details on our work: the dataset, our methods and modified algorithms, and results from our experiments. We demonstrate that our modified community detection algorithms with natural language features performs better in certain cases.

2 Related Work

2.1 Overlapping Community Detection

2.1.1 Link Communities Reveal Multiscale Complexity (Ahn et al, 2010)

Ahn et al. propose a method of community detection in which communities are viewed as groups of links (edges) rather than node [1]. Their approach is to use hierarchical clustering with similarity between links to build a dendrogram, where each leaf is a link and branches represent link communities. Using a partition density threshold, they then cut the dendrogram at various thresholds; now, nodes belong to the multiple, overlapping communities of their links.

They evaluated this method against other existing node clustering algorithms (InfoMap, clique percolation, greedy modularity optimization), and found that link communities revealed more about a network's metadata and were more relevant. Thus, they concluded that link communities are able to incorporate overlaps in communities, and reveal the hierarchical organization of the network.

2.1.2 Overlapping Community Detection at Scale (Yang & Leskovec, 2013)

Yang and Leskovec aim to design a model to detect both overlapping and non-overlapping communities in large scale networks in their paper [8]. They discuss the fact that most community detection algorithms assume networks with sparsely-overlapping communities, while in reality communities tend to have denser connections when they overlap with other communities than when they don't overlap.

They propose the BigClam algorithm which is a variant of nonnegative matrix factorization designed to take a network G(V,E) and generate a bipartite graph between nodes and communities such that the model works for non-overlapping, overlapping and nested communities. They evaluate their model against six datasets with ground truth community data including Orkut, Youtube community and the Amazon product co-purchasing network. BigClam outperforms other community detection algorithms including Link Clustering and Clique Percolation Method in both the accuracy and runtime performance metrics.

2.1.3 Hierarchical Structure and the Prediction of Missing Links (Clauset et al, 2008)

In their paper [3], Clauset et al. present a technique to infer hierarchical structure from network data. The technique combines a Monte Carlo sampling algorithm with a maximum likelihood approach, and builds upon the assumption that networks can be assortative and/or disassortative. In an assortative network, nodes are more likely to form a connection if the height of their lowest common ancestor is very low; in this situation, probabilities decrease as you move up. In a disassortative network, nodes are more likely to form a connection if their lowest common ancestor is very high up probabilities increase as you move up the dendrogram. We leverage the insights gained from hierarchical structure detection to improve the performance of link clustering and BigClam.

2.2 Discussion

To our knowledge, detecting overlapping communities in the Wikipedia dataset with link clustering or BigClam has not been done yet. In addition, much of the literature on soft clustering, or overlapping community detection, seem to conflict on whether edge clustering [1] or node clustering like BigClam [8] performs better [4]. While Ahn et al. claim that link clustering is better performing than node clustering, Yang and Leskovec demonstrate that BigClam performs better than link clustering. Specifically, BigClam performs an average of 79% higher than link clustering over six performance metrics including overlap quality, community quality and community coverage [8]. However, it is noted that there still lacks conclusive evidence on whether edge or node clustering is better, and it's possible that it is mainly dataset dependent [4]. We evaluated node versus edge clustering on the Wikipedia dataset, and bring some more insight into this problem.

There are a few weaknesses to note in Ahn et al.'s link clustering paper. First, as discovered by Yang and Leskovec, the overlaps between communities might actually be a lot denser than expected, which means that it's possible for edges to also belong in more than one community. Second, there lacked conceptual insight into why edge clustering performed better than the existing node clustering algorithms.

Lastly, none of the work we saw used natural language features to improve their models. We believe

that this will help tremendously in correctly detecting categories for the Wikipedia dataset since pages which have semantic similarity tend to be grouped together.

3 Dataset

We gathered data from Wikispeedia, which has several compiled datasets made up of page links and article categories available online [6, 7]. The Wikispeedia datasets have been cleaned and processed to be structured well for use. For example, article categories specify particular areas of discipline (e.g. British History, Ancient History, Archaeology), in addition to which major disciplines they came from (e.g. History, Science). This dataset contains over 4500 articles and over 106,000 links, with articles labeled with one or more categories from over 130 choices of different categories.

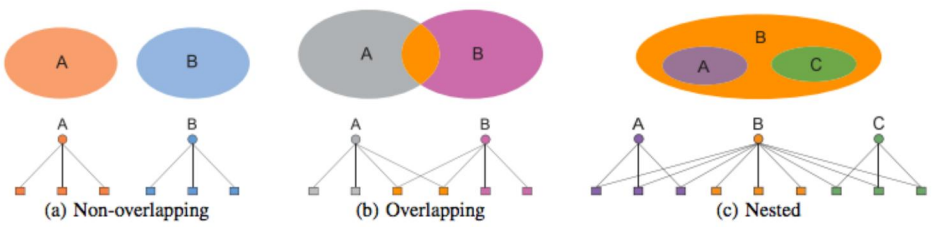
Because Wikipedia articles come from a very diverse range of disciplines, we focused our attention to detecting communities within a single major discipline, as advised by Austin. This was a beneficial approach for community detection because when there are a lot of sparse and overlapping links to other disciplines, it can be difficult to estimate the number of communities, which can consequentially make it difficult to predict communities.

Thus, we focused on the "History" discipline and created a history sub-graph by randomly sampling 10,000 links from a set of links where *at least one end* of the link belongs to a History category. After doing some initial evaluation, we realized that this was a poor sub-graph for community detection because 10,000 links are distributed among 129 communities, which can lead to our algorithms predicting too many small communities (2-3 articles) due to sparse links.

Therefore, we obtained our final dataset by create a sub-graph by only including links if *both* articles belong to a History category, guaranteeing all articles are within the History sector. This dataset contains 495 articles and 3180 links. From this subgraph, we performed a preliminary analysis to see how densely overlapped the communities are in the dataset. There are a total of 138 articles that belong to multiple communities: the most common overlap was between the categories "British History - 1500 and before" and "Monarchs of Great Britain" with 16 overlapping articles, followed by "Classical History and Mythology" and "Historical Figures" with 11 overlapping articles.

3.1 Network Representation

We model Wikipedia articles as nodes, and in/out links between pages as edges in our graph. For community detection, we aimed to generate a bipartite graph between categories (communities) and articles (nodes), as seen in the figure below (taken from Yang and Leskovec) [8]



4 Methods

4.1 Community Detection

After conducting a literature review we thought applying various community detection algorithms to the Wikipedia dataset would be an way to determine which method is best suited for the data.

We implemented two algorithms other papers have shown to be effective on different networks to see how they fare on Wikipedia articles:

1. Edge Clustering (Link Communities)
2. Node Clustering (BigClam)

For edge clustering, we used the source code provided by Ahn et al.[1], and applied the algorithm in their paper. For node clustering (BigClam), we used the implementation provided by Snap to detect community membership of nodes.

4.1.1 Natural Language Features

After evaluating the existing models on our data-set, we found our poor performance was due to focusing only on links between pages, rather than the language similarities pages may share. We decided to supplement the link clustering model with natural language features to see if semantic similarities improve performance. In the link clustering algorithm, we cluster edges by using the Jaccard similarity of two edges as described in Ahn's paper[1] (which we call the link clustering score). With our new community detection algorithm, we compute a semantic similarity score and average it with the link clustering score and use this as the scoring value for our modified link clustering model. We experimented with two ways of computing semantic similarity:

1. **TF-IDF cosine similarity:** For a given edge, we intersect the words in the texts of the two nodes that form the edge (after removing stop words and applying NLTK's Porter stemming algorithm[2]), and compute the term frequency - inverted document frequency score (TFIDF) for each word such that an edge is represented as a vector. Then, for a given pair of edges, x, y, to compute their semantic similarity score we compute their cosine similarity of their tfidf vectors as follows:
$$cos(\boldsymbol{x}, \boldsymbol{y}) = \frac{\boldsymbol{x} \cdot \boldsymbol{y}}{||\boldsymbol{x}|| \cdot ||\boldsymbol{y}||}$$
2. **Unigram Jaccard Similarity:** For a given edge, we intersect the words in the texts of the two nodes that form the edge(removing stop words), such that each edge is represented as a set of words. Then, for a given pair of edges, represented by sets A, B respectively, we compute their semantic Jaccard similarity as follows:
$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$
We evaluate performance based on the entire text of an article, and on just the header text (as Wikipedia's articles are structured as Headers with Paragraph text underneath). We also

evaluate performance of these features in combination with the link clustering score (which we call Augmented Link Clustering) as described above and also by themselves.

We hypothesize that the natural language features will perform better than the link clustering score, which does not take semantic similarity into account, which is a key attribute of our Wikipedia dataset.

4.1.2 K-means and Word2Vec

Another clustering method we used was K-means. We represent each node (article) in our graph as a set of words made up of its full article text, and apply Word2Vec to each word. By applying Word2Vec to each word, we transform each word into a vector-space trained on a pre-defined vocabulary. This transformation is called a word embedding and as a result, every word gets represented by a vector of length 100. For a given node, we average the vectors for all words to get its final word embedding. We then apply kmeans clustering too all nodes to find the optimal cluster assignments:

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x \in S_i} ||x - \mu_i||^2$$

where μ_i is the average vector of cluster S_i. This algorithm initializes k random nodes as cluster centers and with every iteration assigns nodes to each cluster such that their Euclidean distance to that cluster is minimized as compared to other centers. New cluster centers are picked as the average of all the nodes assigned to that cluster, and the algorithm repeats until convergence. We performed kmeans on $k = 47$, since we have 47 communities in our ground-truth category data for the history sub-graph. We evaluate k-means by its F1 score, similar to the other models thus far.

4.1.3 BigClam

The next algorithm we use is BigClam, which is based on clustering nodes instead of clustering edges as we mentioned in the previous methods. BigClam is a tractable and more optimized version of AGM (Affiliation Graph Model) and relies on estimating parameters specified by the matrix composed of affinities of individual nodes to each community. It *guesses* the number of communities by adding edges with probabilities given by the affiliation matrix. We leverage this method by running the Snap implementation using different initializations. For example, in cases where we have a fixed number of communities to detect, (e.g. history by continent), we can leverage it to compute specific initialization values and see which can lead to better results.

4.2 Evaluation Metrics

To evaluate the algorithms we implement, we computed node-similarity within a community and also used the ground truth data provided by Wikispeedia. Specifically, Wikispeedia lists article-category pairs which we can use to group which articles fall under the same category (where we allow overlapping categories.) To evaluate the communities formed by our various algorithms, we used two scoring metrics:

1. **Tanimoto Coefficient (Similarity Score):** One way to evaluate a community distribution to see how similar nodes in a given community are and take the average similarity across all communities. To compute this similarity we use the Tanimoto coefficient between two nodes A,B in a given community, where A and B contain their links to other nodes in the graph:

$$TC(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

We apply this pairwise to all nodes in a given community, average all of them and then average these averages across all communities.

2. **F1 Score:** Since we have ground-truth category data of all the nodes in our data-set, we leverage these to evaluate our communities. Wikipedia categories follow a hierarchical structure: for example if one node has the category "British-History" and another has "British-History.19th-Century", it would appear that the two aren't in the same category, even though the latter is a subset of the former. Therefore when computing the F1 score we make split categories so that in this example the first node's category set is still "British-History" but the second's is ["British-History","19th-Century"]. After doing this, we go through every node in a predicted community X and label it by all its ground-truth categories. For that community we then find the most popular category, Y. We calculate precision as the percentage of nodes in X which were labeled Y by the algorithm:[5]

$$P = \frac{|X \cap Y|}{|X|}$$

And recall as the percentage of nodes in Y which are covered by x:

$$R = \frac{|X \cap Y|}{|Y|}$$

We then calculate the F1 of that community as:

$$F1 = 2 * \frac{PR}{P + R}$$

We average the F1-scores of all communities to calculate the overall F1 for the given algorithm.

5 Results and Analysis

As we hypothesized, modifying link clustering with natural language features has better community detection performance. The similarity score (Tanimoto Coefficient) is higher for all modified link clustering algorithms, whether it's augmenting with the TF-IDF cosine similarity of headers/all article text, or using the Unigram Jaccard Similarity (see Figure 1).

We were also interested in seeing how the link clustering algorithm would perform if we only used natural language features for the similarity score. Since the link clustering score only has information about the links, and Wikipedia is largely text-based, it's possible that using similarity from Cosine Similarity and Unigram Jaccard Similarity will be sufficient on its own in Ahn's edge clustering algorithm.

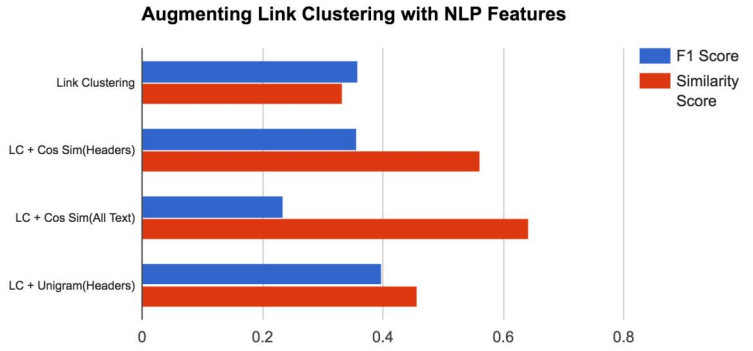


Figure 1: Modifying the link clustering algorithm with natural language features produces better performance. In particular, the Unigram Jaccard Similarity performs the best.

As we can see in Figure 2, link clustering with only natural language features performs better than when the natural language features are combined with the original link clustering score described in Ahn's paper. In particular, edge clustering with the cosine similarity on *all text* in Wikipedia articles performs the best, with an F1 score of 0.4385.

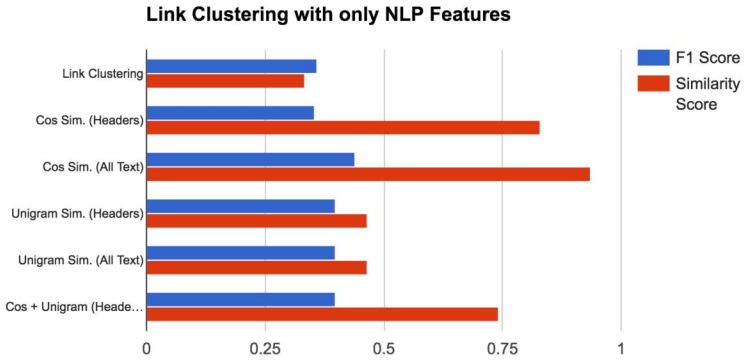


Figure 2: Link clustering with just the cosine similarity performs better than link clustering with the link clustering score. The F1 score for just cosine similarity is 0.4385, while the F1 score for link clustering on its own is 0.3592.

Below in Figure 3, we can see that BigClam has the best performance for a solely linked based community detection algorithm. Additionally, since K-means, which is solely based on language features performs the worst, this indicates that in order to best do community detection for the Wikipedia data-set we must combine both link and language features.

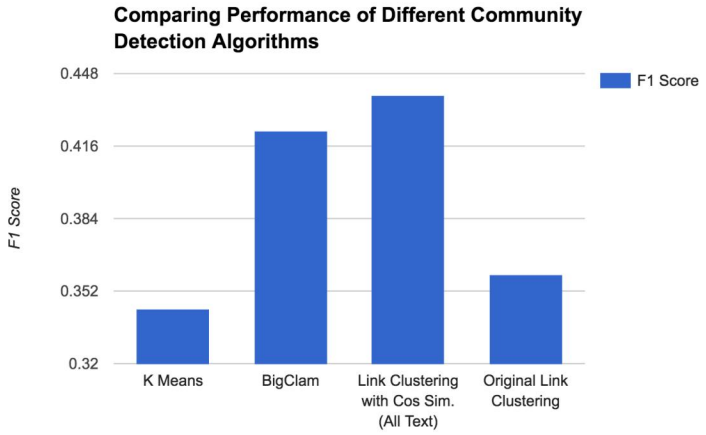


Figure 3: Here we compare the best models for each algorithm. We see that link clustering with cosine-similarity language features on the article text performs the best.

5.1 Error Analysis

In link clustering, there is no way to specify the number of desired communities, meaning that in addition to forming communities, the algorithm also determines the optimal number of communities to properly classify all nodes. Because of this, a lot of errors came from communities that contained only 2-3 nodes. We believe this is because the algorithm was too specific and grouped nodes together which might be a subset of a larger community. We now look at examples which had low F1 scores as compared against the ground truth category data from our best performing model (link clustering with cosine similarity language features):

Example 1

Community	Jorvik, 10th century
Ground-truth	Jorvik: *British History 1500 and before including Roman Britain*
	10th century: *General History*

In this example, we have falsely detected Jorvik and 10th century as a community. Taking a closer look, we can understand why our modified community detection algorithm with cosine similarity from all the text makes this error. Below, we see snippets from the text taken from the Jorvik article and the 10th century article. Theres actually a semantic similarity between the two, as Scandinavian York occurred during the period of the late 9th century and first half of the 10th century which overlaps with the idea and text 10th century.

Jorvik	Scandinavian York (also referred to as Jorvik) or Danish Norwegian York is a term used by historians for the south of Northumbria (modern day Yorkshire) during the period of the late 9th century and first half of the 10th century, when it was dominated by Norse warrior-kings; in particular, used to refer to the city (York) controlled by these kings.
10th century	The 10th century is the period from 901 to 1000 in accordance with the Julian calendar, and the last century of the 1st millennium.

In this case, we see how computing cosine similarity from all of the text in the Wikipedia article is more useful than computing it from the header text. In the 10th century article, the headers are split geographically, and include words like Africa, Americas and so on. In the Scandinavian York article, the headers are more sparse and include words like History and Kings of Jorvik.

This error comes from the ground-truth categories, as 10th century isnt a category, and there are no categories which both nodes share. For future, perhaps we can use another metric that doesnt rely on ground truth categories. We also see that the metric of similarity score, rather than F1 score, helps determine how good our community detection is, as our modified algorithm had an extremely high similarity score of 0.934435.

Example 2

Community 1:	1st, 2nd, 3rd, 4th, 5th, 6th, 7th, 8th century
Community 2:	13th century, 14th century

These pages were put in different communities, though since they are all centuries we would assume they should be put in the same History.General History community. We think the algorithm would have classified them this way because when we compare the topics talked about in these articles, they differ pretty widely. In the 1st-8th century the articles focus highly on civilizations in Asia and the Middle East, while the 13th and 14th century pages discuss the middle ages and Europe. The NLP features of our model must have picked up these differences and used them to split up these documents into different communities. This implies that our algorithm couldnt detect the overarching theme between all these pages, and also that perhaps the ground truth category might not be the best since the history of these two communities is quite different.

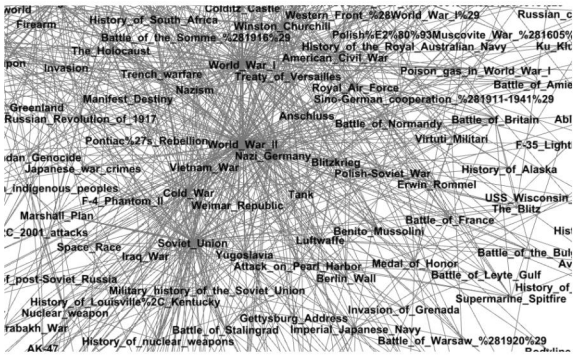


Figure 4: Magnified version of the network visualization (see next page for entire network). Notice that articles related to each of the World Wars form communities amongst each other.

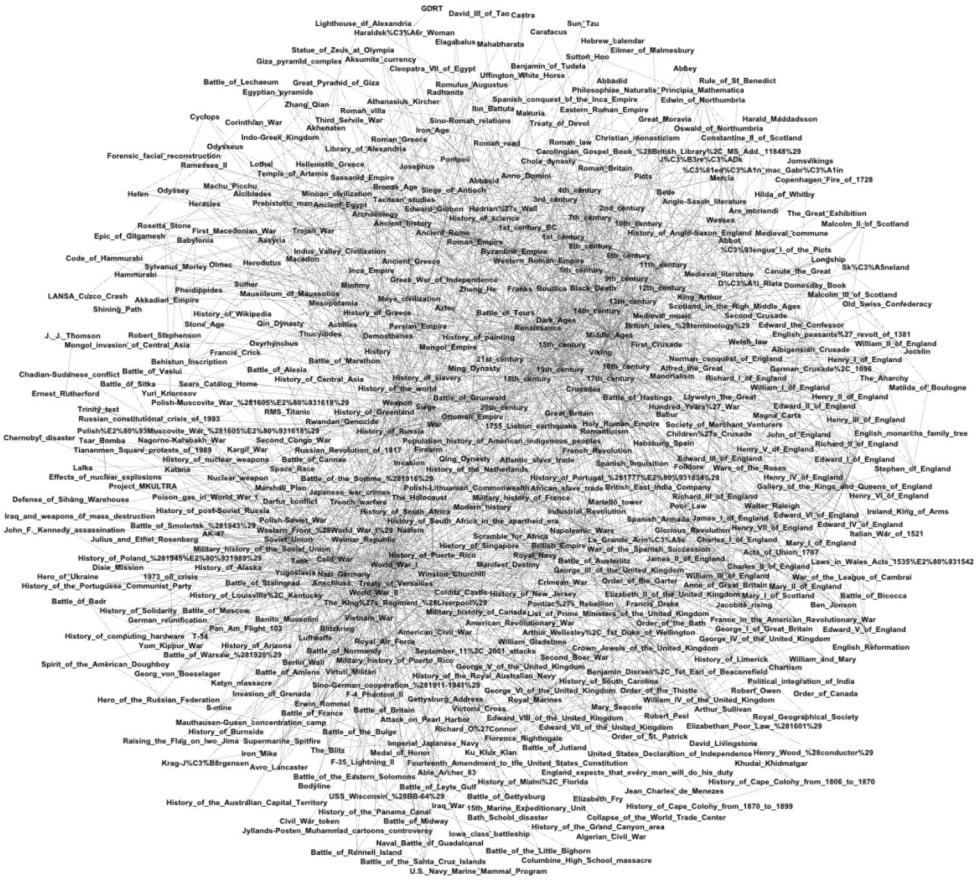


Figure 5: Visualization of the communities detected using the BigClam *algorithm. Darker areas of the network indicate a detection of community.*

6 Conclusion

Community detection is a difficult problem in Wikipedia, because some articles have more than one category, and thus form overlapping communities. In this paper, we've explored different methods of categorizing the Wikipedia articles, from BIGCLAM to edge clustering, and also trying out k-means clustering with word2vec. Taking advantage of the semantic uniqueness of Wikipedia, we incorporated natural language features by modifying the link clustering algorithm. Our experiments have shown that combining link clustering and semantic understanding improves community detection for the Wikipedia data-set. After evaluating both densely and sparsely overlapping community detection algorithms and using word embeddings with kmeans, we have shown that the performance of all algorithms is boosted when semantic features are added.

References

[1] Yong-Yeol Ahn, James P Bagrow, and Sune Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466(7307):761–764, 2010.

[2] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python.* " O'Reilly Media, Inc.", 2009.

[3] Aaron Clauset, Cristopher Moore, and Mark EJ Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101, 2008.

[4] Santo Fortunato and Darko Hric. Community detection in networks: A user guide. *Physics Reports*, 2016.

[5] Giulio Rossetti, Luca Pappalardo, and Salvatore Rinzivillo. A novel approach to evaluate community detection algorithms on ground truth. In *Complex Networks VII*, pages 133–144. Springer, 2016.

[6] Robert West and Jure Leskovec. Human wayfinding in information networks. In *Proceedings of the 21st international conference on World Wide Web*, pages 619–628. ACM, 2012.

[7] Robert West, Joelle Pineau, and Doina Precup. Wikispeedia: An online game for inferring semantic distances between concepts. In *IJCAI*, pages 1598–1603, 2009.

[8] Jaewon Yang and Jure Leskovec. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 587–596. ACM, 2013.