An Evaluation of Set Similarity Measures

by Team Jaccard

NIKOLA DANEVSKI, University of Rochester
PATRICK PHILLIPS, University of Rochester
JUSTIN FARGNOLI, University of Rochester
BENJAMIN CARLETON, University of Rochester

Abstract: Set similarity is a pervasive concept in mathematics and computer science which is applied extensively in the databases and data mining fields. There are multiple different measures of set similarity. In this paper we do comparative testing on the most notable and commonly used ones.

We perform our evaluation by first translating the problem of set similarity from its abstract set theory domain to a concrete graph theory problem and then exhaustively testing the most common set similarity measures. Our results show that using the Tversky index as a similarity measure yields the best results. This measure depends on two coefficients, α and β, and we find their optimal values to be 1.0 and 0.01, respectively.

Additional Key Words and Phrases: Set Similarity, Jaccard Index, Sørensen-Dice Coefficient, Tversky Index, Overlap Coefficient, Set Containment, Cosine Coefficient

1 INTRODUCTION

1.1 Motivation

Set similarity is a pervasive concept in mathematics and computer science. For example, set similarity is used in the field of databases to find similar attributes of relations to join on, in internet searching to compare the similarity of documents, and in genomics to find the similarity of genetic data sequences. Despite the prevalent use of set similarity, it is unclear which of the many different metrics is the "best" metric, or under what circumstance different metrics ought to be applied. Thus this paper sets out to develop and formalize a systematic way of evaluating set similarity metrics.

1.2 Problem Statement

A set similarity metric, denoted $S(X, Y)$ where X and Y are arbitrary sets, can be interpreted as the degree to which X is similar to Y, where X is the subject of the comparison and Y is the referent [9]. We pose the problem of comparing set similarity metrics. However, with quite literally an infinite number of different set similarity metrics, we limit our evaluation to the following handful of representative similarity metrics:

$$\textbf{Jaccard}(X,Y) \quad \frac{|X \cap Y|}{|X \cup Y|} \qquad \textbf{Sørensen}(X,Y) \quad \frac{2|X \cap Y|}{|X| + |Y|} \qquad \textbf{Ovelap}(X,Y) \quad \frac{|X \cap Y|}{\min(|X|, |Y|)}$$

$$\textbf{Cosine}(X,Y) \quad \frac{|X \cap Y|}{\sqrt{|X| \cdot |Y|}} \qquad \textbf{Set-Containment}(X,Y) \quad \frac{|X \cap Y|}{|X|} \qquad \textbf{Tversky}(X,Y) \quad \frac{|X \cap Y|}{|X \cap Y| + \alpha|X - Y| + \beta|Y - X|}$$

Comparing set similarities is challenging due to the nature of the problem. Each similarity metric maps two sets to a number ranging from 0 to 1. However, it is not the case that one can just compare these numbers and draw a conclusion. Even though all of the metrics measure the same concept of similarity, they fundamentally count different things. Thus we face the challenge of evaluating and comparing set similarity scores without actually directly comparing the output of one set similarity metric against the other.

Authors' addresses: Nikola Danevski, ndanevsk@u.rochester.edu, University of Rochester; Patrick Phillips, pphill10@u.rochester.edu, University of Rochester; Justin Fargnoli, jfargno2@u.rochester.edu, University of Rochester; Benjamin Carleton, bcarlet2@u.rochester.edu, University of Rochester.

1.3 Overview

We have outlined our motivation and our goal; we aim to compare the performance of various set similarity metrics. In Section 2 we formalize our method of evaluating set similarity metrics. Then, in Section 3, we dive into an evaluation of the performance of set similarity metrics. Next, in Section 4, we compare our results to that of related work while highlighting some other important features of set similarity metrics. Finally, in Section 5, we give a summary of our contributions, namely an overview of our novel framework for evaluating similarity metrics, as well as some key takeaways from the results we find in our data.

2 SOLUTION TECHNIQUE

2.1 Evaluating Metric Performance

It is certainly the case that one can artificially create tests in which one similarity metric can outperform the others. However, we wanted to test the metrics from a practical point of view. Thus, in order to perform our evaluation, we decided to use real-world datasets. This led us to the Stanford Network Analysis Project, SNAP. SNAP is a general purpose, high performance system for analysis and manipulation of large networks [5]. The project maintains a substantial collection of datasets, [4] with topics varying from social networks, citation networks, web graphs and so on. SNAP's library, originally written in C++, is the main tool we used to load these large datasets. It uses a graph representation of the datasets, and we decided to translate our problem into one with graphs. Namely, we consider the similarity of neighbor sets of vertices.

If nodes A and B are connected, and nodes A and C are not, then we would expect the similarity of the neighbors of A and B to be greater than that of A and C. (This intuitive assumption is a commonly used heuristic for link prediction [6]). Therefore, we say that a similarity metric performs well when it assigns a high value to the neighbor sets of two connected nodes, and when it assigns a low value to the neighbor sets of two unconnected nodes. We have thus reduced the problem of comparing set similarity metrics to the problem of evaluating their respective performance on the link prediction problem.

We now formally define the similarity of two nodes in a graph according to each of the metrics we are testing. In order to define similarity between nodes A and B, with neighbor sets X and Y, all we need to know is

- $|X|$
- $|Y|$
- $|X \cap Y|$

Given those, we can represent the other information needed to compute our set similarity metrics as follows:

- $|X \cup Y| = |X| + |Y| - |X \cap Y|$
- $|X - Y| = |X| - |X \cap Y|$
- $|Y - X| = |Y| - |X \cap Y|$

Thus, the needed information can be simply obtained as follows:

- $|X| = \text{count\_neighbors}(A)$
- $|Y| = \text{count\_neighbors}(B)$
- $|X \cap Y| = \text{count\_mutual\_neighbors}(A, B)$

With these definitions, we can now compute the similarity of two nodes in a graph according to the definitions in Section 1.2.

Now, the naive approach to coming up with a scoring system for set similarity metrics would be to simply compute the similarity between two nodes that are neighbors and score the metrics based on how large a value they return. This approach, however, would provide an inaccurate measure of the performance of similarity metrics, because again the numerical values of various metrics are not directly comparable. Concretely, if similarity metric S assigns a value of 0.5 to nodes A and B, and metric S' assigns a value of 0.4 to A and B, we may not necessarily conclude that S believes that A and B are more similar than S' does. We may only conclude that if S assigns a value of 0.4 to nodes A and C, then S believes that A and B are more similar than A and C.

So instead we come up with a different method. First, we call some arbitrary node A the query node, and pick two reference nodes B and C. We pick these reference nodes such that B is a neighbor of the query node, but C is not. We then compute the similarity between the set of neighbors of A and B and the set of neighbors of A and C. That is, we compute $S(\mathrm{n}(A), \mathrm{n}(B))$ and $S(\mathrm{n}(A), \mathrm{n}(C))$ using similarity metric S, where $\mathrm{n}(A)$ denotes the set of neighbors of A. If $S(\mathrm{n}(A), \mathrm{n}(B)) > S(\mathrm{n}(A), \mathrm{n}(C))$ we count that as a success for metric S, and otherwise as a failure. After performing many rounds of testing on many query and reference nodes, we can calculate the overall success rate of the metric.

2.2 Test Datasets

We compute the success rate of each of the similarity metrics on datasets collected from the SNAP database. We decided to do exhaustive testing through all the edges in each graph dataset in order to create easily repeatable results. We do so by iterating over all edges (A, B) of the graph, and comparing $S(\mathrm{n}(A), \mathrm{n}(B))$ with $S(\mathrm{n}(A), \mathrm{n}(C))$ where again C will be some node not connected to A. We test using every C such that the minimum-length path from A to C has length exactly two. This ensures that A and C are unconnected but that their neighbor sets are not disjoint.

The results from our experiment are discussed in Section 3. The datasets we test on from SNAP are described in the following table:

Name	**Nodes**	**Edges**	**Description**
ca-AstroPh	18,772	198,110	A collaboration network that covers scientific collaborations between authors papers submitted to Astro Physics category.
ca-CondMat	23,133	93,497	A collaboration network that covers scientific collaborations between authors papers submitted to Condensed Matter category.
ca-GrQc	5,242	14,496	A collaboration network that covers scientific collaborations between authors papers submitted to General Relativity and Quantum Cosmology category
ca-HepPh	12,008	118,521	A collaboration network that covers scientific collaborations between authors papers submitted to High Energy Physics—Phenomenology category.
ca-HepTh	9,877	25,998	A collaboration network that covers scientific collaborations between authors papers submitted to High Energy Physics—Theory category.
com-Amazon	334,863	925,872	Products frequently bought together via Amazon.
ego-Facebook	4,039	88,234	This dataset consists of “circles” (or “friends lists”) from Facebook.
musae-Facebook	22,470	171,002	This webgraph represents verified Facebook sites and links between them.

oregon1_010331	10,670	22,002	Autonomous Systems peering information inferred from Oregon route-views.
oregon1_010407	10,729	21,999	Autonomous Systems peering information inferred from Oregon route-views.
cit-HepPh	34,546	421,578	A citation network between papers submitted to High Energy Physics—Phenomenology category.
cit-HepTh	27,770	352,807	A citation network between papers submitted to High Energy Physics—Theory category.

Table 1. Test Datasets

3 EVALUATION

The Tversky index depends on coefficients α and β. In order to compare it properly to the other metrics, we first wanted to find the optimal values for α and β. We do this by searching through the space of coefficients $H(\alpha, \beta)$. To determine the bounds of our search space, we first performed a search over a broad range of values. The results of this broad search for the ca-GrQc dataset are shown in Figure 1. The results of the broad search indicated that the best results are generally

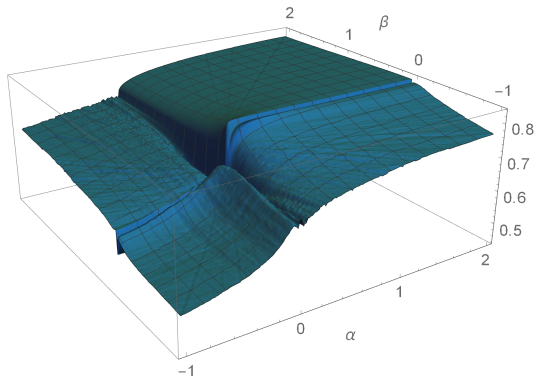


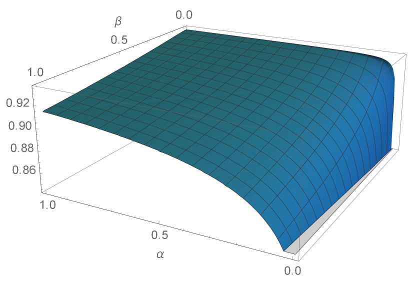
Fig. 1. Tversky Optimization: ca-GrQc

found in the space of coefficients satisfying $0 \leq \alpha, \beta \leq 1$. Hence we limited the search space to this range for future tests.

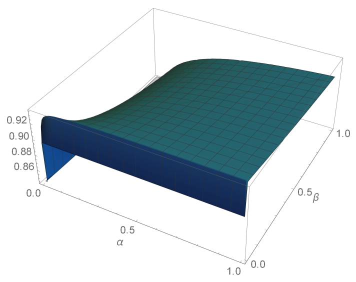
We performed exhaustive tests on eight datasets, varying the parameters in steps of 0.01. All datasets yielded similarly shaped success-rate curves. Figure 2 shows detailed views of one such example.

In more than half of the cases, the global maximum was achieved when $(\alpha, \beta) = (1.0, 0.01)$. Furthermore, all results show that the best results are achieved when $\alpha \gg \beta$. These results are summarized in Table 2. Notice that when $(\alpha, \beta) = (1.0, 0.0)$ then the Tversky index is equivalent to Set-Containment. Most curves show a steep drop-off in performance when $\beta = 0$. (See Figure 2b.) From this testing, we selected $(1.0, 0.01)$ as the optimal set of parameters.

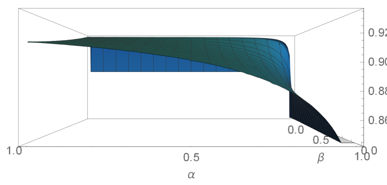
After obtaining the optimal Tversky index parameters, we tested these parameters against five other metrics, and one other set of Tversky index parameters. We computed the success rate for each of these seven metrics across twelve different datasets. To show relative performance, the results within each dataset were then standardized. The results are shown in Figure 3. The absolute measurements are also included in Appendix A.



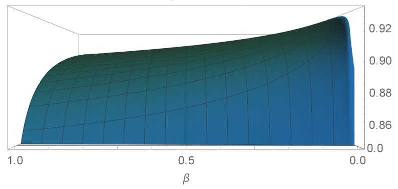
(a) Full view



(b) Drop-off at $\beta = 0$



(c) α-axis detail



(d) β-axis detail

Fig. 2. Tversky Optimization: ca-CondMat

Dataset(s)	**Maximizing Parameters** (α, β)
ca-CondMat, com-Amazon, oregon1_010331, cit-HepPh, cit-HepTh	$(1.0, 0.01)$
ca-GrQc	$(0.97, 0.02)$
ca-HepTh, ego-Facebook	Multiple (each satisfies $\alpha \gg \beta$, $\beta \neq 0$)

Table 2. Results of Tversky Optimization

The results from our experiment can be summarized as follows:

- The most important result from our analysis is that the Tversky index runs the best at its optimal point with $(\alpha, \beta) = (1.0, 0.01)$, and outperforms the other metrics. In addition, running the Tversky index with values $(\alpha, \beta) = (0.75, 0.25)$ produces, on average, the second best results out of the metrics we tested.
- The Jaccard index and Sørensen-Dice coefficient perform exactly the same in all test cases. We can show mathematically that this must be the case. For two arbitrary nodes, let j be the value of their Jaccard similarity and s be the value of their Sørensen similarity. Then some manipulation of the definitions shows that j and s are related by $s = 2j/(1+j)$. The function of j given by $2j/(1+j)$ is strictly increasing on the interval $[0, 1]$, so for any two pairs of nodes with Jaccard similarities j and j' respectively, the relative ordering of the corresponding Sørensen similarities s and s' will be the same.
- Set-Containment did not perform as well as the optimized Tversky index even though there is a very slight change in their definitions: the $0.01|Y - X|$ term in the denominator of the Tversky index. This result is an interesting and an unexpected one. A conclusion can be drawn that in asymmetric tests there are better measures than Set-Containment.
- The Overlap Coefficient performed the worst. This result is also not that surprising given its asymmetric definition. Even though it is not the only one that is asymmetric, it is the

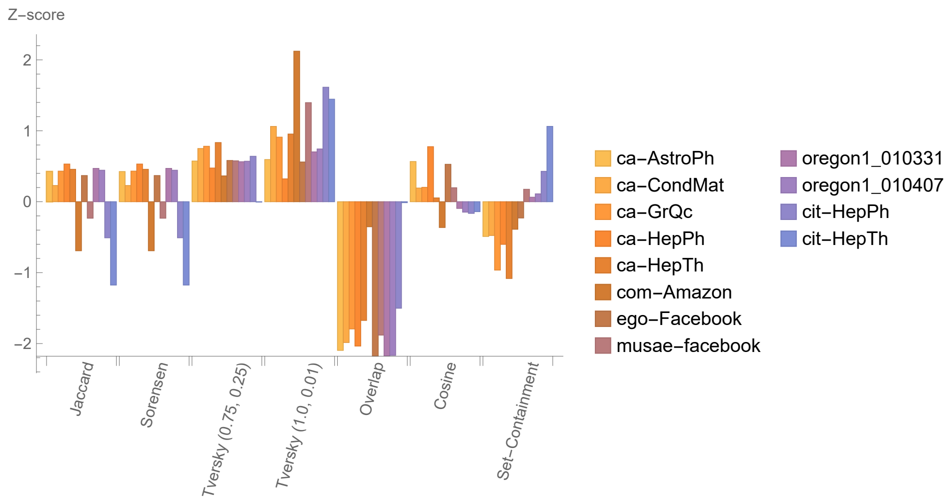


Fig. 3. Results of Comparative Testing

only one whose denominator can change according to query sets. This inconsistency is of great importance in our tests, because we always test one query node with two reference nodes, and thus, the node in the denominator can change. This is not the case for the other asymmetric metrics (Tversky and Set-Containment) and is the reason why Overlap indeed performs the worst.

4 RELATED WORK

There is a good deal of literature on both applications of set similarity and computation of set similarity [1], [2], [7], [10]. There is significantly less work pertaining to evaluating and comparing different set similarity metrics themselves as we do here, though there is some that we will look at [8], [9].

4.1 Tversky

Amos Tversky's seminal 1977 paper on similarity is the most prominent example of work that really looked into evaluating similarity metrics in and of themselves. Tversky came up with the widely accepted notion regarding set similarity metrics that "The value of $S(X, Y)$ can be interpreted as the degree to which X is similar to Y, where X is the subject of the comparison and Y is the referent." In this paper Tversky introduced two different general models of set similarity, the Tversky Index and the Tversky Contrast Model. The Tversky Index given by

$$S(X,Y) = \frac{|X \cap Y|}{|X \cap Y| + \alpha|X - Y| + \beta|Y - X|}$$

was looked at throughout this paper. The Tversky Contrast Model is given by

$$S(X,Y) = \gamma|X \cap Y| + \beta|X - Y| + \beta|Y - X|$$

where now $\alpha, \beta, and\ \gamma$ are parameters of this metric. One feature of similarity metrics that Tversky highlights is symmetry. A set similarity measure S is symmetric iff $S(X,Y) = S(Y,X)$. The two previous similarity metrics are symmetric if either $\alpha = \beta$ or if $|Y - X| = |X - Y|$; Tversky goes into some scenarios where you would or would not want a symmetric measure of similarity [9].

We have found that a clearly asymmetric measure (the Tversky Index with $\alpha = 1.0$ and $\beta = 0.01$) performs the best in our graph connection task.

4.2 Approximation of Set Similarity Metrics

Computing the intersection, union, and difference of sets are not particularly costly operations; all can be done in $O(n)$ time and space (or even faster [3]). Thus all of these set similarity metrics that consist of only these operations can also be found in $O(n)$. In practice however $O(n)$ is often not fast enough (a problem we ran into when testing our algorithm on larger datasets). A practical example of this is table joining. Imagine if you want to compute the similarity between *all pairs of columns* of a massive table with tens of thousands or even millions of entries per column. These exact methods become intractable, and we need a more efficient solution.

This brings us to the large body of literature on approximation of set similarity measures that we mentioned earlier. Most of the algorithms for quickly computing measures of set similarity rely on locality sensitive hashing in Hamming Space [2], [7], [10]. Hamming Space, in the case of sets, consists of all the vectors (or strings) $\{0, 1\}^d$ where each vector represents a particular set. Here d is the total number of possible unique elements across all sets and the bit 0 or 1 in any given vector indicates whether or not a given element is in the set represented by the vector. We won't go into depth with these approximation algorithms, as they were not the focus of our research, however it is worthwhile to note that these similarity metrics are **not** all computable (or at least closely approximated) in the same amount of time.

4.3 Other Evaluations of Set Similarity Metrics

Methods other than ours have been devised to evaluate set similarity metrics. One such method was developed by Spertus, Sahami, and Buyukkokten. Their idea was to give a user some recommendation to a social network community based on a similarity metric, and then score the similarity metric according to whether or not the user actually clicked on the recommendation given. Their paper finds that the cosine coefficient is the most successful in this task [8].

Zhu et. al. also discuss set similarity metrics in reference to the application of table joining (choosing which columns are most similar to join tables on). In their paper they formalize their problem as having some query set that they compare against several other reference sets that represent potential columns to join a table on. They argue that Set Containment is the most appropriate set similarity metric in this application, as other set similarity metrics which consider the size of the reference set are biased towards smaller reference sets [10].

Now recall that we have found the set similarity metric that performs best in our task is the Tversky Index with $\alpha = 1.0$ and $\beta = 0.01$. As mentioned earlier, Set Containment is the Tversky Index with $\alpha = 1.0$ and $\beta = 0.0$. In light of the commentary of Zhu et. al., the superior performance of the Tversky Index begins to make more sense. More specifically, why would we want to consider the size of the set of neighbors of the reference node? In our task we do not have any reason to be biased towards reference nodes with a smaller set of neighbors; we are just considering whether or not there will be a link between the query node and reference node at all.

Let's make sense of this with an example. Suppose we have a social network graph where vertices are people and edges are friendships. Three of the people represented in this graph are Joe, John, and Jack, and their friends are given as:

Joe: Noah, Elizabeth, Hayley, Caitlin
John: Noah, Hayley
Jack: Noah, Elizabeth, Hayley, Michaela, Kevin, Owen, Caitlin, Lindsay, Robin, Eliza, Rivera, Laura

Let our query node be Joe, and our reference nodes be John and Jack. Then our task is to predict whether it is more likely that Joe and John or Joe and Jack will end up being friends using set similarity comparisons on these sets of friends. The Jaccard similarity for Joe and John's sets of friends is 0.33, and the Jaccard similarity for Joe and Jack's sets of friends is 0.25. Set Containment for Joe and John's sets of friends is 0.50, and Set Containment of Joe and Jack's sets of friends is 1.0. Thus we get different results; Jaccard predicts Joe and John are more likely to be friends and Set Containment predicts that Joe and Jack are more likely to be friends. What's going on is that, just as Zhu et. al. argued, Jaccard similarity is biased away from the larger set and towards the smaller set. In this example, and in our link prediction task, we clearly don't want this bias. In fact in this case, the large size of the set of Jack's friends is perhaps indicating he is popular or extroverted, either way making it even *more likely* he and Joe are friends. Generalizing this result, it makes sense that Jaccard similarity or other metrics that normalize by some function of the reference node's set of neighbors performed poorly, and moreover that the Tversky index performed best with β close to 0.

5 SUMMARY

In this paper we first related the problem of comparing set similarity metrics to the problem of evaluating performance on the link prediction task. We then measured the success rate of commonly used set similarity metrics in this task. We found that there were clear winners and losers. The Overlap Coefficient consistently yielded the worst success rate, and we had that for $\alpha = 1.0$ and $\beta = 0.01$ the Tversky Index produced the highest success rates among all of the set similarity metrics as shown in Figure 3. Finally, we compared these results to those of other evaluations of set similarity metrics.

6 ACKNOWLEDGEMENTS

We would like to first thank Professor Fatemeh Nargesian for the background she gave us on the problem of set similarity and its application to the field of databases. We would also like to thank those who created and contribute to the SNAP project as it greatly eased the technical challenges of carrying out our research. Finally, we would like to thank Professors Chen Ding and Randal Nelson for teaching CSC 200H and providing us and other students with the opportunity to gain experience with computer science research and collaborative work in general.

REFERENCES

[1] Moses S Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*. 380–388.

[2] Tobias Christiani and Rasmus Pagh. 2017. Set similarity search beyond minhash. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. 1094–1107.

[3] Bolin Ding and Arnd Christian König. 2011. Fast set intersection in memory. *Proceedings of the VLDB Endowment* 4, 4 (2011), 255–266.

[4] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.

[5] Jure Leskovec and Rok Sosič. 2016. SNAP: A General-Purpose Network Analysis and Graph-Mining Library. *ACM Transactions on Intelligent Systems and Technology (TIST)* 8, 1 (2016), 1.

[6] David Liben-Nowell and Jon Kleinberg. 2003. The Link Prediction Problem for Social Networks. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management* (New Orleans, LA, USA) *(CIKM '03)*. Association for Computing Machinery, New York, NY, USA, 556–559. https://doi.org/10.1145/956863.956972

[7] Jianbin Qin, Yaoshu Wang, Chuan Xiao, Wei Wang, Xuemin Lin, and Yoshiharu Ishikawa. 2018. GPH: Similarity search in hamming space. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 29–40.

[8] Ellen Spertus, Mehran Sahami, and Orkut Buyukkokten. 2005. Evaluating similarity measures: a large-scale study in the orkut social network. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. 678–684.

[9] Amos Tversky. 1977. Features of similarity. *Psychological review* 84, 4 (1977), 327.
[10] Erkang Zhu, Fatemeh Nargesian, Ken Q Pu, and Renée J Miller. 2016. LSH ensemble: Internet-scale domain search. *arXiv preprint arXiv:1603.07410* (2016).

A ABSOLUTE MEASUREMENTS

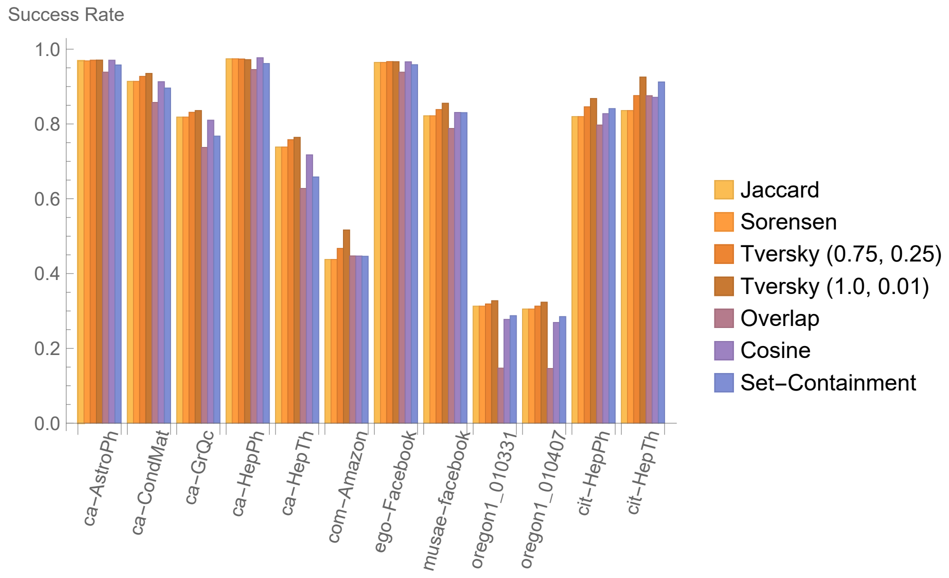


Fig. 4. Measured Success Rates, Grouped by Dataset