Hypergraph Network Models: Learning, Prediction, and Representation in the Presence of Higher-Order Relations

A THESIS

SUBMITTED FOR THE DEGREE OF

Doctor of Philosophy

IN THE

Faculty of Engineering

BY

Govind Sharma

Computer Science and Automation
Indian Institute of Science
Bangalore – 560 012
Karnataka (INDIA)

December, 2020

This page has intentionally been left blank.

Declaration of Originality

I, **Govind Sharma**, with SR No. **04-04-00-10-12-12-1-09805** hereby declare that the material presented in the thesis titled

Hypergraph Network Models: Learning, Prediction, and Representation in the Presence of Higher-Order Relations

represents original work carried out by me in the **Department of Computer Science and Automation** at **Indian Institute of Science, Bangalore, India** during the years **2012–2020**.
With my signature, I certify that:

- I have not manipulated any of the data or results.
- I have not committed any plagiarism of intellectual property. I have clearly indicated and referenced the contributions of others.
- I have explicitly acknowledged all collaborative research and discussions.
- I have understood that any false claim will result in severe disciplinary action.
- I have understood that the work may be screened for any form of academic misconduct.

Date: Student Signature

In my capacity as supervisor of the above-mentioned work, I certify that the above statements are true to the best of my knowledge, and I have carried out due diligence to ensure the originality of the report.

Advisor Name: Advisor Signature

This page has intentionally been left blank.

This page has intentionally been left blank.

DEDICATED TO

My sister Shweta

Acknowledgements

Please bear with me if this section sounds philosophical; a "tl;dr" version would be simply the bold-faced names of people. I feel it is mandatory to acknowledge people who have contributed towards the possibility of this thesis, hence making the arduous journey joyful. For it would be unfair to claim that this thesis is my sole effort – which is also not true. Even if we ignore the ethical reasons, there is a much higher purpose of this "public display of gratitude" as well: *there are not enough examples of gratitude in this world*, which could be punishing. It is necessary for gratitude to be expressed explicitly, and the expression to reach the person/people concerned. As I complete my thesis, the world is in a lock-down mode due to a pandemic, and lots of lives are getting lost every day. We might have heard about the article that enlists "top 10 things dying people have regrets about", but have we thought enough about *what regrets living people have about people they have lost?* – it's the opportunity to make amends, say thanks, and express their thoughts about them while they still could. Moreover, one act of expressing gratitude might make a huge difference to the people concerned, and might keep them happy and motivated.

With that prologue, let me start with my academic experiences, influences, and support. First and foremost, I'd like to thank my guide **Prof. M. Narasimha Murty** (Professor, CSA, IISc), who with his knowledge in machine learning, genuine concern as a research guide, and inherent urge to *uncover truth* – something he calls the main goal of any research exercise – has enabled the end-to-end evolution of work done in this thesis. I am equally thankful to him for maintaining patience and a high energy environment despite the numerous pieces of our experiments and discussions that could not see light of the day. Our laboratory – Topic Analysis and Synthesis Lab (TASL) – has been my second home, and as every PhD student does, I have had interactions with generations of graduate students. I have met them, have worked alongside them reading papers and designing experiments, have spent days and nights with them on failing code/experiments, have celebrated successes with them, and finally, have bidden them goodbyes, followed by a new, similar cycle — I thank **all present and past members of TASL**. Few people who had stayed for long, and have been my, loosely speaking, "partners-in-crime" are **Dr. Sharad Nandanwar** (Applied Scientist, A9.com), **Dr. Madhav Ram Nimishakavi** (Machine Learning Engineer, Facebook AI), and **Mr. Sambaran Bandyopadhyay** (PhD candidate, CSA, IISc,

and Advisory Research Engineer, IBM IRL).

As far as the research in this thesis is concerned, I can't thank enough my current colleagues at CSA: **Mr. Prasanna Patil**, **Mr. Paarth Gupta**, **Mr. Swyam Prakash Singh** (Master of Engineering students, CSA, IISc), **Dr. Aditya Challa** (Postdoctoral Researcher, CSA, IISc), and **Dr. V. Susheela Devi** (Principal Research Scientist, CSA, IISc). The area of hyperlink prediction had been largely unexplored, and with the help of Prasanna, we were able to devise solutions to three problems therein. With the help of Paarth and Aditya we were able to tame the hugely abstract idea about the "effect of hypergraphs in link prediction", and push the boundaries of link prediction further than they currently are. We were able to put together our ideas on various aspects of bipartite hypergraphs with huge efforts from Swyam's and Dr. Susheela's side. I have been fortunate to have friends in the academic fraternity at large and Computer Science in specific. Their role in listening, criticising, and suggesting ideas – mostly bombarding me with *constructive* criticism – has been great. I owe a great deal to Sharad, Madhav, and **Mr. Kartik Asooja** (Senior Data Scientist, UnitedHealth Group) for their consistent help.

I cannot go without mentioning faculty members at IISc – some for their courses, some for advise/motivation, and some for both. My learning through all the courses I undertook — something IISc calls the Research Training Program, but I have gained more than that from them — defines me today as a researcher. The very first core courses on Information Retrieval by Prof. MNM and Probabilistic Graphical Models by **Dr. Indrajit Bhattacharya** (Senior Scientist, TCS Innovation Labs) made me taste the true flavours of the sweet-sounding AI. The most interesting courses were the ones on Natural Language Understanding and Cognition & Machine Intelligence by **Prof. C. E. Veni Madhavan** (Professor, CSA, IISc), and Discrete Structures by **Prof. Dilip P. Patil** (Professor, Mathematics, IISc) – the former ones for their exciting experiments on real text corpora and on an eye-tracker machine, and the latter one for its sheer role in enhancing my mathematical reading, writing, and pedagogical skills. In addition to that, Linear Algebra and Real & Functional Analysis by **Prof. Vittal R. Rao** (Professor, IISc), Probability & Statistics by **Prof. Ambedkar Dukkipati** (Associate Professor, CSA, IISc) and Dr. Indrajit, and Computational Methods of Optimization by **Prof. Chiranjib Bhattacharyya** (Professor, CSA, IISc) were some of the first courses I had credited, which built a strong academic foundation for machine learning. An unusual course (for me) was on Real Analysis taught by **Prof. Gautam Bharali** (Professor, Mathematics, IISc), which questioned my very understanding of Mathematics, and introduced me to the world of rigor and abstraction.

Professors who have been monumental in keeping my PhD journey motivating are: **Prof. Matthew Jacob** (Professor, CSA, IISC), **Prof. K. Gopinath** (Professor, CSA, IISC), **Prof. Vijay Natarajan** (Professor, CSA, IISC), **Prof. Aditya Kanade** (Professor, CSA, IISC), **Prof. Shirish Shevade** (Professor, CSA, IISC), and **Prof. N. Viswanadham** (INSA Senior Scientist, CSA, IISC). I would

like to express my gratitude towards **Prof. Anurag Kumar** (Professor, ECE, IISc, and Director, IISc), **Prof. Y. N. Srikant** (Professor, CSA, IISc, and Dean of Engineering, IISc), **Prof. Shalabh Bhatnagar** (Professor and Chairman, CSA, IISc), and the members of various evaluation committees for my research **Prof. Joy Kuri** (Professor, DESE, IISc), **Prof. Rajesh Sundaresan** (Professor, ECE, IISc), **Prof. P. S. Sastry** (Professor, EE, IISc), **Prof. A. G. Ramakrishnan** (Professor, EE, IISc), **Prof. Y. Narahari** (Professor, CSA, IISc, and Chair, EECS Division, IISc), and **Prof. Jayant Haritsa** (Professor, CSA, IISc) for their role in taking the time to review, evaluate, and eventually supporting me in completing my PhD thesis.

On the personal side, there is seemingly no need to thank my family and friends, for any amount of gratitude cannot encapsulate their contributions. Nevertheless, it is my duty to verbalise my thoughts for them since they are a huge part of this endeavour. A PhD for sure, is a tiring journey that at times becomes demotivating. To add further, it also tests the researcher to her/his limits, whose effects could be observed on the personal life as well. My parents, **Mr. Kishore Sharma** & **Mrs. Shashi Sharma** and **all my family members (both paternal and maternal side)** have played a huge role in – for the lack of a better word – *supporting* me, by not only absorbing the side-effects of a PhD pursuer, but also rebuilding the much-needed strength to pick oneself up and keep working. My wife **Dr. Nidhi Sharma** (including her parents **Mr. Kamal Kishore Sharma** & **Mrs. Aruna Sharma** and **the rest of her family**) has been standing by me for all the times I remember; she has been proofreading my drafts and scripts, showing me the light after every paper rejection, and acting as a shock-absorber. My family's support has been enormous, and had kept me sane during the development of this piece of research.

Some of my best friends – **Aayush Gupta**, **Abhimanyu Rathore**, **Kartik Asooja**, **Gopal Sharma**, **Nishant Tripathi**, **Harish Sharma**, **Poonam Tripathi**, **Saurav Mittra**, **Gaurangi Sharma**, **Saurav Joshi**, **Pooja Mittra**, **Jitendra Sharma**, and **Syed Saud** – have literally held my hand (over the phone) and kept me from falling in times of distress! The amount of time they have spent on me and the pieces of advise they have bestowed upon me is priceless. Last but not least, the **administrative staff of IISc in general and CSA in specific** have been strenuously working towards providing a smooth environment for research.

Abstract

The very thought about "relating" objects makes us assume the relation would be "pairwise", and not of a "higher-order" — involving possibly more than two of them at a time. Yet in reality, *higher-order relations do exist* and are spread across multiple domains: medical science (*e.g.*, co-existing diseases/symptoms), pharmacology (*e.g.*, reacting chemicals), bibliometrics (*e.g.*, collaborating researchers), the film industry (*e.g.*, cast/crew), human resource (*e.g.*, a team), social sciences (*e.g.*, negotiating/conflicting nations), and so on. Since a collection of intersecting higher-order relations *lose context when represented by a graph*, "hypergraphs" – graph-like structures that allow edges (called "hyperedges"/"hyperlinks") spanning possibly more than two nodes – capture them better. In a quest to better understand such relations, in this thesis we focus on solving a few network-science oriented problems involving hypergraphs.

In the first of three broad parts, we study the *behavior of usual graph-oriented networks that have an otherwise-ignored hypergraph underpinning*. We particularly establish the *skewness a hypergraph introduces into its induced graphs*, and the effect of these biases on the structure and evaluation of the well-known problem of link prediction in networks. We find that an underlying hypergraph structure makes popular heuristics such as common-neighbors *overestimate their ability to predict links*. Gathering enough evidence – both theoretical and empirical – to support the need to reestablish the evaluations of link prediction algorithms on hypergraph-derived networks, we propose *adjustments that essentially undo the undesired effects* of hypergraphs in performance scores. Motivated by this observation, we *extend graph-based structural node similarity measures to cater to hypergraphs* (although still, for similarity between pairs of nodes). To be specific, we first establish mathematical transformations that could transfer any graph-structure-based notion of similarity between node pairs to a hypergraph-structure-based one. Using exhaustive combinations of the newly established scores with the existing ones, we could *show improvements in the performance of both structural as well as temporal link prediction*.

For the second part of our thesis, we turn our attention towards a more central problem in hypergraphs — the "hyperlink/hyperedge prediction" problem. It simply refers to developing models to predict the occurrence of missing or future hyperedges. We first study the effect of "negative sampling"

(sampling the negative class) – an exercise performed due to the extreme intractability of the set of all non-hyperlinks, also known as the class imbalance problem – on hyperlink prediction, which has never been studied in the past. Since we observe hyperlink prediction algorithms *performing differently under different negative sampling techniques*, our experiments help the seemingly unimportant procedure gain some significance. Moreover, we contribute towards *two benchmark negative sampling algorithms* that would help standardize the step. Moving on from the negative sampling problem to predicting hyperlinks themselves, we work on two different approaches: a "clique-closure" based approach, and a "sub-higher-order" oriented one. While in the former, we develop and successfully test the *clique-closure hypothesis – that hyperlinks mostly form from cliques or near-cliques* – and are able to utilize it for *hyperlink prediction via matrix completion* (C3MM), the latter approach works on a novel *information flow* model in hypergraphs. More precisely, we introduce the concept of "sub-hyperedges" to capture the sub-higher-order notion in relations, and utilize an *attention-based neural network model called SHONeN* focusing on sub-hyperedges of a hyperedge. Owing to SHONeN's computational complexity, we propose a *sub-optimal heuristic* that is able to perform better than its baselines on the downstream task of predicting hyperedges.

The third and final part of our thesis is dedicated exclusively to "bipartite hypergraphs": structures that are used to capture higher-order relations between two disjoint node sets, *e.g.*, *a patient's diagnosis* (possibly multiple diseases and symptoms), *a movie project* (multiple actors and crew members), *etc*. We first capture the structure of real-world such networks using "per-fixed" bipartite hypergraphs (those where the set of left and right hyperedges is fixed beforehand), and then focus on the "bipartite hyperlink prediction" problem. Since existing self-attention based approaches meant for usual hypergraphs *do not work for bipartite hypergraphs* — a fact that our experiments validate, we propose a "cross-attention" model for the same, and use the notion of set-matching over collections of sets to solve for bipartite hyperlink prediction. As a result, we develop a *neural network architecture called CATSETMAT* that performs way better than any of the other approaches meant to solve the bipartite hyperlink prediction problem. Last but not least, we also explain how, owing to an observation we call the "positive-negative dilemma", existing state-of-the-art algorithms fail on bipartite hypergraphs.

Publications based on this Thesis

Published / Accepted for publication

1. Govind Sharma, Prasanna Patil, and M Narasimha Murty. C3MM: Clique-Closure based Hyperlink Prediction. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI)*. AAAI Press, 2020.

2. Prasanna Patil, Govind Sharma, and M. Narasimha Murty. Negative Sampling for Hyperlink Prediction in Networks. In Hady W. Lauw, et al., editors, *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pages 607–619, Cham, 2020. Springer International Publishing.

Preprints / Under review

1. Govind Sharma, Prasanna Patil, and M Narasimha Murty. SHONeNs: Sub-higher-order Neural Networks for Hyperedges. (Preprint for an upcoming conference).

2. Govind Sharma, Aditya Challa, Paarth Gupta, and M Narasimha Murty. Higher Order Relations Skew Link Prediction in Graphs. (Preprint for an upcoming conference).

3. Govind Sharma, Swyam Singh, V Susheela Devi, and M Narasimha Murty. The CAT SET on the MAT: Cross Attention for Set Matching in Bipartite Hypergraphs. (Preprint for an upcoming conference).

4. Govind Sharma, Paarth Gupta, and M Narasimha Murty. Love tHy Neighbour: Remeasuring Local Structural Node Similarity in Hypergraph-Derived Networks. (Preprint for an upcoming conference).

This page has intentionally been left blank.

Contents

Chapter 1

Introduction

"More is not necessarily better. Better is better." ∼ Julie Newmar

MORE *is better* – a saying we have all heard is true, but in most contexts including computer science, *more is expensive* is as true. If we settle on the proposition that *more is both better as well as expensive*, we could start our journey with *relations of a higher order*. First things first, by "higher-order relations", we do not refer to relations over relations (as Armstrong [6] does), but to relations involving possibly multiple entities (popularly known as higher-degree, higher-arity, higher-rank, or higher-adicity relations).[1]

1.1 Motivation

Broadly speaking, the first step in a data analysis pipeline is that of *data representation* – we borrow tools called *vectors* from the field of linear algebra to represent individual data points. The data representation step focuses on an individual data point X and identifies it in isolation, by, say an n-dimensional vector $\boldsymbol{x} \in \mathbb{R}^n$. Once every single data point is satisfactorily represented in isolation, we move to exploring the *relation* between two of them. An example of a relation concerning two data points X and Y represented by vectors $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{y} \in \mathbb{R}^n$ respectively, is based on the cosine similarity $\frac{\boldsymbol{x}^\top \boldsymbol{y}}{|\boldsymbol{x}||\boldsymbol{y}|}$ between them. While cosine similarity is numerical, relations could also be Boolean in nature, *e.g.*, those connecting two data points by answering the question "Are two data points related?" by a simple "Yes" or "No". Moreover, relations between data points could also be in a "may be" state, when not explicitly related. We discuss this point in detail in this section.

[1] Though we have many options to use, we stick to the qualifier "higher order" to be consistent with the concept of *higher-order learning* propagated by Agarwal et al. [3].

1.1.1 The Dyadic Relation Assumption (DRA)

Nevertheless, the choice of relating *two* data points (and not more) owes itself to the sheer convenience – both philosophical [71] as well as mathematical – it provides to the researcher. In other words, it is customary to postfix the phrase "relation between" with something of the form "two items" or "items A and B". Moreover, even if one says "relation between items of set S" or "relation over set S", it generally is understood as a pairwise relation defined over pairs of items $\mathsf{A}, \mathsf{B} \in S$. For example, "a set S of siblings" is simply interpreted as "$\mathsf{A}, \mathsf{B} \in S \implies$ A and B are siblings." We call this the "*dyadic relation assumption (DRA)*".

While DRA is not problematic for most relations (namely friends, relatives, colleagues, *etc.*), a substantial variety of useful relations are meant to be kept as those being of a higher order. For instance, in some networks pertaining to biological sciences (protein interaction, disease comorbidity, drug abuse warning, *etc.*), we see relations adopting a higher order. Although whether to view them as pairwise relations is still a matter of choice, such an assumption could be hugely lossy, since some pairs of entities may not even be related directly. Take a protein interaction network [25], wherein proteins that are known have a physical contact mutually are connected using a higher-order relation.

Let $\mathcal{F}$ denote the set of all such observed/known protein interactions (*i.e.*, relations), and pick a specific relation $F := \{\mathsf{P}, \mathsf{Q}, \mathsf{R}, \mathsf{S}\} \in \mathcal{F}$ that relates four proteins that are known to be interacting. In essence, the mere existence of the relation F in $\mathcal{F}$ implies that proteins $\mathsf{P}, \mathsf{Q}, \mathsf{R}$, and S are known to mutually interact. However, going by DRA, we get a set $\mathcal{P}_2(F)$[1] of protein pairs, defined by $\mathcal{P}_2(F) := \{F' \subseteq F : |F'| = 2\} = \{\{\mathsf{P}, \mathsf{Q}\}, \{\mathsf{P}, \mathsf{R}\}, \{\mathsf{P}, \mathsf{S}\}, \{\mathsf{Q}, \mathsf{R}\}, \{\mathsf{Q}, \mathsf{S}\}, \{\mathsf{R}, \mathsf{S}\}\}$. Now, DRA says "$\{\mathsf{A}, \mathsf{B}\} \in \mathcal{P}_2(F) \implies$ A and B interact with each other." But this might not necessarily be true for all pairs, unless all of them occur as individual relations in the original collection $\mathcal{F}$ of relations (just as F does). In other words, it could very well be true that, say, proteins Q and R cannot interact in the absence of P – a fact that would mean the protein interaction $\{\mathsf{Q}, \mathsf{R}\}$ "does not belong to" $\mathcal{F}$. Hence, the erroneous assumption that all pairs of proteins interact is not true in reality. This shows how by merely observing an interacting quadruplet $\{\mathsf{P}, \mathsf{Q}, \mathsf{R}, \mathsf{S}\}$, *the dyadic relation assumption (DRA) can tag some non-interacting protein pairs as interacting*. It could therefore be stated that the dyadic relation assumption (DRA) could be potentially misleading in real-world applications.

1.1.2 The Gestalt Viewpoint

We have discussed how the dyadic relation assumption (DRA) distorts the view a higher-order relation holds, by simply connecting all pairs of involved nodes with their individual relations. But it still does not satisfactorily answer the question: "Why not simply be done with the higher-order perspective

[1] Let $\mathcal{P}(S)$ denote the ***powerset*** (set of all subsets), and $\mathcal{P}_2(S)$, the ***k-powerset*** (all k-sized subsets) of S.

and accept the DRA view outright"? In other words, what could go wrong if a higher-order relation is replaced by multiple pairwise ones, as per DRA? After all, the status of multiple entities being connected still remains intact in the form of pairs of connections. Let us take a different example before discussing this further.

Imagine there is a political organization called BAKERSTREET JOINT PARTY containing members SHERLOCK HOLMES, JOHN WATSON, and MYCROFT HOLMES. Now, since SHERLOCK, WATSON, and MYCROFT are members of the same party, it implies that the trio is *mutually compatible*. However, would it be right to deduce(!) that they are *pairwise compatible* as well (a fact demanded by DRA)? Or more specifically, should one assume from the three-order relation $\{$SHERLOCK, WATSON, MYCROFT$\}$ that SHERLOCK and MYCROFT gel well with each other? Not necessarily; let us suppose that they have conflicting ideologies[1]. But if that is the case, then why on earth would they join the same party? The answer lies in WATSON, who acts as the "compatibility enhancer" between the two. In other words, the trio would have formed a party mainly because of WATSON, and so while the pairs $\{$SHERLOCK, WATSON$\}$, $\{$MYCROFT, WATSON$\}$ may be valid relations, the pair $\{$SHERLOCK, MYCROFT$\}$ is not.

Our aim with this example is not to reestablish the shortcomings of DRA (since we have already done that in the foregoing section), but to illustrate an important concept called the *Gestalt psychology* [58, 29, 53]: "*The whole is more than the sum of its parts*." That between SHERLOCK, WATSON, and MYCROFT, the principles governing their *triadic* and the *dyadic* interactions are not the same (and more so, not derivable from each other), owes itself to the Gestalt principle. Restating this point more concretely, we have that "The higher-order relation $F = \{$SHERLOCK, WATSON, MYCROFT$\}$ (*i.e.*, the *whole*) cannot be replaced merely by the set of its pairwise combinations $\mathcal{P}_2(F) = \{\{$SHERLOCK, WATSON$\}$, $\{$WATSON, MYCROFT$\}$, $\{$SHERLOCK, MYCROFT$\}\}$ (*i.e.*, the sum of its *parts*), but has more to it." This is true mainly since the kind of political underpinnings involved in the functioning of a group could be more than just pairwise interactions of its members. Since the Gestalt principle – although not so popularly discussed these days – applies to multiple domains of machine learning [75, 116, 129, 138, 87], it motivates us in our pursuit of understanding and utilizing higher-order relations better.

1.1.3 Where DRA holds good

The Dyadic Relation Assumption (DRA), however, is not all that bad. This is so since for most practical purposes, it is the only tractable technique. It is true that it distorts the real picture, but the extent of distortion varies from dataset to dataset and application to application. For example, the ill-effects of clique expansion would be worse in bigger hyperedges (say of size 10) rather than in

[1] Fans of the works of fiction on Sherlock Holmes by Arthur Conan Doyle [27] would *know* that this is actually true!

smaller ones (of size 3–4). Also, for a hypergraph clustering application, clique expansion seems good enough [10, 106, 65]. After all, retaining a higher-order structure only caters to subtle changes as compared to following DRA.

1.2 Scope of the Thesis

In the current thesis, we have worked towards expanding the horizons of network science to cater to higher-order relations. To capture such relations conveniently, we use combinatorial structures called "hypergraphs" [13, 14, 18]. For the unintroduced reader, a hypergraph is exactly like a graph, only it allows edges involving an arbitrary number of vertices (as opposed to just two). In other words, given a set V of vertices, while a graph collects pairs (*i.e.*, 2-sized subsets) of them as *edges* $\mathcal{E} \subseteq \mathcal{P}_2(V) := \{V' \in 2^V : |V| = 2\}$, a hypergraph is composed of arbitrary-sized subsets of vertices as *hyperedges* $\mathcal{F} \subseteq \mathcal{P}(V) := 2^V$. To represent the protein interaction data discussed in Section 1.1.1, if V denotes the set of proteins, then the set $\mathcal{F}$ ($\subseteq \mathcal{P}(V)$) of protein interactions (*i.e.* hyperedges) would include subsets of proteins that are known to interact. Particularly, the set $\mathcal{F}$ would contain the hyperedge $F = \{\mathsf{P}, \mathsf{Q}, \mathsf{R}, \mathsf{S}\}$ as well, that captures the interaction between the four proteins $\mathsf{P}, \mathsf{Q}, \mathsf{R}, \mathsf{S} \in V$. Apart from hypergraphs, other higher-order paradigms such as Galois structures, simplicial complexes, *etc.* have also been used in the literature [48] for complex systems. Though the literature on hypergraphs talks about various varieties thereof, we restrict ourselves to undirected, homogeneous, and unweighted hypergraphs, both unipartite and bipartite (see Chapter 2 for more details). We divide our thesis into three broad parts: *hypergraphs and pairwise links* (Chapters 3–4), *predicting higher-order relations* (Chapters 5–7), and *higher-order bipartite relations* (Chapter 8). Let us briefly introduce them one by one.

1.2.1 Hypergraphs and Pairwise Links

In practical scenarios, it is customary to convert a hypergraph network into a graph via *clique expansion* [3], *i.e.* replacing each hyperedge by a clique over its containing nodes. For example, a co-authorship network is treated as a graph, wherein two collaborating authors form an edge. The same is true for a number of other networks (protein interaction networks, co-actor networks, co-citation networks, etc.), where, going by DRA, it is taken for granted that every relation would have two entities; we call such relations "pairwise links". Moreover, even though most networks (excepting "pure" pairwise ones such as friend-friend social networks) have an underlying higher-order structure, they are seldom treated as so, and are instead converted into graphs at the very beginning of an analysis pipeline itself. Conclusively, the higher-order structure of these networks is ignored at large, and they are dealt with using mere graphs.

We take knowledge of such networks' underlying higher-order structure and for the first time,

study the latter's effect on the subsequent graph-based network analysis in Chapter 3. As a result, we make some surprising observations regarding neighborhood-based link prediction techniques in these networks. For one, *the underlying higher-order structure of hypergraph networks meddles with the predictability of links via well-known link prediction heuristics*. Motivated by the effect of hyperedges on link prediction, in Chapter 4, *we exploit the higher-order neighborhood structure directly to improve existing node similarity heuristics and show improvements in link prediction scores*. In addition, we also introduce mathematical formulation to convert graph based node similarity scores to hypergraph based ones. In summary, in this part, we have merely touched the field of hypergraphs, and have focused on their effect on usual graph-oriented networks.

1.2.2 Predicting Higher-order Relations

Predicting the missing/future occurrence of a higher-order relation between a group of entities (also referred to as the *hyperedge/hyperlink prediction problem*[1] in the literature [121, 132, 11, 133]) is as important as the well-known link prediction problem in networks [66]. For example, questions such as the following:

- "Will four workers W_1, W_2, W_3, W_4 in a corporation be part of the same email in future"?
- "Will three authors A_1, A_2, A_3 write a paper together eventually"?
- "Is a combination of three medicines M_1, M_2, M_3 (but not a pair of them) harmful when prescribed together"?
- "Will four movie actors A_1, A_2, A_3, A_4 work in the same movie"?

are of prime importance in their respective domains. However, the extent of predictability of a missing or a non-existent future higher-order relation (or hyperedge) might vary from domain to domain, and from dataset to dataset. Existing techniques to solve the hyperedge prediction problem also deal with hypergraph embedding (latent-space embedding of nodes and/or hyperedges) as a by-product.

We first *extensively discuss the effect of negative sampling on hyperlink prediction* in Chapter 5, since the problem is usually posed as a binary classification problem wherein hyperedges form the positive class and non-hyperedges the negative class. Further, we also *provide novel techniques to sample the negative class* and hence show how a seemingly insignificant pre-experimental negative sampling step could make huge differences in the evaluation of prediction algorithms in the context of hypergraphs. Then, in Chapters 6 and 7, we provide a couple of solutions – one based on matrix completion, and the other a neural-network based one – to the hyperlink prediction problem. The

[1] We use terms "hyperlink prediction" and "hyperedge prediction" alternatively for higher-order relation prediction.

first approach (the one in Chapter 6) provides a *matrix completion based hyperedge prediction model called C3MM*, where we define and statistically establish an important hypothesis — that *hyperedges are mainly formed due to closures of open cliques*. For the second technique (of Chapter 7), we propose a novel paradigm for *information flow between nodes and hyperedges of a hypergraph via "node-to-subhyperedge-to-hyperedge"*, which we show is more comprehensive than the usual node-to-hyperedge assumption. This results in a deep learning architecture called *Sub-Higher-Order Neural Networks (SHONeNs)*, which uses a sub-hyperedge-based attention framework, thereby *surpassing the performance of the current state-of-the-art in hyperedge prediction (viz., Hyper-SAGNN [133]) and other baselines by significant margins*.

1.2.3 Higher-order Bipartite Relations

Whether a hypergraph is but a fancy reinvention-of-the-wheel or not, is a dilemmatic debate[1]. Mathematically speaking, every hypergraph $(V, \mathcal{F})$ (where V is a set of vertices and $\mathcal{F} \subseteq \mathcal{P}(V)$ is a collection of hyperedges over them) could also be modeled as a bipartite graph $(V, V', \mathcal{E})$ (a graph with two *disjoint* sets $V \cup V'$ of vertices, wherein only cross-edges from $V \times V'$ are allowed). It is so since if we take $V := V$ and $V' := \mathcal{F}$ (dummy nodes) to be the disjoint vertex sets needed to form a bipartite graph, we get the set of bipartite edges as $\mathcal{E} := \{(v, F) \in V \times \mathcal{F} \mid v \in F\}$, thereby forming the corresponding bipartite graph $(V \cup \mathcal{F}, \mathcal{E})$. Such bipartite graph representations of vertices and higher-order relations over them are known as *Levi graphs* [22, 117] (more informally, *star expansions* [3] of hypergraphs). Moreover, since hypergraphs and their corresponding bipartite graph versions are inter-derivable, it could easily be claimed that star expansions losslessly represent higher-order relations, something that clique expansions (of hypergraphs into pairwise-relation based graphs via DRA) cannot. From the Gestalt perspective, the dummy vertices in the star expansion of a hypergraph could viewed as *reifications* [63] of hyperedges. Nevertheless, the fact that is clear from this discussion is that the hypergraph paradigm could be considered a tool as good as bipartite graphs, to capture higher-order relations.

However, this should not be the case since in the real world, there exist "*higher-order bipartite relations*" as well. For example, "A collection of patients' diagnoses involving (possibly multiple) diseases and (possibly multiple) symptoms", or "A collection of movie credits involving both cast and crew members", *etc*., constitute some higher-order bipartite relations. Had bipartite graphs (and not hypergraphs) been used to denote higher-order relations, what would one resort to for those higher-order relations that are themselves bipartite in the first place? A careful thought would reveal that to capture such relations using graphs, special tripartite structures would be needed. Moreover, modeling the prediction of such relations would also be tedious. We therefore model higher-order

[1] See, for example, this discussion on Mathematics Stack Exchange [57].

bipartite relations using "bipartite hypergraphs" [140, 5] (those wherein each hyperedge is required to contain at least one node from each node set[1]).

In Chapter 8, we first *identify the bipartite hyperedge prediction problem for a special kind of bipartite hypergraph: "per-fixed bipartite hypergraph"* (one where we fix left and right hyperedges throughout the learning paradigm). Then we propose a solution by learning a set-matching model for set-of-sets. In the process, we argue that how prediction models that treat self- and cross-links using the same attention parameters fail to discriminate between a positive class pattern (a bipartite hyperedge) and a negative class one (a bipartite non-hyperedge). Eventually, *we propose a cross-attention model for hyperedge prediction in such hypergraphs, which hugely outperforms state-of-the-art algorithms and other baselines*. The apparent failure in discriminating between two classes when self- and cross-attention parameters are shared, is due to a phenomenon we call the "positive-negative dilemma", which we discuss in detail in Chapter 8.

1.3 The Logical Flow of the Thesis

The realm of relations is vast, a fact that we depict in the sankey diagram in Figure 1.1. In other words, the width of the top of the block (labeled RELATIONS) represents all existing concepts pertaining to relations. It could be bifurcated broadly into HIGHER-ORDER, and TWO-ORDER RELATIONS, as labeled in the diagram. While the proportion of areas allocated to higher- and two-order relations could be inaccurate, we know for a fact that the former spans a broader and deeper set of concepts than the latter. Chapters 5–8 exclusively deal with higher-order relations, where on the other hand, Chapters 3 and 4 comprise of almost equal proportions of both higher-order as well as two-order relations. As seen in the flow diagram, concepts from higher-order relations, flow into four different tributaries, which have little-to-nothing to do with pairwise relations. They have been described as follows:

- *Bipartite Hypergraphs*: We spend an ample amount of interest to deal with bipartite hypergraphs, which forms the whole of Chapter 8.

- "*Out of Scope*": The width of this block (Out of Scope) could be debated upon, and we do not claim authority over it. But it sure denotes a huge portion from the area of higher-order relations that we ***do not*** cover in this thesis.

  For example, simple concepts such as *community detection in hypergraphs*, *node centrality in hypergraphs*, *node classification in hypergraphs*, *more applications of hypergraphs*, *directed*

[1]Since a bipartite hypergraph is so scarcely studied in the literature, a single definition does not exist – a fact that can be verified from a couple of references [140, 5] that define it in their own way. However, we insist on the "at least one node from each node set" requirement for every hyperedge, and provide a formal definition in Chapter 2.

hypergraphs, *multi-way data analysis*, *tensor analysis for uniform hypergraphs*, *etc*. to more complicated ones such as *heterogeneous hypergraphs*, *multi-hypergraphs*, *bipartite bihypergraphs*, *hypergraph coloring*, and many more concepts involving discrete mathematical structures are beyond the scope of this thesis.

- *Hyperlink Prediction*: A huge portion of concepts from higher-order relations flow into the domain of HYPERLINK PREDICTION — modeling and predicting the occurrence of higher-order relations. We further divide it into negative sampling (Chapter 5), C3MM (Chapter 6), and SHONeNs (Chapter 7), all of which exclusively deal with predicting hyperlinks.

- The fourth tributary has not been labelled; basically, it indicates the flow of concepts from

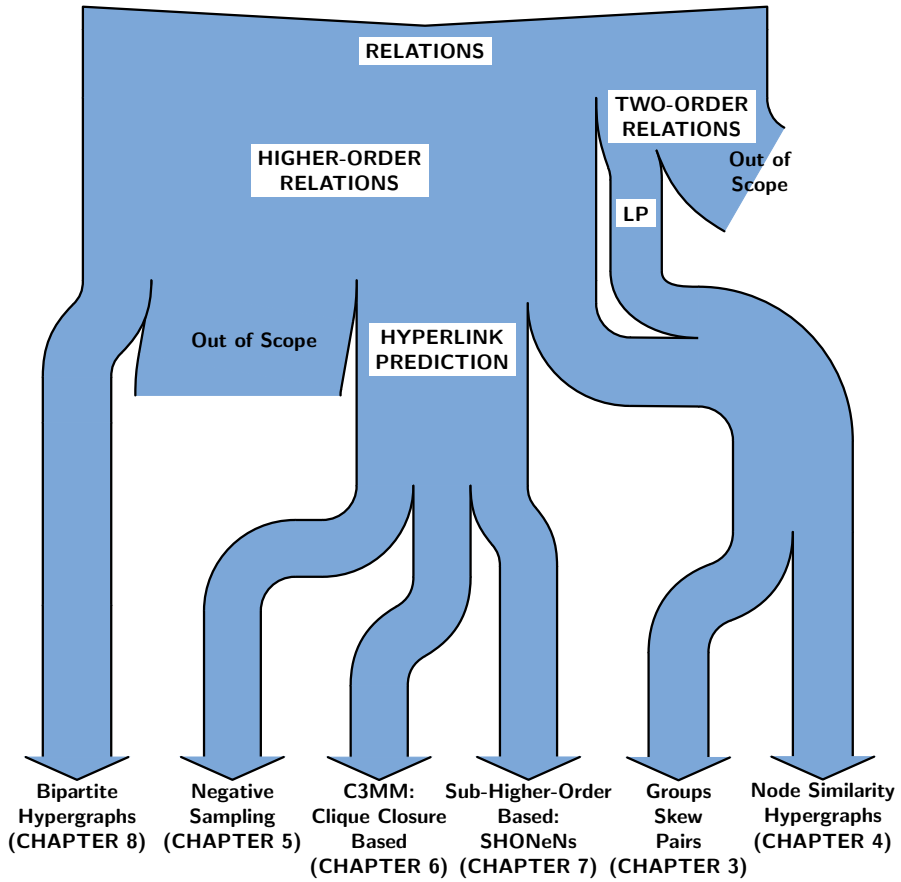


Figure 1.1: A sankey diagram showing the logical flow of concepts in this thesis. The flow has been extensively explained in the running text, with the two "Out of Scope" blocks uncovered into a few example topics in Section 1.3. Please note that in actuality, the flow diagram is more comprehensive. For example, neither do we cover *all* bipartite hypergraphs, nor *all* topics in hyperlink prediction, something which the diagram says we do. We take liberty in not over-representing such details in this diagram.

higher-order relations to pairwise ones, coming from the two-order relations (right) part.

The last two pipeline flows indicate the involvement of higher-order relations in explaining the working of two-order ones, of which, only link prediction (LP) has been focused on. More specifically, in Chapters 3 and 4, we study the effect of underlying hyperedges in the well-known problem of link prediction involving pairwise links. A huge portion of two-order relations that involve merely graphs is out of the scope of this thesis (denoted by Out of Scope on the right), covering topics such as *node centrality, node classification, graph algorithms, community detection, graph embedding, etc*.

1.4 Research Contributions

1. **Hypergraphs and Pairwise Links**

   1 (a) Elucidate both theoretically as well as empirically, that underlying ***higher-order relations have a dramatic effect on their pairwise counterparts***, in that evaluation on the latter becomes biased; also, provide ***solutions to undo the effect by re-basing evaluation criteria***.

   1 (b) Formulate a mechanism to ***utilize higher-order neighborhood to aid node similarity heuristics, thereby grounding high-level concepts and performing extensive experiments to support the thesis***; furthermore, exhibit ***novel data preparation techniques for fairness***.

2. **Predicting Higher-order Relations**

   2 (a) Emphasize the importance and analyze the ***effect of negative sampling in higher-order relation prediction***, contributing also towards ***two highly useful algorithms*** for the same.

   2 (b) ***Propose and rigorously test a clique-closure based hypothesis***; ***then utilize it for predicting higher-order relations***, thereby improving the state-of-the-art.

   2 (c) Establish a novel ***sub-higher-order paradigm for information flow in hypergraphs***, ***formulate a sub-optimal heuristic*** for the same, and finally, further ***improve higher-order relation prediction via a deep learning model based on sub-hyperedges***.

3. **Higher-order Bipartite Relations**

   3 (a) For the first time, ***introduce higher-order bipartite relations in network science, introduce datasets for higher-order bipartite relations and propose a cross-attention based relation prediction model*** which works way better than its competitors.

   3 (b) Uncover and reason for the ***shortcomings in usual models when positive and negative classes share features – a trait we observe in prediction of bipartite higher-order relations***.

Table 1.1: Notations used in this thesis

Symbol	Definition
$\mathcal{P}(\cdot)$, $\mathcal{P}_k(\cdot)$	Power set and k-power set of a set
$<_{slex}$	Shortlex order for subsets of a set
$V = \{v_1, \ldots, v_{\lvert V\rvert}\}$	Set of (left) nodes
$\mathcal{E} = \{e_1, \ldots, e_{\lvert\mathcal{E}\rvert}\} \subseteq \mathcal{P}_2(V)$	Set of edges
$\mathcal{G} = (V, \mathcal{E})$	Graph
$\mathcal{G} = (V, \mathcal{E}, w)$	Weighted graph
$\mathcal{G} = (V, \mathcal{E}, \tau)$	Temporal graph
$\mathcal{F} = \{F_1, \ldots, F_{\lvert\mathcal{F}\rvert}\} \subseteq \mathcal{P}(V)$	Set of hyperedges
$\mathcal{H} = (V, \mathcal{F})$	Hypergraph
$\mathcal{H} = (V, \mathcal{F}, \tau)$	Temporal hypergraph
$\boldsymbol{A} \in \mathbb{R}^{\lvert V\rvert \times \lvert V\rvert}$	Adjacency matrix
$\boldsymbol{S} \in \mathbb{R}^{\lvert V\rvert \times \lvert\mathcal{F}\rvert}$	Incidence matrix of hypergraph
$\Gamma : V \to \mathcal{P}(V)$	Set of neighbors of a node
$d : V \to \mathbb{R}$	Degree of a node
$\tilde{\Gamma} : V \to \mathcal{P}(\mathcal{F})$	Set of hyperneighbors of a node
$\tilde{d} : V \to \mathbb{R}$	Hyperdegree of a node
$\eta(\mathcal{H})$, $\eta^*(\mathcal{H})$	Clique and star expansions of hypergraph $\mathcal{H}$
$V' = \{v'_1, \ldots, v'_{\lvert V'\rvert}\}$	Set of right nodes
$\mathcal{B}_{sim} = \{B_1, \ldots, B_{\lvert\mathcal{B}_{sim}\rvert}\} \subseteq \mathcal{P}(V \cup V')$	Set of simple bipartite hyperedges
$\mathcal{H}_{sim} = (V \cup V', \mathcal{B}_{sim})$	Simple bipartite hypergraph
ρ_{tr}, ρ_{otr}	Train and observed split-ratio
ν	Negative-to-positive ratio
T, T_{tr}, T_{te}	Timeline, train-period, test-period
T_{otr}, T_{utr}	Observed/Unobserved train periods
$\mathcal{E}_{tr}, \mathcal{E}_{te}$	Train and test edges
$\mathcal{E}_{otr}, \mathcal{E}_{utr}$	Observed/Unobserved train edges
$\hat{\mathcal{E}}, \hat{\mathcal{E}}_{te}, \hat{\mathcal{E}}_{utr}$	Total, test, and unobserved train non-edges
$\mathbb{D}_{unsup} = (\mathcal{G}_{tr}, \mathcal{E}_{te}, \hat{\mathcal{E}}_{te})$	LP dataset for unsupervised learning
$\mathbb{D}_{sup} = (\mathcal{G}_{otr}, \mathcal{E}_{utr}, \hat{\mathcal{E}}_{utr}, \mathcal{E}_{te}, \hat{\mathcal{E}}_{te})$	LP dataset for supervised learning
$\pi : \mathcal{E}_{te} \cup \hat{\mathcal{E}}_{te}$	Unsupervised link predictor
$\pi_\theta : \mathcal{E}_{te} \cup \hat{\mathcal{E}}_{te}$	Supervised link predictor with parameters θ
$\mathcal{F}_{tr}, \mathcal{F}_{utr}, \hat{\mathcal{F}}_{utr}, \mathcal{F}_{te}, \hat{\mathcal{F}}_{te}$	Train/test hyperedges/non-hyperedges
$\tilde{\mathbb{D}}_{unsup} = (\mathcal{H}_{tr}, \mathcal{F}_{te}, \hat{\mathcal{F}}_{te})$	HLP dataset for unsupervised learning
$\tilde{\mathbb{D}}_{sup} = (\mathcal{H}_{otr}, \mathcal{F}_{utr}, \hat{\mathcal{F}}_{utr}, \mathcal{F}_{te}, \hat{\mathcal{F}}_{te})$	HLP dataset for supervised learning
$\tilde{\pi} : \mathcal{F}_{te} \cup \hat{\mathcal{F}}_{te}$	Unsupervised hyperlink predictor
$\tilde{\pi}_\theta : \mathcal{F}_{te} \cup \hat{\mathcal{F}}_{te}$	Supervised hyperlink predictor with parameters θ

Chapter 2

Background Review

> *"The mathematical concept of a set can be used as the foundation for all known mathematics. ... What may be surprising is not so much that sets may occur as elements, but that for mathematical purposes no other elements need ever be considered."*
>
> ∼ Paul Richard Halmos

HIGHER-ORDER relations have not been studied in the literature as aggressively as pairwise relations, and so, mere intuitive notions might hinder our progress. Moreover, there are more than one mathematical tools to ground the concept of higher-order relations, and their usage varies from one piece of work to another. For example, while Xu et al. [121], Tu et al. [107], Zhang et al. [133], *etc.* use hypergraphs, Benson et al. use simplicial complexes in one of their works [11], and sequences-of-sets in another [12]. We in this thesis, use "hypergraphs" as a tool to capture higher-order relations and use the terms "hypergraph network", "higher-order network" almost exchangeably. Similar is our handling of the terms "higher-order relation" and "hyperedge", both of which refer to a simultaneous connection between multiple objects. In this chapter, we introduce and fix certain definitions and notations used in the whole thesis. However, wherever definitions and notations different from what we introduce here arise in a particular chapter, we explicitly mention them in the chapter. For one, while the symbol "V" denotes the set of vertices in the whole thesis, its elements could be denoted by either "$\{v_1, v_2, \ldots, v_n\}$" or by "$\{1, 2, \ldots, n\}$" – whichever seems convenient. Nevertheless, such minor change of notation would be explicitly mentioned in the respective chapters at appropriate places. More specific notations – that involving core concepts of our research – have already been summarized in Table 1.1.

2.1 Generic Concepts

For a set S, we denote the collection of all of its subsets – its ***power set*** – by $\mathcal{P}(S)$, and the collection of its k-sized subsets by $\mathcal{P}_k(S)$. We also define two forms of the ***indicator function***: one for a logical

statement S, and the other for a set S, as follows:

$$\mathbb{1}(\mathtt{S}) = \begin{cases} 0, & \text{if S is FALSE} \\ 1, & \text{if S is TRUE} \end{cases} \qquad \mathbb{1}_S(x) = \begin{cases} 0, & \text{if } x \notin S \\ 1, & \text{if } x \in S \end{cases} \tag{2.1}$$

We also define a variant of the "Shortlex" order [100], as follows:

Definition 2.1 (***Shortlex Order***). *For a set X with elements indexed as $X = \{x_1, \ldots, x_n\}$, the **shortlex order** $<_{slex}$ on its powerset 2^X is defined as $(X_1, \ldots, X_{2^n})$, where for $X_i, X_j \subseteq X$, we have:*

1. *If $|X_i| \neq |X_j|$, then $X_i <_{slex} X_j \iff |X_i| < |X_j|$.*
2. *If $|X_i| = |X_j| = s$, and if $X_i := \{x_{i_1}, \ldots, x_{i_s}\}$ and $X_j = \{x_{j_1}, \ldots, x_{j_s}\}$, s.t. $1 \leq i_1 < \cdots < i_s \leq n$ and $1 \leq j_1 < \cdots < j_s \leq n$, then we have:*

$$X_i <_{slex} X_j \iff (i_1, \ldots, i_s) <_{lex} (j_1, \ldots, j_s),$$

where $<_{lex}$ is the lexicographic order.

2.2 Graphs and Hypergraphs

Given a set V of ***vertices*** or ***nodes***, a ***graph*** over it is defined as an ordered pair $\mathcal{G} := (V, \mathcal{E})$, where $\mathcal{E} \subseteq \mathcal{P}_2(V)$ denotes the collection of ***edges*** over V. Occasionally, a graph could be ***edge-weighted***; this is denoted by $\mathcal{G} = (V, \mathcal{E}, w)$, where $w : \mathcal{E} \to \mathbb{R}_+$ assigns a real-number weight to each edge. A graph is ***non-temporal*** by default, but if there exists a chronological order over its edges, it becomes a ***temporal*** graph, wherein time information is denoted using a mapping $\tau : \mathcal{E} \to \mathbb{R}_+$; a temporal graph is therefore a triplet $\mathcal{G} = (V, \mathcal{E}, \tau)$. Not to mention, a ***weighted temporal graph*** is a quadruplet $\mathcal{G} = (V, \mathcal{E}, w, \tau)$.

A ***hypergraph*** over V, on the other hand, is an ordered pair $\mathcal{H} = (V, \mathcal{F})$, where $\mathcal{F} \subseteq \mathcal{P}(V)$ denotes the collection of ***hyperedges*** over V. A similar temporal version – $\mathcal{H} = (V, \mathcal{F}, \tau)$ – could also be defined, where $\tau : \mathcal{F} \to \mathbb{R}_+$. When not specified explicitly, we have $V = \{v_1, v_2, \ldots, v_{|V|}\}$, $\mathcal{E} = \{e_1, e_2, \ldots, e_{|\mathcal{E}|}\}$, and $\mathcal{F} = \{F_1, F_2, \ldots, F_{|\mathcal{F}|}\}$ denoting nodes, edges, and hyperedges respectively.

Although the definitions of weighted/unweighted temporal/non-temporal graphs and hypergraphs is enough to *capture* relations, some other notions that help study properties thereof follow. The ***adjacency matrix*** $\boldsymbol{A} \in \mathbb{R}^{|V| \times |V|}$ of a graph is defined as $\boldsymbol{A}_{ij} = \mathbb{1}_{\mathcal{E}}(\{v_i, v_j\})$, and if the graph is weighted, a weighted version could also be defined, as $\boldsymbol{A}_{ij} = w(\{v_i, v_j\}) \cdot \mathbb{1}_{\mathcal{E}}(\{v_i, v_j\})$. While the adjacency matrix gives a lossless representation of a graph, for a hypergraph the same loses information about hyperedges. Hence, we use an ***incidence matrix*** $\boldsymbol{S} \in \mathbb{R}^{|V| \times |\mathcal{F}|}$ for it, defined by $\boldsymbol{S}_{ij} = \mathbb{1}_{F_j}(v_i)$, where rows and

columns correspond to vertices and hyperedges respectively. While the ideas of ***neighborhood*** and ***degree*** are straightforwardly defined for graphs, as $\Gamma : V \to \mathcal{P}(V)$, $v \mapsto \{u \in V \mid \{u, v\} \in \mathcal{E}\}$ and $d : V \to \mathbb{R}$, $v \mapsto |\Gamma(v)|$ respectively, for a hypergraph, these concepts ought be more comprehensive. The ***neighborhood*** for a hypergraph is defined as $\Gamma(v) := \{u \in V \mid \exists F \in \mathcal{F} \text{ s.t. } \{u, v\} \subseteq F\}$, and ***degree*** as in graphs. For both graphs as well as hypergraphs, neighbors of a node refer to the set of vertices adjacent to it, and its degree is simply the number of such neighbors. A couple of more neighborhood (rather, hyperneighborhood) notions: ***hyperneighbors*** and ***hyperdegree*** are also defined for a hypergraph, as $\tilde{\Gamma} : V \to \mathcal{P}(\mathcal{F})$, $v \mapsto \{F \in \mathcal{F} \mid v \in F\}$ and $\tilde{d} : V \to \mathbb{R}$, $v \mapsto |\tilde{\Gamma}(v)|$ respectively, that involve incident hyperedges and not adjacent vertices.

While equivalent notions such as Levi graphs had been defined by earlier literature as well Agarwal et al. [3] define many mechanisms to convert hypergraphs into graphs, two of which are ***clique expansion*** and ***star expansion*** of a hypergraph. Formally, they are defined as follows (please note that we use the symbol 'η' (*eta*) to stand for "expansion").

Definition 2.2 (***Clique Expansion***)**.** *The **clique expansion** graph $\eta(\mathcal{H})$ of a hypergraph $\mathcal{H} = (V, \mathcal{F})$ is defined as $\eta(\mathcal{H}) := (V, \eta(\mathcal{F}))$, where $\eta(\mathcal{F}) := \bigcup_{F \in \mathcal{F}} \eta(F)$, where for $F \in \mathcal{F}$, we have $\eta(F) := \mathcal{P}_2(F)$. In other words, the clique expansion of $\mathcal{H}$ is:*

$$\eta(\mathcal{H}) := (V, \eta(\mathcal{F})) := \left(V, \bigcup_{F \in \mathcal{F}} \eta(F)\right) := \left(V, \bigcup_{F \in \mathcal{F}} \mathcal{P}_2(F)\right), \tag{2.2}$$

which is a usual graph with two vertices being connected if and only if they have at least one incidence hyperedge in common.

Definition 2.3 (***Star Expansion***)**.** *The **star expansion** graph $\eta^*(\mathcal{H})$ of a hypergraph $\mathcal{H} = (V, \mathcal{F})$ is defined as a bipartite graph $\eta^*(\mathcal{H}) := (\mathcal{V}^*, \mathcal{E}^*)$, where $\mathcal{V}^* := V \cup \mathcal{F}$ and $\mathcal{E}^* := \{\{v, F\} \mid v \in V, F \in \mathcal{F} \text{ s.t. } v \in F\}$. In other words, the star expansion of $\mathcal{H}$ is:*

$$\eta^*(\mathcal{H}) := (\mathcal{V}^*, \mathcal{E}^*) := (V \cup \mathcal{F}, \{\{v, F\} \mid v \in V, F \in \mathcal{F} \text{ s.t. } v \in F\}), \tag{2.3}$$

which is a bipartite graph with the original vertices and hyperedges as left and right node-sets (see Section 2.6) respectively, and a vertex gets connected to a hyperedge if and only if the latter is incident on the former.

2.3 Node Similarity Heuristics

Here, we define certain popular node-similarity metrics for graphs, which are also used as link prediction heuristics [66, 41].

Definition 2.4 (***Common Neighbor (CN) Score***)**.**

$$CN(\{u,v\}) := |\Gamma(u) \cap \Gamma(v)|\,. \tag{2.4}$$

Definition 2.5 (***Jaccard Coefficient (JC) Score***)**.**

$$JC(\{u,v\}) := \begin{cases} 0 & \textit{if } d(u) = d(v) = 0 \\ \dfrac{|\Gamma(u) \cap \Gamma(v)|}{|\Gamma(u) \cup \Gamma(v)|} & \textit{if } d(u) > 0 \textit{ or } d(v) > 0. \end{cases} \tag{2.5}$$

Definition 2.6 (***Association Strength (AS) Score***)**.**

$$AS(\{u,v\}) := \begin{cases} 0 & \textit{if } d(u) = 0 \textit{ or } d(v) = 0 \\ \dfrac{|\Gamma(u) \cap \Gamma(v)|}{d(u) \cdot d(v)} & \textit{if } d(u) > 0 \textit{ and } d(v) > 0. \end{cases} \tag{2.6}$$

Definition 2.7 (***Cosine Similarity (CS) Score***)**.**

$$CS(\{u,v\}) := \begin{cases} 0 & \textit{if } d(u) = 0 \textit{ or } d(v) = 0 \\ \dfrac{|\Gamma(u) \cap \Gamma(v)|}{\sqrt{d(u) \cdot d(v)}} & \textit{if } d(u) > 0 \textit{ and } d(v) > 0. \end{cases} \tag{2.7}$$

Definition 2.8 (***NMeasure (NM) Score***)**.**

$$NM(\{u,v\}) := \begin{cases} 0 & \textit{if } d(u) = 0 \textit{ and } d(v) = 0 \\ \dfrac{|\Gamma(u) \cap \Gamma(v)|}{\sqrt{d(u)^2 + d(v)^2}} & \textit{if } d(u) > 0 \textit{ or } d(v) > 0. \end{cases} \tag{2.8}$$

Definition 2.9 (***MinOverlap (MnO) Score***)**.**

$$MnO(\{u,v\}) := \begin{cases} 0 & \textit{if } d(u) = 0 \textit{ or } d(v) = 0 \\ \dfrac{|\Gamma(u) \cap \Gamma(v)|}{\min\{d(u), d(v)\}} & \textit{if } d(u) > 0 \textit{ and } d(v) > 0. \end{cases} \tag{2.9}$$

Definition 2.10 (***MaxOverlap (MxO) Score***)**.**

$$MxO(\{u,v\}) := \begin{cases} 0 & \textit{if } d(u) = 0 \textit{ and } d(v) = 0 \\ \dfrac{|\Gamma(u) \cap \Gamma(v)|}{\max\{d(u), d(v)\}} & \textit{if } d(u) > 0 \textit{ or } d(v) > 0. \end{cases} \tag{2.10}$$

Definition 2.11 (***Preferential Attachment (PA) Score***)**.**

$$PA(\{u,v\}) := d(u) \cdot d(v). \tag{2.11}$$

Definition 2.12 (***Adamic/Adar (AA) Score***)**.**

$$AA(\{u,v\}) := \sum_{w \in \Gamma(u) \cap \Gamma(v)} \frac{1}{\log d(w)}. \tag{2.12}$$

Definition 2.13 (***Resource Allocation (RA) Score***)**.**

$$RA(\{u,v\}) := \sum_{w \in \Gamma(u) \cap \Gamma(v)} \frac{1}{d(w)}. \tag{2.13}$$

Definition 2.14 (***SimRank (SR) Score***)**.**

$$SR(\{u,v\};C) := \begin{cases} 1 & \textit{if } u = v \\ 0 & \textit{if } d(u) = 0 \textit{ or } d(v) = 0 \\ \dfrac{C}{d(u) \cdot d(v)} \displaystyle\sum_{\substack{u' \in \Gamma(u) \\ v' \in \Gamma(v)}} SR(\{u',v'\}) & \textit{if } u \neq v. \end{cases} \tag{2.14}$$

Definition 2.15 (***Pearson Correlation (PC) Score***)**.**

$$PC(\{u,v\}) := \begin{cases} 0 & \textit{if } d(u) = 0 \textit{ or } d(v) = 0 \\ \dfrac{|V| \cdot |\Gamma(u) \cap \Gamma(v)| - d(u) \cdot d(v)}{\sqrt{d(u) \cdot d(v) \cdot (|V| - d(u)) \cdot (|V| - d(v))}} & \textit{if } d(u) > 0 \textit{ and } d(v) > 0. \end{cases} \tag{2.15}$$

2.4 Link Prediction

2.4.1 Temporal Graphs

Given a temporal graph $\mathcal{G} = (V, \mathcal{E}, \tau)$, define its ***timeline*** as $T := \tau(\mathcal{E}) = \{\tau(e) \mid e \in \mathcal{E}\}$. Now, define a ***train split-ratio*** $\rho_{tr} \in [0,1]$ used to partition the timeline into train and test periods T_{tr} and T_{te} respectively in a way that $|T_{tr}| \approx \rho_{tr} \cdot |T|$ and $(t_{tr}, t_{te}) \in T_{tr} \times T_{te} \implies t_{tr} < t_{te}$. Next, the set $\mathcal{E}$ of edges is split into ***train edges*** $\mathcal{E}_{tr} = \{e \in \mathcal{E} \mid \tau(e) \in T_{tr}\}$ and ***test edges*** $\mathcal{E}_{te} = \{e \in \mathcal{E} \mid \tau(e) \in T_{te}\}$. Further, we split the train period into observed and unobserved train periods T_{otr} and T_{utr} respectively using another parameter ***observed split-ratio*** $\rho_{otr} \in [0,1]$, that partitions it in the same manner as above (*i.e.*, in a way that $|T_{otr}| \approx \rho_{otr} \cdot |T_{tr}|$ and $(t_o, t_u) \in T_{otr} \times T_{utr} \implies t_o < t_u$). This partitions

the train edges $\mathcal{E}_{tr}$ into ***observed train edges*** $\mathcal{E}_{otr} := \{e \in \mathcal{E}_{tr} \mid \tau(e) \in T_{otr}\}$ and ***unobserved train edges*** $\mathcal{E}_{utr} := \{e \in \mathcal{E}_{tr} \mid \tau(e) \in T_{utr}\}$. Hence, we have three different kinds of edges with us – observed train ($\mathcal{E}_{otr}$), unobserved train ($\mathcal{E}_{utr}$), and test ($\mathcal{E}_{te}$); not-to-mention, there is a set of train edges $\mathcal{E}_{tr} = \mathcal{E}_{otr} \cup \mathcal{E}_{utr}$ as well. We also define two graphs $\mathcal{G}_{otr} = (V, \mathcal{E}_{otr})$ and $\mathcal{G}_{tr} = (V, \mathcal{E}_{tr})$. In real-world networks, the non-edges – node pairs not connected directly by an edge – are much higher in number than edges, and need be sub-sampled for a practical link prediction exercise to be possible; we fix a ***negative-to-positive ratio*** $\nu \in \mathbb{N}$ that decides how many of them to sample. More specifically, the set of all available non-edges could be defined as $\hat{\mathcal{E}} := \mathcal{P}_2(V) \setminus \mathcal{E}$, of which $\nu \cdot |\mathcal{E}_{te}|$ could be randomly sampled as ***test non-edges*** $\hat{\mathcal{E}}_{te}$, and $\nu \cdot |\mathcal{E}_{utr}|$ of them as ***unobserved train non-edges*** $\hat{\mathcal{E}}_{utr}$.

Once we have these items with us, we can prepare two kinds of ***LP dataset*** "packages": (i) $\mathbb{D}_{unsup} := (\mathcal{G}_{tr}, \mathcal{E}_{te}, \hat{\mathcal{E}}_{te})$ for unsupervised LP (with no parameters to be learnt) and (ii) $\mathbb{D}_{sup} := (\mathcal{G}_{otr}, \mathcal{E}_{utr}, \hat{\mathcal{E}}_{utr}, \mathcal{E}_{te}, \hat{\mathcal{E}}_{te})$ for supervised LP. Now, we are in an ideal position to define the temporal link prediction problem.

2.4.2 Non-Temporal Graphs

Given a non-temporal graph $\mathcal{G} = (V, \mathcal{E})$, and the train split-ratio $\rho_{tr} \in [0, 1]$, *randomly* sample $\lceil \rho_{tr} \cdot |\mathcal{E}| \rceil$ of the edges from $\mathcal{E}$ as ***train edges*** $\mathcal{E}_{tr}$, and the remaining as ***test edges*** $\mathcal{E}_{te} := \mathcal{E} \setminus \mathcal{E}_{tr}$. With the help of another parameter called observed split-ratio $\rho_{otr} \in [0, 1]$, randomly sample the train edges into $\rho_{otr} \cdot |\mathcal{E}_{tr}|$ ***observed train edges*** $\mathcal{E}_{otr}$ and the rest are collected as ***unobserved train edges*** $\mathcal{E}_{utr} := \mathcal{E}_{tr} \setminus \mathcal{E}_{otr}$. Non-edges $\hat{\mathcal{E}}_{utr}$ and $\hat{\mathcal{E}}_{te}$ are sampled as before. Ultimately, we package ***LP datasets*** $\mathbb{D}_{unsup}$ and $\mathbb{D}_{sup}$ as before.

Definition 2.16 (***Unsupervised Link Prediction*** [66])**.** *Given LP dataset $\mathbb{D}_{unsup} = (\mathcal{G}_{tr}, \mathcal{E}_{te}, \hat{\mathcal{E}}_{te})$, the **unsupervised link prediction** problem is to develop a link predictor $\pi : \mathcal{E}_{te} \cup \hat{\mathcal{E}}_{te} \to \mathbb{R}$ that uses the train graph $\mathcal{G}_{tr}$ to compute LP scores $\pi(\{u, v\})$ for each pair $\{u, v\}$ in such a way that edges get higher scores than non-edges. That is,*

$$\max_{\pi \in \mathbb{R}^{\mathcal{E}_{te} \cup \hat{\mathcal{E}}_{te}}} Pr(\pi(\{u, v\}) > \pi(\{\hat{u}, \hat{v}\}) \mid \{u, v\} \in \mathcal{E}_{te}, \{\hat{u}, \hat{v}\} \in \hat{\mathcal{E}}_{te}) \tag{2.16}$$

Definition 2.17 (***Supervised Link Prediction*** [4])**.** *Given LP dataset $\mathbb{D}_{sup} = (\mathcal{G}_{otr}, \mathcal{E}_{utr}, \hat{\mathcal{E}}_{utr}, \mathcal{E}_{te}, \hat{\mathcal{E}}_{te})$, the **supervised link prediction** problem is to train a link prediction model $\pi_\theta : \mathcal{E}_{te} \cup \hat{\mathcal{E}}_{te} \to \mathbb{R}$ parameterized by θ using the observed train graph $\mathcal{G}_{otr}$ and training data $(\mathcal{E}_{utr}, \hat{\mathcal{E}}_{utr})$ such that each test edge gets a higher score than a test non-edge. That is,*

$$\max_{\pi_\theta \in \mathbb{R}^{\mathcal{E}_{te} \cup \hat{\mathcal{E}}_{te}}} Pr(\pi_\theta(\{u, v\}) > \pi_\theta(\{\hat{u}, \hat{v}\}) \mid \{u, v\} \in \mathcal{E}_{te}, \{\hat{u}, \hat{v}\} \in \hat{\mathcal{E}}_{te}) \tag{2.17}$$

Be it supervised or unsupervised, if the data is prepared from a temporal graph, we call the LP problem a ***temporal link prediction*** problem, and for a non-temporal graph, it is called a ***non-temporal/structural link prediction*** problem.

2.5 Hyperlink/Hyperedge Prediction

Given a temporal/non-temporal hypergraph, notions similar to the ones for link prediction (ref. Section 2.4) could be defined. Essentially, at the end of the data preparation pipeline, we get **hyperlink prediction datasets** $\tilde{\mathbb{D}}_{unsup} = (\mathcal{H}_{tr}, \mathcal{F}_{te}, \hat{\mathcal{F}}_{te})$ and $\tilde{\mathbb{D}}_{sup} = (\mathcal{H}_{otr}, \mathcal{F}_{utr}, \hat{\mathcal{F}}_{utr}, \mathcal{F}_{te}, \hat{\mathcal{F}}_{te})$ for unsupervised and supervised scenarios respectively.

Definition 2.18 (***Unsupervised Hyperlink Prediction***)**.** *Given hyperlink prediction dataset* $\tilde{\mathbb{D}}_{unsup} = (\mathcal{H}_{tr}, \mathcal{F}_{te}, \hat{\mathcal{F}}_{te})$*, the* ***unsupervised hyperlink prediction*** *problem is to develop a hyperlink predictor* $\tilde{\pi} : \mathcal{F}_{te} \cup \hat{\mathcal{F}}_{te} \to \mathbb{R}$ *that uses the train graph* $\mathcal{H}_{tr}$ *to compute LP scores* $\tilde{\pi}(F)$ *for each subset* $F \subseteq V$ *in such a way that hyperedges get higher scores than non-hyperedges. That is,*

$$\max_{\tilde{\pi} \in \mathbb{R}^{\mathcal{F}_{te} \cup \hat{\mathcal{F}}_{te}}} Pr(\tilde{\pi}(F) > \tilde{\pi}(\hat{F}) \mid F \in \mathcal{F}_{te}, \hat{F} \in \hat{\mathcal{F}}_{te}) \tag{2.18}$$

Definition 2.19 (***Supervised Hyperlink Prediction***)**.** *Given LP dataset* $\tilde{\mathbb{D}}_{sup} = (\mathcal{H}_{otr}, \mathcal{F}_{utr}, \hat{\mathcal{F}}_{utr}, \mathcal{F}_{te}, \hat{\mathcal{F}}_{te})$*, the* ***supervised hyperlink prediction*** *problem is to train a hyperlink prediction model* $\tilde{\pi}_\theta : \mathcal{F}_{te} \cup \hat{\mathcal{F}}_{te} \to \mathbb{R}$ *parameterized by* θ *using the observed train hypergraph* $\mathcal{H}_{otr}$ *and training data* $(\mathcal{F}_{utr}, \hat{\mathcal{F}}_{utr})$ *such that each test hyperedge gets a higher score than a test non-hyperedge. That is,*

$$\max_{\tilde{\pi}_\theta \in \mathbb{R}^{\mathcal{F}_{te} \cup \hat{\mathcal{F}}_{te}}} Pr(\tilde{\pi}_\theta(F) > \tilde{\pi}_\theta(\hat{F}) \mid F \in \mathcal{F}_{te}, \hat{F} \in \hat{\mathcal{F}}_{te}) \tag{2.19}$$

Be it supervised or unsupervised, if the data is prepared from a temporal hypergraph, we call the hyperlink problem a ***temporal hyperlink prediction*** problem, and for a non-temporal hypergraph, it is called a ***non-temporal/structural hyperlink prediction*** problem.

2.6 Bipartite Structures

Given two disjoint sets V and V' of nodes for bipartite relations to be defined, we name them "***left***" and "***right***" node-sets respectively. In this scenario, the set $\mathcal{E} \subseteq \{\{v, v'\} \mid v \in V, v' \in V'\}$ refers to a collection of ***bipartite edges***. Hence, we get a ***bipartite graph*** $\mathcal{G} = (V \cup V', \mathcal{E})$, which essentially connects a left node to a right one, forming cross-relations between the two node sets. Contrasting it with higher-order bipartite relations, we define the following:

Definition 2.20 (***Simple Bipartite Hypergraph***)**.** *Given left and right node sets* V *and* V' *respectively,*

*a **simple bipartite hypergraph** is defined as an ordered triplet $\mathcal{H}_{sim} = (V \cup V', \mathcal{B}_{sim})$, where*

$$\mathcal{B}_{sim} \subseteq \{B \in \mathcal{P}(V \cup V') \mid B \cap V \neq \emptyset \text{ and } B \cap V' \neq \emptyset\} \tag{2.20}$$

*denotes a collection of **simple bipartite hyperedges**.*

Chapter 3

Groups Skew Pairs: Effect of Hypergraphs on Pairwise Links

"People most strenuously seek to evaluate performance by comparing themselves to others, not by using absolute standards." ∼ `Leon Festinger`

LINK prediction in networks is a well-known problem, and has been researched upon for about two decades now [2, 66]. The most widely used approaches to solving this problem are based on heuristics such as Common Neighbors (CN) – *more the common neighbors of a pair of nodes, higher their linkage probability*. In this chapter, we investigate this problem in the presence of higher-order relations (essentially, hypergraphs). What has been surprising is that neighborhood based link prediction heuristics such as CN work very well – and even better in the presence of higher-order relations. However, as we prove later, this is due to the CN-heuristic overestimating its prediction abilities in the presence of higher-order relations. We prove this statement by considering a theoretical latent space model for higher-order relations and by showing that area under ROC (AUC) scores for CN are higher than could be achieved from the model. Empirically, we observe that the presence of underlying hyperedges indeed skews prediction scores for link prediction heuristics. Moreover for simple examples, we also provide theoretical justifications. Further, we extend our observations to other similar link prediction algorithms such as Adamic Adar, and observe that the mere presence of hyperedges in a network as an underlying component distorts its AUC. Finally, we use these insights to propose an *adjustment factor* to the evaluation by taking into conscience that a random graph would only have a best AUC of 50%. Eventually, we see that this adjustment factor allows for a better estimation of generalization scores by providing an *adjusted AUC* for each method-dataset pair.

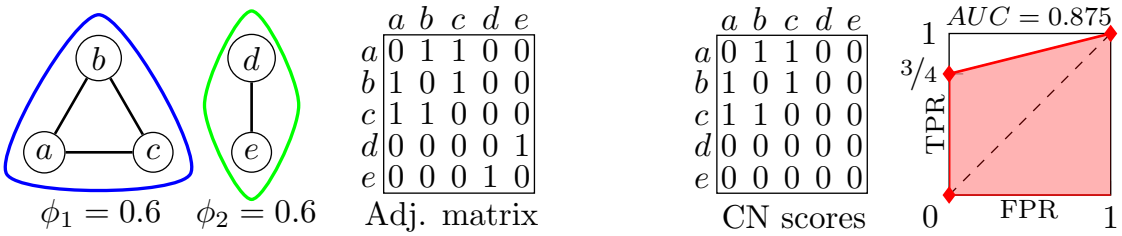
3.1 Introduction

The problem of link prediction (LP) is described as follows: "Given a set of objects V, and a set $\mathcal{E} \subseteq \mathcal{P}_2(V)$ of (partial) links among them, predict new/missing links among V". This is naturally modelled as a simple graph $G = (V, \mathcal{E})$. Ever since the seminal work on this problem [66], it has seen constant advancements [113, 72]. Standard LP algorithms are based on heuristics such as Common Neighbors (CN) [78], which posits that more number of common neighbors imply a higher chance of link between a pair of nodes, or Adamic Adar (AA) [2], a normalized version of the CN approach. These heuristics are known to work dramatically well for simple datasets [66, 89, 21]. In this chapter, we consider the LP problem (and algorithms for the same) in the presence of *higher-order relations*.

We show that in the presence of higher-order relations, LP algorithms do not generalize well. Moreover, we prove that evaluation of LP algorithms in such scenarios overestimates their prediction capability. As a simple example, consider the network in Figure 3.1. Let it consist of 5 vertices named a–e. Also assume that it consists of two hyperedges $\{a, b, c\}$ and $\{d, e\}$, appearing with probabilities $\phi_1 = 0.6$ and $\phi_2 = 0.6$ respectively. This has been depicted in Figure 3.1 using the blue and green enclosures respectively. It is to be noted that by "probability of edge $\{b, c\}$ (i.e., ϕ)", we do not mean the predicted probability, but the real probability of existence of this edge as per a (Erdos-Renyi) random graph. Let us apply CN over this example by predicting links using the *leave-one-out* method of evaluation and subsequently compute its AUC score.

In clearer words, we first assume that all edges $\{a, b\}$, $\{b, c\}$, $\{a, c\}$, and $\{d, e\}$ exist. Then we take all pairs from $\{a, b, c, d, e\}$ and find their predicted scores (the CN matrix Fig. 3.1b) using leave-one-out method. And we finally find the AUC by comparing it with the ground truth (the adjacency matrix). Hence, CN asserts that link $b \sim c$ is more probable than link $d \sim e$ since the former has more common neighbors than the latter. However this is *not* the case since both these links occur with probability 0.6. Moreover, the evaluation of CN for this example *does not* take this into account, thereby estimating the predictive AUC score to be 0.875. While actually, one can only obtain an AUC of 0.5 – a fact that can be verified empirically. Thus CN overestimates its own predictive capabilities. In this chapter, we formalize these notions and provide both theoretical and empirical support to these observations. Moreover, we also provide a novel evaluation method, proposing an *adjustment-factor* to correct the predictive scores.

In Section 3.2, we propose a simple mathematical model for higher-order relations, which is used for analysis in the rest of the chapter. For completeness, we compare this to existing latent space models for link prediction as well. Then in Section 3.3, we use the model in Section 3.2 to prove empirically that the LP heuristics CN and AA overestimate their generalization-ability. This is justified theoretically considering simple cases. In Section 3.4, we propose a new evaluation scheme which



(a) (b)

Figure 3.1: Figure illustrating the effect of higher-order relations. (a) A toy hypergraph where set of vertices is $\{a, b, c, d, e\}$. $\{a, b, c\}$ denotes one hyperedge (in blue) which appears with probability $\phi_1 = 0.6$ and $\{d, e\}$ is another one (in green) appearing with the same probability $\phi_2 = 0.6$. The four arcs $a \sim b$, $b \sim c$, $a \sim c$, and $d \sim e$ (in black) denote edges of the graph formed by the hypergraph's clique expansion [3], whose adjacency matrix has also been shown. (b) CN scores are calculated using the *leave-one-out* method and are shown as a matrix. Also shown is the ROC curve of CN *w.r.t.* the Adjacency matrix in (a), which has an AUC of 87.5%.

takes into account the higher-order relations. Finally in Section 3.6 we discuss the implications of the work done in this chapter.

3.1.1 Key Contributions

1. We prove that higher-order relations skew link prediction. In particular, we show that standard heuristics such as CN and AA do not generalize well in the presence of higher-order relations. Moreover, we show that the evaluation of these methods also do not take this into consideration, thereby overestimating their ability to predict links.

2. To provide better estimates of the generalization performance, we propose a novel approach to compute an *adjustment factor* to correct the generalization scores.

3.2 A Mathematical Model for Higher-Order Relations

In this section, we provide a simple model for modelling higher-order relations, which is used in the rest of the chapter for analysis and simulation. This has been adapted from Turnbull et al. [108], the main difference being our assumption that the latent space is fixed. Recall that to specify a hypergraph, one needs to specify the set of objects V and the subsets of V chosen to be hyperedges, $\mathcal{F} \subseteq \mathcal{P}(V)$.

3.2.1 Model Formulation

Let $V = \{1, 2, \cdots, n\}$ denote a set of objects, wherein for each element $i \in V$, assume there exists an underlying vector $\boldsymbol{u}_i$ in the *latent space* $\mathbb{R}^d$. To model the hyperedges in this space, we assume

that their sizes/cardinalities (number of objects in a hyperedge) lie in the set $\{1, 2, \cdots, k\}$, where $1 \leq k \leq n$. Let $r_1, r_2, \cdots, r_k \in \mathbb{R}$ be called as the *radii* corresponding to hyperedge sizes $1, 2, \cdots, k$ respectively. As per our model, we define a subset $F \in \mathcal{P}(V)$ to be a "potential hyperedge" if and only if there exists a ball of radius $r_{|F|}$ encompassing the set of latent vectors of its containing objects. That is for a given subset $F \in \mathcal{P}(V)$, we have:

$$F \in \overline{\mathcal{F}} \iff \exists \boldsymbol{x} \in \mathbb{R}^d \text{ s.t. } \{\boldsymbol{u}_i\}_{i \in F} \subseteq B\left(\boldsymbol{x}, r_{|F|}\right), \tag{3.1}$$

where $\overline{\mathcal{F}}$ denotes the set of *potential hyperedges*. It is unlikely that all the potential hyperedges belong to the final hypergraph. Hence, we introduce probabilities $\phi_1, \phi_2, \cdots, \phi_k \in [0, 1]$ corresponding to hyperedge sizes $1, 2, \cdots, k$ respectively. The final set of hyperedges would then be given by:

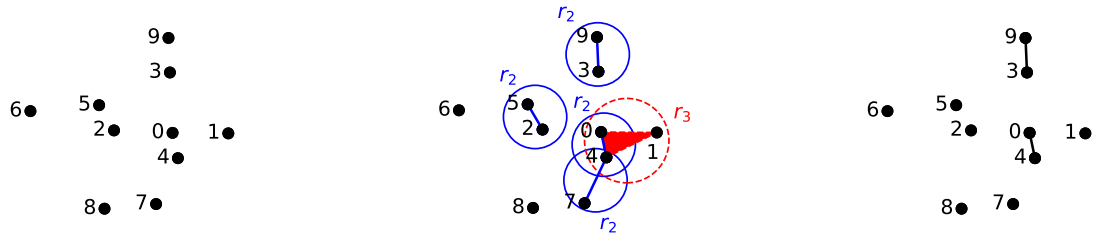
$$F \in \mathcal{F} \iff F \in \overline{\mathcal{F}} \text{ and Bernoulli}(\phi_{|F|}) = 1, \tag{3.2}$$

that is, a potential hyperedge would be in the final set of hyperedges with probability $\phi_{|F|}$.

To generate a hypergraph using the model above, one can start with an set of arbitrary representation vectors $U = \{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n\}$, one for each node, say from a normal distribution with mean $\mathbf{0}_d$ (zero vector of size d) and co-variance $\mathbf{I}_d$ (identity matrix of size $d \times d$), and pick r to be the fixed percentiles of all the pairwise distances. The hyperedges are generated using:

1. Set $s = 2$, and $\overline{\mathcal{F}} = \{\}$.
2. Start with radius r_s, and select all groups with distance $\leq 2r_s$ as s-sized hyperedges and add them to $\overline{\mathcal{F}}$.
3. Obtain the cliques of size s in this hypergraph.
4. Repeat steps 2 and 3 above for radii of a higher order ($s > 2$) to obtain the set of potential hyperedges for all sizes $2, 3, \ldots, k$, and ultimately get $\overline{\mathcal{F}}$.
5. Finally, select each potential hyperedge $F \in \overline{\mathcal{F}}$ with a probability $\phi_{|F|}$ to get the set of final hyperedges $\mathcal{F}$.

Refer to Figure 3.2 for an illustration of the foregoing procedure. This procedure generates a *Vietoris–Rips Complex* [84], which is equivalent to a Čech complex [36, 28, 108]. Moreover, there exist faster algorithms to achieve this as well [139]. We do not further elaborate on the Čech complex nature of our model, which could be found in Turnbull et al [108], and limit our study to its role in explaining hypergraphs' effect on link prediction.



(a) $U = \{\boldsymbol{u}_i\}_{i=0}^{9}$ (b) $\overline{\mathcal{F}}(U, r)$ (c) $\mathcal{F}(U, r, \Phi)$

Figure 3.2: An illustration of the *hypergraph latent space model* described in Section 3.2 for ten vertices 1–10. (a) The 2-dimensional representation U of *vertices*. (b) Radii $r = (r_2, r_3)$ are picked and *potential hyperedges* $\overline{\mathcal{F}}$ are generated: four 2-sized hyperedges $\{0, 4\}$, $\{2, 5\}$, $\{3, 9\}$, $\{4, 7\}$ *w.r.t.* radius r_2, and one 3-sized hyperedge $\{0, 1, 4\}$ *w.r.t.* radius r_3. Note the balls of radii r_2 (blue solid) and r_3 (red dotted) encompass vertices that form the hyperedges. (c) Finally, *hyperedges* $\mathcal{F}$ are sampled from $\overline{\mathcal{F}}$ via distribution Φ; in this case, we have $\mathcal{F} = \{\{0, 4\}, \{3, 9\}\}$.

3.2.2 Relation to Hoff's Latent Space Model

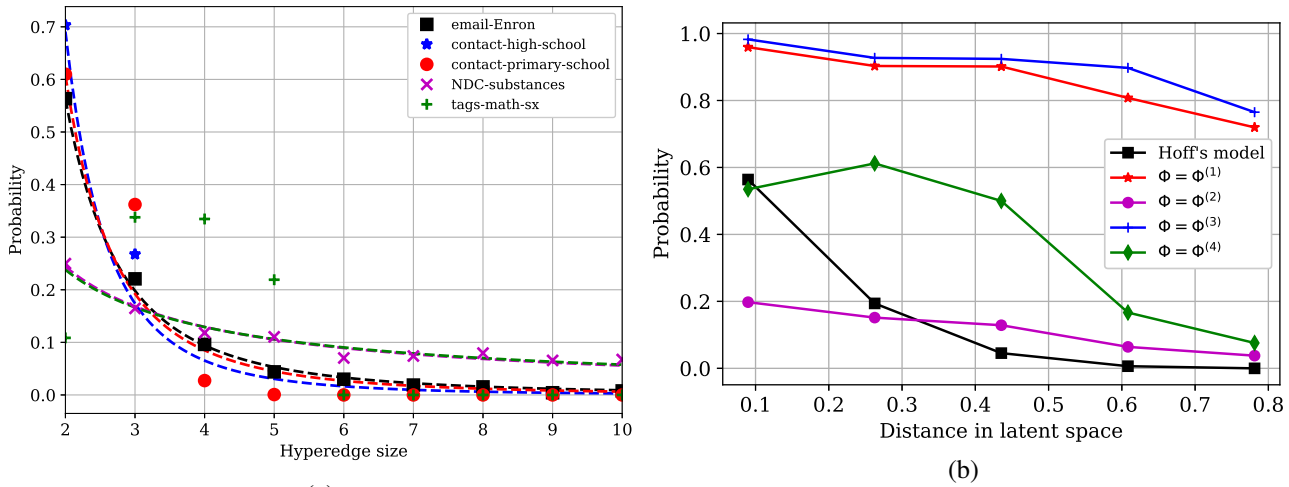
Classically, link prediction in simple graphs has been modelled using what we call the *Hoff's model*, which is described in Hoff et al. [45]. Authors in Sarkar et al. [89] use this model to provide theoretical justifications to three LP heuristics. However, as we shall shortly see, Hoff's model underestimates the higher-order relations. Basically, it assumes that two vertices i and j are linked to each other with probability

$$P(i \sim j) = \frac{1}{1 + \exp(\alpha(\|\boldsymbol{u}_i - \boldsymbol{u}_j\| - \gamma))}, \tag{3.3}$$

where α and γ are the model's parameters and $\boldsymbol{u}_i$ and $\boldsymbol{u}_j$, the vertices' latent vectors. Thus, a hyperedge $F \in \mathcal{F}$ would show its existence via Hoff's model if all 2-subsets of F (*i.e.*, edges comprising the clique over nodes in F) get selected by the model. We have,

$$P(i \sim j, \forall i, j \in F, i \neq j) = \prod_{\substack{i,j \in F \\ i \neq j}} P(i \sim j) = \prod_{\substack{i,j \in F \\ i \neq j}} \frac{1}{1 + \exp(\alpha(d_{ij} - \gamma))}, \tag{3.4}$$

where $d_{ij} := \|\boldsymbol{u}_i - \boldsymbol{u}_j\|$. Observe that if $|F| = k$, then this has $k(k-1)/2$ factors in the product and thus reduces as $\mathcal{O}(1/C^{k(k-1)/2})$ for some constant $C > 1$. Thus, the number of hyperedges reduces exponentially *w.r.t.* hyperedge size k according to Hoff's model. However, in most real-world hypergraphs, the number of hyperedges have been observed to follow a power law, $1/k^{\zeta}$ as shown in Figure 3.3a, where $\zeta > 0$ varies from domain to domain. Hence, we know that Hoff's model does not capture higher-order relations well. Another implication of this observation is that Hoff's model also underestimates number of long distance edges. To illustrate this, we compare the probabilities of generating an edge of distance d by both the models. We show this for four choices of Φ in Figure 3.3b.



(a) (b)

Figure 3.3: (a) Hyperedge size ($|F| = k$) distribution in a few real-world datasets. Each distribution is fitted with a power-law $1/(k^{\zeta})$ truncated between 2 and 10, with best-fit ζ-values being as follows: *email-Enron*: 2.58, *contact-high-school*: 3.43, *contact-primary-school*: 2.83, *NDC-substances*: 0.91. More details about the datasets are available in Table 3.1. (b) Edge-generation probabilities plotted against distance between (the latent space representations of) incident nodes thereof. We consider different choices for ϕ_k and look at the distribution of the edges: $\phi_k^{(1)} = 1/k^2$, $\phi_k^{(2)} = 1/(1 + e^{\alpha(r_k - \gamma)})$, $\phi_k^{(3)} = 0.1$, $\phi_k^{(4)}$ as per $\overline{\mathcal{F}}$. The latent space U is generated using a 2-dimensional normal distribution with mean $\mathbf{0}_2$ and covariance $\mathbf{I}_2$ (identity matrix of size 2×2). The radii are picked to be the 1-, 5-, 9-, and 13-percentiles of all the distances between the points. Observe that for large distances (ranging between 0.4–0.6), Hoff's model underestimates the number of edges.

Remark: The choices of Φ are dictated by conventional wisdom. (i) $\phi_k^{(1)} := 1/k^2$ is used since in real datasets, a power law size distribution is observed. (ii) $\phi_k^{(2)} := 1/(1 + \exp(\alpha(r_k - \gamma)))$ is used since this is the probability that an edge with distance r_k is picked. (iii) For completeness, we also consider $\phi_k^{(3)} = 0.1$. (iv) Another option is to take $\phi_k^{(4)}$ as per the size distribution in $\overline{\mathcal{F}}$.

3.3 Effect on the Evaluation of Link Prediction

In this section, we shall use the model from Section 3.2 to analyze the performance of the LP heuristics - Common Neighbors (CN) and Adamic Adar (AA). Specifically, we show that *these heuristics overestimate their ability to predict links*.

3.3.1 Capturing the Generalization Error

Recall that the hypergraph model uses a triplet (U, r, Φ) of vertex representation vectors U, size-specific radii r, and hyperedge selection probability distribution Φ to obtain the hypergraph $\mathcal{H} = (V, \mathcal{F})$. This is converted to a simple graph $\eta(\mathcal{H})$ using *clique expansion* [3]. We then have the following proposition:

Proposition 3.1. *In the model described in Section 3.2, we have*

$$P(i \not\sim j) = \prod_k (1-\phi_k)^{S_k(i,j)}, \tag{3.5}$$

where $S_k(i,j)$ is the number of k-sized hyperedges in $\overline{\mathcal{F}}$ which contain both vertices i and j given U and r. Clearly, we also have

$$P(i \sim j) = 1 - \prod_k (1-\phi_k)^{S_k(i,j)}. \tag{3.6}$$

Proof. Given $i, j \in V$, and $F \in \overline{\mathcal{F}}$, define $S_k(i,j,F) := 1$ if F has $\{i,j\}$ as a subset of its vertices; otherwise, $S_k(i,j,F) = 0$. We hence have $S_k(i,j) = \sum_{F\in\overline{\mathcal{F}}} S_k(i,j,F)$. Note that the event $(i \not\sim j)$ holds if and only when all the events $(i \not\sim j, S_k(i,j,F) = 1)$ for all k hold and each of these events are mutually independent, a fact known from the model. Hence,

$$P(i \not\sim j) = \prod_{F\in\overline{\mathcal{F}}} P(i \not\sim j, S_k(i,j,F) = 1). \tag{3.7}$$

Now, since $S_k(i,j,F)$ is either 1 or 0, and depends on U and r, it is independent of the event $(i \not\sim j)$. Hence we have,

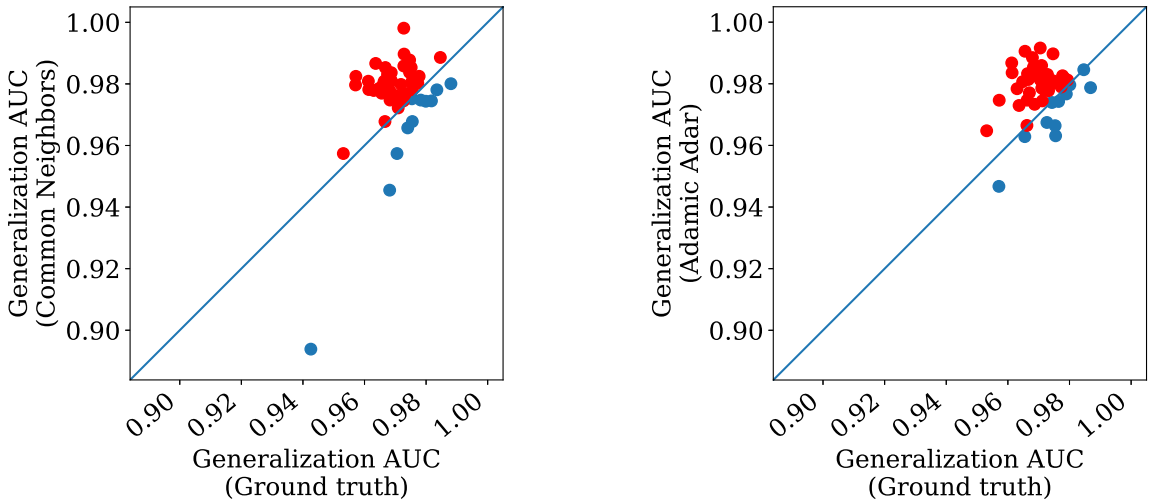
$$\begin{aligned} P(i \not\sim j) &= \prod_{F\in\overline{\mathcal{F}}} P(i \not\sim j \mid S_k(i,j,F) = 1) \cdot S_k(i,j,F) \\ &= \prod_k (1-\phi_k)^{S_k(i,j)}. \end{aligned}$$

□

Using Proposition 3.1 one can compute the probability of a link between two vertices i and j.

3.3.2 The Behavior of Link Prediction Heuristics

On the other hand, the LP heuristics CN and AA dictate that this probability is proportional to the number of common neighbors. Figure 3.4 shows the scatter plots between the AUC scores obtained from the model and those by the LP heuristics. Observe that in several cases (marked red) the generalization performance as estimated by the LP heuristics is *higher* than the ground truth probabilities. However, theoretically, *the generalization performance of any algorithm cannot be better than the one obtained using the estimates in eq.* (3.7). This shows empirically that LP heuristics such



(a) Common Neighbors (CN)

(b) Adamic Adar (AA)

Figure 3.4: Generalization performances of heuristic LP methods. Observe that in several cases (the ones marked in red), the generalization performance predicted by both CN and AA is higher than the actual one derived from the model. This shows that in presence of hyperedges, existing methods overestimate the generalization performance.

as CN/AA overestimate their ability to predict links.

To understand better why this is the case, consider two scenarios: (a) A simple graph without higher-order relations as shown in Figure 3.5a, and (b) A hypergraph with a single hyperedge of size 3 as shown in Figure 3.5b.

(a)

(b)

Figure 3.5: Example hypergraphs. Both the examples consist of 3 vertices a, b, c. Assume that the set of potential hyperedges is $\overline{\mathcal{F}} = \{\{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$, and hyperedge-selection probabilities are $\Phi = (\phi_2, \phi_3)$. (a) The hypergraph where the three 2-edges are possible with probability $\phi_2 > 0$ (and ϕ_3 is fixed to zero). (b) The hypergraph where the 3-edge is possible with probability $\phi_3 > 0$ (and ϕ_2 is set to zero). We have that the AUC scores of the model for CN match in (a), which is equal to 0.5. However in (b), we have that the AUC score of CN is 1, while the AUC score from the model is just 0.5. Thus, CN overestimates its ability to predict the links in presence of higher-order relations.

3. GROUPS SKEW PAIRS: EFFECT OF HYPERGRAPHS ON PAIRWISE LINKS

We now closely analyze the AUC scores of the CN heuristic. For this, we require the following notation: Let $Z_{\sim} := (CN \mid (i \sim j))$ denote the random variable which counts the number of common neighbors between i and j when it is known that $i \sim j$ (i is linked to j). Similarly, let $Z_{\not\sim} := (CN \mid (i \not\sim j))$ be a random variable counting common neighbors when $i \not\sim j$. Using this notation and a result from Fawcett et al. [30], we have that the AUC score can be computed using:

$$AUC = P(Z_{\sim} > Z_{\not\sim} \mid (Z_{\sim} \neq Z_{\not\sim})). \tag{3.8}$$

Now, in the case of Figure 3.5a, which is essentially a graph with no higher-order relations, we have,

$$Z_{\sim} = Z_{\not\sim} = \begin{cases} 1 & \textit{w. p. } \phi_2^2, \\ 0 & \textit{w. p. } 1 - \phi_2^2 \end{cases} \tag{3.9}$$

So, it is easy to see that we have an AUC of 0.5 as can be computed from eq. (3.8). In other words, the CN heuristic estimates that it cannot predict well whether node a would be linked to node b. This prediction matches with the ground-truth, since in the first place, the link between node a and node b is randomly present with probability $\phi_2 > 0$. Now, consider the scenario when a higher-order relation is present in the network as shown in Figure 3.5b. In this case, the link between nodes a and b appears randomly with probability ϕ_3, and hence, any heuristic should not be able to predict the link with an AUC score of more than 0.5. However, in this case observe that:

$$Z_{\sim} = \begin{cases} 1 & \textit{w. p. } 1, \\ 0 & \textit{w. p. } 0 \end{cases} \quad \text{and } Z_{\not\sim} = \begin{cases} 1 & \textit{w. p. } 0, \\ 0 & \textit{w. p. } 1. \end{cases} \tag{3.10}$$

And from eq. (3.8), we have that AUC score is 1. In other words, the CN heuristic estimates it can predict perfectly whether the link between nodes a and b exists or not. But, it is known that this is not possible since the existence of a link between nodes a and b is, by construction, random with probability $\phi_3 > 0$. Hence, this justifies the empirical observation that LP heuristics CN and AA overestimate their ability to predict links in presence of higher-order relations (hyperedges).

We state and prove a similar argument for a more generic hypergraph in Theorem 3.1, making our case even stronger.

Theorem 3.1. *Let (U, r, ϕ) denote the hypergraph model, where $\phi_2 = 0$ and $\phi_i > 0$ for all $i > 3$. Then the AUC score of CN is strictly greater than 0.5.*

Note that since any link $i \sim j$ can occur only with probability $\phi_{|F|}$ where $F \supset \{i, j\}$, the best possible score can only be 0.5.

Proof. Let $\{i, j\}$ denote arbitrary but fixed pair of points such that there exists a unique hyperedge $F \supset \{i, j\}$. We prove that in this case,

$$P\left(Z_{i\sim j} > Z_{i\not\sim j} \mid Z_{i\sim j} \neq Z_{i\not\sim j}\right) > 0.5 \tag{3.11}$$

The proof for the generic case is similar.

Claim 1 : We first show that,

$$P(Z_{i\sim j} = 0) < P(Z_{i\not\sim j} = 0) \tag{3.12}$$

and for all $k \geq 1$,

$$P(Z_{i\sim j} > k) > P(Z_{i\not\sim j} > k) \tag{3.13}$$

Recall that $\overline{\mathcal{F}}$ denotes the set of all possible hyperedges. Now, let E denote an event that essentially selects a subset of $\overline{\mathcal{F}}$ based on some constraints. Let $E_{i\not\sim j}$ denote the event where $i \not\sim j$. Then, one can construct the event $E_{i\sim j}$ by ensuring that one picks a hyperedge $F \in \overline{\mathcal{F}}$. So, if $P(E_{i\not\sim j}) = p$, then, $P(E_{i\sim j}) = p \cdot \phi_{|F|}/(1 - \phi_{|F|})$. Now,

$$\begin{aligned} P(E_{i\not\sim j} \mid i \not\sim j) &= \frac{P(E_{i\not\sim j})}{P(i \not\sim j)} = \frac{p}{1 - \phi_{|F|}} \\ &= \frac{p \cdot \phi_{|F|}}{\phi_{|F|} \cdot (1 - \phi_{|F|})} = \frac{P(E_{i\sim j})}{P(i \sim j)} \\ &= P(E_{i\sim j} \mid i \sim j) \end{aligned} \tag{3.14}$$

Moreover, we have that $CN(E_{i\sim j})$ (common neighbor count of $\{i, j\}$ when $i \sim j$) is greater than or equal to $CN(E_{i\not\sim j})$. Now, consider $P(Z_{i\sim j} = 0)$. Since, if $i \sim j$, then F is selected and $|F| > 2$, we must have that $P(Z_{i\sim j} = 0) = 0$ that is there exists at least one common neighbor for $\{i, j\}$ when $i \sim j$. Clearly, it is possible that $Z_{i\not\sim j} = 0$ and hence

$$P(Z_{i\sim j} = 0) < P(Z_{i\not\sim j} = 0) \tag{3.15}$$

Now, consider the case when $k \geq 1$,

$$P(Z_{i\not\sim j} > k) = \sum_{E_{i\not\sim j}} I\{CN(E_{i\not\sim j}) > k\} P(E_{i\not\sim j} \mid i \not\sim j) \tag{3.16}$$

$$< \sum_{E_{i\sim j}} I\{CN(E_{i\sim j}) > k\} P(E_{i\sim j} \mid i \sim j) \tag{3.17}$$

$$= P(Z_{i\sim j} > k) \tag{3.18}$$

Since, as shown earlier, for each $E_{i\not\sim j}$, one can construct a corresponding $E_{i\sim j}$ such that $CN(E_{i\sim j})$ is greater than or equal to $CN(E_{i\not\sim j})$. The strong inequality holds since one can construct atleast one case such that $CN(E_{i\sim j})$ is strictly greater than $CN(E_{i\not\sim j})$.

Now, let $p_k = P(Z_{i\sim j} = k)$ and $p'_k = P(Z_{i\not\sim j} = k)$

$$\begin{aligned} AUC &= P\left(Z_{i\sim j} > Z_{i\not\sim j} \mid Z_{i\sim j} \neq Z_{i\not\sim j}\right) && (3.19)\\ &= \sum_k \frac{P\left(Z_{i\sim j} > Z_{i\not\sim j}\right)}{P\left(Z_{i\sim j} \neq Z_{i\not\sim j}\right)} && (3.20)\\ &= \sum_k \frac{P\left(Z_{i\sim j} > k, Z_{i\not\sim j} = k\right)}{\sum_{k'} P\left(Z_{i\sim j} \neq k', Z_{i\not\sim j} = k'\right)} && (3.21)\\ &= \sum_k \frac{P\left(Z_{i\sim j} > k\right) P\left(Z_{i\not\sim j} = k\right)}{\sum_{k'} P\left(Z_{i\sim j} \neq k'\right) P\left(Z_{i\not\sim j} = k'\right)} && (3.22)\\ &> \sum_k \frac{P\left(Z_{i\not\sim j} > k\right) P\left(Z_{i\not\sim j} = k\right)}{\sum_{k'} P\left(Z_{i\sim j} \neq k'\right) P\left(Z_{i\not\sim j} = k'\right)} && (3.23)\\ &= \sum_k \frac{P\left(Z_{i\not\sim j} < k\right) P\left(Z_{i\not\sim j} = k\right)}{\sum_{k'} P\left(Z_{i\sim j} \neq k'\right) P\left(Z_{i\not\sim j} = k'\right)} && (3.24)\\ & && (3.25) \end{aligned}$$

Now, for every event $E_{i\sim j}$ one can construct an event $E_{i\not\sim j}$ by simply removing F. Moreover, if $CN(E_{i\sim j}) < k$ then, it implies that $CN(E_{i\not\sim j}) < k$. Hence, $P\left(Z_{i\not\sim j} < k\right) > P\left(Z_{i\sim j} < k\right)$. So, we have

$$\begin{aligned} AUC &> \sum_k \frac{P\left(Z_{i\not\sim j} < k\right) P\left(Z_{i\not\sim j} = k\right)}{\sum_{k'} P\left(Z_{i\sim j} \neq k'\right) P\left(Z_{i\not\sim j} = k'\right)} && (3.26)\\ &> \sum_k \frac{P\left(Z_{i\sim j} < k\right) P\left(Z_{i\not\sim j} = k\right)}{\sum_{k'} P\left(Z_{i\sim j} \neq k'\right) P\left(Z_{i\not\sim j} = k'\right)} && (3.27)\\ &= P\left(Z_{i\sim j} < Z_{i\not\sim j} \mid Z_{i\sim j} \neq Z_{i\not\sim j}\right) && (3.28) \end{aligned}$$

Hence we have that,

$$AUC = P\left(Z_{i\sim j} > Z_{i\not\sim j} \mid Z_{i\sim j} \neq Z_{i\not\sim j}\right) > 0.5 \quad (3.29)$$

as,

$$P\left(Z_{i\sim j} > Z_{i\not\sim j} \mid Z_{i\sim j} \neq Z_{i\not\sim j}\right) + P\left(Z_{i\sim j} < Z_{i\not\sim j} \mid Z_{i\sim j} \neq Z_{i\not\sim j}\right) = 1 \quad (3.30)$$

□

3.4 Better Evaluation of Link Prediction Methods

In the previous section, we have proved that higher-order relations skew scores provided by LP heuristics. More particularly, we saw that they tend to overestimate their capability of generalization. In this section, we provide a method to better estimate this generalization-ability.

Theorem 3.2. *On an Erdos-Renyi graph, the best AUC a link prediction method can achieve is of* 0.5.

Proof. Given $n \in \mathbb{N}$ and $p \in \mathbb{R}$, we have $V = \{1, 2, \ldots, n\}$ and $P(i \sim j) = p$. Suppose there is a link predictor $\pi : \mathcal{P}_2(V) \to \mathbb{R}$. If $Z_\sim := (\pi(\{i,j\}) \mid i \sim j)$ and $Z_\nsim := (\pi(\{i,j\}) \mid i \nsim j)$, we have $AUC = P(Z_\sim > Z_\nsim \mid Z_\sim \neq Z_\nsim)$. Now, whatever be the logic the value of π depends upon, it would not differentiate between links and non-links since the "environment" for link prediction was formed at random. In other words, $\pi(\{i,j\})$ would follow the same distribution for both links and non-links, giving us $P(Z_\sim = k) = P(Z_\nsim = k)\ \forall k$ (assuming them to be discrete random variables; a similar argument holds for a continuous one as well). So, we have $P(Z_\sim > Z_\nsim \mid Z_\sim \neq Z_\nsim) = P(X > Y \mid X \neq Y) = 0.5$ (where X and Y are two random variables from the same distribution). □

The main idea relies on the foregoing premise: "*On a random version of a given graph, the best AUC a link prediction method can achieve is of* 0.5". It can be seen from the following Thus, given a hypergraph $\mathcal{H}$, one can construct a *randomized version* $\mathcal{H}_{rand}$ of $\mathcal{H}$, and expect the link prediction AUC on its clique-expanded graph $\eta(\mathcal{H}_{rand})$ to be around 0.5. Now, as noted in the previous section, we know that a typical LP algorithm gives a higher-than-expected AUC score on any graph expanded from a hypergraph. Thus, we compute an *adjustment factor* AF, which we define as the ratio of AUC score $AUC(\mathcal{H}_{rand})$[1] obtained on the randomized hypergraph and the ideally expected score *viz.*, 0.5 on it. Finally, this adjustment factor is used to compute an *adjusted AUC score* $AUC_{adj}(\mathcal{H})$ on the original hypergraph $\mathcal{H}$.

To achieve this, we first make multiple runs of a *hyperedge relocation algorithm* (Algorithm 1) on a given hypergraph $\mathcal{H}$ to obtain multiple relocated versions $\mathcal{H}_{rel}$ of the same. Basically, for each hyperedge in the original hypergraph $\mathcal{H}$, we add a same-sized random hyperedge to $\mathcal{H}_{rel}$. This ensures that the core statistics of the network remains the same. However, since the hyperedges are added randomly, any LP algorithm should only have achieved a score of 0.5. The *adjustment factor* and accordingly, an *adjusted AUC score* can then be computed using relocated AUC $AUC_{rel} := AUC(\mathcal{H}_{rel})$ as:

$$AF(\mathcal{H}) = \frac{AUC_{rel}}{0.5} \qquad\qquad AUC_{adj}(\mathcal{H}) = \frac{AUC(\mathcal{H})}{AF(\mathcal{H})} \tag{3.31}$$

[1]For a hypergraph $\mathcal{H}$, $AUC(\mathcal{H})$ denotes the AUC score obtained on its clique-expanded graph $\eta(\mathcal{H})$.

Algorithm 1: Generates randomized version of a hypergraph $\mathcal{H}$, $\mathcal{H}_{rel}$. This is referred to as the *relocation algorithm*. Every hyperedge in $\mathcal{H}$ is relocated to a randomly selected new one, thereby constructing a *relocated hypergraph* $\mathcal{H}_{rel}$. Multiple runs of this algorithm on the same network is used to estimate baselines for an LP algorithm.

Input: Original hypergraph, $\mathcal{H} = (V, \mathcal{F})$
LP algorithm, X
Output: $\mathcal{H}_{rel}$

$\mathcal{F}_{rel} \leftarrow \{\}$
for $F \in \mathcal{F}$ **do**
  $F_{rel} \leftarrow \text{RANDOMSUBSET}(V, |F|)$
  $\mathcal{F}_{rel} \leftarrow \mathcal{F}_{rel} \cup \{F_{rel}\}$
$\mathcal{H}_{rel} \leftarrow (V, \mathcal{F}_{rel})$
return $\mathcal{H}_{rel}$

3.5 Results and Discussion

Table 3.1 shows the AUC scores obtained on an original hypergraph $\mathcal{H}$ and its relocated versions $\mathcal{H}_{rel}$, the adjustment factors AF, and the adjusted-AUC scores AUC_{adj} for real-world datasets taken from Benson et al. [11]. We perform *five* relocations, and hence report the mean and standard-deviation values for AUC_{rel}. Although there are varieties of algorithms available in the literature: neighborhood-based [127], path-based [72], eigenvector-based [41], evolutionary [15], matrix-factorization-based [74], matrix-completion-based [83], entropy-based [80], deep-learning-based [130], etc., in this chapter we focus on the neighborhood- and path-based ones. Evolutionary algorithms too have shown to be promising in the recent past in various problems involving networks, including link prediction [124] and community detection [90, 104]. Following are the LP algorithms we use in this chapter: *Preferential Attachment* (*PA*) [78, 50], *Adamic Adar* (*AA*) [2], *Common Neighbors* (*CN*) [78], *Jaccard Coefficient* (*JC*) [66], *Resource Allocation* (*RA*) [136], and *SimRank* (*SR*) [49].

The following key observations can be made from Table 3.1:

- For `NDC-substances`, wherein we predict drug interactions, the effect of higher-order relations is the highest. Without adjustment, all heuristics estimate that they would be able to predict around 96–99% of links. However, the randomized (relocated) hypergraph also gives a really high score (except for *PA* and *SR*). In reality, for most heuristics, the score is only around 50–55% as obtained after adjustment (again, *PA* is an exception).

- Interestingly the adjustment factors are *proportional to number of hyperedges of higher orders*. That is, higher the number of higher order relations, larger the adjustment factor. For instance, it is clear from Figure 3.3a that dataset `contact-primary-school` has the least number of

Table 3.1: Popular neighborhood-based link prediction (LP) algorithms' AUC scores (%) for five real-world hypergraphs ($\mathcal{H}$) and their relocated versions ($\mathcal{H}_{rel}$), providing new baselines (*i.e.*, relocated AUCs AUC_{rel}) and corrected performance scores (adjusted AUCs, AUC_{adj}) for them. Observe that the newer baselines are different across different datasets per method. Using the mean relocated AUC scores, we compute adjustment factors AF and ultimately report an adjusted AUC score AUC_{adj} as per eq. (3.31). Apart from this, we have results (t and p) of a t-test at $5+5-2=8$ degrees of freedom for the null hypothesis "Relocated hypergraphs exhibit random behavior (i.e., relocated AUCs $= 50\%$)", wherein it could be observed that $p \ll 1\%$ in most cases (except for in `contact-primary-school` where p is not so small). Also shown are number of nodes $|V|$ and hyperedges $|\mathcal{F}|$, whose size distributions are depicted in Figure 3.3. All datasets have been taken from Benson et al. [11]. Adjustment factors that are lowest for a single dataset are **bold-faced**. In addition, four algorithm-pairs have been marked using superscripts a, b, c, and d; these show performance-reversals *w.r.t.* AUC and AUC_{rel}.

Dataset $\mathcal{H}=(V,\mathcal{F})$	LP algorithm	Orig. AUC AUC	Relocated AUC AUC_{rel}	t	p	AF	Adj. AUC AUC_{adj}
`email-Enron` $\lvert V\rvert=148$ $\lvert\mathcal{F}\rvert=1{,}436$	*PA*	82.46	$\mathbf{67.6000 \pm 0.3123}$	112.7	10^{-13}	**1.35**	61.08
	AA	94.22	69.2780 ± 0.3741	103.1	10^{-13}	1.39	67.78
	CN	93.12	69.1860 ± 0.3855	99.5	10^{-13}	1.38	67.48
	JC	95.36	69.5460 ± 0.4242	92.2	10^{-13}	1.39	68.60
	RA	96.41	69.4460 ± 0.3569	109.0	10^{-13}	1.39	69.36
	SR	88.82	68.2940 ± 0.2667	137.2	10^{-14}	1.37	64.83
`contact-high-school` $\lvert V\rvert=327$ $\lvert\mathcal{F}\rvert=7{,}818$	*PA*	67.05	56.7160 ± 0.1669	80.5	10^{-12}	1.13	59.34
	AA	93.69	54.8560 ± 0.1876	51.8	10^{-11}	1.10	85.17
	CN	93.55	54.8240 ± 0.1940	49.7	10^{-11}	1.10	85.05
	JC	93.20	$\mathbf{53.1520 \pm 0.2081}$	30.3	10^{-9}	**1.06**	87.92
	RA	93.81	54.8580 ± 0.1955	49.7	10^{-11}	1.10	85.28
	SR	92.50	56.7240 ± 0.1166	115.3	10^{-13}	1.13	81.86
`contact-primary-school` $\lvert V\rvert=242$ $\lvert\mathcal{F}\rvert=12{,}704$	*PA*	72.20	56.1760 ± 0.2117	58.3	10^{-11}	1.12	64.46
	AA	86.95	52.3760 ± 0.1852	25.7	10^{-8}	1.05	82.81
	CN	86.27	52.3560 ± 0.1929	24.4	10^{-8}	1.05	82.16
	JC	88.94	$\mathbf{49.3960 \pm 0.1983}$	6.1	10^{-4}	**0.99**	89.84
	RA	88.73	52.3860 ± 0.1822	26.2	10^{-8}	1.05	84.50
	SR	86.28	55.0620 ± 0.2876	35.2	10^{-9}	1.10	78.44
`NDC-substances` $\lvert V\rvert=5{,}556$ $\lvert\mathcal{F}\rvert=4{,}525$	*PA*^{a}	96.86	$\mathbf{66.8960 \pm 0.1153}$	293.1	10^{-17}	**1.34**	72.28
	AA^a	99.13	96.4960 ± 0.0806	1153.7	10^{-21}	1.93	51.36
	CN	98.95	96.1080 ± 0.0966	954.6	10^{-21}	1.92	51.54
	JC	98.67	98.4800 ± 0.0245	3957.6	10^{-26}	1.97	50.09
	RA^b	99.57	97.7540 ± 0.0206	4636.3	10^{-26}	1.96	50.80
	SR^b	98.88	88.9620 ± 0.0479	1626.8	10^{-23}	1.78	55.55
`tags-math-sx` $\lvert V\rvert=1{,}554$ $\lvert\mathcal{F}\rvert=22{,}274$	*PA*^{c}	90.48	$\mathbf{56.0940 \pm 0.0585}$	208.3	10^{-16}	**1.12**	80.79
	AA^c	94.88	64.8120 ± 0.0611	484.8	10^{-18}	1.30	72.98
	CN	94.31	64.6880 ± 0.0601	488.8	10^{-18}	1.29	73.11
	JC	89.43	65.2260 ± 0.0700	435.0	10^{-18}	1.30	68.79
	RA^d	96.04	64.8480 ± 0.0631	470.6	10^{-18}	1.30	73.88
	SR^d	94.78	60.0960 ± 0.0422	478.5	10^{-18}	1.20	78.98

higher-sized hyperedges, and also the least of the adjustment factors (1–1.12). On the other hand, dataset `NDC-substances` has a higher number of higher-sized hyperedges, and hence the largest of adjustment factors (1.34–1.97).

- To better understand the need for an adjustment factor and an adjusted AUC for a hypergraph, we consider two link prediction algorithms LPA_1 and LPA_2. Now, any comparison of these algorithms assumes *similar* AUC scores in case of randomly generated datasets;[1] this can be considered as a *baseline* for their comparison. This seems true for some algorithm pairs in Table 3.1: *e.g.*, *AA* and *CN* share similar AUC_{rel} scores for each dataset. Contrast this with the pairs marked using superscripts a, b, c, and d. For instance, consider the algorithm pairs (*PA*, *AA*) and (*RA*, *SR*), which on both the randomly relocated hypergraphs `NDC-substances` and `tags-math-sx`, *do not perform* equally, in that *AA* and *RA* decently outperform *PA* and *SR* respectively. Hence, the *baselines are quite different*, and so algorithms *PA* and *AA* are not comparable here. Same is the case for *RA* and *SR*.

- A remarkable observation is that the adjusted AUCs of the a-, b-, c-, and d-marked algorithm-pairs in Table 3.1 show a *performance-reversal*, *i.e.*, the AUC order reverses for algorithm-pairs (*PA*, *AA*) and (*RA*, *SR*) when adjusted. More specifically, for both datasets `NDC-substances` and `tags-math-sx`, we have $AUC(AA) > AUC(PA)$, but $AUC_{adj}(AA) < AUC_{adj}(PA)$ (similarly, $AUC(RA) > AUC(SR)$, but $AUC_{adj}(RA) < AUC_{adj}(SR)$), reversing the performance rating of the algorithms. This is indeed the situation which occurs in presence of higher-order relations, and hence correction is required for proper evaluation. As link prediction is a 2-class problem, the appropriate baseline is indeed 0.5, and one should normalize the scores accordingly. This is achieved by the adjustment factor.

3.6 Conclusion

We set out to study the effect hypergraphs have on their induced graphs – in particular, on link prediction therein. Through a couple of experiments, some observations, and a few theoretical arguments, we have successfully proved that *higher-order relations skew link prediction* in simple graphs. More specifically, we prove this by proposing a simple model for hypergraphs, and using this model, we show that link prediction algorithms such as Common Neighbors and Adamic Adar *do not generalize well* in the presence of higher-order relations. Moreover, we also posit that these algorithms tend to *overestimate their ability to predict links*. To correct the said overestimation, we propose a *new*

[1] Random in the sense that any useful prediction cannot be made for such datasets. Consider a simple classification problem where both classes 0 and 1 come from the same distribution. In such cases, it is known that ***no*** classifier could be successfully learnt.

evaluation approach by computing an *adjustment factor* that amends the performance scores for link prediction, namely, area under ROC curve (AUC).

Chapter 4

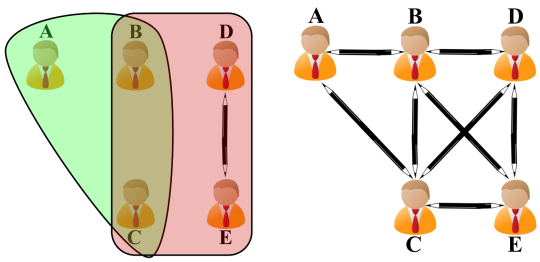
Exploit before Expanding: Leveraging Hypergraphs for Modeling Node Pairs

"It has been said: The whole is more than the sum of its parts. It is more correct to say that the whole is something else than the sum of its parts, because summing up is a meaningless procedure, whereas the whole-part relationship is meaningful." ∼ Kurt Koffka

NODE-similarity in networks has motivated a plethora of such measures between node-pairs, which make use of the underlying graph structure. We have already seen in the previous chapter (Chapter 3) how hypergraphs affect the dynamics of link prediction heuristics, a paradigm catering to pairs of nodes. Measuring proximity between node pairs in the presence of hyperedges calls for a revision in the topological measures of similarity, lest the hypergraph structure remains under-exploited. In this chapter, we propose a multitude of hypergraph-oriented similarity scores between node-pairs, thereby providing novel link prediction heuristics as well, that exploit higher-order relations (or hyperedges). As a part of our proposition, we also provide theoretical formulations to extend graph-topology based scores to hypergraphs. We compare our scores with popular graph-based heuristic similarity scores (over clique-expansions of hypergraphs into graphs). Using a combination of the existing graph-based and the proposed hypergraph-based similarity scores as features for a classifier predicts links much better than using the former solely. Experiments on several real-world datasets and both quantitative as well as qualitative analyses on the same exhibit the superiority of the proposed similarity scores over the existing ones.

4.1 Introduction

Measuring similarity between nodes of a graph has attracted the attention of network science researchers in all domains, be it social [37], biological [9], bibliographic [102], or entertainment [61]. One simple



(a) Hypergraph (b) Derived network

Figure 4.1: A toy example showing the genesis of a co-authorship network from its originally occurring hypergraph (*i.e.*, its higher-order counterpart).

reason why similarity between two nodes is important is to make a decision as to whether two seemingly unconnected nodes should be connected or not – a problem more popularly known as recommendation [32]. While the notion of similarity between two nodes is fairly intuitive when the underlying relational structure of the network is graph-like (*i.e.*, edges connect two nodes), it is a different game altogether when it is not. More specifically, if the underlying relational structure of a network involves higher-order relations, it is quite unclear how close or similar two nodes would be in the presence of such higher sized "edges" (*i.e.*, hyperedges). To make these two points clearer, let us divert our attention to Figure 4.1(a). We see five authors A–E who are related to each other by *co-authorship*, which by nature possesses a higher-order property in that more than two authors can write a publication together. In this case, we see three co-authorship groups: ABC, $BCDE$, and DE, each corresponding to a collaboration between the respective authors. Some illustrative pieces of information that the graph on the right loses are: (1) How many papers were written to start with? (2) Who all collaborated with each other? (3) Which author has the tendency to collaborate with larger teams?

Moreover, on one hand, we see that it makes sense about how close or far away two nodes are to each other in a graph (say, authors A and E in Figure 4.1(b)) in terms of shortest-path or a node similarity mechanism based on the graph-structure. But on the other hand, while similarity between the authors could be calculated for the hypergraph on the left (Figure 4.1(b)) as well, it ought be done using the incident hyperedges lest we miss the subtle differences between nodes A and E, and their respective environments. For one, even though we know that A and E have two and three neighbors respectively, we don't know their "tendency" to collaborate with 3-author and 4-author groups. It is clear that there is a misrepresentation that reduces our ability to find the similarity between two nodes in such an scenario. As would be shown later, standard measures of node similarity do not transfer directly from the graph-domain to hypergraphs. While most of the literature on computing

node-similarity is focused on graphs, we deal with the same problem in the presence of hypergraphs[1]. Moreover we know that any given hypergraph could be converted into a graph (although lossily) by simple heuristics, *weighted/unweighted clique-expansion* (ref. Chapter 2) being one of them.

It is well-known that local similarity measures (*e.g.*, *common neighbors* [78]) prove to be powerful link predictors, either by themselves [66], or as classifier features [4]. However, while computing their values for a network, we usually ignore the underlying hypergraph structure, owing to the dyadic relation assumption (see Chapter 1, Section 1.1.1). This deprives the similarity measure of any extra information such higher-order structure could have otherwise contributed towards. In the present chapter, we exploit the underlying hypergraph structure of a given network and extend popular local (*i.e.*, neighborhood-based) node similarity measures to their higher-order variants. In essence, we leverage the topological properties of the underlying hypergraph of a derived network to aid the process of measuring the similarity between two nodes. We restrict our interest to local neighborhood based similarity measures and argue about the performance of their higher-order variants by explicitly predicting links using them. Our experiments include both temporal as well as non-temporal hypergraphs (see Chapter 2 for formal definitions).

We first provide a generic formulation to convert any neighborhood based pairwise similarity score to hypergraphs in Section 4.2. Then in Section 4.3, we describe procedures to carefully prepare data and hypergraph-oriented features so as to carry out experiments. In Section 4.5, we perform experiments, that include both temporal as well as non-temporal datasets. We also compute mutual information scores to vouch for the relevance of our measures, and devise different feature combinations to test them in a supervised learning scenario. And finally, in Section 4.6, we compare our AUC scores with that of the baselines.

4.1.1 Key Contributions

1. We formulate a **theoretically-backed novel technique** to convert graph topology-based pairwise node similarity measures into hypergraph-topology-based ones.

2. We **extend** the local neighborhood based node similarity scores to their **hypergraph variants**.

3. We propose **fair and unbiased novel data preparation** algorithms, so that similarity computation could be performed on both temporal and non-temporal hypergraph networks.

4. We **improve the quality of structural similarity between nodes** by incorporating hypergraphs and the scores we formulate.

[1]Please note that by "node similarity on hypergraphs", we still refer to pairs of nodes (*links/edges*), ***not*** hyperlinks/hyperedges.

4.2 Formulating Similarities

Social scientists have long been involved in finding metrics that describe relations between entities in a network. It was Liben-Nowell, et al. [66] who first formally accumulated topological similarity scores from the network science literature and showed that they are good measures by themselves. These similarity functions range from the earliest works of Katz [54] and Adamic, et al. [2] to the then recent works by Newman, et al. [76]; and till date, topological link prediction has undergone several advancements [72]. Nevertheless, though there have been several works pertaining to hypergraphs and their applications, no work in the literature (with exceptions to some works [64], which use uniform/heterogeneous hypergraphs) utilizes higher-order relations for similarity computation.

In this section, we formally extend similarity scores in the literature from the graph- to the hypergraph-domain. For the same, we define an end-to-end process of carefully constructing such scores from existing graph-topology based ones. We first generalize graph-based local neighborhood scores via defining set-similarity functions taken from well-known local link prediction heuristics, and then extend them to graphs (which is a usual, adjacency-based topology) and finally to a hypergraph (incidence-based) topology. Our ultimate goal is to be able to predict links between an unlinked pair $\{u, v\}$ of nodes given a hypergraph $\mathcal{H} = (V, \mathcal{F})$. In the literature, we find a plethora of techniques that make use of the existing graph structure, given graph $\mathcal{G} = (V, \mathcal{E})$. Most such techniques are set-based, in that they take some node subsets $S_u, S_v \subseteq V$ corresponding to the nodes u, v in consideration, and assign a prediction score to the pair $\{u, v\}$. Hence, we focus on set-based similarity measures, which we discuss next, taking "common neighbors" as a special case.

4.2.1 The Case of Common Neighbors

For a pair of nodes $\{u, v\}$, the Common Neighbors (CN) technique takes sets S_u and S_v to be *neighbors* of u and v respectively (*i.e.*, $S_u := \Gamma(u)$ and $S_v := \Gamma(v)$), and computes the cardinality of their intersection. In essence, CN makes use of two major concepts: "neighborhood", and "intersection". Now, if one were to compute the *hypergraph-equivalent to CN*, one would have to use a concept equivalent to "neighborhood". A simple option would be to consider "hyperneighborhood" (ref. Chapter 2) instead. In other words, the hypergraph equivalent of CN for u, v could be defined as the *number of hyperedges incident on both* u *and* v; but since the nodes are unlinked in the first place, there would be no common hyperedges! Thus, this option *fails*. This is because because while for graphs we have CN being "the number of common neighbors", for hypergaphs the same cannot be extended since the concept of "neighbor" is ambiguous, in that while in graphs the number of neighboring edges and neighboring nodes would be the same, in a hypergraph the same wouldn't be true. So one needs to define common neighbors explicitly for hypergraphs. Moreover, the CN criterion ought be defined

for hypergraphs in such a way that it could be vacuously equated to the existing CN for graphs (since every graph is a hypergraph). In other words, we still want to use the "common-neighbor" paradigm on hypergraphs.

Hence, first consider pairs of hyperedges $F_u, F_v \in \mathcal{F}$, such that one is incident to u ($u \in F_u$) and the other to v ($v \in F_v$), and count their "intersection" $|F_u \cap F_v|$. And since F_u and F_v are precisely the hyperneighbors ($\tilde{\Gamma}$) of u and v, we have combined the two concepts of "hyperneighborhood" and "intersection", and thus, have extended CN to hypergraphs. But since each choice of F_u, F_v would give a number $|F_u \cap F_v|$, we would have an $|\tilde{\Gamma}(u)| \times |\tilde{\Gamma}(v)|$ matrix of intersection-counts. A suitable *matrix norm* could then be used to convert this matrix into a single numerical quantity, which we would use as a feature for similarity computation.

Extending this formalization to all local-neighborhood [41] based similarity computation scores is the ultimate goal of this section. In order to do that, we define the notions of a *set similarity function* φ, a *node similarity function* ω, and a *node-similarity matrix-function* ψ. For ease of transference of any set-similarity notion to node pairs, we also define two functionals (functions that map functions to functions): an *adjacency functional* α (to use set-similarity functions as similarity computers in graphs) and an *incidence functional* $\|\xi\|$ (to use them as similarity computers in hypergraphs). An intermediate concept: *incidence matrix-functional* ξ has to be defined, so that it could be composed with a matrix norm to obtain incidence-based node-similarity measures.

4.2.2 Extending Similarities to Hypergraphs

Given a hypergraph $\mathcal{H} = (V, \mathcal{F})$, for each vertex pair $\{u, v\} \in \mathcal{P}_2(V)$, we define functions that quantify the proximity between vertices u and v.

Definition 4.1 (***Set Similarity Function***). *A **set similarity function**:*

$$\varphi : \mathcal{P}_2(\mathcal{P}(V)) \to \mathbb{R}, \qquad \{U, U'\} \mapsto \varphi(\{U, U'\}) \tag{4.1}$$

is a function that assigns to an unordered pair of vertex sets, $U, U' \subseteq V$, a real number $\varphi(\{U, U'\})$ corresponding to a measure of similarity between the sets. Let $\Phi := \mathbb{R}^{\mathcal{P}_2(\mathcal{P}(V))}$ represent all set-similarity functions over V.

Definition 4.2 (***Node Similarity Function***). *A **node similarity function** is defined as a function:*

$$\omega : \mathcal{P}_2(V) \to \mathbb{R}, \qquad \{u, v\} \mapsto \omega(\{u, v\}) \tag{4.2}$$

that assigns to a pair of nodes $u, v \in V$, a similarity score $\omega(\{u, v\}) \in \mathbb{R}$. Let $\Omega := \mathbb{R}^{\mathcal{P}_2(V)}$ denote the set of all node-similarity functions over V.

4. EXPLOIT BEFORE EXPANDING: LEVERAGING HYPERGRAPHS FOR MODELING NODE PAIRS

Table 4.1: Set Similarity Functions for a graph $\mathcal{G} = (V, \mathcal{E})$. Here, $U, U' \subseteq V$ and $\Gamma(u)$ represents the neighbors of a node $u \in V$. Refer to Chapter 2 for definitions of these functions.

Common Neighbor, φ_{CN}	$\lvert U \cap U' \rvert$
Jaccard Coefficient, φ_{JC}	$(\lvert U \cap U' \rvert)/(\lvert U \cup U' \rvert)$
Association Strength, φ_{AS}	$(\lvert U \cap U' \rvert)/(\lvert U \rvert \cdot \lvert U' \rvert)$
Cosine Similarity, φ_{CS}	$(\lvert U \cap U' \rvert)/(\sqrt{\lvert U \rvert \cdot \lvert U' \rvert})$
NMeasure, φ_{NM}	$(\lvert U \cap U' \rvert)/(\sqrt{\lvert U \rvert^2 + \lvert U' \rvert^2})$
MinOverlap, φ_{MnO}	$(\lvert U \cap U' \rvert)/(\min\{\lvert U \rvert, \lvert U' \rvert\})$
MaxOverlap, φ_{MxO}	$(\lvert U \cap U' \rvert)/(\max\{\lvert U \rvert, \lvert U' \rvert\})$
Adamic Adar, φ_{AA}	$\sum_{u \in U \cap U'} \frac{1}{\log(\lvert \Gamma(u) \rvert)}$
Pearson Correlation, φ_{PC}	$\frac{\lvert V \rvert \cdot \lvert U \cap U' \rvert - \lvert U \rvert \cdot \lvert U' \rvert}{\sqrt{(\lvert V \rvert \cdot \lvert U \rvert - \lvert U \rvert^2)\,(\lvert V \rvert \cdot \lvert U' \rvert - \lvert U' \rvert^2)}}$
Preferential Attachment, φ_{PA}	$\lvert U \rvert \cdot \lvert U' \rvert$

At this point, we also define an **adjacency functional**:

Definition 4.3 (***Adjacency Functional***)**.** *An adjacency functional is a function*

$$\alpha : \Phi \to \Omega, \qquad \varphi \mapsto \alpha(\varphi) \in \Omega, \qquad \alpha(\varphi)(\{u, v\}) := \varphi(\{\Gamma(u), \Gamma(v)\}), \tag{4.3}$$

that maps each set-similarity function $\varphi \in \Phi$ to a node-similarity function $\omega = \alpha(\varphi) \in \Omega$ defined as $\alpha(\varphi)(\{u, v\}) := \varphi(\{\Gamma(u), \Gamma(v)\})$.

In order to extend this successfully to hypergraphs, we define a **node-similarity matrix-function**:

Definition 4.4 (***Node-similarity Matrix-function***)**.**

$$\psi : \mathcal{P}_2(V) \to \mathcal{M}_{\mathbb{R}}, \qquad \{u, v\} \mapsto \psi(\{u, v\}) \tag{4.4}$$

is defined as a node-similarity matrix-function that assigns to a pair of nodes $u, v \in V$, multiple similarity scores arranged in a real matrix $\psi(\{u, v\}) \in \mathcal{M}_{\mathbb{R}}$, where $\mathcal{M}_{\mathbb{R}} := \bigcup_{m,n \in \mathbb{N}} \mathbb{R}^{m \times n}$ denotes the set of all real-valued finite-dimensional matrices. Let the set of all such functions for V be denoted by $\Psi := \mathcal{M}_{\mathbb{R}}^{\mathcal{P}_2(V)}$.

Then for a hypergraph, we define an **incidence matrix-functional**:

4. EXPLOIT BEFORE EXPANDING: LEVERAGING HYPERGRAPHS FOR MODELING NODE PAIRS

Definition 4.5 (***Incidence Matrix-functional***)**.**

$$\xi : \Phi \to \Psi, \qquad \varphi \mapsto \xi(\varphi) \in \Psi \tag{4.5}$$

that maps each set-similarity function $\varphi \in \Phi$ to a node-similarity matrix-function $\psi = \xi(\varphi) \in \Psi$ defined as:

$$\xi(\varphi)(\{u, v\}) := \begin{bmatrix} \varphi(F_1, F_1') & \cdots & \varphi(F_1, F_n') \\ \vdots & \ddots & \vdots \\ \varphi(F_m, F_1') & \cdots & \varphi(F_m, F_n') \end{bmatrix} \in \mathbb{R}^{m \times n}, \tag{4.6}$$

where $\tilde{\Gamma}(u) =: \{F_1, F_2, \ldots, F_m\}$ and $\tilde{\Gamma}(v) =: \{F_1', F_2', \ldots, F_m'\}$ are hyperneighbors of u and v respectively.

As discussed earlier, multiple matrix norms could be used to convert this matrix to a real number. Some of them are:

1. *Max-norm*: $\|X\|_{max} := \max_{i,j}\{|X_{ij}|\}$
2. *Avg-norm*: $\|X\|_{avg} := \frac{1}{m \cdot n} \sum_{i,j} |X_{ij}|$
3. *L1-norm*: $\|X\|_1 := \sum_{i,j} |X_{ij}|$
4. *L2-norm*: $\|X\|_2 := \sqrt{\sum_{i,j} |X_{ij}|^2}$.

Finally, the composition of a matrix norm with the incidence matrix-functional forms an **incidence functional**.

Definition 4.6 (***Incidence Functional***)**.** *The incidence functional*

$$\|\xi\| : \Phi \to \Omega, \qquad \varphi \mapsto \|\xi(\varphi)\|, \qquad \|\xi(\varphi)\|(\{u, v\}) := \|\xi(\varphi)(\{u, v\})\|, \tag{4.7}$$

is defined as $\varphi \mapsto \|\xi\|(\varphi) := \|\xi(\varphi)\| \in \Omega$, mapping pairs $\{u, v\} \in \mathcal{P}_2(V)$ to incidence-based similarities $\|\xi(\varphi)(\{u, v\})\| \in \mathbb{R}$.

Of the functionals defined above, the adjacency and the incidence functionals make use of *neighbors* and *hyperneighbors* respectively to transfer set-similarity functions to node-similarity in graphs and hypergraphs respectively.

4.2.2.1 Illustration with Common Neighbors

For the sake of further clarity, we demonstrate the case for Common Neighbors. We first pick $\varphi_{CN} \in \Phi$ (as defined in Table 4.1) as the set similarity function. The adjacency functional α maps φ_{CN} to a node similarity function $\omega_{CN} := \alpha(\varphi_{CN}) : \mathcal{P}_2(V) \to \mathbb{R}$, defined by $\{u, v\} \mapsto \varphi_{CN}(\Gamma(u), \Gamma(v)) := |\Gamma(u) \cap \Gamma(v)|$, the usual common-neighbor criterion for similarity computation in graphs. Moving to the incidence (hypergraph) domain, we first use the incidence matrix-functional ξ to map φ_{CN} to a node-similarity matrix-function $\psi_{CN} := \xi(\varphi_{CN}) : \mathcal{P}_2(V) \to \mathcal{M}_{\mathbb{R}}$. If $\tilde{\Gamma}(u) = \{F_1, \ldots, F_m\}$ and $\tilde{\Gamma}(v) = \{F'_1, \ldots, F'_n\}$, then $\xi(\varphi_{CN})(\{u, v\})$ is a matrix whose $(i, j)^{\text{th}}$ entry would be $\varphi_{CN}(F_i, F'_j)$. If a matrix norm such as $\|\cdot\|_{max}$ is used, we get the incidence functional, $\|\xi\|_{max}$, that gives us the node similarity function $\omega_{HCNM}(\{u, v\})$ (where $HCNM$ stands for *hypergraph-common-neighbor-max*) as

$$\|\xi\|_{max}(\varphi_{CN})(\{u, v\}) = \max_{(F,F') \in \tilde{\Gamma}(u) \times \tilde{\Gamma}(v)} \{|\varphi_{CN}(\{F, F'\})|\},$$

where $\varphi_{CN}(F, F') = |F \cap F'|$.

4.2.3 Sanity Check for Hypergraph-based Similarities

For the sake of establishing the sanity of the recently developed mechanism, we have the following lemma.

Lemma 4.1. *The Common-Neighbor set similarity function φ_{CN}, when used to define an incidence-based node similarity function $\|\xi(\varphi)\|$, for a graph $\mathcal{G} = (V, \mathcal{E})$, assigns to each pair $\{u, v\} \in \mathcal{P}_2(V)$, a similarity score that is proportional to a constant power of the original score. That is,*

$$\|\xi(\varphi)\|(\{u, v\}) = \lambda \cdot (\alpha(\varphi)(\{u, v\}))^{\beta}, \tag{4.8}$$

for at least one matrix norm $\|\cdot\|$, and for some scalars $\lambda, \beta > 0$.

Proof. Suppose we have $\mathcal{G} = (V, \mathcal{E})$ as a usual undirected graph. For nodes $u, v \in V$ s.t. $u \neq v$, if $\Gamma(u) := \{x_1, x_2, \ldots, x_m\}$ and $\Gamma(v) := \{y_1, y_2, \ldots, y_n\}$, we get hyperneighbors

$$\tilde{\Gamma}(u) = \{\{u, x\} \mid x \in \Gamma(u)\} \tag{4.9}$$

$$= \{\{u, x_1\}, \{u, x_2\}, \ldots, \{u, x_m\}\}. \tag{4.10}$$

Similarly,

$$\tilde{\Gamma}(v) = \{\{v, y_1\}, \{v, y_2\}, \ldots, \{v, y_n\}\}. \tag{4.11}$$

Now, if we take the matrix norm to be L1 ($\|\cdot\|$), we have:

$$\begin{aligned}\|\xi(\varphi_{CN})(\{u,v\})\|_1 &= \left\|(\varphi_{CN}(\{\{u,x_i\},\{v,y_j\}\}))_{\substack{1\le i\le m\\1\le j\le n}}\right\|_1 \\ &= \left\|(|\{u,x_i\}\cap\{v,y_j\}|)_{\substack{1\le i\le m\\1\le j\le n}}\right\|_1 \\ &= \sum_{\substack{1\le i\le m\\1\le j\le n}} \mathbb{1}(x_i=y_j) = |\Gamma(u)\cap\Gamma(v)| \\ &= \varphi_{CN}(\{\Gamma(u),\Gamma(v)\}) = \alpha(\varphi_{CN})(\{u,v\}) \end{aligned} \tag{4.12}$$

Taking different matrix norms, we get scores as shown in the table below (Table 4.2).

Table 4.2: Similarity scores between u and v (φ_{CN}) when hypergraph is actually a graph.

Norm	$\|\xi(\varphi_{CN})(\{u,v\})\|$
$\|\cdot\|_{max}$	$\mathbb{1}(\varphi_{CN}(\{\Gamma(u),\Gamma(v)\})>0)$
$\|\cdot\|_{avg}$	$\mathcal{D}\frac{1}{\lvert\Gamma(u)\rvert\cdot\lvert\Gamma(v)\rvert}\cdot\varphi_{CN}(\{\Gamma(u),\Gamma(v)\})$
$\|\cdot\|_1$	$\varphi_{CN}(\{\Gamma(u),\Gamma(v)\})$
$\|\cdot\|_2$	$\sqrt{\varphi_{CN}(\{\Gamma(u),\Gamma(v)\})}$

It could be observed that when $\|\cdot\|_1$ is used as matrix norm, φ_{CN} becomes the same for both hypergraphs and graphs (*i.e.*, $\lambda=\beta=1$). Scores from the other norms act as extra features that we get as a result of the "incidence matrix" interpretation of a graph. The same procedure when repeated for φ_{AA}, φ_{JC}, and other similarity scores gives us either the same graph score, or a scalar multiple of a power of it. □

Note: It needs to be understood that *complete equality is not required*, since ultimately, we use graph features along with hypergraph ones (macro/micro combinations for GH, WH, etc.). Also, the hypergraph based scores act as new features that come from the incidence matrix interpretation of the hypergraph (even if it is a graph).

4.3 Methodology

4.3.1 Data Preparation and Preprocessing

Given a hypergraph (temporal or non-temporal), we need to convert it into a form that is consumable in the similarity computation setting, so that we are able to calculate both graph- and hypergraph-based features readily. We prepare data separately for temporal and structural similarity computation settings.

Algorithm 2: STRUCTURALSPLIT($\mathcal{H}$, r) for hypergraph data

Input : Hypergraph $\mathcal{H} = (V, \mathcal{F})$
Split ratio $r \in [0, 1]$
Output : Train hyperedges $\mathcal{F}_{tr}$
Test links $\mathcal{E}_{te}$

$\mathcal{E} \leftarrow \{\}$
for $F \in \mathcal{F}$ **do**
  for $e \in \mathcal{P}_2(F)$ **do**
    if $e \notin \mathcal{E}$ **then**
      $\mathcal{E} \leftarrow \mathcal{E} \cup \{e\}$
$\mathcal{E}_{te} \leftarrow$ SAMPLE($\mathcal{E}$, $\lceil (1-r) \cdot |\mathcal{E}| \rceil$)
$\mathcal{F}_{tr} \leftarrow$ CLEANHYPEREDGES($\mathcal{F}$, $\mathcal{E}_{te}$)
return $\mathcal{F}_{tr}$, $\mathcal{E}_{te}$

4.3.1.1 Temporal Processing

In the temporal setting, we have a timed hypergraph, $\mathcal{H} = (V, \mathcal{F}, T_{\mathcal{H}})$ with us. Unweighted and weighted clique expansions of $\mathcal{H}$ give $\mathcal{G} = (V, \mathcal{E}, T_{\mathcal{G}})$ and $\mathcal{G}_w = (V, \mathcal{E}, T_{\mathcal{G}}, w)$ respectively. In short, we have timed graphs $\mathcal{G}$ and $\mathcal{G}_w$ with us now. Now, as described in Chapter 2, we prepare datasets $\mathbb{D}_{unsup}$ and $\mathbb{D}_{sup}$ ready for link prediction. The typical running time of a temporal preprocesser would be $\mathcal{O}(|\mathcal{F}| \cdot 2^{s_{max}})$, where $\mathcal{F}$ is the set of all hyperedges and $s_{max} := \max_{F \in \mathcal{F}} |F|$.

4.3.1.2 Structural Processing

Structural processing is easier than temporal, since hyperedges are non-timed. Formally, we start with a hypergraph $\mathcal{H} = (V, \mathcal{F})$, which gets converted into graphs $\mathcal{G} = (V, \mathcal{E})$ and $\mathcal{G}_w = (V, \mathcal{E}, w)$ as before. A similar split-ratio $\rho \in [0, 1]$ is selected, and if $m := |\mathcal{E}|$, we randomly delete $m_{te} := \lceil \rho \cdot m \rceil$ number of edges from the graph, which has to be predicted later. In other words, a random sample $\mathcal{E}_{te} \subseteq \mathcal{E}$ is selected such that $|\mathcal{E}_{te}| = m_{te}$. As a result, we get the set of test edges $\mathcal{E}_{te}$.

We now discuss the preparation of the train hypergraph, whose topology would be used while predicting links. In the temporal case, we simply ignored hyperedges $\mathcal{F}_{te}$ from the test period and what remained was $\mathcal{F}_{tr}$. But here, we have no temporal information, and the train-test split is done at random, which successfully separates $\mathcal{E}_{tr}$ from $\mathcal{E}_{te}$, but not $\mathcal{F}_{tr}$ from $\mathcal{F}_{te}$, since there are no well-defined concepts of "train period" or "test period" here.

Let us analyze the situation closely. Before continuing further, let us extend the hyperneighborhood function to edges: $\tilde{\Gamma} : \mathcal{E} \to \mathcal{P}(\mathcal{F})$ defined as $e \mapsto \tilde{\Gamma}(e) := \{F \in \mathcal{F} \mid e \subseteq F\}$. The question is: which hyperedges should be included in the train set so that information from them could be used while predicting test links $\mathcal{E}_{te}$?

Algorithm 3: CLEANHYPEREDGES($\mathcal{F}$, $\mathcal{E}$) to remove edge-information in $\mathcal{E}$ from hyperedges $\mathcal{F}$

Input : Set of hyperedges $\mathcal{F}$
Set of edges $\mathcal{E}$
Output : Cleaned-up hyperedges $\mathcal{F}_{-\mathcal{E}}$

$\mathcal{EF} : \mathcal{E} \to 2^{\mathcal{F}}$
$\mathcal{F}_{-\mathcal{E}} \leftarrow \mathcal{F}$
for $\{u, v\} \in \mathcal{E}$ **do**
  $\mathcal{EF}[\{u, v\}] \leftarrow \{F \in \mathcal{F} \mid u, v \in F\}$
for $\{u, v\} \in \mathcal{E}$ **do**
  for $F \in \mathcal{EF}[\{u, v\}]$ **do**
    $F_{-u} \leftarrow F \setminus \{u\}$
    $F_{-v} \leftarrow F \setminus \{v\}$
    $\mathcal{F}_{-\mathcal{E}} \leftarrow (\mathcal{F}_{-\mathcal{E}} \setminus \{F\}) \cup \{F_{-u}, F_{-v}\}$
    $\mathcal{EF}[\{u, v\}] \leftarrow \mathcal{EF}[\{u, v\}] \setminus \{F\}$
    for $w \in F \setminus \{u, v\}$ **do**
      if $\{u, w\} \in \mathcal{E}$ **then**
        $EF[\{u, w\}] \leftarrow (\mathcal{EF}[\{u, w\}] \setminus \{F\}) \cup \{F_{-v}\}$
      if $\{v, w\} \in \mathcal{E}$ **then**
        $\mathcal{EF}[\{v, w\}] \leftarrow (\mathcal{EF}[\{v, w\}] \setminus \{F\}) \cup \{F_{-u}\}$
return $\mathcal{F}_{-\mathcal{E}}$

Choosing all hyperedges $\mathcal{F}$ as $\mathcal{F}_{tr}$ would ***trivialize*** the very task of similarity computation and we would end up predicting all links with a 100% accuracy using only one feature: "common hyperneighbors"! And, on the other hand, using only those hyperedges that are not supersets of any test edge, i.e., $\mathcal{F}_{tr} = \{F \in \mathcal{F} \mid F \not\supseteq e \; \forall e \in \mathcal{E}_{te}\}$ would ***deprive*** us of many links that a "hyperedge *minus* a test edge" would have otherwise provided. We go with neither of the options and choose to "strip" each test edge off of a potential train hyperedge. A detailed procedure has been described in Algorithm 2, which in turn uses Algorithm 3 to clean away information about any test edge from the hyperedges, finally giving us a rich train hypergraph for similarity computation.

Finally, we get train hypergraph $\mathcal{H}_{tr} := (V, \mathcal{F}_{tr})$, and test edges $\mathcal{E}_{te}$. The similarity computation problem would be to predict new links (i.e., those not already present in $\mathcal{E}_{tr}$) using information from the hypergraph topology $\mathcal{H}_{tr}$; predictions will later be evaluated using test set $\mathcal{E}_{te}$. The running time of structural processing is much higher than that of temporal: $\mathcal{O}(|\mathcal{F}| \cdot 2^{s_{max}} + (1-r) \cdot |\mathcal{F}|^2 \cdot 2^{s_{max}}) = \mathcal{O}((1-r) \cdot |\mathcal{F}|^2 \cdot 2^{s_{max}})$

4.3.2 Computing Graph Features

We had earlier listed certain set-similarity functions in Table 4.1. Let use take the corresponding link predictors (let us call them ***base predictors***) from the literature [66, 41] and hence get ten different similarity computation scores for each pair of nodes in a given dataset. More specifically, we take the adjacency node similarity function $\alpha(\varphi_i)$ where α and φ are used as per Section 4.2 (where base predictor, $i \in \{AA, JC, AS, CS, NM, MnO, MxO, AA, PC, PA\}$) to find scores for each pair. We repeat this exercise for the edge-weighted version of the graph (using weighted scoring functions defined in [41]). Finally, for each hypergraph, corresponding to each base predictor, we have two different graph-based topological scores per node pair, which we denote by $\mathtt{G}$ (for unweighted graph) and $\mathtt{W}$ (for weighted graph) respectively.

4.3.3 Computing Hypergraph Features

Similar to Section 4.3.2, we also compute scores for the hypergraph-variations of the base-predictors. This involves computing the node-similarity matrix-function ψ for each of the set similarity functions mentioned in Table 4.1, followed by the application of the four matrix norms defined earlier to obtain a single numeric score for each pair. In summary, we compute the incidence node similarity function via $\|\xi(\varphi_i)\|$, where φ, ξ, and $\|\cdot\|$ are as defined earlier, and Table 4.1. For each hypergraph, corresponding to each base predictor, we have four different hypergraph-based topological scores per node pair, which we denote by $\mathtt{H_m}$, $\mathtt{H_a}$, $\mathtt{H_1}$, and $\mathtt{H_2}$ corresponding to the four matrix norms identified.

4.4 Related Work

Computing similarity scores has a vast literature, and covering it in whole is beyond the scope of the present chapter. The reader is redirected to some excellent review works [113, 70, 72], which provide an intelligible coverage of the similarity computation ecosystem. Although the concept wasn't new to network scientists, and there have been vintage works on predicting new relations in networks ([54]), the first formal work on similarity computation via link prediction could be credited to Liben-Nowell, et al. [66]. They brought together multiple similarity scores to solve the problem, scores both new and existing [54, 2, 78]. Ever since, many interesting directions to solve the similarity computation problem in networks were taken.

However, almost all works that use hypergraph networks (with the exception of Li et al. [64], who deal with heterogeneous, uniform hypergraphs only) do not consider the underlying hypergraph structure after the network gets expanded to a graph.

4.5 Experiments

4.5.1 Datasets

We use a multitude of hypergraph datasets, mainly from Benson, et al. [11], from where we pick six datasets.

4.5.2 Preprocessing Data and Computing Scores

We perform a lot of link-prediction experiments on a number of hypergraph datasets belonging to multiple real-world domains. Since data preparation is both a crucial step as well as one of our main contributions, it forms a major part in our methodology (Section 4.3.1) itself. We fix the split-ratio to be $r = 0.2$, and choose to *randomly* generate $p = 5$ times as many negative samples (non-links) as positive samples (links). For each hypergraph, we perform *both* temporal and structual link prediciton (ignoring the time information for the latter). We get train hyperedges $\mathcal{F}_{tr}$, test links $\mathcal{E}_{te}$, and test non-links $\hat{\mathcal{E}}_{te}$ as defined above.

For each pair $\{u, v\} \in \mathcal{E}_{te} \cup \hat{\mathcal{E}}_{te}$, we compute the ten base predictor scores, as mentioned in Section 4.3.2, taking $\mathcal{E}_{tr} := \eta(\mathcal{F}_{tr})$ (both weighted and unweighted clique expansion as described in Chapter 2, Section 3.6) as information for edges, hence preparing our baselines. Then, as explained in Section 4.3.3, we compute hypergraph-topology based scores that we have proposed. Towards the end, for each base predictor, we have a total of six different scores per node pair: *graph* ($\mathtt{G}$), *weighted-graph* ($\mathtt{W}$), *hypergraph-max* ($\mathtt{H_m}$), *hypergraph-avg* ($\mathtt{H_a}$), *hypergraph-L1* ($\mathtt{H_1}$), and *hypergraph-L2* ($\mathtt{H_2}$). And since there are a total of ten base predictors: *AA*, *AS*, *CN*, *Cos*, *PA*, *JC*, *MxO*, *MnO*, *NM*, and *Prn*, we finally get $6 \times 10 = 60$ different scores per node pair.

4.5.3 Calculating Mutual Information

Mutual information [91] has been shown to play a major role in similarity computation [103]. But we use it here in the classical sense, in that for each dataset, we find the mutual information score for each individual feature by binning its values via a log-binning (where consecutive bins are assigned on the base-10 log scale) mechanism since they are continuous values, with all of them being power-law distributed as opposed to normal. We monitor the MI scores for various number of bins and found that beyond a sufficiently large number of bins, the relative rank of the similarity computation features does not change. Hence, we fix the number of bins to be 2000.

4.5.4 Performing Link Prediction

Finally, we evaluate our similarity measures by performing link prediction in three different modes, which have been described as follows:

1. ***Standalone features***: In this mode, we simply use the predictor scores (`G`, `W`, $\texttt{H}_\texttt{m}$, $\texttt{H}_\texttt{a}$, $\texttt{H}_1$, $\texttt{H}_2$) calculated in Section 4.5.2 for similarity computation, *i.e.*, predict links via the unsupervised similarity computation paradigm similar to Liben-Nowell et al. [66]. At the end, we would have a total of 60 standalone scores. Although we did not expect to do better than the baselines in this mode, we still observe decent performances.

2. ***Micro-feature combination***: Here, we take various feature combinations, treating each of the ten base predictors separately. We have a total of five different feature combinations per base predictor: `mic-G`, `mic-W`, `mic-H`, `mic-GH`, `mic-WH`, where the first two correspond to singleton features `G` and `W`, and the last three to taking `H` individually, `G` and `H` together, and `W` and `H` together respectively (ref. Sections 4.3.2 and 4.3.3). In all, we have $10 \times 5 = 50$ micro feature combinations for each dataset.

3. ***Macro-feature combination***: This is similar to micro-feature combination, except *all* base predictors are taken together for each combination. That is, we take all graph-based features (`mac-G`), all weighted-graph-based features (`mac-W`), all hypergraph-based features (`mac-H`), and their combinations `mac-GH` and `mac-WH`. We have totally 5 macro-feature combinations for each dataset.

In case of *micro* and *macro* modes, we learn a classifier[1] (using both XGBoost [20] as well as Logistic Regression separately) to predict links (and get one classifier per feature combination), and in the *standalone* mode, the scores themselves are used as predictions. For the classification, we randomly split the prepared data $\mathcal{E}_{te} \cup \hat{\mathcal{E}}_{te}$ further into train and test, this time for classification[2]. Once we have the predictions by a feature combination, for evaluating performance, the predictions are compared with the labels (link/non-link) and ROC curves [24] are derived, which are finally summarized using Area Under ROC (AUC).

4.6 Results and Discussion

We perform the experiments listed in the previous section on all the six datasets, all base predictors. For *micro* and *macro* modes, we get a total of 50 and 5 classifiers respectively (one per feature combination), and the same number of AUC scores, and for the *standalone* mode, we have 60 different AUC scores. We run these experiments for a total of *five* times, so as to monitor the variance across different runs, since each experiment has at least one random step, *viz.*, sampling of non-links.

[1] We use a couple of specific classifiers to illustrate the fact that one combination of features work better than another. In practice, it is advisable to try out a number of different classifiers and choose the best performing one(s).

[2] Earlier, we had performed a train-test split in a temporal or a structural sense, which was a data preparation step. But here, the usual, supervised-learning oriented split of the *prepared* data into train and test has been performed

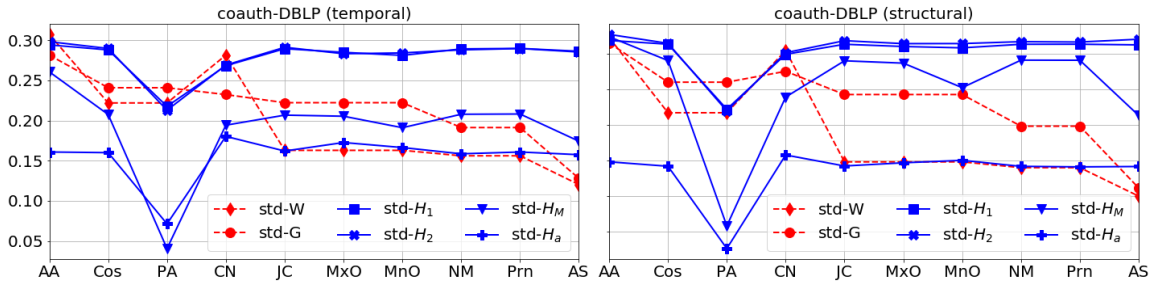


Figure 4.2: Mutual information scores denoting importance of six features (for all ten base predictors) in classifying links vs. non-links, computed on the coauth-DBLP hypergraph.

4.6.1 Mutual Information for Link Prediction

Treating each standalone score as a feature in a supervised setting, we compute their mutual information (MI) *w.r.t.* the positive (links) and negative (non-links) classes. For the dataset coauth-DBLP, we plot MI scores for both temporal and structural similarity computation for each base predictor. As could be observed, in the temporal case, except for AA, PA, and CN, where graph or weighted-graph MI outperforms the others, at least two hypergraph MI scores are better than the graph ones. This only means that hypergraph based scores have the potential to better *explain* links vs. non-links. We chose this dataset since it is the largest hypergraph we have used.

4.6.2 Micro Feature Combination Performances

As per the description of the *micro feature combination* mode in Section 4.5.4, we report AUC scores for the contact-high-school data in Table 4.3. It has to be interpreted as per various micro-feature combinations. As is clear from the highlighted numbers, except for Cos, JC, and MxO in the temporal similarity computation case (which perform best with `mic-W`), in all other cases, feature combinations involving hypergraphs (`mic-H`, `mic-GH`, `mic-WH`) work best.

A similar trend could be seen from the rank-performance table of the *micro* mode (Table 4.4), where at least one combination involving `H` ranks higher than the rest in each row. As compared with the analysis in the *standalone* mode, where individual features were used, the *micro* mode gives better scores; more so, when hypergraph features are involved.

4.6.3 Macro Feature Combination Performances

Finally, partitioning the features as per the *macro* mode in Section 4.5.4 gives us a total of five feature combinations, all of whose performances have been listed in Table 4.5. The hypergraph based features perform much better with these feature combinations. Even though `mac-H` underperforms the last two columns, compared with the purely graph oriented feature combinations (*mac-G* and *mac-W*), except

Table 4.3: AUC scores (%) for structural (-s) and temporal (-t) link prediction using micro-feature-combination for *contact-high-school* (*i.e.*, dataset B). Row ids AA–Prn represent base predictors.

	mic-G	mic-W	mic-H	mic-GH	mic-WH
AA-s	93.0±0.7	92.8±0.8	93.3±0.6	**93.4±0.5**	**93.4±0.6**
AS-s	91.5±0.8	88.3±0.6	93.3±0.4	**93.5±0.4**	93.4±0.5
CN-s	93.0±0.7	92.6±0.9	92.9±0.3	**93.2±0.4**	**93.2±0.4**
Cos-s	92.9±0.8	93.0±0.6	93.1±0.3	93.2±0.4	**93.5±0.5**
PA-s	62.3±0.9	60.9±1.6	62.3±1.5	62.6±1.2	**63.7±1.5**
JC-s	92.8±0.5	92.8±0.4	93.1±0.3	**93.3±0.2**	**93.3±0.3**
MxO-s	92.6±0.4	92.5±0.4	93.2±0.4	**93.3±0.4**	**93.3±0.3**
MnO-s	92.6±0.9	91.5±0.6	93.0±0.2	**93.3±0.7**	93.1±0.3
NM-s	92.8±0.5	92.5±0.3	93.2±0.3	93.3±0.3	**93.4±0.4**
Prn-s	90.9±0.6	90.8±0.8	93.2±0.3	93.2±0.3	**93.3±0.4**
AA-t	86.3±2.4	86.7±2.4	87.3±2.0	87.4±1.8	**87.9±2.2**
AS-t	85.9±1.3	84.0±1.4	**87.5±1.9**	86.9±1.8	87.2±1.9
CN-t	87.3±2.0	86.8±1.8	86.4±1.9	86.8±2.1	**87.4±2.1**
Cos-t	87.5±1.6	**88.1±2.0**	87.4±1.8	87.3±1.9	87.5±2.1
PA-t	53.6±2.1	54.0±2.3	52.4±3.6	55.0±2.7	**57.2±3.2**
JC-t	87.5±1.9	**88.4±1.9**	87.5±1.8	87.4±1.7	88.0±1.8
MxO-t	86.9±1.9	**87.7±1.3**	87.4±1.5	87.2±1.6	87.5±1.3
MnO-t	86.9±1.2	86.6±1.5	86.6±2.2	86.5±1.9	**87.7±1.8**
NM-t	86.6±2.0	87.6±1.4	87.3±1.8	87.2±1.7	**87.9±1.8**
Prn-t	84.0±2.6	83.7±2.2	87.3±2.0	87.1±2.2	**87.5±2.1**

Table 4.4: Rank-performances *w.r.t.* AUC scores from Table 4.3 across all datasets.

	mic-G	mic-W	mic-H	mic-GH	mic-WH
email-Enron-s	3.6±0.5	5.0±0.0	3.2±0.7	**1.4±0.5**	1.8±0.7
contact-high-school-s	4.0±0.4	4.9±0.3	3.0±0.4	1.7±0.5	**1.4±0.4**
NDC-substances-s	4.0±0.2	5.0±0.2	2.3±0.5	**1.8±0.2**	**1.8±0.2**
tags-math-sx-s	4.3±0.5	4.7±0.5	2.6±0.4	**1.3±0.5**	2.1±0.4
threads-math-sx-s	4.2±0.2	4.8±0.2	**1.9±0.3**	2.0±0.2	2.0±0.2
coauth-DBLP-s	3.1±0.3	3.2±0.6	3.0±0.0	**2.8±0.4**	**2.8±0.4**
email-Enron-t	4.8±0.4	4.1±0.5	3.0±0.4	2.0±0.4	**1.1±0.3**
contact-high-school-t	3.7±1.1	3.0±1.5	3.3±1.2	3.6±1.1	**1.4±0.5**
NDC-substances-t	3.3±0.6	3.3±0.6	2.8±0.3	2.8±0.3	**2.7±0.6**
tags-math-sx-t	4.3±0.5	4.5±0.9	2.8±0.7	2.1±0.5	**1.2±0.5**
threads-math-sx-t	4.2±0.7	4.7±0.4	2.2±0.5	2.1±0.5	**1.8±0.6**
coauth-DBLP-t	3.2±0.6	3.1±0.3	3.0±0.0	2.9±0.3	**2.8±0.6**

Table 4.5: AUC scores for link prediction performed using XGBoost on various feature combinations: G, W, H, GH, WH. Also shown are the t and p values for t-tests (with $5+5-2=8$ degrees of freedom) performed over the best graph-based (among mac-G and mac-W) and the best hypergraph-based (among mac-H, mac-GH, and mac-WH) link predictors for each dataset. As could be seen, for a 1% significance level, most datasets support the hypothesis that hypergraph-based scores outperform graph-based ones.

	mac-G	mac-W	mac-H	mac-GH	mac-WH	t	p
email-Enron-s	93.10±1.00	93.06±1.05	93.89±0.41	93.90±0.52	**94.10±0.54**	1.76	0.12
contact-high-school-s	93.40±0.46	93.54±0.41	93.46±0.65	93.59±0.72	**93.68±0.44**	0.47	0.65
NDC-substances-s	98.77±0.12	98.73±0.12	98.87±0.11	98.88±0.10	**98.89±0.15**	1.25	0.25
tags-math-sx-s	95.16±0.08	95.35±0.12	96.56±0.12	**96.60±0.09**	96.56±0.11	16.67	10^{-7}
threads-math-sx-s	96.90±0.15	96.86±0.14	97.19±0.14	**97.20±0.16**	97.19±0.15	2.74	0.03
coauth-DBLP-s	97.79±0.02	97.79±0.02	99.51±0.00	**99.52±0.00**	99.51±0.00	173.00	10^{-15}
email-Enron-t	74.44±1.50	78.29±1.64	79.05±2.14	79.56±1.83	**84.76±1.40**	6.00	10^{-3}
contact-high-school-t	86.64±1.87	87.92±1.67	87.31±1.93	86.96±1.81	**88.46±1.68**	0.46	0.66
NDC-substances-t	58.89±0.06	59.15±0.05	61.01±0.07	61.08±0.06	**61.41±0.07**	52.54	10^{-11}
tags-math-sx-t	90.80±0.40	91.63±0.35	91.31±0.34	91.53±0.33	**92.23±0.30**	2.60	0.03
threads-math-sx-t	84.66±0.25	84.95±0.27	90.59±0.14	90.59±0.13	**90.77±0.15**	37.69	10^{-10}
coauth-DBLP-t	85.29±0.04	86.00±0.04	87.93±0.04	88.00±0.04	**88.44±0.05**	76.21	10^{-12}

for B-s, B-t, and D-t, it performs better. Moreover, we see that the choice of classifier does not affect the scores consistently across datasets, from Table 4.6, and hence it is recommended to try out a bunch of classification algorithms before finalizing on the best performing one.

4.6.4 Standalone Feature Performances

For link prediction experiments in the *standalone* mode (Section 4.5.4), we show results only for a single dataset: *contact-high-school* (dataset B) in Table 4.7.

Although we did not expect to do better than the baselines in the standalone mode, since individual hypergraph scores might not be powerful link predictors, yet we observe decent performances in the last four columns (the only ones that correspond to hypergraph-based scores). To make this point clearer, we know that hypergraphs are more informative than graphs, but we do not make use of the granular information up to the extent graphs do; hence the standalone scores are not expected to perform by themselves. Moreover, our goal was not to "replace" the existing graph-topology based scores with hypergraph-topology ones, but to "aid" the graph-based scores. In other words, we have suggested some new features for the link prediction problem which do not outperform the existing ones when seen individually, but when considered together, they contribute towards a better performance. Going by a base predictor individually (row-wise), graph-versions (std-G) of Adamic Adar (AA) for structural- and Cosine Similarity (Cos) for the temporal-mode perform best.

We consolidate these results for all datasets by finding the mean (over all base predictors) "rank" among all standalone modes (std-G, std-W, std-H_m, std-H_a, std-H_1, std-H_2) in Table 4.8. This is

Table 4.6: AUC scores for Logistic Regression (LR) for the six datasets, and corresponding t and p values for one-sided t-tests (with $5 + 5 - 2 = 8$ degrees of freedom) performed over the best XGBoost (XGB) scores from Table 4.5 and the best Logistic Regression (LR) based ones. As could be seen, all datasets (except for one for which we have a negative t value) support the hypothesis "LR performs better than XGB" on an average. Moreover, since we have low p values in the temporal case, it could also be said that LR outperforms XGB by a greater margin in these cases. The conclusion we arrive at is that different classification methods could either perform better or worse than each other, but on an average, the hypergraph scores outperform the graph ones.

	mac-H	mac-GH	mac-WH	t	p
email-Enron-s	93.52 ± 0.48	94.32 ± 0.46	93.32 ± 0.5	0.62	0.55
contact-high-school-s	92.85 ± 0.63	94.61 ± 0.76	93.52 ± 0.35	2.12	0.07
NDC-substances-s	98.22 ± 0.14	98.13 ± 0.12	99.08 ± 0.18	1.62	0.14
tags-math-sx-s	94.47 ± 0.03	96.51 ± 0.02	96.72 ± 0.07	2.10	0.07
threads-math-sx-s	94.92 ± 0.17	96.89 ± 0.2	96.54 ± 0.22	-2.42	0.04
coauth-DBLP-s	99.95 ± 0.0	99.76 ± 0.0	99.21 ± 0.0	860000.00	10^{-44}
email-Enron-t	78.03 ± 2.14	78.87 ± 1.78	85.27 ± 1.34	0.53	0.61
contact-high-school-t	91.58 ± 1.88	86.4 ± 1.82	88.88 ± 1.67	2.47	0.04
NDC-substances-t	63.49 ± 0.1	62.88 ± 0.13	59.03 ± 0.02	34.08	10^{-9}
tags-math-sx-t	91.84 ± 0.32	92.03 ± 0.31	93.9 ± 0.24	8.69	10^{-5}
threads-math-sx-t	92.1 ± 0.05	91.4 ± 0.18	89.9 ± 0.08	16.82	10^{-7}
coauth-DBLP-t	91.11 ± 0.06	88.01 ± 0.02	88.26 ± 0.0	68.37	10^{-12}

Table 4.7: AUC scores (%) for structural (-s) and temporal (-t) link prediction using standalone features for *contact-high-school*. Row ids AA–Prn represent base predictors.

	std-G	std-W	std-H_m	std-H_a	std-H_1	std-H_2
AA-s	**93.0±0.5**	92.8±0.3	89.1±0.3	92.1±0.3	92.4±0.4	92.6±0.4
AS-s	91.2±0.3	88.1±0.2	69.7±0.3	91.9±0.3	92.6±0.4	**92.8±0.4**
CN-s	**92.8±0.5**	92.4±0.3	77.3±0.4	92.0±0.3	92.2±0.4	92.2±0.4
Cos-s	**92.8±0.4**	**92.8±0.2**	77.9±0.3	92.0±0.3	92.4±0.4	92.6±0.4
PA-s	**63.6±0.6**	62.0±0.8	55.0±0.4	56.0±1.2	62.6±0.8	62.1±0.9
JC-s	**92.8±0.4**	**92.8±0.3**	77.9±0.3	92.0±0.3	92.5±0.4	92.7±0.4
MxO-s	92.6±0.4	92.6±0.3	77.7±0.3	92.0±0.3	92.5±0.4	**92.7±0.4**
MnO-s	**92.4±0.3**	91.2±0.1	77.3±0.4	92.0±0.3	92.3±0.4	**92.4±0.4**
NM-s	**92.8±0.4**	92.7±0.3	77.9±0.3	92.0±0.3	92.5±0.4	92.6±0.4
Prn-s	90.6±0.4	90.1±0.2	77.9±0.3	92.0±0.3	92.4±0.4	**92.6±0.4**
AA-t	87.9±0.3	87.8±0.3	83.5±0.2	**88.0±0.2**	86.7±0.3	87.0±0.3
AS-t	87.5±0.2	84.8±0.3	67.1±0.3	**88.0±0.2**	87.1±0.3	87.5±0.3
CN-t	87.7±0.3	87.4±0.3	72.2±0.3	**87.9±0.2**	86.5±0.3	86.5±0.3
Cos-t	**88.5±0.2**	**88.5±0.2**	73.3±0.3	88.0±0.2	86.8±0.3	87.1±0.3
PA-t	**53.8±0.3**	**53.8±0.3**	51.2±0.4	50.3±0.4	52.5±0.4	52.3±0.4
JC-t	**88.4±0.2**	**88.4±0.2**	73.2±0.3	88.0±0.2	86.9±0.3	87.2±0.3
MxO-t	87.9±0.2	**88.0±0.2**	72.6±0.3	**88.0±0.2**	86.9±0.3	87.2±0.3
MnO-t	**88.2±0.2**	87.1±0.2	72.2±0.3	88.0±0.2	86.7±0.3	86.8±0.3
NM-t	**88.3±0.2**	88.2±0.2	73.2±0.3	88.0±0.2	86.8±0.3	87.1±0.3
Prn-t	85.8±0.2	85.0±0.2	73.3±0.3	**88.0±0.2**	86.8±0.3	87.1±0.3

Table 4.8: Rank-performances *w.r.t.* AUC scores from Table 4.7 (standalone mode) across all datasets. Here, -s and -t refer to structural and temporal respectively.

	std-G	std-W	std-H_m	std-H_a	std-H_1	std-H_2
email-Enron-s	**1.3±0.6**	3.8±1.0	3.0±1.5	3.2±1.3	5.4±1.3	4.3±1.0
contact-high-school-s	**1.8±1.2**	3.2±1.3	6.0±0.0	4.5±0.8	3.2±0.9	2.2±1.0
NDC-substances-s	3.2±1.1	4.1±1.3	**1.4±1.2**	5.8±0.6	3.7±1.0	2.8±0.9
tags-math-sx-s	4.0±1.1	4.3±1.6	3.4±0.8	5.7±0.6	2.2±0.6	**1.4±0.7**
threads-math-sx-s	4.1±1.3	3.8±0.9	3.0±1.1	5.8±0.6	**2.2±1.0**	**2.2±0.7**
coauth-DBLP-s	3.4±0.4	3.4±0.2	3.6±0.4	3.8±0.8	**3.2±0.8**	3.6±0.2
email-Enron-t	2.9±0.7	**1.6±1.2**	3.8±1.2	2.1±0.7	5.8±0.4	4.8±0.4
contact-high-school-t	**2.0±0.9**	2.7±1.3	5.9±0.3	2.2±1.5	4.4±0.8	3.6±0.8
NDC-substances-t	3.2±1.4	3.1±1.6	**2.6±1.0**	4.4±0.9	4.1±1.2	3.5±1.0
tags-math-sx-t	4.0±0.9	4.0±1.6	3.6±0.9	5.8±0.6	**1.8±0.5**	**1.8±0.7**
threads-math-sx-t	4.1±1.3	3.6±1.1	3.6±0.9	5.8±0.6	**1.6±0.8**	2.2±0.6
coauth-DBLP-t	3.4±0.4	**3.2±0.8**	3.6±0.4	3.8±0.8	3.4±0.2	3.6±0.2

how it has to be interpreted: for example, for dataset A, in the structural mode rank of `std-G` being 1.3 ± 0.6 means out of the six standalone modes, `std-G` stands at a mean position of 1.3 (with variance 0.6), when evaluated across all base predictors.

4.7 Conclusion

Structural (topological) node similarity scores have a long history in similarity computation, and have been equally successful as well. Also, hypergraph networks are very frequently used in works involving similarity computation, albeit not being exploited for the task per se. We set out to use the underlying hypergraph structure of networks to generate new features for similarity computation. Apart from establishing a strong theoretical foundation by devising functional templates that could help standard similarity computation scores getting translated from graphs to hypergraphs, we are also able to elucidate hypergraphs' contribution in predicting links. We perform a number of experiments to show the importance of using hypergraph-based topological features for similarity computation, including showing a mutual-information based perspective. A few take-away messages are:

1. Higher-order structure does have richer information than graphs.
2. When available, using the underlying hypergraph structure would term fruitful in link prediction.
3. Various matrix norms combine hyperedge information in different ways; the best bet is to use multiple norms and choose the best.
4. Unless the similarity computation model overfits, all hypergraph features should be used, if possible.

Chapter 5

Sample Hard, Learn Harder: Negative Sampling in Hyperlink Prediction

> *"I bow to the saints, who are even-minded towards all and have no friend or foe, just as a flower of good quality placed in the palm of one's hands communicates its fragrance alike to both the hands. . . . Again, I greet with a sincere heart the malevolent class, who are hostile without purpose even to the friendly, to whom others' loss is their own gain, and who delight in others' desolation and wail over their prosperity."*
>
> ∼ *Goswami Tulasīdās*

HYPERLINK prediction refers to the problem of predicting higher-order relations (hyperedges) in hypergraphs. Being a two-class classification problem, it treats hyperlinks and non-hyperlinks as "positive" and "negative" classes respectively. However, just as is the case with link prediction, hyperlink prediction too suffers from the problem of extreme class imbalance. Given this context, "negative sampling" – under-sampling the negative class of non-hyperlinks – becomes mandatory for performing hyperlink prediction. No prior work on hyperlink prediction [121, 132, 133] deals with this problem; in this chapter, we deal with it explicitly. More specifically, we leverage graph sampling techniques for sampling non-hyperlinks in hyperlink prediction. Our analysis clearly establishes the effect of random sampling, which is the norm in both link- as well as hyperlink-prediction. Further, we formalize the notion of "hardness" of non-hyperlinks via a measure of density, and analyze its distribution over various negative sampling techniques. We experiment with some real-world hypergraph datasets and provide both qualitative and quantitative results on the effects of negative sampling. We also establish its importance in evaluating hyperlink prediction algorithms.

5.1 Introduction

Although the problem of hyperlink prediction (HLP) has not been explored much, we have enough literature on the topic [11, 121, 132, 131] (and much more on its graph-variant, *viz.*, link prediction (LP) [66, 69, 72, 114]) to vouch for its importance. When posed as a supervised learning problem, where "presence of hyperedge" and "absence of hyperedge" are the *positive* and *negative* classes respectively, HLP suffers from extreme class imbalance (ECI), with positive class being the minority one. ECI haunts LP too, and has been thoroughly discussed in the literature as well [33, 67, 68, 126], but with HLP, the situation is much worse owing to the arbitrariness in the number of nodes allowed in a hyperedge. Hence, the solutions provided to combat ECI in usual networks (graphs) for LP could not be directly extended to HLP, at least not without a careful analysis thereof.

We consider the Southern women club (SWC) social hypergraph from Davis et al. [23] illustrated in Figure 5.1 that connects eighteen women through twelve hyperlinks, each corresponding to an event they had attended together. All non-hyperlinks from the hypergraph have been plotted in Figure 5.2, with the color shade in each vertical bar denoting edge-density (ref. Definition 5.1) distribution for a given hyperedge size. Although this being a dense hypergraph is atypical of real-world hypergraphs,

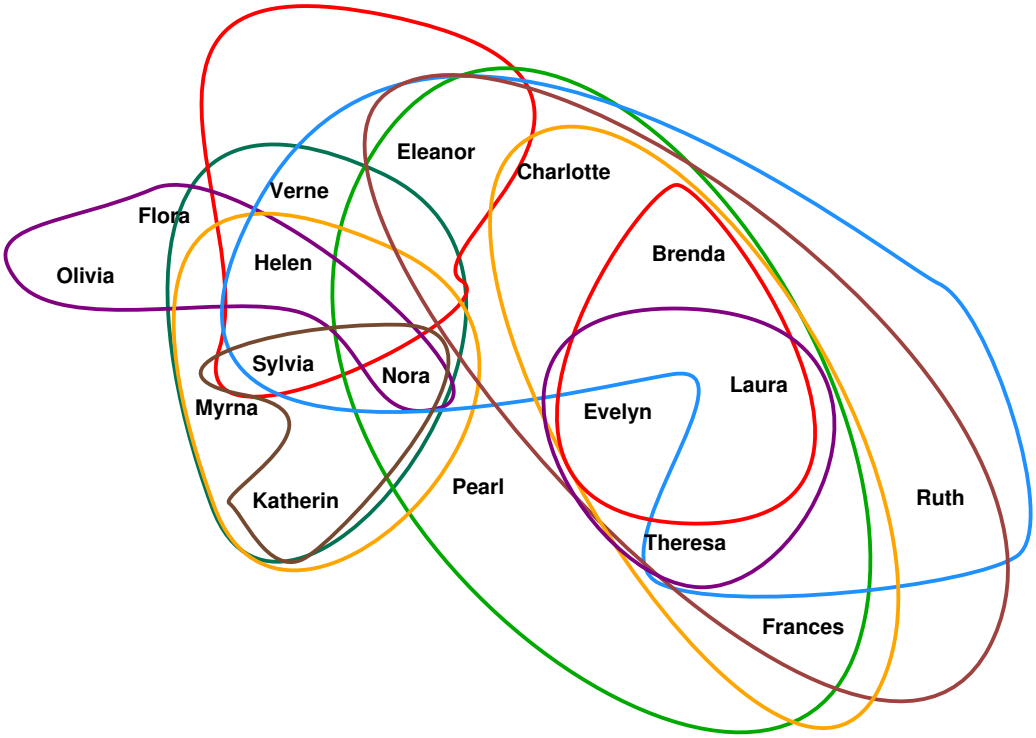


Figure 5.1: The Southern Women Club (SWC) hypergraph by Davis et al. [23]. For a more fruitful analysis, we have excluded one 11-sized hyperedge from the original hypergraph.

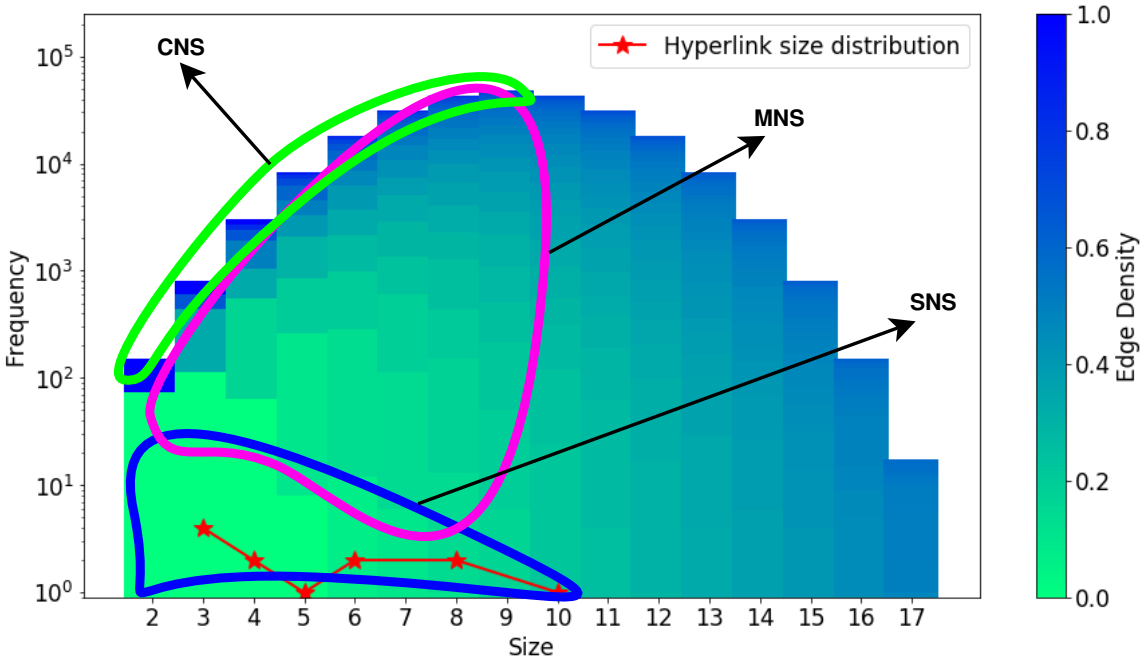


Figure 5.2: Edge density and size distribution of non-hyperlinks in the SWC events hypergraph.

we could notice the existence of zero-density non-hyperlinks in the left bottom corner of the plot. However, even for such small a hypergraph, one could compare the positive class size (denoted by red asterisks) *w.r.t.* that of the negative class (the entire histogram), only to reinforce the existence of ECI in a HLP problem.

Of all the solutions available in the literature to treat ECI, *majority-class sub-sampling* (here, *negative sampling* (NS)) is the one that has been prescribed strongly. Other methods (*e.g.*, minority-class over-sampling [19]) further increase the burden on HLP by necessitating computation of prediction scores for each point in the over-sampled positive class as well as those in the negative class (which is already huge in number). Where on one hand, NS makes the HLP problem computationally tractable, on the other, it poses the danger of misinterpretability of results (comparing two HLP algorithms, for instance) due to test set undersampling. The threat has been thoroughly argued about by Lichtenwalter et al. [67, 68] and Yang et al. [126] for LP. One technique that works towards nullifying this effect is "multiple" negative samplings.

In this chapter, we provide an extensive analysis of NS for HLP, but since a hypergraph has enormous number of negative patterns, our analysis is limited to a handful of NS algorithms. We propose four different approaches for NS: *Uniform Negative Sampling* (UNS), *Sized Negative Sampling* (SNS), *Motif Negative Sampling* (MNS), and *Clique Negative Sampling* (CNS), with the last three of them focused on the regions bounded by blue, pink and green boundaries in Figure 5.2. Of the four,

UNS and SNS are both motivated by random NS in LP [4], and have already been used in the literature to predict new recipes [132] and new email interactions [121]. We derive MNS from a motif-based representative subgraph sampling [52, 1] and CNS is our attempt towards developing a 1-hop based equivalent of NS [67] to HLP. For our analysis, we consider the "temporal" HLP problem, wherein we take a "past" snapshot of hypergraph $\mathcal{H}$ (or *observed* hypergraph), and predict "future" (*unobserved*) hyperlinks.

5.1.1 Key Contributions

1. We show the **importance of negative sampling in hyperlink prediction**, something that hasn't been done in the past.

2. We contribute towards **two negative sampling benchmarks** that could be used by hyperlink prediction practitioners in the future.

3. We perform **thorough cross-validation analysis of models learnt using various negative sampling techniques** to show their effect on hyperlink prediction.

5.2 Methodology

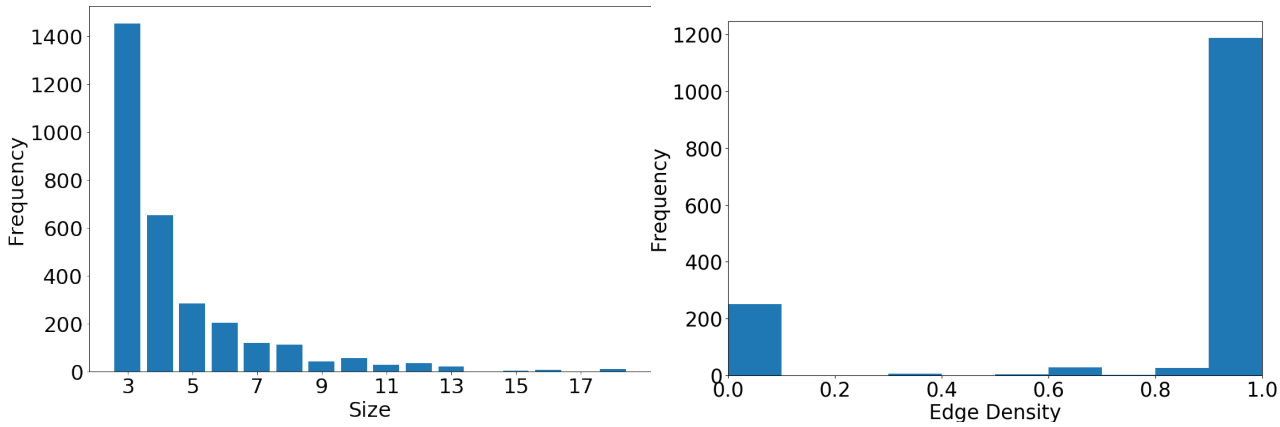
5.2.1 Characterizing Hardness of Prediction

Yang et al. [126] suggest avoiding sampling the test data as much as possible, so that LP could be evaluated fairly. But under unavoidable circumstances, test set has to be sampled, although we propose doing so not without acknowledging some notion of "hardness" in predicting hyperlinks.

Benson et al. [11] point out several properties of a hyperlink $F = \{v_1, \cdots, v_s\}$ that play a key role in its evolution in a hypergraph $\mathcal{H} = (V, \mathcal{F}, \tau)$ over time: (i) the *connectivity* among its incident nodes $v_1, \cdots, v_s$ in the projected graph $\eta(\mathcal{H})$ right *before* F was formed, and (ii) the *strength* of these connections. These observations can be generalized to arbitrary-sized hyperlinks through the notion of a hyperlink's *edge-density* (ED) defined as per Definition 5.1.

Since ED plays an important role for hyperlink evolution, it could be used to characterize the "hardness" of HLP. In layman terms, hardness in predicting the true class of a test non-hyperlink $F \in \hat{\mathcal{F}}$ denotes how hard it is to predict F as a pattern from the negative class. Let us formally define it as follows:

Definition 5.1 (***Hardness of a non-hyperlink***). *Given a HLP dataset* $(\mathcal{H}_{tr}, \mathcal{F}_{te}, \hat{\mathcal{F}}_{te})$*, the hardness* $h : \hat{\mathcal{F}} \rightarrow [0, 1]$ *of predicting the true class of a non-hyperlink* $\hat{F} \in \hat{\mathcal{F}}_{te}$ *is defined as one being*



(a) Size distribution (SD)

(b) Edge-density distribution (EDD)

Figure 5.3: Size distribution (SD) and edge-density distribution (EDD) of hyperlinks – the positive class – in `email-Enron`.

proportional to its ***edge-density*** $ed(\hat{F}; \mathcal{H}_{tr})$ *defined as:*

$$h(\hat{F}) \;\propto\; ed(\hat{F}; H_{tr}) := \frac{2 \cdot \left|\eta(\hat{F}) \cap \eta\left(\mathcal{F}_{tr}\right)\right|}{|\hat{F}| \cdot \left(|\hat{F}| - 1\right)}, \tag{5.1}$$

Figures 5.3a and 5.3b respectively show the hyperedge size distribution (SD) and the edge-density distribution (hardness) as defined in eq. (5.1).

5.2.2 Uniform Negative Sampling (UNS)

This is the easiest of the four NS algorithms we describe in this section. For a hypergraph $\mathcal{H} = (V, \mathcal{F})$, the UNS algorithm picks a sample of k non-hyperlinks $\hat{\mathcal{F}}_{sam}$ uniformly at random from the set of all non-hyperlinks $\hat{\mathcal{F}}$. The non-hyperlink sizes of $\hat{\mathcal{F}}_{sam}$ are expected to be binomially distributed, which could be validated by Figure 5.4, which shows the size-distribution (SD) of non-hyperlinks in $\hat{\mathcal{F}}_{sam}$ for one dataset (`email-Enron`). Figures 5.3a and 5.4 show SD of the positive class and negative class (as sampled by UNS) respectively, which can be compared to see how different they are from each other.

As is clear from Figure 5.4, UNS substantially "blows-up" the non-hyperlink sizes, where its median would be around $|V|/2$, which for a 1000-node network amounts to 500-node non-hyperlinks, which is impractical for almost all applications. In `email-Enron` alone, one one hand, the *largest* observed hyperedge contains 18 nodes (see Figure 5.3a), whereas the *mean* non-hyperedge size is a whopping 72 (see Figure 5.4). Such a scenario, when seen from the perspective of HLP, ends up with a single trivial feature discriminator, *viz.*, "hyperlink size", which successfully separates the positive

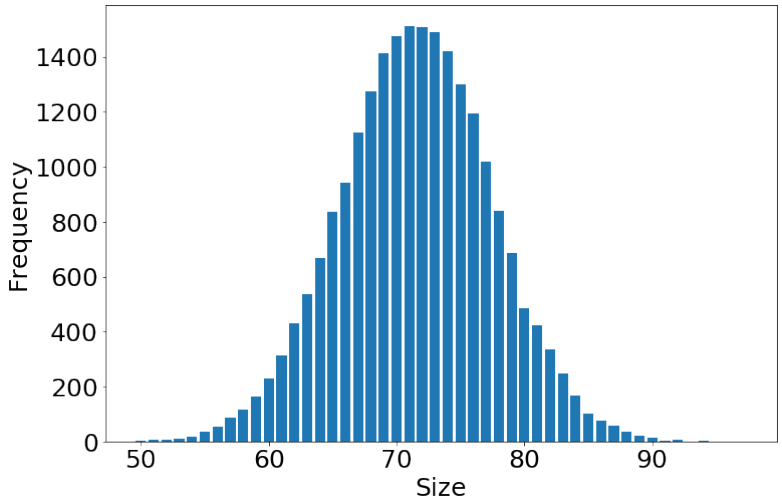


Figure 5.4: Size distribution (SD) of the sampled negative class of non-hyperlinks sampled via UNS for `email-Enron`. It is to be compared with the size distribution of the positive class of hyperlinks plotted in Figure 5.3a.

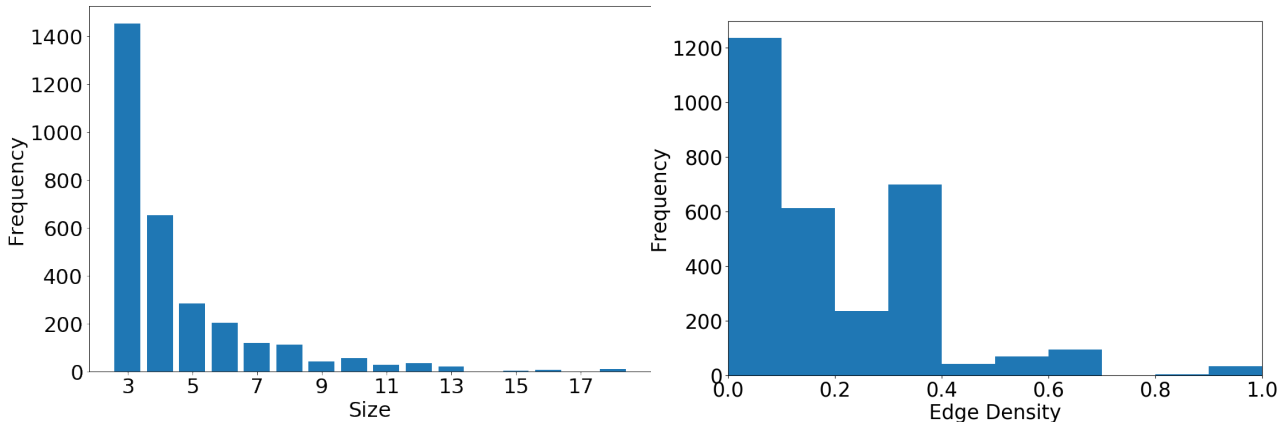
class from the negative, since the size-distribution for the former is fairly left-skewed, and for the latter, it is binomial. Since this renders UNS as a technique that generates one of the easiest-to-discriminate negative samplings, we limit our discussion on UNS merely to theory, and recommend ***it never to be used in practice***. Neither do we conduct any HLP experiments for UNS in this thesis.

If it were for UNS, it would sample non-hyperlinks akin to the negative class shown in Figure 5.2, uniformly at random.

5.2.3 Sized Negative Sampling (SNS)

SNS overcomes the shortcomings of UNS by sampling non-hyperlinks such that their SD matches that of hyperlinks. SNS is a slight variant of UNS in that the target SD (*i.e.*, that of the sampled negative class) $Pr_{-}(S = s)$ is fixed to that of the positive class ($Pr_{+}(S = s)$), and not a binomial as per Figure 5.4. Once a size s has been sampled according to $Pr_{+}(S = s)$, a non-hyperlink is sampled randomly. The SD of non-hyperlinks sampled with SNS exactly follows the positive class SD (see Figure 5.5a).

Since SNS fixes the "size-blow-up" issue in sampled non-hyperlinks, it should ideally be the one-stop solution to negative sampling. But there is yet another problem – a subtler one at it: since real-world hypergraphs are heavily sparse (much sparser than graphs), sampling non-hyperlinks via SNS biases the binary classification problem *w.r.t.* the challenge in predicting the true class of a test non-hyperlink. As we have already characterized the *hardness* of predicting the true class of a non-hyperlink in Definition 5.1 via ED, we could monitor the edge-density distribution (EDD) of



(a) Size distribution (SD)

(b) Edge-density distribution (EDD)

Figure 5.5: Size distribution (SD) and edge-density distribution (EDD) of non-hyperlinks – the negative class – in `email-Enron` sampled using Sized Negative Sampling (SNS).

non-hyperlinks sampled via SNS against hyperlinks.

Figure 5.5b shows the edge-density distribution (EDD) of non-hyperlinks sampled via SNS. It can be seen that most of these non-hyperlinks (negative patterns) have low ED, which makes it "easy" for a HLP algorithm to reject them as positive patterns, whose EDD has been plotted in Figure 5.3b. Since most hyperlinks have a high ED, it could be assumed that ED among an arbitrary set of nodes has a positive correlation with their probability of forming hyperlinks in the future. The positive class EDD (Figure 5.3b) shows that in most cases, incident nodes of a test hyperlink are well-connected with each other – a pattern not observed for non-hyperlinks sampled via SNS (Figure 5.5b). Hence, ***an SNS based positive-negative split not only poses little challenge to a predictor trained on such a dataset, but also misleads HLP evaluation.***

An SNS algorithm would sample non-hyperlinks from within the "blue" enclosure depicted in the bottom left corner of Figure 5.2. This paves way for yet another NS algorithm: MNS.

5.2.4 Motif Negative Sampling (MNS)

The hardness of predicting a non-hyperlink $\hat{F} \in \hat{\mathcal{F}}$ to be of the negative class (*i.e.*, True Negative Rate) depends upon the intra-connectivity structure of $\hat{F}$. We have seen that ***SNS trivializes the HLP problem by sampling low-density non-hyperlinks*** thereby skewing the EDD for negative class towards the left (Figure 5.5b). This makes it easy for an HLP algorithm to discriminate it with the positive class (for which the EDD is skewed towards the right (Figure 5.3b)). To address this issue, we propose an approach that samples *connected subgraph components* (CCs) from the clique-expanded graph $\eta(\mathcal{H}_{tr})$ of $\mathcal{H}_{tr}$. The nodes of these CCs then form the sampled non-hyperedge.

5. SAMPLE HARD, LEARN HARDER: NEGATIVE SAMPLING IN HYPERLINK PREDICTION

Algorithm 4: The MNS algorithm

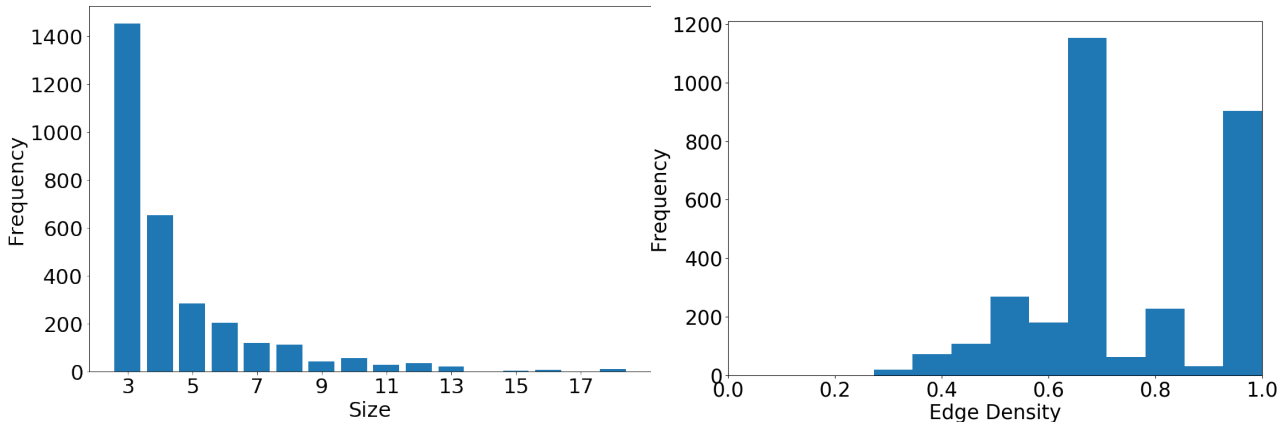
Input: A hypergraph $\mathcal{H} = (V, \mathcal{F})$ and size s of the non-hyperlink to be sampled
Output: Sampled non-hyperlink $\hat{F}$

$\mathcal{E} = \eta(\mathcal{F})$ // Edges of induced graph $\eta(\mathcal{H})$
$e_0 = \text{RANDOMCHOICE}(\mathcal{E})$ // sample initial edge uniformly at random
$\hat{F} = \{u \mid u \in e_0\}$ // set $\hat{F}$ to nodes of initial edge e_0
while $|\hat{F}| < s$ **do**
  $\mathcal{E}' = \{e \in \mathcal{E} : |e \cap \hat{F}| = 1\}$
  if $\mathcal{E}' = \emptyset$ **then**
    go to 2
  $e = \text{RANDOMCHOICE}(\mathcal{E}')$
  $\hat{F} = \hat{F} \cup \{u \mid u \in e\}$
return $\hat{F}$

We propose Motif Negative Sampling (MNS) that uses *Mfinder* [52], which is a stochastic algorithm used to estimate the *concentration* of a particular motif in a graph without exhaustive enumeration. Our aim here is to sample non-hyperlinks that are harder to reject by an HLP algorithm, as compared to those sampled by SNS.

Algorithm 4 samples a non-hyperlink of size s by sampling a s-connected component from the underlying graph $\eta(\mathcal{H})$ of a hypergraph $\mathcal{H}$. It first selects an edge at random and adds its incident nodes to the sample; and keeps adding more and more nodes adjacent to the already-sampled ones. At each step, all edges that contain exactly one sampled node are considered as the *candidate* set of edges for that step, and keeps getting updated. The sampling process stops when there are exactly s nodes, which ultimately form the sampled non-hyperlink (it not being a hyperlink is ensured by rejection sampling). The sampling is done in a way such that the nodes of $\hat{F}$ would form a connected component (*i.e.*, would have at least $s-1$ links in $\eta(\mathcal{H})$). Note that there could be more links between a sampled set of nodes than those chosen by the MNS algorithm, and all of them ultimately form the non-hyperlink.

Figure 5.6b shows the distribution of edge-density of non-hyperlinks sampled via MNS, which should immediately be compared with that of the positive class from Figure 5.3b. It is evident that the number of non-hyperlinks having high ED in Figure 5.6b is quite high as compared to that using SNS (Figure 5.5b). Moreover, it is clear to see that ED of any non-hyperlink $\hat{F}$ sampled using MNS (Algorithm 4) satisfies the following: $\frac{2}{|\hat{F}|} \leq ed(\hat{F}; \mathcal{H}) \leq 1$, since the minimum edge-density for an s-node sample would occur for a line-subgraph (*i.e.*, when s nodes would be linked by $s-1$ links), thereby resulting in an edge density of $\frac{2 \cdot (s-1)}{s \cdot (s-1)} = \frac{2}{s}$. This owes to the fact that MNS gives



(a) Size distribution (SD)

(b) Edge-density distribution (EDD)

Figure 5.6: Size distribution (SD) and edge-density distribution (EDD) of non-hyperlinks – the negative class – in `email-Enron` sampled using Motif Negative Sampling (MNS).

connected subgraphs, and hence, the ED of small-sized non-hyperlinks is likely to be high.

However, it could be noted that MNS is not completely unbiased, in that high degree nodes have a higher probability of getting sampled. Such bias is also introduced when random walks over graphs are used to find a representative sub-graph of original graph and there have been approaches in the literature to rectify such biases, one of them being the Metropolis Hastings algorithm [86]. However it cannot be used reliably in case of negative sampling in hypergraphs since the number of edges to be sampled is quite small and hence convergence guarantees cannot be ensured.

Non-hyperlinks sampled via MNS would occupy the "pink" region indicated in Figure 5.2.

5.2.5 Clique Negative Sampling (CNS)

Where one extreme NS technique that makes prediction *easy* for an HLP algorithm is UNS, another extreme is to make it *tough*, by sampling cliques from the clique-expanded graph $\eta(\mathcal{H})$ of a hypergraph $\mathcal{H}$. This ensures the edge density of sampled hyperedges to always be *unity*, which, according to our measure of hardness (Definition 5.1), returns the ***hardest-to-classify*** set of non-hyperlinks. However, since clique-finding in a graph is an NP-complete problem, we do not compute them directly. Instead, motivated by the geodesic-distance based NS technique by Lichtenwalter et al. [67], we develop a hypergraph equivalent of their "1-hop" sampling approach via a simple heuristic to efficiently sample non-hyperlinks as per Algorithm 5. Since a hyperlink F (positive pattern) forms a clique in the induced graph $\eta(\mathcal{H})$ of $\mathcal{H}$, this very information could be exploited to sample a non-hyperlink $\hat{F}$ such that $\hat{F}$ too follows F. To be precise, we replace a node $v_0 \in F_0$ (the existing hyperlink) with a node v_1 which is a common neighbor to all of the remaining nodes in $F_0 \setminus \{v_0\}$, to result in a new set of nodes

Algorithm 5: The CNS algorithm

Input: A hypergraph $\mathcal{H} = (V, \mathcal{F})$, s = size of non-hyperlink to be sampled
Output: Sampled non-hyperlink $\hat{F}$

1 $F_0 = \text{RANDOMCHOICE}(\mathcal{F})$ `// Randomly sample a hyperlink`
2 $V_F = \{u \mid u \in F_0\}$ `// Nodes of` F_0
3 $v_0 = \text{RANDOMCHOICE}(V_F)$ `// Randomly sample a node for removal`
`/* Randomly select a node from the neighborhood of` $F_0 \setminus \{v_0\}$ `*/`
4 $V_n = \{u \in V \mid \exists F \in \mathcal{F} \text{ s.t. } \{u, v\} \subseteq F,\ \forall v \in F_0 \setminus \{v_0\}\}$
5 **if** $V_n = \emptyset$ **then**
6 **go to** 1
7 $v_1 = \text{RANDOMCHOICE}(V_n)$
8 $\hat{F} = (F_0 \setminus \{v_0\}) \cup \{v_1\}$
9 **return** $\hat{F}$

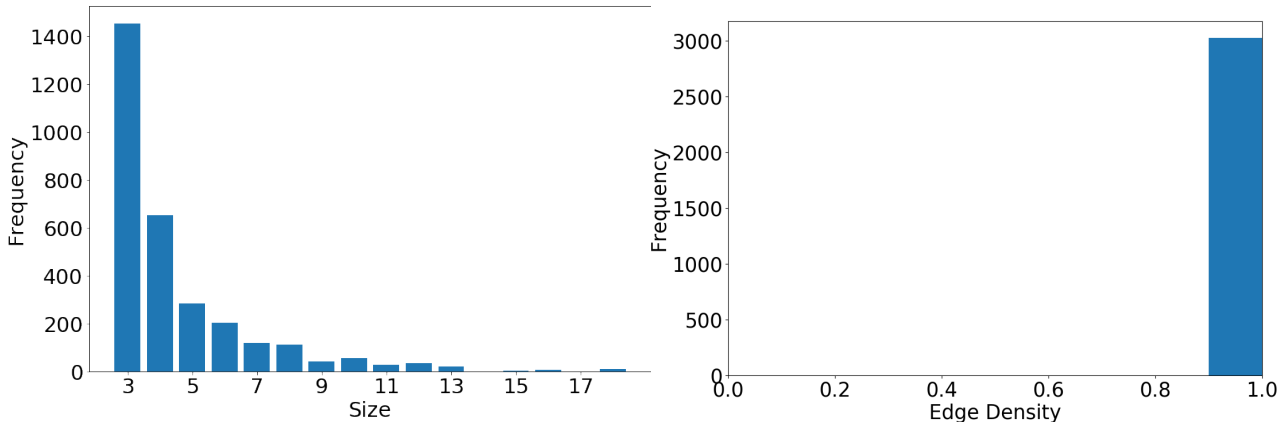
$\hat{F} := (F_0 \setminus \{v_0\}) \cup \{v_1\}$. This new node-set $\hat{F}$ forms a clique in $\eta(\mathcal{H})$ and can be considered as a negative pattern (non-hyperlink) as long as it is not a hyperlink. The exact procedure for CNS has been described in Algorithm 5.

Note that although this heuristic does not guarantee the existence of such common neighbor nodes v_1 (step 7 in Algorithm 5), we, however, empirically observe that such nodes do exist. Extensions to CNS (*e.g.*, to add/remove multiple nodes at once, *etc.*) could also be implemented. Moreover, by no means does Algorithm 5 sample all possible cliques; it only gives a sample which we use for HLP. CNS ensures all sampled non-hyperlinks to have a *unit* edge density (ref. Figure 5.7b), which is much different from SNS (ref. Figure 5.5b), where most of them have extremely low ED (if not zero). Hence, ***CNS provides the hardest of non-hyperlinks*** whereas the hardness of those sampled from MNS lies in the moderate range (ref. Figure 5.6). One could compare the SD and EDD for non-hyperlinks sampled using CNS (Figure 5.7) with those of the positive class (Figure 5.3) and see that they closely match with each other.

Non-hyperlinks sampled by CNS gives patterns from the "green" region marked at the top in Figure 5.2. In summary, there is a whole *spectrum* of NS algorithms that could sample non-hyperlinks, and we have explored four of them, *viz.*, UNS, SNS, MNS, and CNS.

5.3 Related Work

A rigorous study of HLP began with the near-seminal works [3, 135] on hypergraph Laplacian and spectral clustering methods. Xu et al. [121] explore the latent representation of hyperlinks obtained via those of nodes and a novel entropy-based approach to combine them. More recently, Zhang et al. [131, 132] proposed CMM, which is the current state-of-the-art for HLP. Benson et al. [11] study



(a) Size distribution (SD)

(b) Edge-density distribution (EDD)

Figure 5.7: Size distribution (SD) and edge-density distribution (EDD) of non-hyperlinks – the negative class – in `email-Enron` sampled using Clique Negative Sampling (CNS). When compared with the positive class in Figure 5.3, both ED and SDD closely match.

the evolution of hyperlinks of size 3 and 4 in a hypergraph.

As mentioned before, HLP in hypergraphs is analogous to LP in graphs. Though sampling non-links are necessary when LP is posed as a classification problem, it has received little attention in the literature [66, 4, 67]. Most works randomly sample non-links, which is still justified since the space of possible non-links is *polynomial* in $|V|$. However for hypergraphs, where the space of all possible non-hyperlinks is *exponential* in $|V|$, carefully devised non-hyperlink sampling approaches are mandatory.

Sampling non-hyperlinks is akin to subgraph sampling, which is commonly performed to sample frequent patterns from graphs. More recently, there has been enough attention on mining frequent patterns called *motifs* in a graph to understand the evolution of edges therein. A motif of size k is a k-connected component of the graph. There exists randomized methods such as Mfinder [52] and GUISE [1] to mine these frequent motifs. One of our NS methods (MNS) has been inspired from such motif sampling techniques.

5.4 Experiments

We take seven different temporal hypergraph datasets – `email-Enron`, `contact-high-school`, `contact-primary-school`, `tags-math-sx`, `MAG-Geo`, `coauth-DBLP`, `NDC-substances` – from Benson et al. [11] and perform various HLP experiments on them. Also, we use the same k-core based sampling technique as used by Liben-Nowell et al. [66] to reduce the size of `MAG-Geo` and `coauth-DBLP` datasets since they are huge hypergraphs. More specifically, we retain only those nodes

which have *hyperdegree* (number of incident hyperedges) greater than a threshold $k = 16$.

We use five different HLP algorithms: Bayesian Sets (BS) [35], Factorization Machines (FM) [85], Hyper Katz (Katz) [54], Hyper Common Neighbors (CN) [132, 78], and Coordinated Matrix Minimization (CMM) [132]. To evaluate an HLP algorithm, we use the area under ROC curve (AUC) metric. In addition, we also report certain statistics on multiple NS techniques, which ultimately gives insights into which technique works best.

All of the datasets used are temporal in nature. Following the data preparation step described in Chapter 2, we perform a temporal split of 80:20 (*i.e.*, $\rho_{tr} = 0.8$) where hyperedges are sorted according to their timestamp[1] and first 80% of hyperedges are used for training and feature extraction, whereas the remaining 20% are used for testing. The NS ratio (*i.e.*, ratio of negative samples (non-hyperlinks) to positives (hyperlinks) ν from Chapter 2) is fixed to 10:1 ($\nu = 10$), except for `NDC-substances` where it is 5:1 ($\nu = 5$, since on an average, the data has has bigger hyperedges). We use the AUC score for the evaluation and comparison of HLP algorithms, since it is a standard metric that has been widely used in the LP literature. We experiment with multiple NS ratios (ν) to analyse its impact on the evaluation metric. We perform five different negative samplings per dataset per hyperlink prediction approach.

5.5 Results and Discussion

This section provides detailed analyses of the impact of different NS techniques on the performance of HLP algorithms. We also provide a comprehensive analysis of comparing some properties of non-hyperlinks sampled using different methods with those of hyperlinks.

5.5.1 Hyperlink Prediction Performance

The AUC scores obtained by applying each of the five HLP algorithms on the seven datasets have been populated in Table 5.1, which has been divided into three parts corresponding to negative sampling techniques SNS, MNS, and CNS. In each row, AUC score for the best performing algorithm has been underlined. But since our main aim is not to compare between HLP algorithms, scores for NS algorithms that give the best performance for a given HLP algorithm has been **bold-faced**. The first observation we make is that except for CMM [132], all other HLP algorithms perform their best when compared against a SNS-sampled negative class. ***CMM, which is supposed to be the current state-of-the-art in HLP, performs its best when evaluated against either MNS or CNS based negative sampling***. Another striking point that Table 5.1 reveals is that ***no dataset has a unanimous best performing HLP algorithm, and instead varies with the NS algorithm***. For example, according to SNS, MNS, and CNS, the best algorithm for the `tags-math-sx` (`tms`) dataset turns out to be Katz,

[1]Multiple timestamps are resolved by using the earliest one.

Table 5.1: AUC scores (%) for HLP using BS, FM, Katz, CN, and CMM on seven datasets, where NS is performed via SNS, MNS, and CNS. Avg. reduction:
SNS→MNS: BS=21%, FM=12%, Katz=36%, CN=43%, CMM=−28%
SNS→CNS: BS=35%, FM=36%, Katz=44%, CN=44%, CMM=−33%.

	Sized NS (SNS)					Motif NS (MNS)					Clique NS (CNS)				
Dataset	BS	FM	Katz	CN	CMM	BS	FM	Katz	CN	CMM	BS	FM	Katz	CN	CMM
email-Enron	**72.7**	**81.8**	**70.1**	**66.0**	55.2	69.3	77.3	29.0	24.3	39.9	37.9	44.8	35.5	27.8	**59.7**
contact-high-school	**64.6**	**69.9**	**99.4**	**99.2**	57.8	49.9	63.9	77.0	77.4	64.8	47.5	65.8	62.2	66.4	**65.9**
contact-primary-school	**71.1**	**60.2**	**93.9**	**93.4**	49.2	54.1	54.4	67.5	73.1	54.2	49.8	59.6	57.2	62.4	**61.8**
coauth-DBLP	**63.9**	**69.9**	**71.2**	**74.9**	38.8	38.3	46.8	16.8	21.9	**61.9**	37.6	38.4	22.1	29.9	54.7
NDC-substances	**95.9**	**80.0**	**85.0**	**94.5**	60.6	89.9	75.7	81.2	23.1	**74.2**	73.8	60.4	79.0	58.1	65.7
tags-math-sx	**95.8**	**75.2**	**99.2**	**98.9**	22.8	75.5	64.9	75.7	78.5	53.5	63.9	62.4	51.9	57.3	**59.0**
MAG-Geo	**78.4**	**53.9**	**98.1**	**97.5**	26.7	49.4	50.5	46.5	54.3	**50.2**	41.6	45.7	40.1	51.2	47.1

CN, and BS respectively. One final point we want to make *w.r.t.* this table is the general trend of reduction in AUC scores as we move from the leftmost block (SNS) to the rightmost one (CNS). The average reduction has been indicated in the table caption, according to which, ***simple extensions of link prediction such as CN and Katz have the maximum average reduction (of*** $\sim 44\%$***)***, and the CMM algorithm, which actually "learns" to *pick* hyperlinks out of a bag of hyperlinks and non-hyperlinks sees an *increment* of 33% as we go from SNS to CNS sampling.

5.5.2 Edge Density Distribution

We plot the edge-density distributions (EDD) for the email-Enron dataset in Figures 5.5b, 5.6b, and 5.7b, whereas EDD for hyperlinks have been plotted in Figure 5.3b. For a discussion, see Sections 5.2.2 – 5.2.5.

5.5.3 Common Neighbor vs. Edge Density

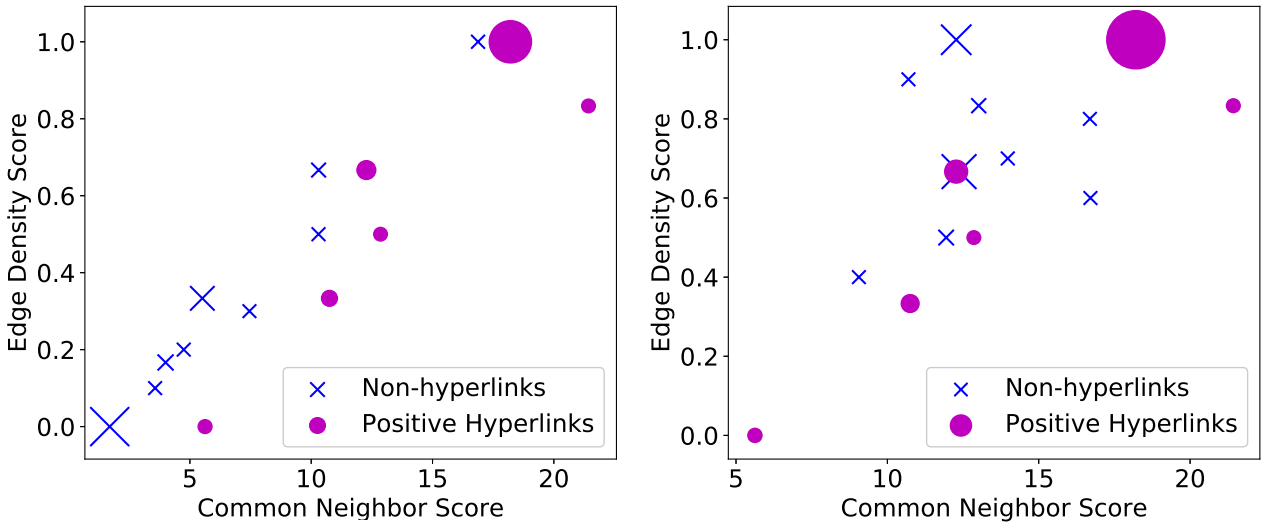
Figure 5.8 shows the scatter plot of common-neighbor (CN) scores and edge-densities (ED) for each test pattern in the contact-high-school dataset, where the blue crosses and pink discs represent non-hyperlinks and hyperlinks respectively. It is clear from these plots that while SNS sampled non-hyperlinks have lower ED values and lower CN scores as well, the MNS algorithm samples non-hyperlinks in a way that CN is not able to distinguish between the two classes.

5.5.4 True Negative Rates

To better explain the impact of NS algorithms on HLP, we perform HLP via supervised learning, *i.e.*, by using CN scores as a single feature to learn a Logistic Regression classifier (LRC). We perform a cross validation by first preparing three different validation sets, each formed by sampling the negative class by a different NS algorithm. We then train one LRC per NS algorithm (with NS performed

via the respective algorithm to generate training data) which is subsequently tested on all three cross validation sets. The performance of classifiers in terms of true negative rates is shown in Figure 5.9. An LRC trained using MNS (Figure 5.9a) or CNS (Figure 5.9b) can easily predict negatives from SNS truly. However, the same is not true for a classifier trained on SNS samples as shown in Figure 5.9c. A typical CN score distribution for three different validation sets defined above and positive hyperlinks is shown in Figure 5.9d. Cross validation results of LRC models are evident from this distribution as most of the SNS sampled hyperlinks have a low CN score whereas MNS and CNS sampled hyperlinks have CN scores that are comparable with the CN scores positive hyperlinks.

Please note that in this chapter, we do not propose a new method to perform hyperlink prediction. Instead, as all studies in class imbalance do, we have proposed techniques to balance the data and have discussed in each of the cases, the effects they have on the hyperlink prediction problem. The main take-home message is that "One should not believe a model where the negative (or majority) class was randomly undersampled", since this would be an over-promising model. Moreover, we "have" to perform negative sampling – something that's inevitable. For unsupervised learning, it's not too problematic since there are no parameters to be learnt. But for supervised learning, the way we sample the negative class while training a model decides how "hard" the model trains itself. In other words, this decides the generalizability of our model. If it's trained on a "simple" negative sampling (e.g., uniformly randomly sampled), we get a "lazy" classifier that won't work in practise, since it learns the wrong parameters. On the other hand, using a "hard" negative sampling challenges the classifier to

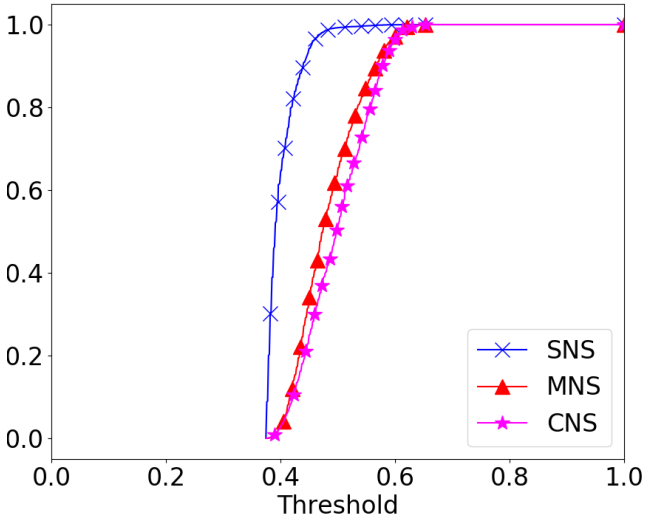


(a) SNS-sampled non-hyperlinks

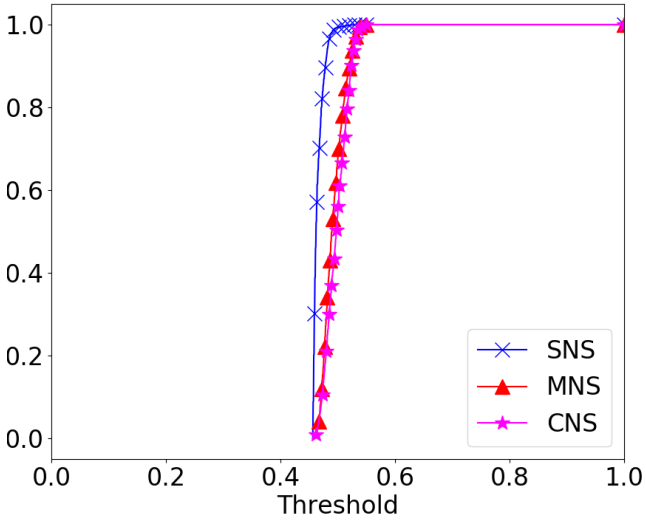
(b) MNS-sampled non-hyperlinks

Figure 5.8: Common neighbor score (CN) vs. edge-density (ED) scatter-plots for hyperlinks (pink discs ●) and non-hyperlinks (blue crosses ×); marker size is proportional to frequency count.

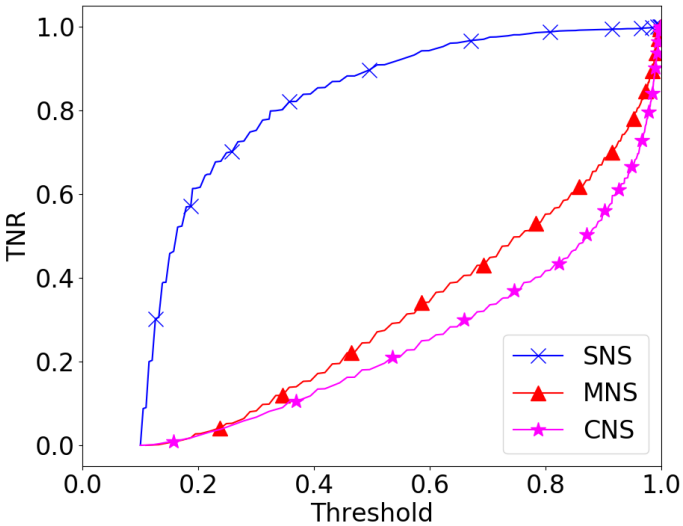
a greater extent, and learns better rules for hyperlink vs. non-hyperlink. Hence, ***a model trained on MNS or CNS would be able to generalize better*** than that trained on SNS.



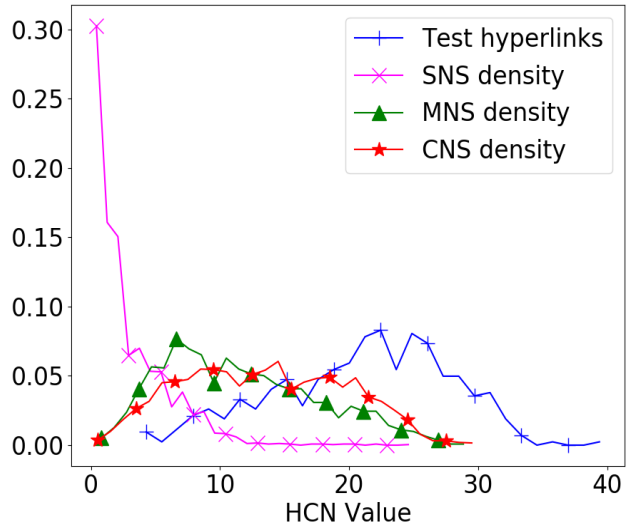
(a) Classifier trained via MNS.



(b) Classifier trained via CNS.



(c) Classifier trained via SNS.



(d) CN score distribution for test data.

Figure 5.9: (a–c): True Negative Rates (TNR) for CN-based Logistic Regression HLP classifiers trained for one NS algorithm and tested on all. (d): Common neighbor (CN) score density plots for test hyperlinks (positive class, blue pluses +), and test non-hyperlinks (negative class) sampled using SNS (pink crosses ×), MNS (green wedges ▲), and CNS (red stars ⋆). All plots are for `contact-high-school`.

5.6 Conclusion

Under-sampling the majority class in class-imbalanced scenarios is a common practice. But hyperlink prediction (HLP) is atypical, in that there exists extreme class imbalance, with the set of non-hyperlinks being the majority class. We set out to analyze four negative sampling (NS) techniques for HLP, *viz.*, Uniform (UNS), Sized (SNS), Motif (MNS), and Clique (CNS) based NS. We analyzed size, edge-density, and a usual predictor score (CN) distribution for candidate hyperlinks extracted via all NS techniques and found that ***while UNS is completely useless for HLP***, ***SNS makes the negative class follow the same size distribution as the positive class***. But ***MNS and CNS go one step further and focus on matching their edge-density distributions as well, making the HLP problem challenging in nature***. While the evaluation of an HLP algorithm on test sets sampled via SNS, MNS, and CNS is found to vary drastically, a specialized cross-validation of HLP via the supervised learning paradigm further shows that ***only MNS and CNS generalize well for HLP***. In essence, ***we prescribe using either MNS or CNS*** for sampling non-hyperlinks for HLP, since they learn fair and generalized HLP predictors that would perform as expected in practical scenarios.

Chapter 6

Clique and Shut: A New Model to Predict Hyperedges

"In the long history of humankind (and animal kind, too) those who learned to collaborate and improvise most effectively have prevailed." ∼ `Charles Darwin`

AKIN to the link prediction problem in graphs, we deal with hyperlink (higher-order link) prediction in hypergraphs. With a handful of solutions in the literature that seem to have merely scratched the surface, we provide improvements for the same using a matrix completion based algorithm. Motivated by observations in recent literature – Benson et al. [11] – we first formulate a "clique-closure" hypothesis (*viz.*, hyperlinks are more likely to be formed from near-cliques rather than from non-cliques), test it on real hypergraphs, and then exploit it for our very problem. In the process, we generalize hyperlink prediction on two fronts: (1) from small-sized to arbitrary-sized hyperlinks, and (2) from a couple of domains to a handful. We perform experiments (both the hypothesis-test as well as the hyperlink prediction) on multiple real datasets, report results, and provide both quantitative and qualitative arguments favouring better performances *w.r.t.* the state-of-the-art.

6.1 Introduction

Hyperlink prediction refers to predicting future/missing hyperlinks in a given hypergraph [121, 132, 11] (ref. Chapter 2, Definition 2.5). We draw inspirations from recent literature and the existing state-of-the-art, *i.e.*, Coordinated Matrix Minimization (CMM) [132], to solve the problem of predicting arbitrary-sized hyperlinks in networks. We first formulate a ***C**lique-**C**losure **H**ypothesis* (**CCH**), which can be summarized as follows: ***Hyperlinks in a network are more likely to be formed from closures of cliques (and near-cliques) rather than those of non-cliques***. In simpler terms, we hypothesize that for a given hyperlink, prior to its first occurrence, its incident nodes should have had more interactions

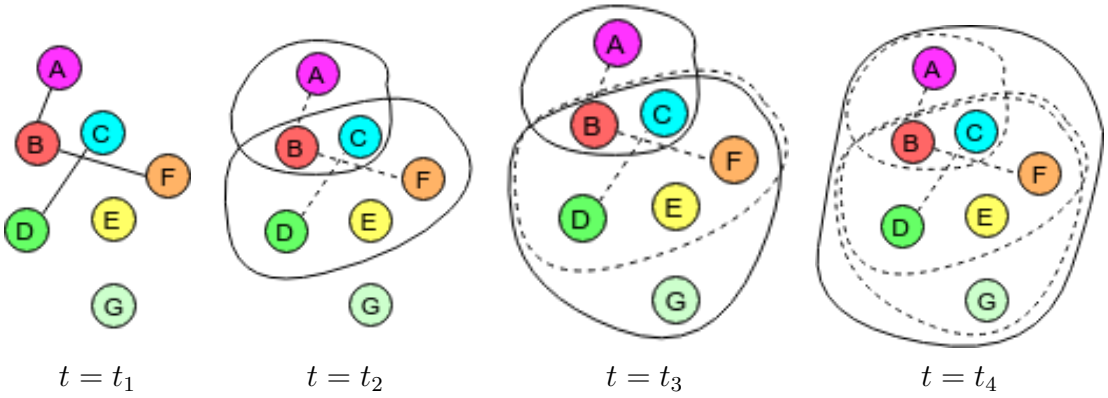


Figure 6.1: A toy example: For $i = 1, 2, 3, 4$, solid hyperlinks formed exactly at $t = t_i$, are eventually shown as dotted connections. The clique-closure hypothesis (CCH) we propose says that for a current hyperlink (solid) it is highly likely that its nodes had been densely connected via past connections (dotted).

than a set of arbitrary number of nodes usually does. *I.e.*, we expect every hyperlink to have *evolved gradually*, rather than having spontaneously "sprung-up". Consider the example shown in Figure 6.1, where it could be noted how at any $t = t_i$, smaller hyperlinks from the past (dotted) combine together to form larger ones (solid) in the present.

We first test CCH on real datasets, and then use it for hyperlink prediction via a method we term *"**C**lique-**C**losure based **C**oordinated **M**atrix **M**inimization"* (**C3MM**). We ingest CCH into the objective function of CMM to get C3MM, and then solve it in a similar fashion. Choosing datasets from different domains, we note significant improvements over CMM. Major improvements come from the fact that C3MM gives a chance to those hyperlinks that could explain existing relations.

In Section 6.2, we introduce, formally define, and give intuitive explanations our clique-closure hypothesis (CCH). Then, embed CCH into the hyperlink prediction process in Section 6.3, resulting in our prediction algorithm C3MM, post which we discuss the state-of-the-art in hyperlink prediction in Section 6.4. Finally, we describe our experimental process in Section 6.5 and discuss the results on both CCH's hypothesis test as well as hyperlink prediction using C3MM in Section 6.6, before we conclude in Section 6.7.

6.1.1 Key Contributions

1. We formulate and test a **clique-closure hypothesis (CCH) for hypergraph network evolution**. As a result, we provide novel insights into hyperlink evolution.

2. We provide a **hyperlink prediction algorithm C3MM that significantly improves upon CMM**.

3. We **extend hyperlink prediction** for hyperlinks of arbitrary **size** and to multiple **domains**.

6.2 The Clique Closure Hypothesis

We first formulate our clique-closure hypothesis (CCH), which argues that nodes in a hyperlink remain very densely-connected in the hypergraph right before the hyperlink's formation. We provide hypothesis test for validation of CCH in real world datasets and come up with a method for hyperlink prediction which exploits it to improve performance of the current state-of-the-art. Occurrence of a hyperlink marks the collaboration among multiple entities via a single common event. It is intuitive that subsets of these entities would have interacted in some form in the past, rather than the hyperlink getting formed spontaneously. In formal terms, in a temporal hypergraph $\mathcal{H} = (V, \mathcal{F}, \tau)$, corresponding to a hyperlink $F \in \mathcal{F}$ formed at a given time $\tau(F)$, we could expect to find some hyperlinks $F' \in \mathcal{F}_{<\tau(F)}$ that overlap densely with subsets of nodes incident on F. Since if that were not true, there is not much explanation – at least not any using the hypergraph topology – as to why the relation F is formed in the first place. In the projected graph $\eta(\mathcal{H}_{<\tau(F)})$, this translates as densely connected subgraphs (near-cliques) or sometimes even cliques. In simple words, ***CCH states that with high probability, nodes of a hyperlink were part of dense subgraphs before they formed hyperlinks***. We formally define CCH in this section, but before doing so, we need to keep certain concepts well-defined, since their equivalents do not exist in the literature.

6.2.1 Some Salient Concepts

We define ***hypergraph density*** of a hypergraph $\mathcal{H}$ as $hd(\mathcal{H}) := gd(\eta(\mathcal{H}))$, the density of its clique-expanded graph. Similarly, ***subgraph density*** $sgd(F, \mathcal{H})$ of any set of nodes $F \subseteq V$ (where F need not be a hyperlink) is defined as $sgd(F, \mathcal{H}) := gd(\eta(\mathcal{H})|_F)$. Note that $sgd(F, \mathcal{H}) = 1$ for all hyperlinks $F \in \mathcal{F}$. We define a slight modification of this notion for temporal and non-temporal hypergraphs, *viz.*, ***pre-hyperlink density*** $hd_{pre}(F, \mathcal{H}) := sgd(F, \mathcal{H}_{<\tau(F)})$ and ***punctured hyperlink density*** $hd_{punc}(F, \mathcal{H}) := sgd(F, \mathcal{H}_{-F})$ respectively. The prefixes *punctured-* and *pre-* here refer to the fact that density is calculated on the hypergraph that existed *without* and *before* hyperlink F respectively. A higher pre-hyperlink density for a hyperlink would mean it evolved from near-cliques. Moreover, a hyperlink F evolving from cliques would have $hd_{pre}(F, \mathcal{H}) = 1$, and those having an underlying clique structure would have $hd_{punc}(F, \mathcal{H}) = 1$. In other words, hd_{punc} is used as an alternative for hd_{pre} in a non-temporal hypergraph, where the concept of *evolution* (*i.e.* order of hyperedge discovery is irrelevant) does not exist.

Let the ***clique-fraction*** $cf(\mathcal{H})$ of hypergraph $\mathcal{H}$ be defined as $cf(\mathcal{H}) := \dfrac{|\{F \in \mathcal{F} : hd_x(F, \mathcal{H}) = 1\}|}{|\mathcal{F}|}$, *the fraction of hyperlinks that formed from cliques*, where hd_x denotes hd_{punc} and hd_{pre} for temporal and non-temporal hypergraphs respectively. Since $cf(\mathcal{H})$ is expected to be too low for non-temporal

datasets, we also define a constant ***minimum-clique-fraction*** cf_{min} and fix it to be $cf_{min} = 0.05$, which is a little more than the maximum hypergraph density among all non-temporal hypergraphs (ref Table 6.3). Finally, we define ***cliqueness*** of a hyperlink $F \in \mathcal{F}$ as follows:

$$\chi(F, \mathcal{H}) := hd_x(F, \mathcal{H}) \cdot \max(cf_{min}, cf(\mathcal{H})) \tag{6.1}$$

where hd_x denotes hd_{punc} and hd_{pre} for temporal and non-temporal hypergraphs respectively. We are now ready with a well-defined measure – ***cliqueness*** – to capture the notion of how dense is the region from which a given hyperlink is formed. Cliqueness captures both clique-fraction, as well as density, thereby catering to both clique- as well as near-clique-structure of a given hyperlink.

6.2.2 Stating and Refining the Hypothesis

Hypothesis 1 (***CCH: Clique Closure Hypothesis***). *Given a hypergraph $\mathcal{H} = (V, \mathcal{F})$ (or $(V, \mathcal{F}, \tau)$), the null and alternate hypotheses for CCH are defined as follows for a hyperlink $F \in \mathcal{F}$:*

$$\mathbb{H}'_0 : \chi(F, \mathcal{H}) \leq \mathbb{E}[sgd \mid \mathcal{H}], \quad \mathbb{H}'_1 : \chi(F, \mathcal{H}) > \mathbb{E}[sgd \mid \mathcal{H}], \tag{6.2}$$

where $\mathbb{E}[sgd \mid \mathcal{H}] := \dfrac{1}{|\mathcal{P}(V)|} \cdot \displaystyle\sum_{F' \in \mathcal{P}(V)} sgd(F', \mathcal{H})$, *the mean subgraph density over all subsets of* V.

In order to simplify CCH, and to make it more deterministic, we have the following result in place.

Theorem 6.1. *Mean subgraph density of $\mathcal{H}$ over all subsets of V is equal to its hypergraph-density. In other words,* $\mathbb{E}[sgd \mid \mathcal{H}] = hd(\mathcal{H})$.

Proof. Let X be a random variable denoting *hyperlink size*. And let $\mathbb{E}[sgd \mid \mathcal{H}, X = k]$ be the mean hyperedge-density of all k-sized hyperedges. We have:

$$\mathbb{E}[sgd \mid \mathcal{H}, X = k] = \frac{1}{\binom{n}{k}} \sum_{F' \in \mathcal{P}_k(V)} \left(\frac{1}{\binom{k}{2}} \sum_{e \in \mathcal{P}_2(F')} \mathbb{1}_{\eta(\mathcal{F})}(e) \right) \tag{6.3}$$

$$= \frac{k!(n-k)!}{n!} \cdot \frac{2!(k-2)!}{k!} \cdot \binom{n-2}{k-2} \sum_{e \in \mathcal{P}_2(V)} \mathbb{1}_{\eta(\mathcal{F})}(e) \tag{6.4}$$

$$= \frac{(n-k)! \cdot 2! \cdot (k-2)!}{n!} \cdot \frac{(n-2)!}{(k-2)!(n-k)!} \cdot |\eta(\mathcal{F})| \tag{6.5}$$

$$= \frac{2 \cdot |\eta(\mathcal{F})|}{n \cdot (n-1)} = gd(\eta(\mathcal{H})) = hd(\mathcal{H}). \tag{6.6}$$

Symbol	Definition
$\mathcal{F}_{<t}$	Hyperlinks observed *before* time t
$\mathcal{H}_{<t}$	Hypergraph observed *before* time t
$\mathcal{H}_{-F}$	Hypergraph *punctured w.r.t.* (or *without*) F
$gd(\mathcal{G})$	Density of graph $\mathcal{G}$
$\mathcal{G}\|_F$	Subgraph of $\mathcal{G}$ *w.r.t.* nodes in $F \subseteq V$
$hd(\mathcal{H})$	Hypergraph density of $\mathcal{H}$
$sgd(F, \mathcal{H})$	Subgraph density of $F \subseteq V$ w.r.t. $\mathcal{H}$
$hd_{pre}(F, \mathcal{H})$	Pre-hyperlink density of $\mathcal{H}$ before F
$hd_{punc}(F, \mathcal{H})$	Punctured hyperlink density of $\mathcal{H}$ w.r.t. F
$cf(\mathcal{H})$	Clique-fraction of $\mathcal{H}$
cf_{min}	Minimum-clique-fraction (a constant)
$\chi(F, \mathcal{H})$	Cliqueness of F w.r.t. $\mathcal{H}$
$\mathbb{H}'_0$	Null hypothesis for CCH
$\mathbb{H}'_1$	Alternate hypothsis for CCH
$\mathbb{H}_0$	Null hypothesis for CCH restated
$\mathbb{H}_1$	Alternate hypothesis for CCH restated

Table 6.1: Extra notations used in this chapter. Others have already been tabulated in Chapter 2.

Finally, we have:

$$\begin{aligned}\mathbb{E}[sgd \mid \mathcal{H}] &= \mathbb{E}_X[\mathbb{E}[sgd \mid \mathcal{H}, X]] \\ &= \sum_k \mathbb{E}[sgd \mid \mathcal{H}, X = k] \cdot P(X = k) \\ &= \sum_k hd(\mathcal{H}) \cdot P(X = k) \\ &= hd(\mathcal{H}) \cdot \sum_k P(X = k) = hd(\mathcal{H})\end{aligned}$$

□

Hence, we could restate the CCH hypothesis as follows:

Hypothesis 2 (*CCH restated*).

$$\mathbb{H}_0 : \chi(F, \mathcal{H}) \leq hd(\mathcal{H}), \quad \mathbb{H}_1 : \chi(F, \mathcal{H}) > hd(\mathcal{H}), \tag{6.7}$$

where $hd(\mathcal{H})$ and $\chi(F, \mathcal{H})$ denote density of hypergraph $\mathcal{H}$ and cliqueness of hyperlink F therein.

For ease of reference, we compile a list of all notations used hitherto in Table 6.1. We test CCH on a given temporal hypergraph H using Algorithm 6 and report results in Table 6.1.

Algorithm 6: An algorithm to test CCH on a temporal hypergraph $\mathcal{H} = (V, \mathcal{F}, \tau)$. Each hyperlink $F = \{v_1, \ldots, v_{|F|}\} \in \mathcal{F}$ is evaluated *w.r.t.* connections $\eta(\mathcal{H}_{<\tau(F)})$ in its past based on how densely nodes $v_1, \ldots, v_{|F|}$ are connected.

Input: Temporal hypergraph, $\mathcal{H} = (V, \mathcal{F}, \tau)$
Output: p-value of $\mathbb{H}_0$ for hypergraph $\mathcal{H}$

1 $\mathcal{F}_{>2} \leftarrow \{F \in \mathcal{F} : |F| > 2\}$
2 $d_{\mathcal{H}} \leftarrow \dfrac{2 \cdot |\eta(\mathcal{F})|}{|V| \cdot (|V| - 1)}$ // hypergraph density
3 $N_c \leftarrow 0$ // no. of cliques
4 $D \leftarrow \{\}$ // hyperlink density map
5 **for** $F \in \mathcal{F}_{>2}$ **do**
6 $\quad \mathcal{E}_F \leftarrow \{\{u, v\} \mid \forall\, u, v \in f\}$ // projected edges of F
7 $\quad t \leftarrow \tau(F)$
8 $\quad \mathcal{F}_{<t} \leftarrow \tau^{-1}([0, t))$ // hyperlinks before F
9 $\quad \mathcal{E}_{<t} \leftarrow \eta(\mathcal{F}_{<t})$ // projected edges before F
10 $\quad D[F] \leftarrow \dfrac{|\mathcal{E}_F \cap \mathcal{E}_{<t}|}{|\mathcal{E}_F|}$ // $hd_{pre}(F, \mathcal{H})$
11 $\quad$ **if** $D[F] == 1$ **then** // i.e., if $\mathcal{E}_F \cap \mathcal{E}_{<t}$ is a clique
12 $\quad\quad N_c \leftarrow N_c + 1$
13 $cf \leftarrow \max(cf_{min}, N_c/|\mathcal{F}_{>2}|)$ // clique-fraction
14 $N_{CCH} \leftarrow 0$ // no. of hyperlinks satisfying CCH
15 **for** $F \in \mathcal{F}_{>2}$ **do**
16 $\quad \chi_F \leftarrow D[F] \cdot cf$ // cliqueness
17 $\quad$ **if** $\chi_F \leq d_{\mathcal{H}}$ **then**
18 $\quad\quad N_{CCH} \leftarrow N_{CCH} + 1$
19 $p \leftarrow N_{CCH}/|\mathcal{F}_{>2}|$
20 **return** p

6.2.3 Hypothesis Test

Applying CCH to a non-temporal hypergraph would be futile, since there's no concept of *evolution* per se defined for it. For instance, reactions (hyperlinks) in a metabolite hypergraph [132] cannot be arranged in a chronological order. However, we attempt to test CCH for such networks using a proxy mechanism, in that we set $sgd_x = hd_{punc}$ in eq. 6.1 while calculating cliqueness $\chi(F, \mathcal{H})$. Making few changes to lines 7–10 in Algorithm 6, so as to calculate $D[F] \leftarrow hd_{punc}(F, \mathcal{H})$ for each hyperlink F, we could find p-values for a non-temporal hypergraph as well. The idea is to validate whether for a hyperedge, its incident nodes are well-connected even without its presence. Finally, we present the results in Table 6.3 for both temporal and non-temporal datasets.

In the literature, Benson et al. [11], who restrict themselves to 3- and 4-sized hyperlinks only, refer (although implicitly) to a similar phenomenon, wherein they argue that a clique (an open simplex) eventually forms a hyperlink (a closed simplex). The results of evaluating CCH on various datasets are tabulated and discussed in Section 6.6.1.

As would be discussed later, ***temporal hypergraphs strongly satisfy CCH***, *i.e.*, it is evident that ***most hyperlinks were cliques or near-cliques (densely connected) in the projected graph before they become hyperlinks.***

6.3 C3MM: CCH based Hyperlink Prediction

We exploit this unique characteristic of clique-closure to predict hyperlinks. The approach is similar to *Coordinated Matrix Minimization* (CMM) by Zhang et al. [132]. We call our method *Clique-Closure based CMM* (C3MM). Let us formulate C3MM here, for which we remain consistent with CMM [132] in defining concepts and the C3MM objective function; although, some notations would differ from what we define in Chapter 2, owing to being consistent with CMM.

6.3.1 Theoretical Formulation of C3MM

We know that for a given hypergraph $\mathcal{H} = (V, \mathcal{F})$ $\boldsymbol{S} \in \mathbb{R}^{|V| \times |\mathcal{F}|}$ is its *incidence matrix*. Let $\Delta\mathcal{F} \subseteq \mathcal{P}(V) \setminus \mathcal{F}$ represent the hyperlinks that are *missing* from (or *yet to occur* in) $\mathcal{H}$. Clearly, $\mathcal{F} \cap \Delta\mathcal{F} = \emptyset$. Let $\Delta\boldsymbol{S} \in \mathbb{R}^{|V| \times |\Delta\mathcal{F}|}$ be the incidence matrix corresponding to $\Delta\mathcal{F}$. Let $\mathcal{H}' := (V, \mathcal{F} \uplus \Delta\mathcal{F})$ be the *completed hypergraph*, whose incidence matrix $\boldsymbol{S}' \in \mathbb{R}^{|V| \times (|\mathcal{F}| + |\Delta\mathcal{F}|)}$ could be represented as follows (here, $[A; B]$ denotes column-wise concatenation of matrices A and B):

$$\boldsymbol{S}' := [\boldsymbol{S};\ \Delta\boldsymbol{S}]. \tag{6.8}$$

Adjacency matrix for projected graph $\eta(\mathcal{H})$ is defined as $\boldsymbol{A} := \eta(\boldsymbol{S}) := \boldsymbol{S}\boldsymbol{S}^T \in \mathbb{R}^{|V| \times |V|}$. Similarly, $\boldsymbol{A}' := \boldsymbol{S}'\boldsymbol{S}'^T$ refers to the adjacency matrix of $\eta(\mathcal{H}')$, for which we have:

$$\boldsymbol{A}' = \boldsymbol{S}'\boldsymbol{S}'^T = [\boldsymbol{S}; \Delta\boldsymbol{S}][\boldsymbol{S}; \Delta\boldsymbol{S}]^T = \boldsymbol{S}\boldsymbol{S}^T + \Delta\boldsymbol{S}\Delta\boldsymbol{S}^T = \boldsymbol{A} + \Delta\boldsymbol{A}, \tag{6.9}$$

where $\Delta\boldsymbol{A}$ refers to the links (edges) in adjacency space that get projected by missing hyperlinks $\Delta\mathcal{F}$ represented by $\Delta\boldsymbol{S}$.

Let $\mathcal{F}_{univ} = \{F_1, F_2, \ldots, F_{|\mathcal{F}_{univ}|}\}$ represent the set of *universal hyperlinks* (or candidate hyperlinks), forming our test set. Of $\mathcal{F}_{univ}$, $\Delta\mathcal{F}$ corresponds to true hyperlinks (the positive class); the remaining hyperlinks, $\mathcal{F}_{univ} \setminus \Delta\mathcal{F}$, called as *non-hyperlinks*, can be represented by $\Delta\hat{\mathcal{F}}$. And let $\Delta\hat{\boldsymbol{S}} \in \mathbb{R}^{|V| \times |\Delta\hat{\mathcal{F}}|}$ be the corresponding incidence matrix.

Let $\boldsymbol{U} = [\boldsymbol{u}_1\ \boldsymbol{u}_2\ \cdots\ \boldsymbol{u}_{|\mathcal{F}_{univ}|}] \in \mathbb{R}^{|V|\times|\mathcal{F}_{univ}|}$ to be the incidence matrix for the of set of candidate hyperlinks $\mathcal{F}_{univ}$, where $\boldsymbol{u}_i \in \mathbb{R}^{|V|}$ represents the incidence vector of the i^{th} hyperlink $F_i \in \mathcal{F}_{univ}$. Once the adjacency matrix $\Delta\boldsymbol{A}$ of the missing links is predicted, the next step would be to pick those hyperlinks from $\mathcal{F}_{univ}$, that best *explain* $\boldsymbol{A}$ and $\Delta\boldsymbol{A}$. For a given diagonal matrix $\boldsymbol{\Lambda_U} = diag(\lambda_1, \lambda_2, \ldots, \lambda_{|\mathcal{F}_{univ}|}) \in \mathbb{R}^{|\mathcal{F}_{univ}|\times|\mathcal{F}_{univ}|}$, the product incidence matrix $\boldsymbol{U\Lambda_U}$ would "select" exactly those columns $\boldsymbol{u}_i$ from $\boldsymbol{U}$, for which $\lambda_i = 1$. The corresponding adjacency matrix is then $\boldsymbol{U\Lambda_U}(\boldsymbol{U\Lambda_U})^T = \boldsymbol{U\Lambda_U^2U}^T = \boldsymbol{U\Lambda_UU}^T$. Hence $\boldsymbol{\Lambda_U}$ functions as ***hyperlink selector*** or predictor.

6.3.2 Redefining the Objective

For the purpose of link prediction, any feasible link prediction method can be used. Here, we use a Common Neighbor (CN) [78, 66] based link prediction technique. We first complete the adjacency matrix using the CN score, and then achieve its low rank approximation via Symmetric NMF [59]. The matrix $\boldsymbol{A}_{CN} = \boldsymbol{A}^2 - diag(\boldsymbol{A}^2)$ captures the common neighbor information of the projected graph $\eta(\mathcal{H})$ of $\mathcal{H}$. To predict missing links $\Delta\boldsymbol{A}$ we first approximate $\boldsymbol{A} + \boldsymbol{A}_{CN}$ with a low-rank matrix $\boldsymbol{W} \in \mathbb{R}^{|V|\times k}$, where $k < |V|$, such that $\boldsymbol{A} + \boldsymbol{A}_{CN} \approx \boldsymbol{WW}^T$.

$$\min_{\boldsymbol{W}\in\mathbb{R}_+^{|V|\times k}} \left\|\boldsymbol{A} + \boldsymbol{A}_{CN} - \boldsymbol{WW}^T\right\|_F^2. \tag{6.10}$$

The representation capability of $\boldsymbol{W}$ is low and hence such approximation ends up removing noisy links which might have been introduced due to $\boldsymbol{A}_{CN}$. Thus we define the predicted links as $\Delta\boldsymbol{A} := \boldsymbol{WW}^T - \boldsymbol{A}$.

Next step is to predict the missing hyperlinks $\Delta\mathcal{F}$ from the predicted missing links $\Delta\boldsymbol{A}$. Since $\boldsymbol{U}$ contains missing hyperlinks as well as non-hyperlinks, the diagonal matrix $\boldsymbol{\Lambda_U}$ should be such that the hyperlinks selected by $\boldsymbol{U\Lambda_U}$ correspond to the links in $\Delta\boldsymbol{A}$ when they are projected on graph. This can be obtained by optimizing $\boldsymbol{\Lambda_U}$ *w.r.t* following objective function:

$$\min_{\boldsymbol{\Lambda_U}\in diag\left(\{0,1\}^{|\mathcal{F}_{univ}|}\right)} \left\|\Delta\boldsymbol{A} - \boldsymbol{U\Lambda_UU}^T\right\|_F^2 \tag{6.11}$$

This is where we bring CCH into the picture. According to CCH, links in $\boldsymbol{A}$ also play a major role in formation of future hyperlinks. Hence, the predicted hyperlinks in $\Delta\mathcal{F}$ should not only explain missing links of $\Delta\boldsymbol{A}$ but also existing links in $\boldsymbol{A}$ (through clique and near-clique closure). However, as we already know, links in $\boldsymbol{A}$ are formed by $\mathcal{F}$ and hence can always be explained by $\boldsymbol{S}$. Then

hyperlinks can be predicted by optimizing $\mathbf{\Lambda_U}$ *w.r.t* following objective function:

$$\min_{\substack{\mathbf{\Lambda_U} \in diag\left(\{0,1\}^{|\mathcal{F}_{univ}|}\right) \\ \mathbf{\Lambda_S} \in diag\left(\{0,1\}^{|\mathcal{F}|}\right)}} \left\|\boldsymbol{A} - \boldsymbol{S}\mathbf{\Lambda_S}\boldsymbol{S}^T - \boldsymbol{U}\mathbf{\Lambda_U}\boldsymbol{U}^T\right\|_F^2 + \|\mathbf{\Lambda_S}\|_1. \tag{6.12}$$

The L1-penalty imposed on $\mathbf{\Lambda_S}$ is important to avoid a trivial solution where $\mathbf{\Lambda_S} = \boldsymbol{I}$ (the identity matrix) and $\mathbf{\Lambda_U} = \mathbf{0}$ (the zero matrix).

Satisfying the objective functions specified in eqs. (6.12) and (6.11) leads to a joint optimization problem for hyperlink prediction, which we formulate next. We note that since the problems in eqs. (6.11 – 6.12) fall into the integer programming paradigm, and since such problems are NP-complete, we relax the domains of $\mathbf{\Lambda_U}$ and $\mathbf{\Lambda_S}$ to the unit interval $[0, 1]$ instead of $\{0, 1\}$, just as Zhang et al. [132] do. Hence our final problem boils down to the following:

$$\min_{\substack{\mathbf{\Lambda_U} \in diag\left([0,1]^{|\mathcal{F}_{univ}|}\right) \\ \mathbf{\Lambda_S} \in diag\left([0,1]^{|\mathcal{F}|}\right)}} \left\|\boldsymbol{A} - \boldsymbol{S}\mathbf{\Lambda_S}\boldsymbol{S}^T - \boldsymbol{U}\mathbf{\Lambda_U}\boldsymbol{U}^T\right\|_F^2 + \left\|\Delta\boldsymbol{A} - \boldsymbol{U}\mathbf{\Lambda_U}\boldsymbol{U}^T\right\|_F^2 + \|\mathbf{\Lambda_S}\|_1 . \tag{6.13}$$

In summary, ***we have exploited our clique-closure hypothesis by explicitly forcing the objective function to consider cliques and near-cliques from the projected graph of the observed hyperlinks, as well as new information $\Delta\boldsymbol{A}$ simultaneously, and predict hyperlinks that explain them both.***

6.3.3 Alternating Minimization

Finding an optimal solution to the problem (6.13) can be done by minimizing it alternatively – first for $\boldsymbol{W}$, and then for $\mathbf{\Lambda_S}$ and $\mathbf{\Lambda_U}$.

This leads to an alternating optimization problem, where we first predict missing links with the help of $\boldsymbol{W}$ obtained as per eq. (6.10) and then predict missing hyperlinks by solving the optimization problem in eq. (6.13). At the end of each iteration, we update $\boldsymbol{A}$ with the new predicted links by adding $\boldsymbol{U}\mathbf{\Lambda_U}\boldsymbol{U}^T$. Overall, C3MM predicts hyperlinks by performing following steps alternatively:

Step 1: For fixed $\mathbf{\Lambda_U}$ from Step 2 below (or by fixing it to be a random matrix for the first iteration), solve for $\boldsymbol{W}$:

$$\min_{\boldsymbol{W} \in [0,\infty)^{|V| \times |V|}} \left\|\boldsymbol{A} + \boldsymbol{A}_{CN} + \boldsymbol{U}\mathbf{\Lambda_U}\boldsymbol{U}^T - \boldsymbol{W}\boldsymbol{W}^T\right\|_F^2 \tag{6.14}$$

Step 2: Defining $\Delta\boldsymbol{A} := \boldsymbol{W}\boldsymbol{W}^T - \boldsymbol{A}$ for $\boldsymbol{W}$ fixed from Step 1 above, find the optimal $\mathbf{\Lambda_U}$ according to eq. (6.13).

Since both Step 1 and Step 2 are convex optimization problems, we solve them by alternatively minimizing them for matrices $\mathbf{\Lambda_U}$, $\mathbf{\Lambda_S}$ and $\boldsymbol{W}$, and finally use $\mathbf{\Lambda_U}$, that denotes the newly predicted hyperlinks.

6.4 Related Work

The hyperlink prediction problem focuses on predicting unknown/unseen interactions between a set of nodes, whose analogue in usual networks is the link prediction problem. Here, we given a brief overview of the related work in both link- as well as hyperlink-prediction.

Although research in hyperlink prediction has been limited, its literature is convincing enough to vouch for its importance. Ever since the near-seminal works by Agarwal et al. [3] and Zhou et al. [135] that unite the fields of hypergraphs and machine learning, there has been four major works focusing on hyperlink prediction [11, 121, 132, 131]. Xu et al. [121] and both works by Zhang et al. [132, 131] deal with specific domains, *viz.*, email and metabolite networks respectively. Benson et al. [11], on the other hand, bring a multitude of domains to the table (see Section 6.5 for more details). While Zhang et al. [132] introduce a matrix completion based solution called Coordinated Matrix Minimization (CMM) that works well for a [11] restrict the problem to that of predicting the closure of a 3–4 sized open simplex, which is a problem temporal in nature.

Researchers have previously worked on the task of predicting links in heterogeneous and bipartite networks as well [60]; however, their relevance to the work in this chapter is limited since hyperlink prediction parallels neither to link prediction on such networks, nor their one-mode projections.

6.5 Experiments

We test our algorithm (C3MM) on both structural as well as temporal link prediction problems, and report results on diverse datasets, using a few baselines to compare against. But before that, we test our hypothesis (CCH) on these datasets, and elucidate that it holds statistically significantly for most of them. Let us describe the datasets we have used.

6.5.1 Datasets

We have performed our experiments on altogether *ten* datasets, of which *four* are *temporal* hypergraphs and we use the *six non-temporal metabolite* hypergraphs from Zhang et al. [132]. We suggest the reader to refer to Benson et al. [11] for an extensive analysis of the four (and more) *temporal* datasets, and to Zhang et al. [132] for the six *metabolites* datasets. A summary has been tabulated in Table 6.2.

6.5.2 Baselines

Coordinated Matrix Minimization (CMM) [132] as well as baseline algorithms mentioned in their paper form the baselines for our experiments. More specifically, we use the following methods as our baselines: Bayesian Sets (BS) [35], Spectral Hypergraph Clustering (SHC) [135], Factorization Machines (FM) [85], Katz [54, 11], and Hyper Common Neighbors (CN) [132, 66]. For more

Table 6.2: Temporal and non-temporal datasets that we use in our experiments.

Dataset	Temporal?	$\lvert V\rvert$	$\lvert\mathcal{F}\rvert$
`contact-primary-school`	Yes	242	11,161
`contact-high-school`	Yes	327	6,700
`MAG-Geo`	Yes	1,876	9,471
`tags-math-sx`	Yes	862	9,098
iJO1366	No	1,805	2,583
iAF1260b	No	1,668	2,388
iAF692	No	628	690
iHN637	No	698	785
iIT341	No	485	554
iAB_RBC_283	No	342	469

information we refer to Zhang et al. [132] or the respective references therein. To evaluate the performance of hyperlink prediction algorithms, we make use of the area under ROC curves (AUC) metric.

6.5.3 Data Preparation

We use Clique Negative Sampling (CNS) to under-sample the negative class of non-hyperlinks, as discussed in the previous chapter (Chapter 5). We sample 15 times as many non-hyperlinks as there are hyperlinks in the unobserved hypergraph for all of the temporal hypergraphs (*i.e.*, $\nu = 15$). For the non-temporal hypergraphs (*i.e.*, the *Metabolites* datasets), Zhang et al. [132] already refer to a manually curated negative class (or non-hyperlinks)[1]; hence there is no need to generate any negative samples. We fix the size of latent dimension for symmetric NMF in (6.10) to be $k = 30$ for the all the datasets, just as Zhang et al. [132] do as a default choice for CMM.

6.6 Results and Discussion

6.6.1 CCH Hypothesis Testing

We test our hypothesis (CCH) on a total of ten datasets, four of which are temporal, while remaining are non-temporal (Table 6.3). On temporal datasets, we test CCH using Algorithm 6, whereas for non-temporal ones the variation as mentioned in Section 6.2 is used. More specifically, for each of our datasets, we report the values of $hd(\mathcal{H})$, $cf(\mathcal{H})$, and also p-values of $\mathbb{H}_0$ (Hypothesis 2) over all hyperlinks $F \in \mathcal{F}$.

The first set of results (first four rows of Table 6.3) show that all temporal hypergraphs satisfy the hypothesis by a decent margin, in that $p < \alpha$. One can infer that in these settings, it is highly required for a group of nodes to have had dense lower-order interactions before the group evolves

[1]Owing to the knowledge domain experts have about "impossible" metabolic reactions.

Dataset	$hd(\mathcal{H})$	$cf(\mathcal{H})$	p-**value**	**Result** ($\alpha = 0.1$)
`contact-primary-school`	0.285	0.92	**0.001**	Rejects $\mathbb{H}_0$
`contact-high-school`	0.109	0.91	**0.000**	Rejects $\mathbb{H}_0$
`MAG-Geo`	0.014	0.27	**0.059**	Rejects $\mathbb{H}_0$
`tags-math-sx`	0.028	0.52	**0.021**	Rejects $\mathbb{H}_0$
iJO1366	0.009	0.07	0.102	Fails to reject $\mathbb{H}_0$
iAF1260b	0.008	0.07	**0.033**	Rejects $\mathbb{H}_0$
iAF692	0.027	0.08	0.571	Fails to reject $\mathbb{H}_0$
iHN637	0.028	0.03	0.658	Fails to reject $\mathbb{H}_0$
iIT341	0.034	0.04	0.813	Fails to reject $\mathbb{H}_0$
iAB_RBC_283	0.030	0.04	0.591	Fails to reject $\mathbb{H}_0$

Table 6.3: CCH Test on temporal and non-temporal datasets with significance level $\alpha = 0.1$. All temporal datasets reject $\mathbb{H}_0$ (*i.e.*, follow CCH) with significance $\alpha = 0.1$, and all but one (*iAF1260b*) non-temporal datasets fail to reject $\mathbb{H}_0$ (*i.e.*, don't follow CCH). We have performed a t-test with 8 degrees of freedom.

into a hyperlink. Also, as the hyperlink size increases, so does its mean pre-hyperedge density. It is therefore observed that three or four authors can relatively easily group together to collaborate on a common work, than bigger groups.

The second set of results (bottom six rows of Table 6.3) clearly show that metabolite datasets, which are non-temporal in nature, show little-to-no support for the hypothesis. Not much could be commented on the relative comparison between datasets since they are all equally low, wherein $cf(\mathcal{H})$ lies in the range of 3–8% and p-value much higher as compared to its temporal counterparts.

In summary, temporal datasets satisfy CCH with high confidences, while non-temporal ones fail miserably. The results we report in the bottom part of Table 6.3 are certain summaries of the static analysis of metabolite networks, which is not bound to follow a particular pattern, at least not the pattern we expect it to (namely, CCH).

6.6.2 Hyperlink Prediction

We present the results for hyperlink prediction on the four temporal datasets in Table 6.4. Table 6.4 reports mean AUC scores for C3MM versus CMM and its other baselines.

In all the temporal datasets, ***C3MM performs better than the other baselines, of which in particular, CMM (an approach that is similar to C3MM) has much lower AUC scores***. This supports the argument that our hypothesis (CCH) has helped identify hyperlinks that the earlier formulation did not. Of the datasets, MAG-G and tags-math-sx have the highest scores, since they are bigger datasets and have formed over a longer time range than the other ones. Of the other baselines, we have BS

Dataset	C3MM	CMM	BS	SHC	FM	Katz	CN
contact-primary-school	**0.590**	0.455	0.580	0.563	0.497	0.324	0.413
contact-high-school	**0.629**	0.382	0.624	0.537	0.490	0.308	0.391
MAG-Geo	**0.639**	0.380	0.637	0.626	0.262	0.274	0.350
tags-math-sx	**0.638**	0.476	0.590	0.549	0.374	0.374	0.430

Table 6.4: AUC scores of hyperlink prediction on temporal datasets. In all cases, C3MM outperforms CMM, our main baseline. The role of CCH in helping to identify hyperlinks better is hence evident.

(Bayesian Sets) that has a decent AUC for all datasets, except for tags-math-sx, and SHC seems to be the third best baseline.

One dataset that has a relatively higher p-value for CCH and despite this fact C3MM performing well is MAG-Geo, where we see most (73%) of the hyperlinks forming from non-cliques. This is possibly due to MAG-Geo being a co-authorship network where one would anticipate future collaboration among authors who have worked together in the past in some form. The higher p-value could be attributed to the fact that we take the hypergraph snapshot of recent 7 years, which ends up ignoring meaningful connections of the past.

At the same time, performance of C3MM drops for most of the non-temporal metabolite datasets. The only dataset which shows better performance for C3MM is *iAF1260b* while for the rest of the datasets performance drop is anywhere between 12% to 1%. Also it should be noted that *iAF1260b* is the only non-temporal dataset that satisfies CCH hypothesis as seen in Table 6.3 while the other datasets don't. This shows that ***C3MM is a better algorithm for hyperlink prediction when the CCH hypothesis is strongly supported by a dataset***.

6.7 Conclusion

Hyperlink prediction is a difficult task to perform, at least more difficult than what link prediction is. This is so both due to the number of possible hyperlinks in a given hypergraph (which is exponential in the number of nodes), as well as lack of multi-way heuristic scores. We set out to improve upon the current state-of-the-art (CMM) by introducing a clique-closure hypothesis into its objective function, ultimately forming C3MM. It is clear from the results on the hypothesis tests that we succeed in validating that it is cliques and co-cliques that close to form hyperlinks, instead of they being formed by co-cliques or disconnected structures. Embedding the hypothesis into the objective function leads to it significantly hunting down more hyperlinks which were missed by CMM. Another conclusion we draw is that hyperlink prediction on temporal and non-temporal datasets works differently, in that the latter predicts the future, and the former, the missing hyperlinks. While CMM works well on non-temporal datasets, C3MM better predicts future links.

This page has intentionally been left blank.

Chapter 7

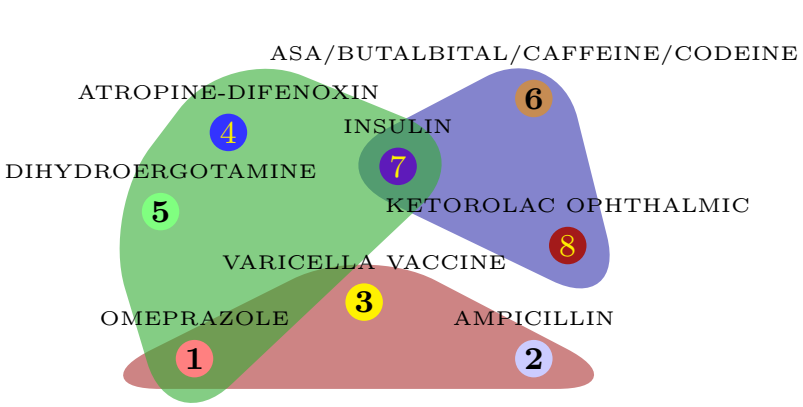
{Parts} > Whole: Sub-Higher Order Models for Hyperedges

> *"The basic thesis of Gestalt theory might be formulated thus: there are contexts in which what is happening in the whole cannot be deduced from the characteristics of the separate pieces, but conversely; what happens to a part of the whole is, in clearcut cases, determined by the laws of the inner structure of its whole."* ~ `Max Wertheimer`

WHILE any higher-order relation in a network has to be captured via a hyperedge (or a set of vertices), its containing vertices need not form its building blocks. In other words, the higher-order (HO) paradigm – that information flows from vertices to hyperedges – need not be true in general, and there could be a latent sub-higher-order (SHO) structure that governs the formation of hyperedges. We propose a novel, SHO-based approach to model the flow of information in a hypergraph by introducing the concept of "sub-hyperedges" and establish the advantage of using the SHO-paradigm. Since an ideal implementation is computationally expensive, we provide a greedy heuristic based on a novel sub-hyperedge scoring metric to reduce complexity. Finally, we use the SHO-based formulations to design a novel neural network architecture called SHONeN to learn better hyperedge embeddings. The higher performance of our model on hyperedge prediction over several real-world datasets exemplifies its superiority over popular baselines for hypergraphs.

7.1 Introduction

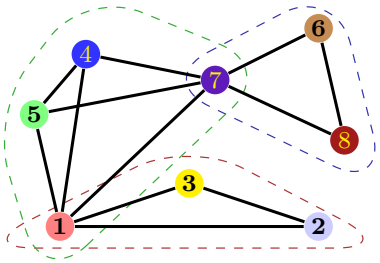
We know that a network comprising of higher-order relations cannot be sufficiently captured via edges (pairwise links), which is why *hyperedges* (sets of vertices) are used instead. Fig. 7.1 shows a *drug abuse warning network* (DAWN) [11] wherein "medications considered harmful when taken together" are grouped into hyperedges. One can see that the drugs `OMEPRAZOLE`, `VARICELLA VACCINE`



	a	b	c
$\emptyset$	1	1	1
$\{1\}$	1	0	1
⋮	⋮	⋮	⋮
$\{8\}$	0	1	0
$\{1,2\}$	1	0	0
⋮	⋮	⋮	⋮
$\{7,8\}$	0	1	0
$\{1,2,3\}$	1	0	0
⋮	⋮	⋮	⋮
$\{6,7,8\}$	0	1	0
⋮	⋮	⋮	⋮
$\{1,\cdots,8\}$	0	0	0

Powerset-Incidence Matrix, $\boldsymbol{P}$

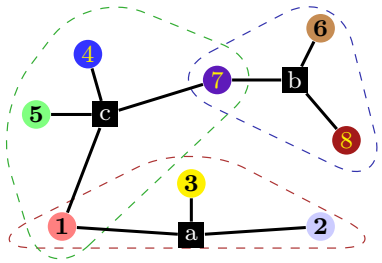
(a) Hypergraph $\mathcal{H}$ and its powerset incidence matrix $\boldsymbol{P}$ (shaded submatrix is $\boldsymbol{S}$ from (c) below)



	1	2	3	4	5	6	7	8
1	0	1	1	1	1	0	1	0
2	1	0	1	0	0	0	0	0
3	1	1	0	0	0	0	0	0
4	1	0	0	0	1	0	1	0
5	1	0	0	1	0	0	1	0
6	0	0	0	0	0	0	1	1
7	1	0	0	1	1	1	0	1
8	0	0	0	0	0	1	1	0

Adjacency Matrix, $\boldsymbol{A}$

(b) Clique expansion $\eta(\mathcal{H})$ and its adjacency matrix



	a	b	c
1	1	0	1
2	1	0	0
3	1	0	0
4	0	0	1
5	0	0	1
6	0	1	0
7	0	1	1
8	0	1	0

Incidence Matrix, $\boldsymbol{S}$

(c) Star expansion $\eta^*(\mathcal{H})$ and its incidence matrix

Figure 7.1: A hypergraph sampled from the Drug Abuse Warning Network (DAWN) dataset [11]. (a) A hypergraph with eight vertices denoting *medical drugs*, and three hyperedges corresponding to *drugs that react adversely when consumed together*. Also, on the right, we have its *powerset-incidence matrix* $\boldsymbol{P}$ (see eq. (7.2)) that introduces the concept of *sub-hyperedges* (one per row). Also, the shaded submatrix corresponds to the usual incidence matrix $\boldsymbol{S}$. (b) Its clique expansion [3], joining two nodes that co-appear in a hyperedge at least once. This forms the 2-order (2O) paradigm: *nodes communicate with each other pair-wise*, and result in the adjacency matrix $\boldsymbol{A}$. (c) Its star expansion [3], reifying each higher-order relation via a superficial node (black squares ■ labeled a, b, and c); this corresponds to a higher-order (HO) paradigm: *nodes communicate to (reified) hyperedges, and vice versa*, resulting in the incidence matrix $\boldsymbol{S}$.

and `AMPICILLIN` are advised not to be co-taken (a fact depicted by the maroon triplet in Fig. 7.1a). However no such advise has been given against taking `OMEPRAZOLE` and `AMPICILLIN` together, a fact that gets misrepresented by its induced graph (depicted in Fig. 7.1b). Had it been for mere graphs, we would have endured with such incorrect drug-abuse warnings. This shows that pair-wise induction of drug-interactions lose context and hence their higher-order ought be retained.

For a hypergraph $\mathcal{H} = (V, \mathcal{F})$, it is commonly assumed that the existence of a k-sized hyperedge $F := \{v_1, \ldots, v_k\} \in \mathcal{F}$ depends either on pairwise links over F's vertices (*e.g.*, *clique-expansion* [3],

as depicted in Fig. 7.1b), or on its individual nodes $v_1, \ldots, v_k$ along with a dummy node u (*star-expansion* [3], as per Fig. 7.1c).[1] The former corresponds to a 2-order (2O) paradigm involving edges, and the latter assumes a *reification* (introduction of a superficial node) of F, a higher-order (HO) paradigm. While multiple hypergraph models rely upon clique-expansion [135, 55], the best performance on standard hypergraph-oriented tasks has been reported by Hyper-SAGNN [133], a vertex-vertex self-attention based neural network. Moreover, typical random walks [135] on hypergraphs alternate between vertices and hyperedges, supporting the HO (or star-expansion) assumption. Hence, all hypergraph-based models essentially assume that flow of information is from nodes to hyperedges and vice-versa.

We hypothesize that ***neither the 2O, nor the HO paradigm fully captures the formation of hyperedges***. Instead, we propose ***a sub-higher-order (SHO) paradigm that is responsible for the evolution of hyperedges***. More specifically, nodes do not directly contribute in the formation of HO hyperedges, but do so via ***sub-hyperedges***, which are themselves smaller sets of nodes. To illustrate, in a co-authorship network (one where all collaborating authors form a hyperedge), the higher-order paradigm supports the view that authors (nodes) contribute individually to form author-groups (hyperedges). But it seems logical to assume that it is not just the individual contributions of authors, but those of even smaller author-groups (sub-hyperedges) that ultimately results in the formation of a hyperedge. This forms our SHO paradigm.

The state-of-the-art techniques for hyperedge embedding and prediction employ deep models for the job [133, 31, 7, 82, 122]. In this chapter, we too design a neural network called **S**ub-**H**igher-**O**rder **Ne**ural **N**etwork (SHONeN) exploiting the SHO argument and show improved performances over baselines for hyperedge prediction. We also analyze the SHO-effect on real-world hypergraphs and show that its contribution is non-trivial. Moreover, since we deal with subsets of hyperedges, the search for the best possible SHO structure is super-exponential (2^{2^k} for a hyperedge of size k), and hence we first define a scoring mechanism called *contextual standalone hyperdegree* (CoSH) and use it to provide a computation-friendly sub-optimal heuristic for hyperedge prediction.

In Section 7.2, we formalize the SHO notion in hypergraphs and devise a greedy heuristic for the same. Then we describe the SHONeN architecture for hyperedge prediction in Section 7.3. We conduct a number of experiments on a few real-world datasets in Section 7.5 and discuss the results in Section 7.6, before we conclude in Section 7.7.

7.1.1 Key Contributions

1. We establish a novel *sub-higher-order (SHO) paradigm* for hypergraphs which could model them better.

[1] More details about hypergraph-to-graph expansions could be obtained from Chapter 2, Section 2.1.

2. We design a novel neural network architecture called *SHONeN for hyperedge prediction* that is able to perform significantly better than its baselines on real-world hypergraphs.

3. We define *contextual standalone hyperdegree* ($CoSH$), a metric that scores node-groups according to their ability to stand without their incident hyperedges. We further use it to define a *sub-optimal heuristic* for better scalability of SHONeNs.

4. We provide *key insights into the working of the SHO-paradigm* by performing a qualitative analysis over real-world hypergraphs.

7.2 The SHO-structure for Hypergraphs

Given a hypergraph $\mathcal{H} = (V, \mathcal{F})$ where $V := \{v_1, \ldots, v_n\}$ and $\mathcal{F} := \{F_1, \ldots, F_m\} \subseteq 2^V$, it is known that it could be uniquely represented (up to a permutation of its columns) using its *incidence matrix* $\boldsymbol{S} \in \mathbb{R}^{n \times m}$ (defined in Chapter 2, Section 2.1), which essentially relates nodes to hyperedges. We first discuss the higher-order paradigm that is based on the incidence matrix.

7.2.1 The higher-order (HO) paradigm

A higher-order (HO) structure assumes a lossless "star" expansion [3] (see Figure 7.1c) of $\mathcal{H}$ into $\eta^*(\mathcal{H}) := (V^*, \mathcal{E}^*)$ defined as in Chapter 2. Since $\eta^*(\mathcal{H})$ is a bipartite graph, each vertex $v \in V$ is expected to pass on any information it has to hyperedges $\tilde{\Gamma}(v)$ incident on v (hyperneighbors); further, each hyperedge $F \in \mathcal{F}$ does the same to each containing vertex $v \in F$. In fact, $\boldsymbol{S}$ also represents the biadjacency matrix of graph $\eta^*(\mathcal{H})$. From the point-of-view of hyperedge prediction, wherein one has to predict whether an arbitrary group of nodes form a hyperedge or not, $\boldsymbol{S}$ could be looked at as an inverted data matrix of *features* (vertices) vs. *patterns* (hyperedges). Hence, the incidence matrix $\boldsymbol{S}$ of a hypergraph gives its HO representation, *i.e.*, relation between its vertices and hyperedges. We now move towards establishing our sub-higher-order (SHO) formulation.

7.2.2 The sub-higher-order (SHO) paradigm

For the SHO paradigm for a hypergraph $\mathcal{H} = (V, \mathcal{F})$, we would need to deal with the powerset $\mathcal{P}(V)$ of vertices V, that represents the set of all possible *sub-hyperedges* in $\mathcal{H}$. For reasons that will be clear soon, let us spend some effort arranging $\mathcal{P}V$ in a particular *fixed* order, numbered from 1 to 2^n, for ease of reference. Please refer to the Shortlex order $<_{slex}$ defined in Chapter 2, Definition 2.1. It could be applied to the powerset $\mathcal{P}(V)$ of V and any of its powerset's subsets. More particularly, we denote

$\mathcal{P}(V)$ as an ordered set $(F'_1, \ldots, F'_{2^n})$, arranged as:

$$\mathcal{P}(V) = (\underbrace{\emptyset}_{F'_1}, \underbrace{\{v_1\}, \ldots, \{v_n\}}_{F'_i:\ i=2,\ldots,n+1}, \underbrace{\{v_1, v_2\}, \ldots, \{v_{n-1}, v_n\}}_{F'_i:\ i=n+2,\ldots,\binom{n}{2}+n+1}, \cdots, \underbrace{V}_{F'_{2^n}}), \tag{7.1}$$

and the same can be inherited for any subset of $\mathcal{P}(V)$ (esp., for the set of hyperedges $\mathcal{F} \subseteq \mathcal{P}(V)$). Some more remarks on the shortlex ordering for the powerset of V follow.

Remarks 7.1. *The first element F'_1 of $\mathcal{P}(V)$ is $\emptyset$. The next n elements $F'_2, \ldots, F'_{n+1}$ are n singletons $\{v_1\}, \ldots, \{v_n\}$ respectively. Then for $n+2 \leq i \leq \binom{n}{2} + n + 1$, we have $\binom{n}{2}$ doubletons sorted in lexicographic order. This pattern continues and finally, we have the last element F'_{2^n} of the ordering to be the whole set V.*

For a hyperedge $F \in \mathcal{F}$, we assume sub-hyperedges ($F' \in \mathcal{P}(F)$), and not vertices ($v \in F$), to be its building blocks. Hence, we see a need to define a ***powerset-incidence matrix*** $\boldsymbol{P} \in \mathbb{R}^{2^n \times m}$ for a hypergraph as follows:

$$\boldsymbol{P}_{ij} := \mathbb{1}_{\mathcal{P}(F_j)}(F'_i) := \begin{cases} 1, & \text{if } F'_i \subseteq F_j, \\ 0, & \text{if } F'_i \not\subseteq F_j. \end{cases} \tag{7.2}$$

The i^{th} row $\boldsymbol{P}_i$ of $\boldsymbol{P}$ refers to the i^{th} sub-hyperedge F'_i as ordered by $<_{slex}$ (see Remarks 7.1). And the j^{th} column denotes the j^{th} hyperedge F_j, again, ordered by $<_{slex}$. Powerset-incidence matrix for the example hypergraph given in Figure 7.1 has been shown in Figure 7.1a (the matrix on the left). Taking Figure 7.1a as a typical example, let us discuss further about $\boldsymbol{P}$ in general.

Remarks 7.2. *The first row $\boldsymbol{P}_1$ of $\boldsymbol{P}$ is an m-dimensional all-one vector $\mathbf{1} := \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}^T$ since $\boldsymbol{P}_{1j} := \mathbb{1}_{\mathcal{P}(F_j)}(\emptyset) = 1\ \forall j$. For a given hypergraph $\mathcal{H} = (V, \mathcal{F})$, we have two matrices: the usual $n \times m$ incidence matrix $\boldsymbol{S}$ and a $2^n \times m$ powerset-incidence matrix $\boldsymbol{P}$ (equation (7.2)). While $\boldsymbol{S}$ gives an n-dimensional representation to each hyperedge, $\boldsymbol{P}$ gives a 2^n-dimensional one. The incidence matrix $\boldsymbol{S}$ connects vertices to hyperedges, and the powerset-incidence matrix $\boldsymbol{P}$ connects sub-hyperedges to hyperedges. More formally, the i^{th} row $\boldsymbol{S}_i$ of $\boldsymbol{S}$ corresponds to the i^{th} vertex $v_i \in V$, and the i^{th} row $\boldsymbol{P}_i$ of $\boldsymbol{P}$, to the i^{th} sub-hyperedge $F'_i \in \mathcal{P}(V)$. Moreover, rows of $\boldsymbol{P}$ would be ordered by $<_{slex}$. On one hand, for the incidence matrix $\boldsymbol{S}$, we have only those columns as "one", that correspond to hyperedges incident on a given vertex (row). While on the other, for the powerset-incidence matrix $\boldsymbol{P}$, we have "ones" in those columns that correspond to hyperedges that are "supersets" of a given sub-hyperedge (row).*

Essentially, we posit that a hyperedge $F \in \mathcal{F}$ does not receive information directly from its containing vertices $v \in F$, but from its sub-hyperedges $F' \subseteq F$. We now make a rank-based statement and

argue that it is due to this that the SHO-representation (as per $\boldsymbol{P}$) is more informative than the HO-one (as per $\boldsymbol{S}$). But before that, a lemma is in place, that would be useful later.

Lemma 7.1. *$\boldsymbol{S}$ is an $n \times m$ submatrix of $\boldsymbol{P}$.*

Proof. By definition of $\boldsymbol{P}$ from eq. (7.2), each row of $\boldsymbol{P}$ corresponds to a sub-hyperedge from $\mathcal{P}(V)$, and so, a submatrix corresponding to all the n *singleton* sub-hyperedges $\{v_1\}, \ldots, \{v_n\} \in \mathcal{P}(V)$ in $\boldsymbol{P}$ exactly matches $\boldsymbol{S}$. To see this, we have from Remarks 7.1 that rows "2 to $n+1$" in $\boldsymbol{P}$ correspond to singleton sub-hyperedges. Take the i^{th} row $\boldsymbol{S}_i$ of $\boldsymbol{S}$ whose j^{th} entry is as follows:

$$\begin{aligned}
\boldsymbol{S}_{ij} &:= \mathbb{1}_{F_j}(v_i) = \mathbb{1}_{\mathcal{P}(F_j)}(\{v_i\}) && (7.3)\\
&= \mathbb{1}_{\mathcal{P}(F_j)}\left(F'_{i+1}\right) && (\because F_{i+1} \text{ denotes singleton } \{v_i\}) \quad (7.4)\\
&= \boldsymbol{P}_{i+1,j} && (\text{from eq. (7.2)}). \quad (7.5)
\end{aligned}$$

Since the choice of j was arbitrary, we have $\boldsymbol{S}_{ij} = \boldsymbol{P}_{i+1,j}$ for each j, $1 \leq j \leq m$. And as per another remark from Remarks 7.1, we know that the first row of $\boldsymbol{P}$ is $\boldsymbol{P}_1 = \mathbf{1}$, an all-one vector. Hence, we have

$$\begin{aligned}
\boldsymbol{P} = \begin{bmatrix} \boldsymbol{P}_1^T \\ \begin{bmatrix} \boldsymbol{P}_2 & \cdots & \boldsymbol{P}_{n+1} \end{bmatrix}^T \\ \begin{bmatrix} \boldsymbol{P}_{n+2} & \cdots & \boldsymbol{P}_{2^n} \end{bmatrix}^T \end{bmatrix} &= \begin{bmatrix} \boldsymbol{P}_1^T \\ \begin{bmatrix} \boldsymbol{S}_1 & \cdots & \boldsymbol{S}_n \end{bmatrix}^T \\ \begin{bmatrix} \boldsymbol{P}_{n+2} & \cdots & \boldsymbol{P}_{2^n} \end{bmatrix}^T \end{bmatrix} \\
&= \begin{bmatrix} \boldsymbol{P}_1 & \boldsymbol{S}^T & \boldsymbol{Q}^T \end{bmatrix}^T
\end{aligned} \tag{7.6}$$

with $\boldsymbol{Q} := \begin{bmatrix} \boldsymbol{P}_{n+2} & \cdots & \boldsymbol{P}_{2^n} \end{bmatrix}^T$. □

Corollary 7.1 (of Lemma 7.1)**.** *The rank of $\boldsymbol{P}$ is greater than or equal to the rank of $\boldsymbol{S}$.*

The proof of the foregoing corollary is straightforward. Intuitively, we have that since $\boldsymbol{P}$ has a higher rank than $\boldsymbol{S}$, it spans a larger vector space, and gets a chance to explain the formation of hyperedges better than $\boldsymbol{S}$. To further see this, let us consider the *full-hypergraph* on a fixed set V of vertices, defined as $\mathcal{H}_{full} := (V, \mathcal{F}_{full} := \mathcal{P}(V))$, *i.e.*, a hypergraph containing *all* possible hyperedges.[1] Also let us define the incidence and powerset-incidence matrices $\boldsymbol{S}_{full} \in \mathbb{R}^{n \times 2^n}$ and $\boldsymbol{P}_{full} \in \mathbb{R}^{2^n \times 2^n}$ for $\mathcal{H}_{full}$, with both rows (sub-hyperedges) and columns (hyperedges) ordered as per $<_{slex}$. An illustration of the structure of $\boldsymbol{P}_{full}$ and $\boldsymbol{S}_{full}$ has been given in Figure 7.2. We now state an important lemma regarding the ranks of these matrices, as follows:

[1] Empty hyperedge is considered purely for theoretical interest, in practice, it is insignificant.

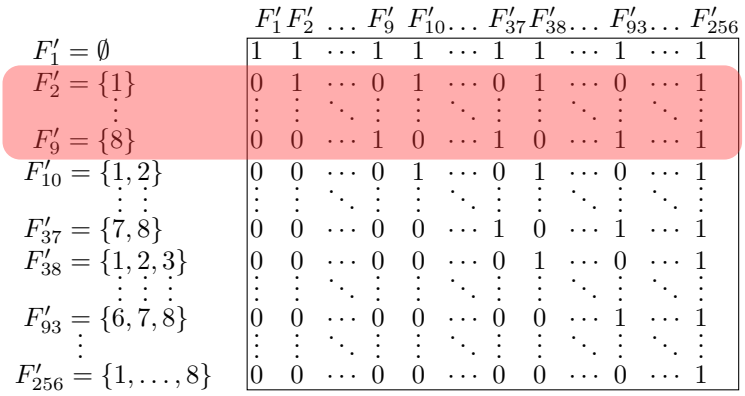


Figure 7.2: Snapshot of the powerset-incidence matrix $\boldsymbol{P}_{full} \in \mathbb{R}^{2^8 \times 2^8}$ and the incidence matrix $\boldsymbol{S}_{full} \in \mathbb{R}^{8 \times 2^8}$ (shaded) for the full hypergraph $\mathcal{H}_{full}$ with $V = \{1, \ldots, 8\}$.

Lemma 7.2. *$\boldsymbol{S}_{full}$ and $\boldsymbol{P}_{full}$ are full-rank matrices, with ranks $\rho(\boldsymbol{S}_{full}) = n$ and $\rho(\boldsymbol{P}_{full}) = 2^n$ respectively.*

Proof. Let us first argue for $\boldsymbol{S}_{full}$. Since its columns are arranged as per $<_{slex}$ (Chapter 2, Definition 2.1), we could ignore the first column that corresponds to the empty set (an all-one vector $\mathbf{1}$). The next n columns (*i.e.*, the 2^{nd} to the $(n+1)^{\text{th}}$ column) correspond to singletons, thereby forming an $n \times n$ identity matrix $\boldsymbol{I}_n$. Thus, $\boldsymbol{S}_{full}$ is of the form:

$$\boldsymbol{S}_{full} = \begin{bmatrix} \mathbf{0} & \boldsymbol{I}_n & \boldsymbol{S}' \end{bmatrix} \tag{7.7}$$

for some appropriate matrix $\boldsymbol{S}'$. Since the identity matrix $\boldsymbol{I}_n$ is a submatrix of $\boldsymbol{S}_{full}$, we have $\rho(\boldsymbol{S}_{full}) = n$.

Coming to the square matrix $\boldsymbol{P}_{full}$, we have the first row and column concerning the empty-set, the next n rows and columns denoting interaction among singletons, the next $\binom{n}{2}$ for double-tons, and so on. In other words, if $\boldsymbol{P}^{(k,l)} \in \mathbb{R}^{\binom{n}{k} \times \binom{n}{l}}$ is the $\binom{n}{k} \times \binom{n}{l}$ submatrix of $\boldsymbol{P}$ denoting powerset-incidence of k-sized sub-hyperedges in 2^V with l-sized hyperedges in $\mathcal{F}_{full}$, we could rewrite $\boldsymbol{P}_{full}$ as $\boldsymbol{P}_{full} = \begin{bmatrix} 1 & \mathbf{1}^T \\ \mathbf{0} & \boldsymbol{P}^* \end{bmatrix}$, where $\boldsymbol{P}^*$ can be expanded block-by-block as:

$$\boldsymbol{P}^* := \begin{bmatrix} \boldsymbol{P}^{(1,1)} & \ldots & \boldsymbol{P}^{(1,n)} \\ \vdots & \ddots & \vdots \\ \boldsymbol{P}^{(n,1)} & \ldots & \boldsymbol{P}^{(n,n)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I}_{\binom{n}{1}} & \ldots & \boldsymbol{P}^{(1,n)} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & 1 \end{bmatrix}, \tag{7.8}$$

which is a full-rank matrix (of rank $2^n - 1$) since it is upper-triangular with blocks of incidence matrices

$I_{\binom{n}{k}}$ as its diagonals. And since $\rho(\boldsymbol{P}_{full}) = \rho(\boldsymbol{P}^*) + 1$, we have $\rho(\boldsymbol{P}_{full}) = 2^n$. □

Although $\mathcal{H}_{full}$ rarely occurs in practice, it successfully illustrates the benefit of using $\boldsymbol{P}$ instead of $\boldsymbol{S}$. To model the information flow between the 2^n hyperedges in $\mathcal{H}_{full}$, while the incidence matrix $\boldsymbol{S}$ uses n vertices, the powerset-incidence matrix $\boldsymbol{P}$ resorts to using 2^n sub-hyperedges. As a result, given the task to span a 2^n-dimensional vector space $\mathbb{R}^{2^n}$, $\boldsymbol{S}$ fails miserably since its row-vectors $\boldsymbol{S}_1, \ldots, \boldsymbol{S}_n$ merely span an exponentially lower (n) dimensional subspace, *viz.*, row-space $\mathcal{R}(\boldsymbol{S}) \subseteq \mathbb{R}^{2^n}$. On the other hand, rows $\boldsymbol{P}_1, \ldots, \boldsymbol{P}_{2^n}$ of $\boldsymbol{P}$, successfully span $\mathbb{R}^{2^n}$ since they form a *basis* for it (owing to it being full-rank). Finally, we could move towards the generic case, where we have the number of hyperedges to be $m \leq 2^n$. Before proceeding, we make some more important remarks as follows.

Remarks 7.3. *The column-sets of the incidence matrix $\boldsymbol{S}$ and the powerset-incidence matrix $\boldsymbol{P}$ of a given hypergraph $\mathcal{H} = (V, \mathcal{F})$ are subsets of the column-sets of $\boldsymbol{S}_{full}$ and $\boldsymbol{P}_{full}$ respectively.*

These remarks give rise to an important lemma:

Lemma 7.3. *The powerset-incidence matrix $\boldsymbol{P}$ of any hypergraph is a full-rank matrix, with rank $\rho(\boldsymbol{P}) = m$.*

Proof. The proof of this is simple: Going by Remarks 7.3, since $\boldsymbol{P}$ is obtained merely by *selecting* m columns from $\boldsymbol{P}_{full}$ (a full-rank matrix itself) – each per hyperedge – its rank is exactly equal to the number of columns thus extracted, *i.e.,* m. □

Thus, we could repeat the argument made above for a generic hypergraph as well. To represent a hypergraph in the HO-paradigm, we use an n rank matrix $\boldsymbol{S}$, and in the SHO-paradigm, we use a m rank matrix $\boldsymbol{P}$. In general, we speculate that since $\boldsymbol{P}$ already captures the flow of information from vertices to hyperedges (via singletons), it also considers other SHO structures from a hyperedge. However, we conjecture that $\boldsymbol{P}$ would capture the HO structure of $\mathcal{H}$ better than what $\boldsymbol{S}$ does because of its higher rank. Moreover, the experiments we perform prove that ***$\boldsymbol{P}$ significantly improves the performance of hyperedge prediction***.

7.2.3 A Greedy Heuristic: T2C2

However, one major concern in implementing the $\boldsymbol{P}$-based SHO-paradigm is its computational complexity. To reduce the complexity of implementing the SHO-based hyperedge representation by a substantial amount, we develop a heuristic for the same. Such a heuristic should give us a subset of $\mathcal{P}(V)$ that could be used for the SHO-based hyperedge representation. Essentially, this would also mean taking a sub-matrix out of the rows of $\boldsymbol{P}$. Lest any vertex of a hyperedge be left-out in our heuristic, let us define a ***cover*** of hyperedge $F \in \mathcal{F}$ to be ***a set $\mathcal{C}_F$ of its sub-hyperedges whose union***

Algorithm 7: An algorithm to collect sub-hyperedges based on the Thresholded Top-CoSH Cover (**T2C2**) Heuristic. It first computes Contextual Standalone Hyperdegree (CoSH) scores for all sub-hyperedges of a hyperedge, sorts them, and then picks sub-hyperedges sequentially until the constraints in Definition 7.2 are satisfied.

Input: Set of hyperedges $\mathcal{F}$, A hyperedge F, A threshold t
Output: **T2C2** based sub-hyperedges

for $F' \subseteq F$ **do**
  $\tilde{\Gamma}_{F'} \leftarrow \{F'' \in \mathcal{F} \mid F' \subseteq F''\}$
  $\tilde{\Gamma}_{F \setminus F'} \leftarrow \bigcup_{u \in F \setminus F'} \{F'' \in \mathcal{F} \mid F' \cup \{u\} \subseteq F''\}$
  $\psi_F(F') \leftarrow \left|\Gamma_{F'} \setminus \Gamma_{F \setminus F'}\right|$
$\psi_F \leftarrow \text{L1NORMALIZE}(\psi_F)$
$\mathcal{F}'_F \leftarrow \text{SORTBYVALUE}(\psi_F)$
$\textbf{T2C2}(F) \leftarrow \{\}$
$\mathcal{C}(F) \leftarrow \{\}$
$CCoSH(F) \leftarrow 0$
for $i \leftarrow 1$ **to** $2^{|F|}$ **do**
  $\textbf{T2C2}(F) \leftarrow \textbf{T2C2}(F) \cup \{\mathcal{F}'_F[i]\}$
  $\mathcal{C}(F) \leftarrow \mathcal{C}(F) \cup \mathcal{F}'_F[i]$
  $CCoSH(F) = CCoSH(F) + \psi_F(\mathcal{F}'_F[i])$
  if $\mathcal{C}(F) \setminus F = \emptyset$ *and* $CCoSH(F) \geq t$ **then**
    break
return $\textbf{T2C2}(F)$

equals F. In other words, a cover of F is nothing but its soft-partition. A heuristic according to which the set of sub-hyperedges of every hyperedge forms its cover, is called a *cover-compliant heuristic*. We know at least two heuristics that are cover-compliant:

Lemma 7.4. *Each of the* $\boldsymbol{S}$*-based and* $\boldsymbol{P}$*-based heuristics are cover-compliant.*

Proof. Rows of $\boldsymbol{S}$ consider singletons $\{v\} \subseteq F$ as sub-hyperedges of F. And since $\bigcup_{v \in F}\{v\} = F$, an $\boldsymbol{S}$-based heuristic is cover-compliant. Rows of $\boldsymbol{P}$ consider all subsets $F' \subseteq F$ as sub-hyperedges of F. And since $\bigcup_{F' \subseteq F} F' = F$, a $\boldsymbol{P}$-based heuristic is cover-compliant. □

We now define a heuristic that we show to work better than a singleton-based HO-heuristic. Our major goal is to come up with a logical means to *select a few sub-hyperedges that improve the representability of* $\mathcal{H}$ *by incorporating its SHO-structure; not-to-mention, it should be cover-compliant*. But before defining such a heuristic, we define a score for each sub-hyperedge of a hyperedge as follows. Then, for each hyperedge, we will use this score to greedily pick sub-hyperedges until it covers the hyperedge.

7. {PARTS} > WHOLE: SUB-HIGHER ORDER MODELS FOR HYPEREDGES

Definition 7.1 (***Contextual standalone hyperdegree***)**.** *Given a hypergraph $\mathcal{H} = (V, \mathcal{F})$, we define the **contextual standalone hyperdegree** (CoSH) $\psi_F(F')$ of a sub-hyperedge $F' \subseteq F$ w.r.t. hyperedge $F \in \mathcal{F}$ as $\psi_F(F') := |\Psi_F(F')|$, where $\Psi_F(F')$ represents the **contextual standalone hyperneighborhood** of sub-hyperedge F' w.r.t. hyperedge F, defined as:*

$$\Psi_F(F') := \tilde{\Gamma}(F') \setminus \bigcup_{u \in F \setminus F'} \tilde{\Gamma}(u),$$

where $\tilde{\Gamma}(v) := \{F \in \mathcal{F} \mid v \in F\}$, and $\tilde{\Gamma}(F') := \{F \in \mathcal{F} \mid F' \subseteq F\}$, the hyperneighbors of a vertex v and a sub-hyperedge F' respectively.

For a given hyperedge F, this score assigns a measure of "standalone-ability" to each of its sub-hyperedges F'. By standalone-ability, we mean the ability of the sub-hyperedge F' to sustain without any other nodes from F. Please note that for the same sub-hyperedge F', its standalone-ability varies from hyperedge to hyperedge; hence the term "contextual" is used. Using the $CoSH$ scores defined as above, we define our heuristic that successfully represents a hyperedge in the sub-higher order domain as follows:

Definition 7.2 (***Thresholded Top-CoSH Cover (T2C2)***)**.** *For a hyperedge $F \in \mathcal{F}$ of size $s = |F|$, if $\mathcal{P}(V) := (F'_1, \ldots, F'_{2^s})$, sorted by $CoSH$ scores ($\psi_F(F'_1) \geq \cdots \geq \psi_F(F'_{2^s})$), consider **T2C2**$(F) := \{F'_1, \ldots, F'_k\}$ to be the **t-threshold top-CoSH heuristic** set of sub-hyperedges if:*

1. $\bigcup_{i=1}^{k} F'_i = F$ *and* $\left(\sum_{i=1}^{k} \psi_F(F'_i)\right) \geq \left(t \cdot \sum_{i=1}^{2^s} \psi_F(F'_i)\right)$,
2. $\bigcup_{i=1}^{k-1} F'_i \subsetneq F$ *or* $\left(\sum_{i=1}^{k-1} \psi_F(F'_i)\right) < \left(t \cdot \sum_{i=1}^{2^s} \psi_F(F'_i)\right)$.

Algorithm 7 computes the **T2C2** sub-hyperedges for each hyperedge in $\mathcal{F}$. Once $CoSH$ scores $\psi_F(\cdot)$ are available for all possible subsets of a hyperedge, the next step is to figure out a soft partition, **T2C2**(F) of F. First, the scores $\psi_F(\cdot)$ are normalized and then sub-hyperedges are sorted according to their $CoSH$ score. Next, we keep adding them to **T2C2**(F) one by one until we have covered all nodes of the hyperedge and their cumulative CoSH score crosses a specified threshold. In addition to this, we also propose *another heuristic*, so as to remain consistent with the literature that deals merely with singletons. The idea is to simply, for each hyperedge F, *forcefully include* its singletons once a cover has been found via **T2C2**(F). We call this heuristic **T2C2S**:

Definition 7.3 (***T2C2S: T2C2+Singleton***)**.** *For a hyperedge $F \in \mathcal{F}$, define **T2C2S**$(F) :=$ **T2C2**$(F) \cup \{\{v\} \mid v \in F\}$.*

Basically, the sub-hyperedges that have a high contextual standalone hyperdegree get selected. More clearly, if there is a sub-hyperedge F' that occurs with high frequency even *without* its superset hyperedge $F \supset F'$ (i.e., F' is also a sub-hyperedge of hyperedges other than F), we say it has a high "standalone-ability" (since it does not need the hyperedge F to stand on its own). Such sub-hyperedges would be ranked higher than others for every hyperedge, and the **T2C2** heuristic would thus select them first. The **T2C2S** heuristic includes all singletons explicitly.

7.2.4 Complexity of the SHO-paradigm

If for a given hypergraph $\mathcal{H} = (V, \mathcal{F})$, we define $s_{max} := \max_{F \in \mathcal{F}} |F|$, the *maximum hyperedge size*, we need in the HO-paradigm, a storage of $\mathcal{O}(n \cdot m)$ for S, which, in the worst case, is $\mathcal{O}(n \cdot 2^n)$. On the other hand, in the SHO-paradigm, we need a storage of $\mathcal{O}(2^n \cdot m)$, which, in the worst case, becomes $\mathcal{O}(2^n \cdot 2^n) = \mathcal{O}(4^n)$. However, exhaustively searching through the space of all subsets of the set of all sub-hyperedges to find an optimal set thereof is much more computationally infeasible since the search-space itself is super-exponential $\mathcal{O}(m \cdot 2^{2^{s_{max}}})$. Hence, we proposed simple greedy heuristics **T2C2** and **T2C2S** to obtain a subset of all sub-hyperedges which can then be used for the information propagation. A nice property of the greedy heuristic is that the number of sub-hyperedges obtained is of the order of hyperedge size, and hence time-complexity reduces from $O(m \cdot 2^{s_{max}})$ to $O(m \cdot s_{max})$, given that we have the $CoSH$ scores, which is a pre-processing step performed only once.

7.3 The SHONeN Architecture

Our ultimate goal is to design a message-passing based neural architecture for hypergraphs, to model the flow of information through sub-hyperedges of a hyperedge. Message-passing based neural architectures have been popular in networks, wherein information flows between neighboring nodes. The key idea of the proposed model is to capture the information flow from nodes to hyperedges through sub-hyperedges; we have termed this the *sub-higher-order (SHO) paradigm*.

The first step is to obtain the $CoSH$ scores $\psi_F(\cdot)$ for all sub-hyperedges of a hyperedge F. Once we have them, we use the simple greedy heuristic called **T2C2** (or **T2C2S**), which has been explained in Section 7.2.3, to obtain sub-hyperedges $\textbf{T2C2}(F) \subseteq 2^F$ (for each hyperedge $F \in \mathcal{F}$). These sub-hyperedges are then used to model the flow of information from nodes to a hyperedge in a neural network. Specifically, information at node-level is passed to the sub-hyperedges and then to hyperedges. Then, information from a hyperedge is distributed back to nodes and this goes on and on over multiple layers.

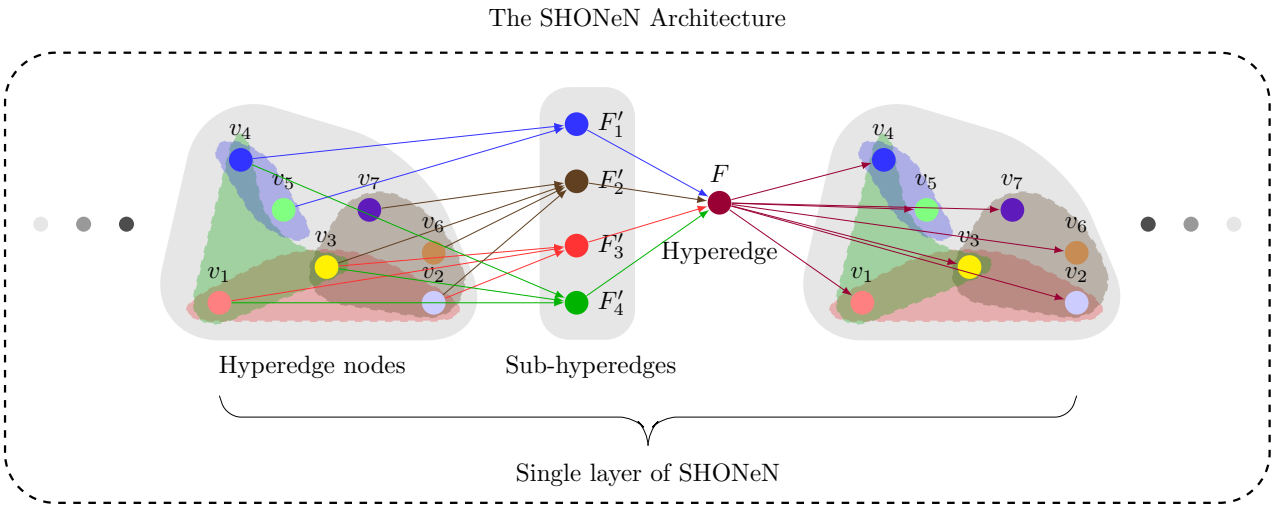


Figure 7.3: Information flow in a single layer of SHONeN. Sub-hyperedges $\texttt{T2C2}(F) = \{F'_1, \ldots, F'_4\}$ of a hyperedge $F = \{v_1, \ldots, v_7\}$ are shown in different lightly shaded regions as subsets $\{v_1, v_3, v_4\}$, $\{v_1, v_2, v_3\}$, $\{v_4, v_5\}$, and $\{v_2, v_3, v_6, v_7\}$ of F. First, the embedding of a sub-hyperedge $F' \subseteq F$ (*i.e.*, $\in \texttt{T2C2}(F)$) is obtained from that of its containing nodes v as specified in eq. (7.9). This is shown as arrows joining the 7 vertices to the 4 sub-hyperedges. These embeddings are then combined using a Multi-head Attention layer as per eq. (7.10) to form a single embedding for the hyperedge F (the four connections from $F'_1, \ldots, F'_4$ to F), which is distributed back to te nodes using eq. (7.11). The information flows in subsequent layers in the same manner and eventually, the final embedding of a hyperedge is obtained by concatenating its embeddings from all layers as shown in eq. (7.12).
Note: This only represents one layer of SHONeN. The actual architecture is similar to Hyper-SAGNN (Figure 7.4) where instead of the attention blocks, we use SHONeN blocks.

7.3.1 Information Flow

The information on nodes, sub-hyperedges and hyperedges is captured in the form of a latent representation, as is usually the case with deep learning models. There are three key aspects to the SHONeN architecture: (i) accumulating node information into sub-hyperedges (ii) accumulating sub-hyperedge information into a hyperedge (iii) distributing hyperedge information back to nodes. Let us discuss them one by one, and follow Fig. 7.3 for reference.

7.3.1.1 Vertex to sub-hyperedge

Let $\boldsymbol{X}_v^{(l)} \in \mathbb{R}^{d(l)}$ be the $d(l)$-dimensional embedding of a node v at the l^{th} hidden layer. Then the embedding $\boldsymbol{h}_{F'}^{(l)} \in \mathbb{R}^{d(l)}$ of a sub-hyperedge F' of F ($F' \subseteq F$) would be captured as, simply:

$$\boldsymbol{h}_{F'}^{(l)} = \frac{1}{|F'|} \sum_{v \in F'} \boldsymbol{X}_v^{(l)}. \tag{7.9}$$

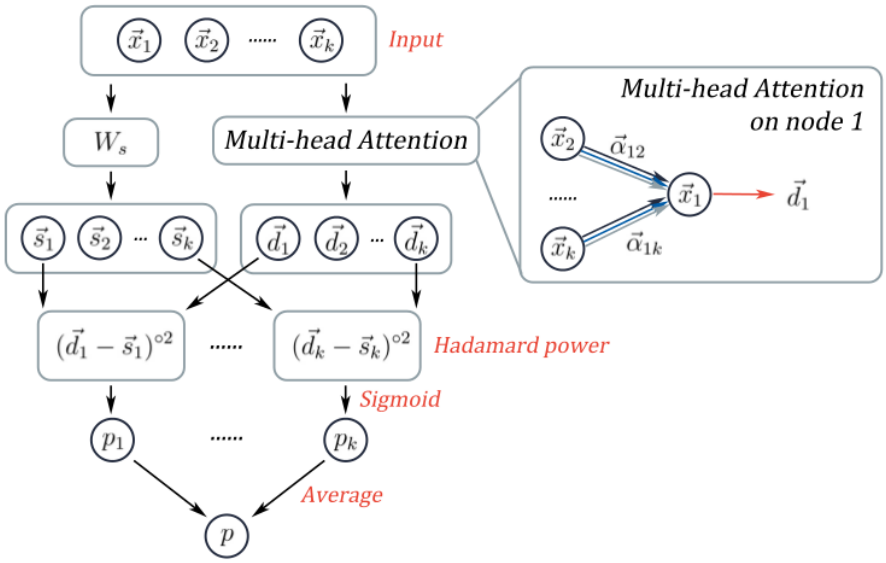


Figure 7.4: The architecture of HyperSAGNN [133], whose self-attention layer (Multi-head Attention) could be replaced by the one shown in Figure 7.3 to give the whole SHONeN architecture.

7.3.1.2 Sub-hyperedge to hyperedge

The next step is to generate the embedding of hyperedge F from those of its sub-hyperedges in $\texttt{T2C2}(F)$. We use *self-attention* between sub-hyperedges to generate a new "attended" embedding of sub-hyperedges which are then pooled using the average operation (AVGPOOL) to generate an embedding of the hyperedge. The self-attention we use is the same as multi-head self-attention (MHATT) described by Vaswani et al. [109]. If $\boldsymbol{h}_F^{(l)} \in \mathbb{R}^{d(l)}$ denotes the embedding of hyperedge F at layer l, we have:

$$\boldsymbol{h}_F^{(l)} := \text{AVGPOOL}\left(\text{MHATT}\left(\left\{\boldsymbol{h}_{F'}^{(l)} \mid F' \in \texttt{T2C2}(F)\right\}\right)\right) \tag{7.10}$$

7.3.1.3 Hyperedge to Vertex

Once we have with us the l-level embedding $\boldsymbol{h}_F^{(l)} \in \mathbb{R}^{d(l)}$ of each hyperedge $F \in \mathcal{F}$, we generate the (next-level) embedding $\boldsymbol{X}_v^{(l+1)}$ of nodes from them as follows:

$$\boldsymbol{X}_v^{(l+1)} := \frac{1}{|\tilde{\Gamma}(v)|} \sum_{F \in \tilde{\Gamma}(v)} ReLU\left(\boldsymbol{W}^{(l)} \boldsymbol{h}_F^{(l)}\right), \tag{7.11}$$

where $\tilde{\Gamma}(v) := \{F \in \mathcal{F} \mid v \in F\}$ and $\boldsymbol{W}^{(l)} \in \mathbb{R}^{d(l+1) \times d(l)}$.

Forward propagation in a single SHONeN layer comprises of the three steps described above (equations (7.9–7.11)). The final embedding of a hyperedge F is obtained by concatenating its

embeddings from all layers and passing it through a feed-forward network, which is shown as follows:

$$\boldsymbol{Z}_F := FFN\left(\boldsymbol{h}_F^{(1)} \parallel \boldsymbol{h}_F^{(2)} \parallel ... \parallel \boldsymbol{h}_F^{(l)}\right). \tag{7.12}$$

We use a single layer of SHONeN which generates node embeddings $\boldsymbol{X}_v^{(1)}$ and hyperedge embeddings $\boldsymbol{h}_F^{(1)}$ as outputs as shown above, for each $v \in V$ and $F \in \mathcal{F}$. The sub-hyperedge embeddings $\boldsymbol{h}_{F'}^{(2)}$ of the second layer as per eq. (7.9), which are then simply averaged (as opposed to applying MHATT) to generate the second layer hyperedge embedding $\boldsymbol{h}_F^{(2)}$.

7.3.2 Relation with Hypergraph NNs

It could be further argued that the message-passing flow modeled by standard hypergraph-based neural networks such as HGNN [31] or HyperGAT [7] form a special case of SHONeNs. To show this, assume that sub-hyperedges of a hyperedge F consist merely of singleton nodes $\texttt{SING}(F) := \{\{v\} \mid v \in F\}$. In eq. (7.10), if we replace the information flow from "sub-hyperedges to hyperedge" by a simple AVGPOOL pooling and remove MHATT attention, then we would have:

$$\boldsymbol{h}_F^{(l)} = \text{AVGPOOL}\left(\{\boldsymbol{h}_v^{(l)} \mid v \in F\}\right) = \frac{1}{|F|}\sum_{v \in F} \boldsymbol{h}_v^{(l)}. \tag{7.13}$$

This, in the light of eq. (7.11), generates the vertex embeddings in the next layer ($\boldsymbol{X}^{(l+1)} \in \mathbb{R}^{n \times d(l+1)}$) to be as follows:

$$\boldsymbol{X}^{(l+1)} = ReLU\left(\boldsymbol{D}_v^{-1}\boldsymbol{S}\boldsymbol{D}_e^{-1}\boldsymbol{S}^T\boldsymbol{X}^{(l)}\boldsymbol{W}^{(l)}\right), \tag{7.14}$$

where $\boldsymbol{S} \in \mathbb{R}^{n \times m}$ is the incidence matrix, $\boldsymbol{D}_v = diag(\text{COLSUM}(\boldsymbol{S})) \in \mathbb{R}^{n \times n}$ and $\boldsymbol{D}_e = diag(\text{ROWSUM}(\boldsymbol{S})) \in \mathbb{R}^{m \times m}$ are normalizing diagonal matrices, $\boldsymbol{X}^{(l)} \in \mathbb{R}^{n \times d(l)}$ are the l^{th} layer embeddings, and $\boldsymbol{W}^{(l)} \in \mathbb{R}^{d(l) \times d(l+1)}$ are model parameters. The neural model in eq. (7.14) is precisely the row-normalized hypergraph convolution network for unweighted hypergraphs proposed by Bai et al. [7]. Moreover, it should be noted that while HGNN [31] uses a symmetrically normalized hypergraph Laplacian, eq. (7.14) models the spectral convolution on hypergraphs through an asymmetrically normalized hypergraph Laplacian. The foregoing analysis shows that our approach for modelling information via the sub-higher-order (SHO) paradigm is indeed much more generic compared to those in the literature. In fact, a singleton-based information flow establishes an equivalence between the two.

SHONeN is a generalizable architecture, in that its pairwise variant is in line with Hypergraph NNs, which are themselves generalizable to Graph NNs for graphs. Moreover, for size-2 hyperedges (edges),

there would be a single sub-hyperedge: the edge itself. So while prediction of hyperedges depends on their relative frequency, what sub-hyperedges provide is a new set of features for the model to learn well.

7.4 Related Work

Quite a few models for hypergraphs have been proposed in the literature, and all of them consider either the 2-order clique-expansion approach, or the higher-order one [3, 135]. The almost-seminal works by Agarwal et al. [3] and Zhou et al. [135] sparked interest in the intersection of machine learning and hypergraphs, and simple models for hypergraphs were proposed [121, 132, 11]. The sudden surge in neural approaches to model networks [38, 55, 110, 16] has paved way for more deep hypergraph models to be formulated. Of them, the popular ones are: Deep Hyperedges [82], DHNE [107], HyperGCN [122], Hyperedge2vec [93], Hyper-SAGNN [133], *etc*. While some deal with uniform hypergraphs, others [123, 7] turn out to be trivial extensions of graph convolution [55].

Hyper-SAGNN by Zhang et al. [133] is a novel application of self-attention to hyperedges, but is limited to connecting pairs of nodes in a hyperedge. Nevertheless, their embeddings have been shown to give the best performance for hyperedge prediction and serves as the current state-of-the-art for the job. Hence, while they assume the flow of information from nodes to hyperedges and work in the higher-order (HO) paradigm, ***we are the first ones to introduce the sub-higher-order (SHO) information flow***. However, similar structures called weighted simplicial complexes have been used in the past to model their recurrence in an evolving simplicial hypergraph [92].

7.5 Experiments

We demonstrate the SHO-based paradigm by performing hyperedge prediction on various real-world hypergraphs, as described in Table 7.1.

Table 7.1: Datasets used in this chapter.

Dataset	$\lvert V\rvert$	$\lvert\mathcal{F}\rvert$
`email-Enron`	1,512	143
`contact-high-school`	7,818	327
`contact-primary-school`	12,704	242
`MAG-Geo`	10,708	1,371
`tags-math-sx`	9,833	862
`coauth-DBLP`	8,529	5,541
`NDC-substances`	9,106	5,043
`DAWN`	7,906	2,558

7.5.1 Datasets and Preprocessing

We use a bunch of homogeneous hypergraphs from Benson et al. [11] that belong to various domains: *email* (`email-Enron`), *contact* (`contact-high-school`, `contact-primary-school`), *co-authorship* (`MAG-Geo`, `coauth-DBLP`), *tags* (`tags-math-sx`), and *medical drugs* (`NDC-substances`, `DAWN`).

As far as processing the datasets is concerned, `NDC-substances` is a huge dataset with many hyperedges having sizes more than 12; so we restrict the size of largest hyperedge to 15, and the maximum sub-hyperedge size to be 8. Moreover, we use the same core-based technique as used by Liben-Nowell et al. [66] to pre-process co-authorship networks such as `MAG-Geo` and `coauth-DBLP`. Also, we only consider hyperedges with multiplicity ≥ 3 for the DAWN dataset (so as to trigger an abuse warning only after 3 or more reaction instances to the same drug combination).

Since we aim to solve a classification problem, negative class subsampling (*i.e.*, sampling non-hyperedges) becomes an important step, since the set of "all" possible non-hyperedges is $O(2^{|V|})$ – a number too huge for our algorithms to run. We use Motif Negative Sampling (MNS) [81] (as described in Chapter 5, Section 5.2.4), wherein to generate the negative class, those connected components from the *clique-expansion* of the hypergraph are sampled, which are not observed as hyperedges. This ensures a fair equivalence between both the classes.

The negative-to-positive class ratio is fixed to be 5:1, and a temporal split of 80:20 is used to split data into train and test hyperedges. Further, we use a 75:25 observed-unobserved split of the train hyperedges; the model is trained on predicting unobserved train hyperedges using information from observed train ones. And it is finally evaluated by predicting test hyperedges. Also, as is customary in imbalanced classification, we use Area Under ROC Curve (AUC) to measure hyperedge prediction performances.

7.5.2 Baselines

We compare the performance of SHONeNs (based on two heuristics: **`T2C2`** and **`T2C2S`**) against three other baselines, two of which are higher-order (HO) oriented and the third uses clique-expansion of hypergraphs (a 2O paradigm). They are described as follows:

1. **HGNN**: HGNN [31] uses spectral convolution on a hypergraph and generates latent embeddings for its nodes. We use average pooling along with multi-head self-attention (see eq. (7.10)).

2. **node2vec-SAG**: node2vec [38] uses clique-expansion of hypergraphs, followed by random walks that generate node embeddings. Again, average-pooling is used along with multi-head self-attention (see eq. (7.10)).

Table 7.2: Hyperedge prediction performance scores (% AUC). SHONeN-**T2C2** and SHONeN-**T2C2S** are the models proposed in Section 7.3, and use sub-hyperedges generated by the **T2C2** and the **T2C2S** heuristics as per Definitions (7.2) and (7.3) respectively. It is clear that for most datasets, one of SHONeN-**T2C2** and SHONeN-**T2C2S** performs the best.

Dataset	SHONeN-T2C2	SHONeN-T2C2S	HGNN	Node2Vec	HyperSAGNN
email-Enron	80.36 ± 1.86	**80.53 ± 1.54**	*77.60 ± 0.18*	76.38 ± 2.69	77.44 ± 1.62
contact-high-school	65.26 ± 1.11	**66.11 ± 1.07**	63.03 ± 0.46	60.55 ± 2.52	*63.47 ± 1.94*
contact-primary-school	**78.94 ± 0.43**	77.50 ± 0.76	68.35 ± 2.69	71.52 ± 2.57	*72.01 ± 2.10*
MAG-Geo	**69.93 ± 1.02**	68.98 ± 0.87	*61.27 ± 0.13*	57.60 ± 1.82	60.60 ± 1.84
tags-math-sx	**79.35 ± 0.28**	79.05 ± 0.41	67.08 ± 1.54	71.64 ± 1.17	*76.58 ± 0.82*
coauth-DBLP	**75.54 ± 0.34**	73.15 ± 0.48	66.01 ± 0.60	69.75 ± 1.06	*74.04 ± 0.72*
NDC-substances	80.68 ± 0.81	**80.71 ± 0.57**	71.10 ± 5.31	71.67 ± 1.71	*76.43 ± 4.05*
DAWN	**90.38 ± 0.27**	90.18 ± 0.44	78.22 ± 0.00	81.52 ± 0.91	*87.01 ± 0.22*

3. **HyperSAGNN**: HyperSAGNN [133] uses node2vec embeddings followed by self-attention.

7.5.3 Configuration of Hyperparameters

Both node- and hyperedge-embedding dimensions are fixed to 64 for all models including SHONeN. We initialize node embeddings for SHONeN with node2vec, as is the case with HyperSAGNN. Node2vec hyperparameters are the same across all models: "window size" $= 10$, "number of random walks" $= 40$ per node, and the rest according to the literature [38, 133]. We use 4 attention heads for self-attention throughout all the models. Adam optimizer is used for all our experiments with a learning rate of 0.001. The threshold t used to select the **T2C2**-based sub-hyperedges (see Definition 7.2 and Algorithm 7) of a hyperedge has been fixed to 0.7. The variation in t affects the choice threshold with which highly ranked sub-hyperedges are picked cumulatively. If we take t to be zero, **T2C2** picks no sub-hyperedges. On the other hand, a value of $t = 1$ would pick all sub-hyperedges. This could be tuned for best results.

7.6 Results and Discussion

We first list down the hyperedge prediction performance scores for SHONeN and other baselines. Then, we also reason for the performance improvement in SHONeNs by showing that hyperedges F having larger-sized sub-hyperedges in their respective **T2C2**(F) are the ones where our SHO-paradigm contributes the most.

7.6.1 Hyperedge Prediction

The performance of SHONeN along with other baselines for hyperedge prediction is shown in Table 7.2. SHONeN-**T2C2** is the same as SHONeN-**T2C2S**, except for the fact that instead of using **T2C2**, the

latter uses the **T2C2S** heuristic (Definition 7.3). SHONeN's superior performance is evident from Table 7.2, where for all datasets, it has the best AUC scores.

With the exception of `coauth-DBLP`, SHONeN-**T2C2** and SHONeN-**T2C2S** have highest and second highest performances for all the datasets. Also note that SHONeN shows significant improvements over HGNN and HyperSAGNN, which consider vertices (and not sub-hyperedges) in their information flow model. This could effectively be thought of as taking singleton sub-hyperedges. Hence, this validates the fact that the SHO-domain in hyperedges is an important paradigm for modeling the information flow therein. Moreover, some datasets have better performance on SHONeN-**T2C2S** as compared to SHONeN-**T2C2**, while for others, the latter does better than the former; however, the difference is nominal. This shows that the addition of singleton groups (as part of **T2C2S** from Definition 7.3) does not help much when the SHO-based heuristic **T2C2** is used. Nevertheless, their addition improves the AUC performances. Moreover, we empirically observed that increasing the number of layers in SHONeN does not lead to any significant improvements as compared to using only a single SHONeN layer.

We further analyse the ***precision*** scores of predicting hyperedges based on the notion of a "size-rank" score for sub-hyperedges in a hyperedge. For this, let us define a ***size-rank score*** $SR(F)$ of a hyperedge F as

$$SR(F) := \sum_{i=1}^{k} \frac{|F'_i|}{\log(i+1)}, \tag{7.15}$$

where $F'_i, i = 1, \ldots, k$ are the sub-hyperedges of hyperedge F fetched by **T2C2**(F) (indexed as per the decreasing order of CoSH scores). Intuitively, for hyperedges F whose **T2C2**(F) has a ***lot of singletons scored above other sub-hyperedges***, we would have a smaller $SR(F)$, whereas $SR(F)$ would be largest for hyperedges having ***highly-scored large-sized sub-hyperedges***. We bin the `email-Enron` hyperedges according to their SR values and calculate the precision for hyperedge prediction within each bin. These precision values are shown in Fig. 7.5b, while Figure 7.5a shows the same for HyperSAGNN on the same dataset. It can be observed that SHONeN gives higher precision value to hyperedges having a high SR value (*i.e.*, those for which CoSH scores for bigger sub-hyperedges is high). The absolute increment in precision for different bins has been shown in Figure 7.5c. Again, increment is higher for hyperedges having high SR values. In summary, this shows that ***the SHO-paradigm has the most effect on hyperedges with a high*** SR ***value, i.e., for which bigger sub-hyperedges have higher CoSH scores***. These are the hyperedges where soft-partition **T2C2**(F) has the most impact as compared to the the ones where singletons are scored higher as per CoSH.

7.6.2 The CoSH Scores

In this section, we analyze the $CoSH$ (Contextual Standalone Hyperdegree) scores that we had defined in Definition 7.1. Essentially, we show that the total information captured by sub-hyperedges $\texttt{T2C2}(F)$ of a hyperedge F is higher for the SHO-paradigm than for HO (*i.e.*, using singleton sub-hyperedges). ***Note that*** $CoSH$ ***scores (***$\psi_F(\cdot)$***) are free parameters that are arbitrarily different from their peers***. This means they are "independent" of each other, which makes them ***additive***, and hence ***cumulative*** $CoSH$ scores ($CCoSH$) of sub-hyperedges present in a soft-partition can be used as a measure of "information captured" by a sub-hyperedges selection heuristic (or alternatively, its "extent of coverage"). Hyperedges are first binned according to their ***mean sub-hyperedge size*** defined as

$$MSS(F) := \sum_{F' \in \texttt{T2C2}(F)} \frac{|F'|}{|\texttt{T2C2}(F)|}. \tag{7.16}$$

Next, the ***information coverage*** for **T2C2** is calculated for each hyperedge as

$$CCoSH_{\texttt{T2C2}}(F) := \sum_{F' \in \texttt{T2C2}(F)} \psi_F(F'). \tag{7.17}$$

For comparison with the HO-equivalent scores, "singleton" sub-hyperedges are used and information coverage for the **SING**-mode is calculated as

$$CCoSH_{\texttt{SING}}(F) := \sum_{u \in F} \psi_F(\{u\}). \tag{7.18}$$

Fig. 7.6 shows the mean information coverage scores (*i.e.* $CCoSH_{\texttt{T2C2}}$ and $CCoSH_{\texttt{SING}}$) for different bins of hyperedges. The information coverage for **T2C2** is significantly higher as compared to the singleton-based sub-hyperedges, *esp.* for the hyperedges having higher MSS values. While for hyperedges having lower MSS values, there is not much difference in the information coverage between the two heuristics. This shows that the SHO-paradigm can be useful for all hyperedges while singleton nodes based one (*i.e.*, the HO-paradigm) fails to capture necessary information for a lot of hyperedges (*i.e.*. those with high MSS values). Hence, a model using **T2C2** has a better hyperedge prediction performance as it is able to obtain more information through the SHO-domain. Not-to-mention, this is also evident from Table 7.2, since SHONeN has significantly better hyperedge prediction performances as compared to other baselines.

7.6.3 Embeddings

Each of node2vec, HyperSAGNN, and SHONeN provide latent embeddings for nodes in a hypergraph network. Since the t-SNE plots did not reveal much, we perform a "node pair similarity" analysis by computing the cosine similarities between embeddings of two nodes, for each algorithm. Few interesting pairs have been tabulated in Table 7.3. The analysis was performed for the "tags" dataset (`tags-math-sx`) since nodes tagged with mathematical concepts are pragmatically interpretable. We first obtain pairwise similarity through the node embeddings generated by the three models SHONeN (S), HyperSAGNN (HS), and Node2Vec (N2V) in the hyperedge prediction experiments. Next, we compare how these pairwise similarity changes across different models.

Table 7.3: Pairwise similarity between node embeddings for SHONeN, HyperSAGNN and Node2Vec.

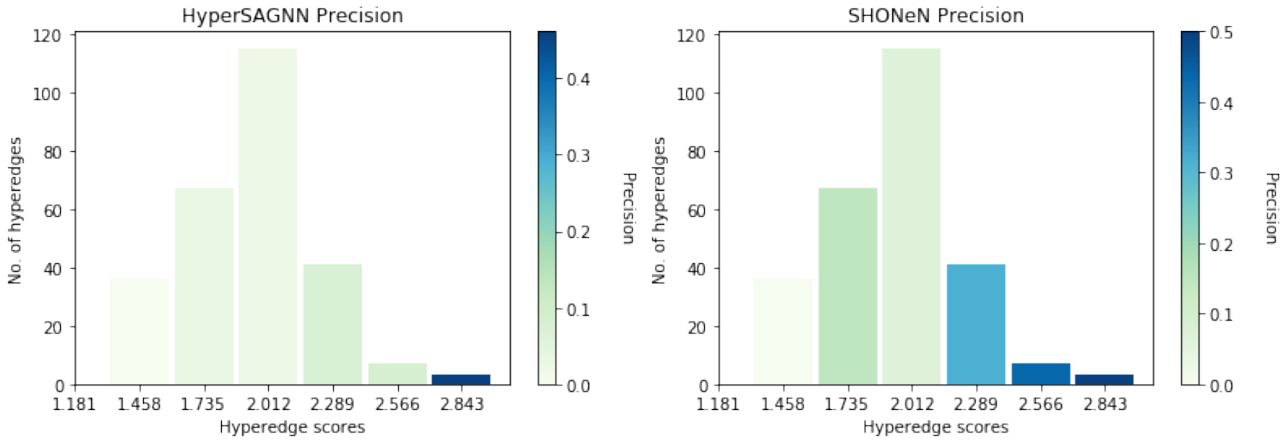
Node Pair	SHONeN	HyperSAGNN	Node2Vec
1: (*elementary-set-theory*, *simple-groups*)	0.91	-0.38	-0.05
2: (*tiling*, *tessellations*)	0.94	0.46	0.17
3: (*matrix-equations*, *uvw*)	0.81	-0.28	0.20
4: (*graph-theory*, *topological-graph-theory*)	0.83	-0.01	0.71
5: (*random-graphs*, *topological-graph-theory*)	0.86	0.02	0.73
6: (*borel-cantelli-lemmas*, *supremum-and-infimum*)	0.90	0.11	0.66
7: (*bayesian*, *estimation-theory*)	0.87	0.87	0.09
8: (*brownian-motion*, *stochastic-analysis*)	0.96	0.96	-0.03
9: (*logic*, *provability*)	0.95	0.95	0.15

We restrict our comparison to pairs with high similarity because most of the pairs with low similarity scores are not interesting and do not make sense to be put together. Table 7.3 clearly shows that SHONeN is superior compared to other baseline algorithms in that it improves the pairwise similarity of pairs which are likely to co-occur as tags in a thread.

It should be noted that both SHONeN and HyperSAGNN use node2vec embeddings as initial node embeddings which makes it interesting to see how these two models change the similarity of pairs *w.r.t* node2vec. SHONeN improves the pairwise similarity even if a pair has a lower similarity initially for node2vec (*e.g.*, pairs 1, 2, 3, 7, 8, and 9), whereas HyperSAGNN either fails to improve the similarity (*e.g.*, pair 3) or ends up reducing it even further (for pairs 1, 3, 4, 5, 6), despite the same pair having high similarity for node2vec. At the same time, for some pairs (*e.g.*, pairs 7, 8, 9), the similarity of SHONeN and HyperSAGNN are higher as compared to node2vec, meaning both models have improved over initial node2vec embeddings. The performance improvement of SHONeNs for hyperedge prediction could be attributed to the improvements of SHONeN in pairwise similarities over baseline models.

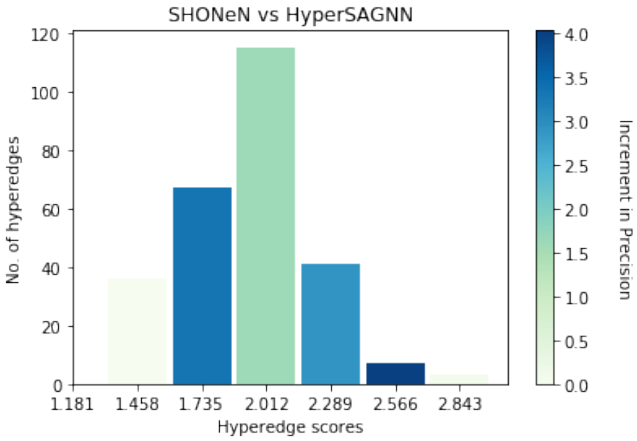
7.7 Conclusion

Hypergraphs help in representing higher-order relations, but modeling the evolution/formation of hyperedges themselves is a complex task. We set out to show that the usual model of information flow in a hypergraph – *i.e.*, from nodes to hyperedges – which we also call the higher-order (HO) paradigm has its limitations, and provide theoretical arguments for the same. For one, the rank of the incidence matrix $\boldsymbol{S}$ (which is bounded by $|V|$) is not high enough for it to even barely span a $|\mathcal{F}|$-dimensional vector space $\mathbb{R}^{|\mathcal{F}|}$, where V and $\mathcal{F}$ represent the set of vertices and hyperedges of a given hypergraph. As a solution to this, we formulated the notion of a sub-higher-order (SHO) paradigm and communicated the idea via a powerset-incidence matrix $\boldsymbol{P}$ that involves *sub-hyperedges*, which has an exponentially better rank than $\boldsymbol{S}$. The theory seems to fall in line with the experiment results, which show significant improvements over AUC scores for hyperedge prediction using SHONeNs (a neural network approach motivated by the SHO-structure). That the heuristic we provide in order to tame computational complexity works in practice is a huge advantage in itself, since we are able to show that we could perform a task worth exponential time-order successfully in polynomial time.



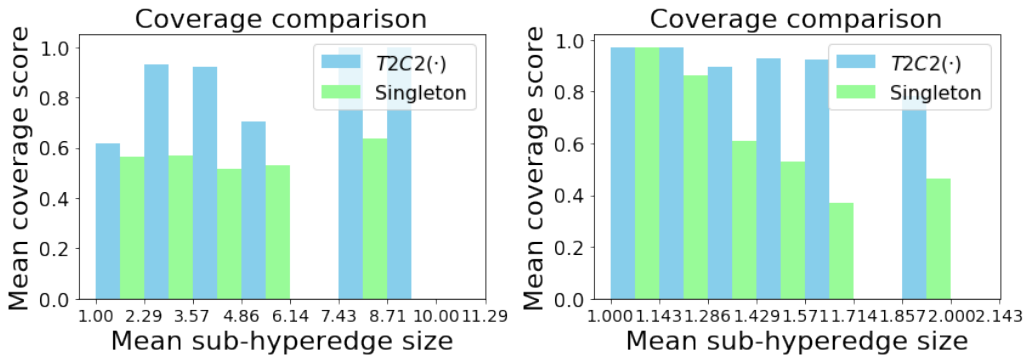
(a) HyperSAGNN Precision

(b) SHONeN Precision



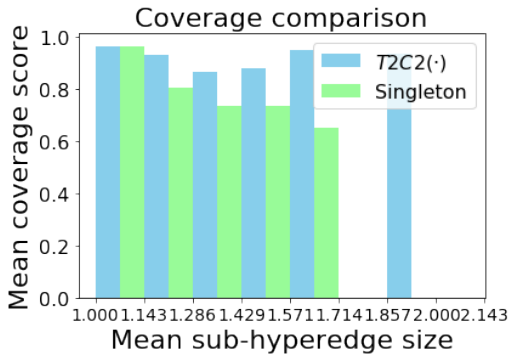
(c) Increment in Precision

Figure 7.5: Precision comparison between SHONeN and HyperSAGNN for the `email-Enron` dataset. Darker color denotes higher precision or an increment therein. The X-axis shows size-rank scores $SR(F)$ for a hyperedge F, which is higher for highly-ranked non-singleton sub-hyperedges, and lower for highly-ranked singleton sub-hyperedges in F. We see a higher precision (a dark blue bar) in the right side of the plots as compared to the left; this shows that the SHO-impact was mostly seen in hyperedges with a higher SR value.

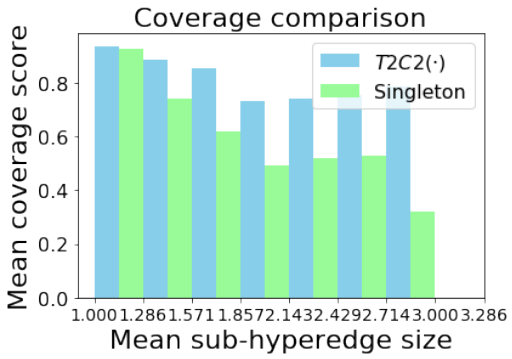


(a) NDC-substances

(b) contact-primary-school



(c) contact-high-school



(d) MAG-Geo

Figure 7.6: Information captured by **T2C2** vs. that by singleton node based partition. The height of each bar (Y-axis) corresponds to mean $CCoSH(F)$ scores, *blue* bars represent scores achieved using **T2C2** partitions ($CCoSH_{\texttt{T2C2}}(F)$) whereas *green* bars represent the same for singleton nodes based partitions ($CCoSH_{\texttt{SING}}(F)$). Hyperedges are binned according to $MSS(F)$, the mean sub-group size of **T2C2**(F), which is denoted on the X-axis.

Chapter 8

Leave Left nor Right: Predicting Bipartite Hyperedges

"You want to know how two chemicals interact, do you ask them? No, they're going to lie through their lying little chemical teeth. Throw them in a beaker and apply heat."

∼ *"Dr. Gregory House"*[1]

RELATIONS of a higher order could be captured using hypergraphs; but those *also* between *two different types of entities* (which we term "left" and "right") – calls for a "bipartite hypergraph". For example, given a left set of symptoms and right set of diseases, the relation between a subset of symptoms (that a patient experiences at a given point of time) and a subset of diseases (that he/she might be diagnosed with) could be well-represented using a bipartite hyperedge. The state-of-the-art in embedding nodes of a hypergraph is based on learning the self-attention structure between node-pairs from a hyperedge. In this chapter, given a bipartite hypergraph, we aim at capturing relations between node pairs from the cross-product between the left and right hyperedges, and term it a "cross-attention" (CAT) based model. More precisely, we pose "bipartite hyperedge link prediction" as a set-matching (SETMAT) problem and propose a novel neural network architecture called CATSETMAT for the same. We perform extensive experiments on multiple bipartite hypergraph datasets to show the superior performance of CATSETMAT, which we compare with multiple techniques from the state-of-the-art. Our results also elucidate information flow in self- and cross-attention scenarios.

8.1 Introduction

Relations between two entities are easily captured by a *graph* [77, 79], wherein a collection of pairwise *edges* (either directed or undirected) encapsulates the relational structure (*e.g.*, friendship relations

[1] Quoted from the series *House M. D., Season 1, Episode 13: "Cursed"* [98].

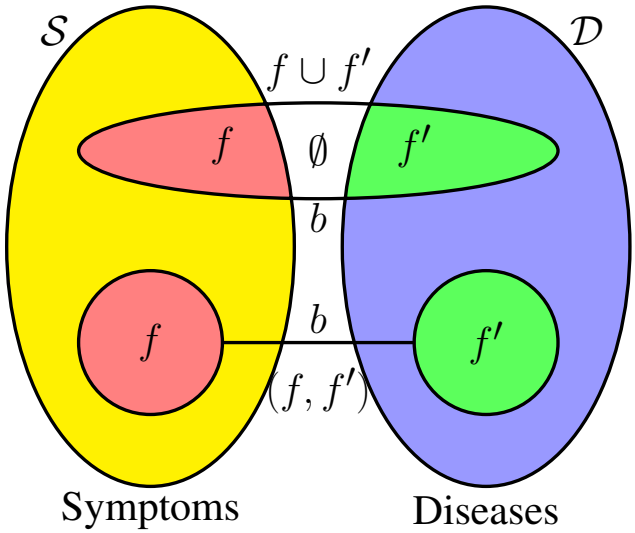


Figure 8.1: Example of a *symptoms-diseases* bipartite hypergraph with left and right node sets being those of symptoms $\mathcal{S}$ and diseases $\mathcal{D}$. Higher-order bipartite relations could be interpreted as a single bipartite hyperedge (above) $b = F \cup F' \in 2^{\mathcal{S} \cup \mathcal{D}}$ intersecting the node sets at F and F'. Such a relation can also be viewed (below) as a set match $(F, F') \in 2^{\mathcal{S}} \times 2^{\mathcal{D}}$ which comes as a result of a set matching between symptoms and diseases.

between two people on a social network [105, 2]). Moreover, *heterogeneous graphs* [101, 115] are used to capture relationship structures between entities of multiple "types" (*e.g.*, a bibliographic network [26] between nodes of type author, paper, venue, *etc.*). However, when the number of types is restricted to two (say, "left" and "right"), and relations exist only *across* (and not *among*) them, we resort to a *bipartite graph* [56] (*e.g.*, an author-paper bibliography network).

Nevertheless, any such relation captured by a usual network – be it homogeneous, heterogeneous, or bipartite – is strictly restricted to a *pair* of entities. For higher-order relations we use hypergraphs, which akin to graphs, have their own heterogeneous versions (those that capture higher-order relations between nodes of different types), which have been used in the literature to capture relations of the type buyer-broker-seller [17], user-location [125], *etc*. While much has been done about such *heterogeneous hypergraphs* in the literature [39, 137], *bipartite hypergraphs* [140] – hypergraphs wherein each hyperedge is required to have at least one node from each one of two disjoint node-sets "left" and "right" – have *seldom* been talked about.

An example would be the *symptoms-diseases* bipartite hypergraph shown in Fig. 8.1, where given a (left) set $\mathcal{S}$ of symptoms and another (right) set $\mathcal{D}$ of diseases, every *non-trivial diagnosis* — one wherein the doctor identifies at least one symptom (*i.e.*, $F \in 2^{\mathcal{S}} \setminus \emptyset$) in a patient and diagnoses him/her with at least one disease (*i.e.*, $F' \in 2^{\mathcal{D}} \setminus \emptyset$) — forms a bipartite hyperedge $F \cup F'$, and a collection $\mathcal{B}$ of such hyperedges forms a bipartite hypergraph $\mathcal{H} = (\mathcal{S} \cup \mathcal{D}, \mathcal{B})$. For the diagnosis F denotes all the

its symptoms and F' denotes all diseases that the patient is suffering from.

Existing models use a self-attention mechanism (Hyper-SAGNN) to predict a heterogeneous hyperedge, but miss the fact that the flow of information has to be across the right and left sets, and not among them individually. In this chapter, we too aim to learn a neural network model for predicting relations, only in a bipartite hypergraph. Going by the symptoms and diseases example in Fig. 8.1, we need not model the existence of a set of symptoms or a set of diseases, but the existence of a relation between a set of symptoms and a set of diseases.

We repose hyperedge prediction as a *set-matching* (SETMAT) problem: *given two set-of-sets, what pair of sets "match" with each other*? In our case, the two sets would be the left and right hyperedges, and we call it a "match" if they are linked by a bipartite hyperedge.

8.1.1 Key Contributions

1. This is the very first work on bipartite hypergraphs in machine learning, along with introduction of some novel **datasets**. Moreover, we introduce the problem of **predicting higher-order bipartite relations** in networks for the first time.

2. Elucidate the drawback of usual hyperedge embedding techniques for bipartite hyperedges via a **alternating positive/negative set pairs** based explanation.

3. Pose the above problem as a **set-matching** prediction problem and show theoretical equivalence of the same.

4. Formulate a **cross-attention framework** based neural network architecture to deal with the set-matching and hence the bipartite hyperedge prediction problem.

8.2 Bipartite Hyperedge Prediction

8.2.1 Bipartite Hypergraphs

Given are two sets of disjoint nodes: **left nodes** V and **right nodes** V'; a **simple bipartite hypergraph** is defined as $\mathcal{H}_{sim} = (V \cup V', \mathcal{B})$, where $\mathcal{B} \subseteq 2^{V \cup V'}$ such that there is at least one node from each node set in the elements of $\mathcal{B}$. Continuing from Chapter 2 Section 2.6, we define a ***simple bipartite hypergraph***, we now define a different kind of bipartite hypergraph: a ***per-fixed***[1] one wherein basically we *fix* the set of "left and right hyperedges" (defined in the next sentence) beforehand. Over the node sets V and V', the set $\mathbb{F} := 2^V \setminus \emptyset$ of ***potential left hyperedges*** and the set $\mathbb{F}' := 2^{V'} \setminus \emptyset$ of ***potential***

[1] We call it "per-fixed" and not pre-fixed, since only left and right hyperedges are fixed.

right hyperedges could be noted. Once we *fix* the set of *actual* ***left hyperedges*** to be $\mathcal{F} \subseteq \mathbb{F}$ and *actual* ***right hyperedges*** to be $\mathcal{F}' \subseteq \mathbb{F}'$, we could define the set of ***potential bipartite hyperedges*** as:

$$\mathbb{B}(\mathcal{F}, \mathcal{F}') := \{F \cup F' \mid F \in \mathcal{F}, F' \in \mathcal{F}'\}. \tag{8.1}$$

Definition 8.1 (PER-FIXED BIPARTITE HYPERGRAPH). *A* ***per-fixed bipartite hypergraph*** *is an ordered set* $\mathcal{H} = (V \cup V', \mathcal{F} \cup \mathcal{F}', \mathcal{B})$ *of left vertices* V*, right vertices* V'*, fixed left hyperedges* $\mathcal{F}$*, fixed right hyperedges* $\mathcal{F}'$*, and* ***bipartite hyperedges*** $\mathcal{B} \subseteq \mathbb{B}(\mathcal{F}, \mathcal{F}')$*. Furthermore,* $\hat{\mathcal{B}} := \mathbb{B}(\mathcal{F}, \mathcal{F}') \setminus \mathcal{B}$ *denotes the set of all* ***bipartite non-hyperedges****.*

A simple bipartite hypergraph is different from a usual (non-bipartite) hypergraph in that it has two disjoint sets of nodes (V, V') instead of one. On the other hand, a per-fixed bipartite hypergraph is different from the two, since we fix the sets of left and right hyperedges $(\mathcal{F}, \mathcal{F}')$ beforehand, and are thence worried only about connections across them (as defined by $\mathcal{B}$). An important consequence of these facts is that ***we need not model the existence of the left or right hyperedges individually, but focus on the cross bipartite relations instead***.

Observation 8.1. *Given a per-fixed bipartite hypergraph* $\mathcal{H} = (V \cup V', \mathcal{F} \cup \mathcal{F}', \mathcal{B})$*, the triplet* $\mathcal{G} := (\mathcal{F} \cup \mathcal{F}', \mathcal{B})$ *forms a bipartite graph over node sets* $\mathcal{F}$ *and* $\mathcal{F}'$*. Also, the triplet* $\mathcal{H}_{sim} := (V \cup V', \mathcal{B})$ *forms a simple bipartite hypergraph.*

Note: Henceforth, unless prefixed with the term "simple", **the phrase "bipartite hypergraph" would refer to a per-fixed bipartite hypergraph** as per Definition 8.1.

8.2.2 The Bipartite Hyperedge Prediction Problem

In this chapter, we aim to solve the problem of bipartite hyperedge prediction (BHP), which we define as follows:

Definition 8.2 (Bipartite Hyperedge Prediction (BHP)). *Given* $\mathcal{H} = (V \cup V', \mathcal{F} \cup \mathcal{F}', \mathcal{B})$*, learn a* ***BHP predictor*** $\varphi : \mathbb{B}(\mathcal{F}, \mathcal{F}') \to \mathbb{R}$ *such that for* $F, \hat{F} \in \mathcal{F}$ *and* $F', \hat{F}' \in \mathcal{F}'$ *and disjoint sets* $\mathcal{P} \subseteq \mathcal{B}$ *and* $\mathcal{N} \subseteq \hat{\mathcal{B}}$ *(see Def. 8.1) denoting the* ***positive*** *and the* ***negative class*** *respectively, we have:*

$$\max_{\varphi \in \mathbb{R}^{\mathbb{B}(\mathcal{F}, \mathcal{F}')}} Pr(\varphi(F \cup F') > \varphi(\hat{F} \cup \hat{F}') \mid F \cup F' \in \mathcal{P} \text{ and } \hat{F} \cup \hat{F}' \in \mathcal{N}) \tag{8.2}$$

It is to be noted that Definition 8.2 considers a per-fixed bipartite hypergraph and one could also define the BHP problem for a simple bipartite hypergraph (let's call it simple-BHP) as well. The only difference between BHP and simple-BHP would be that while for the former, the set of possible

left- and right-hyperedges is fixed even before the problem is defined, for the latter, any possible left- or right-hyperedge could be chosen as arbitrary subset choices from the respective node sets. Nevertheless, the difference shows up only for bipartite non-hyperedges: for simple-BHP, we have $\mathcal{B} \subseteq \mathbb{B}(\mathbb{F}, \mathbb{F}')$ and $\hat{\mathcal{B}} := \mathbb{B}(\mathbb{F}, \mathbb{F}') \setminus \mathcal{B}$, but for per-fixed BHP (as per Definition 8.2, the set of observed left hyperedges and right hyperedges are fixed to $\mathcal{F} \subseteq \mathbb{F}$ and $\mathcal{F}' \subseteq \mathbb{F}'$ respectively, and then hyperedges and non-hypergedges are defined as $\mathcal{B} \subseteq \mathbb{B}(\mathcal{F}, \mathcal{F}')$ and $\hat{\mathcal{B}} := \mathbb{B}(\mathcal{F}, \mathcal{F}') \setminus \mathcal{B}$. ***We will see how existing hyperege prediction algorithms apply only to simple-BHP and fail to cater to the per-fixed BHP problem***.

8.3 Bipartite Hyperedge Set Matching Prediction (BHSMP)

8.3.1 Set Matching (SETMAT)

Definition 8.3 (**Set Matching**). *Given two sets-of-sets $\mathcal{X}$ and $\mathcal{Y}$, a **set matching** is defined as a relation $\mathcal{M} \subseteq \mathcal{X} \times \mathcal{Y}$ that matches a set element $X \in \mathcal{X}$ to another set element $Y \in \mathcal{Y}$. Every $(X, Y) \in \mathcal{M}$ is called a **set match** (or **match** for short). Naturally, the set $\hat{\mathcal{M}} := \mathcal{X} \times \mathcal{Y} \setminus \mathcal{M}$ refers to the corresponding **set anti-matching** and an element $(\hat{X}, \hat{Y}) \in \hat{\mathcal{M}}$ is called a **set anti-match** (or **anti-match** or non-match for short).*

Definition 8.4 (**Set Matching Predicton (SMP)**). *Given two sets-of-sets $\mathcal{X}$ and $\mathcal{Y}$, and a set matching $\mathcal{M} \subseteq \mathcal{X} \times \mathcal{Y}$, the set matching prediction problem learns an **SMP predictor** $\mu : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ such that for $X, \hat{X} \in \mathcal{X}$ and $Y, \hat{Y} \in \mathcal{Y}$, we have:*

$$(X, Y) \in \mathcal{M} \text{ and } (\hat{X}, \hat{Y}) \in \hat{\mathcal{M}} \implies \mu(X, Y) \geq \mu(\hat{X}, \hat{Y}). \tag{8.3}$$

Lemma 8.1. *Every bipartite hypergraph is a set-matching over its left and right hyperedges.*

Proof. Given a bipartite hypergraph $\mathcal{H} = (V \cup V', \mathcal{F} \cup \mathcal{F}', \mathcal{B})$, consider $\mathcal{F}$ and $\mathcal{F}'$ as two sets-of-sets. Now consider the following claim:

> **Claim**: *Sets $\mathbb{B}(\mathcal{F}, \mathcal{F}')$ (see Definition 1) and $\mathcal{F} \times \mathcal{F}'$ are equivalent.*
> *Proof of Claim.* The map $\sigma : \mathbb{B}(\mathcal{F}, \mathcal{F}') \to \mathcal{F} \times \mathcal{F}'$ defined by $x \mapsto (x \cap V, x \cap V')$ is bijective since its inverse σ^{-1} is defined by $(x, y) \mapsto x \cup y$. Hence, the claim. □

Using the same bijective map in the *Proof of Claim* above, we have:

$$\forall b \in \mathcal{B},\ \exists F := b \cap V \in \mathcal{F} \text{ and } \exists F' := b \cap V' \in \mathcal{F}'$$

Hence the relation:

$$\mathcal{M}(\mathcal{H}) := \{(b \cap V, b \cap V') \mid b \in \mathcal{B}\} \subseteq \mathcal{F} \times \mathcal{F}'$$

is a set matching over sets-of-sets $\mathcal{F}$ and $\mathcal{F}'$ as per Definition 3. □

8.3.2 BHP as SMP

Lemma 8.2. *The BHP problem for a bipartite hypergraph can be solved by an SMP predictor for its equivalent set-matching.*

Proof. Given $\mathcal{H} = (V \cup V', \mathcal{F} \cup \mathcal{F}', \mathcal{B})$, fix $\mathcal{P} \subseteq \mathcal{B}$ and $\mathcal{N} \subseteq \hat{\mathcal{B}}$ as positive and negative classes respectively. Consider hyperedges $F, \hat{F} \in \mathcal{F}$, $F', \hat{F}' \in \mathcal{F}'$ be such that $F \cup F' \in \mathcal{P}$ and $\hat{F} \cup \hat{F}' \in \mathcal{N}$.

Now, consider $\mathcal{H}$'s equivalent matching $\mathcal{M}(\mathcal{H})$ and learn an SMP predictor $\mu : \mathcal{F} \times \mathcal{F}' \to \mathbb{R}$. Then, we have $(F, F') \in \mathcal{M}(\mathcal{H})$ and $(\hat{F}, \hat{F}') \in \hat{\mathcal{M}}(\mathcal{H})$. Hence, from eq. (3) in Definition 4, we have $\mu(F, F') \geq \mu(\hat{F}, \hat{F}')$. Now, if we define a derived BHP predictor $\varphi : \mathbb{B}(\mathcal{F}, \mathcal{F}') \to \mathbb{R}$ as $b \mapsto \mu(b \cap V, b \cap V')$, we get:

$$\begin{aligned} \varphi(F \cup F') &= \mu((F \cup F') \cap V, (F \cup F') \cap V') = \mu(F, F') \\ &\geq \mu(\hat{F}, \hat{F}') = \mu((\hat{F} \cup \hat{F}') \cap V, (\hat{F} \cup \hat{F}') \cap V') = \varphi(\hat{F} \cup \hat{F}') \\ \implies \varphi(F \cup F') &\geq \varphi(\hat{F} \cup \hat{F}'). \end{aligned}$$

Hence, since F, F', $\hat{F}$, $\hat{F}'$ were arbitrarily selected as per $\mathcal{P}$ and $\mathcal{N}$, the foregoing arguments make φ satisfy the BHP condition given in eq. (2) from Definition 2. □

The foregoing lemma states an important result: ***that we could solve the BHP problem using SMP***. Since we have an equivalent set matching for a given hypergraph, and that we could solve the BHCP problem using SMP, let us define a new problem called the ***Bipartite Hyperedge Set Matching Prediction (BHSMP)*** problem as follows:

Definition 8.5 (***Bipartite Hyperedge Set Matching Prediction (BHSMP)***)**.** *Given a bipartite hypergraph $\mathcal{H} = (V \cup V', \mathcal{F} \cup \mathcal{F}', \mathcal{B})$, a* ***BHSMP predictor*** *is simply defined as an SMP predictor for the equivalent set matching.*

8.4 The Cross Attention (CAT) Framework

8.4.1 The Problem with usual Self-Attention

Currently, the best known model to predict hyperedges is that of Hyper-SAGNN [133], which uses (1) a self-attention framework to model information flow between nodes of a hyperedge, and (2) a static-vs-dynamic comparative technique to learn hyperedge formation. But the main reason why it does not deem fit for the bipartite hyperedge scenario is that it captures information flow between **all**

the nodes of a usual (non-bipartite) hyperedge. And for a per-fixed bipartite hypergraph, the difference is subtler: "pay heed to the cross-connections, not the self-connections." In other words, one need not model the occurrence of individual left and right hyperedges, and instead focus on the connection *between* them. That is, we need not *predict* left and right hyperedges individually, but only predict the connections between them, since the former is already fixed to be $\mathcal{F}$ and $\mathcal{F}'$. For the symptoms-diseases case, the reason why Hyper-SAGNN does not apply to the BHP prediction problem is that we are interested in modeling/predicting new diagnoses involving observed symptom-sets and disease-sets. Paying simultaneous attention to symptoms, diseases and their connections (diagnoses) leads to a learning process that oscillates between the positive and negative classes. More about this would be discussed in Section 8.9.4.

8.4.2 Usual Hyper-SAGNN based Self-Attention on Bipartite Hypergraphs

Let us see what effect applying Hyper-SAGNN has on a bipartite hyperedge. Consider a bipartite hypergraph $\mathcal{H} = (V \cup V', \mathcal{F} \cup \mathcal{F}', \mathcal{B})$. Now, let us take two sets $F := \{u_1, u_2, \ldots, u_k\} \subseteq V$ and $F' := \{u'_1, u'_2, \ldots, u'_{k'}\} \subseteq V'$ that are "observed" to be hyperedges. Also given are the embeddings of each node: $\boldsymbol{x}_i, \boldsymbol{x}'_{i'} \in \mathbb{R}^d$ for nodes u_i and $u'_{i'}$ respectively ($1 \leq i \leq k$, $1 \leq i' \leq k'$). If $\boldsymbol{W}_Q, \boldsymbol{W}_K \in \mathbb{R}^{d \times d_K}$, and $\boldsymbol{W}_V \in \mathbb{R}^{d \times d_V}$ denote weight matrices for the self-attention framework of Hyper-SAGNN, we have for the potential $(k + k')$-sized hyperedge $F \cup F' = \{u_1, u_2, \ldots, u_k, u'_1, u'_2, \ldots, u'_{k'}\}$,

$$a_{ij} = (\boldsymbol{W}_Q^T \boldsymbol{x}_i)^T (\boldsymbol{W}_K^T \boldsymbol{x}_j), \text{ and } a'_{i'j'} = (\boldsymbol{W}_Q^T \boldsymbol{x}'_{i'})^T (\boldsymbol{W}_K^T \boldsymbol{x}'_{j'}), \qquad (V \text{ to } V \text{ and } V' \text{ to } V') \tag{8.4}$$

$$b_{ij'} = (\boldsymbol{W}_Q^T \boldsymbol{x}_i)^T (\boldsymbol{W}_K^T \boldsymbol{x}'_{j'}), \text{ and } c_{i'j} = (\boldsymbol{W}_Q^T \boldsymbol{x}'_{i'})^T (\boldsymbol{W}_K^T \boldsymbol{x}_j) \qquad (V \text{ to } V' \text{ and } V' \text{ to } V) \tag{8.5}$$

where $1 \leq i, j \leq k$ and $1 \leq i', j' \leq k'$. These values are then normalized using the softmax function as follows:

$$\alpha_{ij} = \frac{\exp(a_{ij})}{z_i}, \; \alpha'_{i'j'} = \frac{\exp(a'_{i'j'})}{z'_{i'}}, \; \beta_{ij'} = \frac{\exp(b_{ij'})}{z_i}, \; \gamma_{i'j} = \frac{\exp(c_{i'j})}{z'_{i'}}, \tag{8.6}$$

where $z_i := \left(\sum_{t=1}^{k} \exp(a_{it}) + \sum_{t=1}^{k'} \exp(b_{it}) \right)$ and $z'_{i'} := \left(\sum_{t=1}^{k'} \exp(a'_{i't}) + \sum_{t=1}^{k} \exp(c_{i't}) \right)$.

The dynamic embeddings of the hyperedge nodes would then be:

$$\boldsymbol{d}_i := \tanh \left(\sum_{l=1}^{k} \alpha_{it} \boldsymbol{W}_V^T \boldsymbol{x}_t + \sum_{l=1}^{k'} \alpha'_{it} \boldsymbol{W}_V^T \boldsymbol{x}'_t \right), \tag{8.7}$$

$$\boldsymbol{d}'_{i'} := \tanh \left(\sum_{t=1}^{k} \beta_{i't} \boldsymbol{W}_V^T \boldsymbol{x}_t + \sum_{t=1}^{k'} \beta'_{i't} \boldsymbol{W}_V^T \boldsymbol{x}'_t \right) \tag{8.8}$$

8.4.3 Cross-Attention for Bipartite Hypergraphs

Going by the cross-attention paradigm, we introduce three extra attention parameters, amounting to parameters $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}'_Q, \boldsymbol{W}'_K \in \mathbb{R}^{d\times d_K}$ and $\boldsymbol{W}_V, \boldsymbol{W}'_V \in \mathbb{R}^{d\times d_V}$. We redefine b, c, β, and γ for $1 \le i \le k$ and $1 \le i' \le k'$ as follows:

$$b_{ii'} = (\boldsymbol{W}_Q^T \boldsymbol{x}_i)^T (\boldsymbol{W}_K'^T \boldsymbol{x}'_{i'}), \text{ and } \beta_{ii'} = \exp(b_{ii'}) \Big/ \sum_{t=1}^{k'} \exp(b_{it}) \,, \tag{8.9}$$

$$c_{i'i} = (\boldsymbol{W}_Q'^T \boldsymbol{x}'_{i'})^T (\boldsymbol{W}_K^T \boldsymbol{x}_i), \text{ and } \gamma_{i'i} = \exp(c_{i'i}) \Big/ \sum_{t=1}^{k} \exp(c_{i't}) \,. \tag{8.10}$$

Now, the cross-attention dynamic embedding of the left and right hyperedge nodes would then be:

$$\boldsymbol{\delta}_i := \tanh\left(\sum_{t=1}^{k'} \beta_{it} \boldsymbol{W}_V'^T \boldsymbol{x}'_t\right), \boldsymbol{\delta}'_{i'} := \tanh\left(\sum_{t=1}^{k} \gamma_{i't} \boldsymbol{W}_V^T \boldsymbol{x}_t\right). \tag{8.11}$$

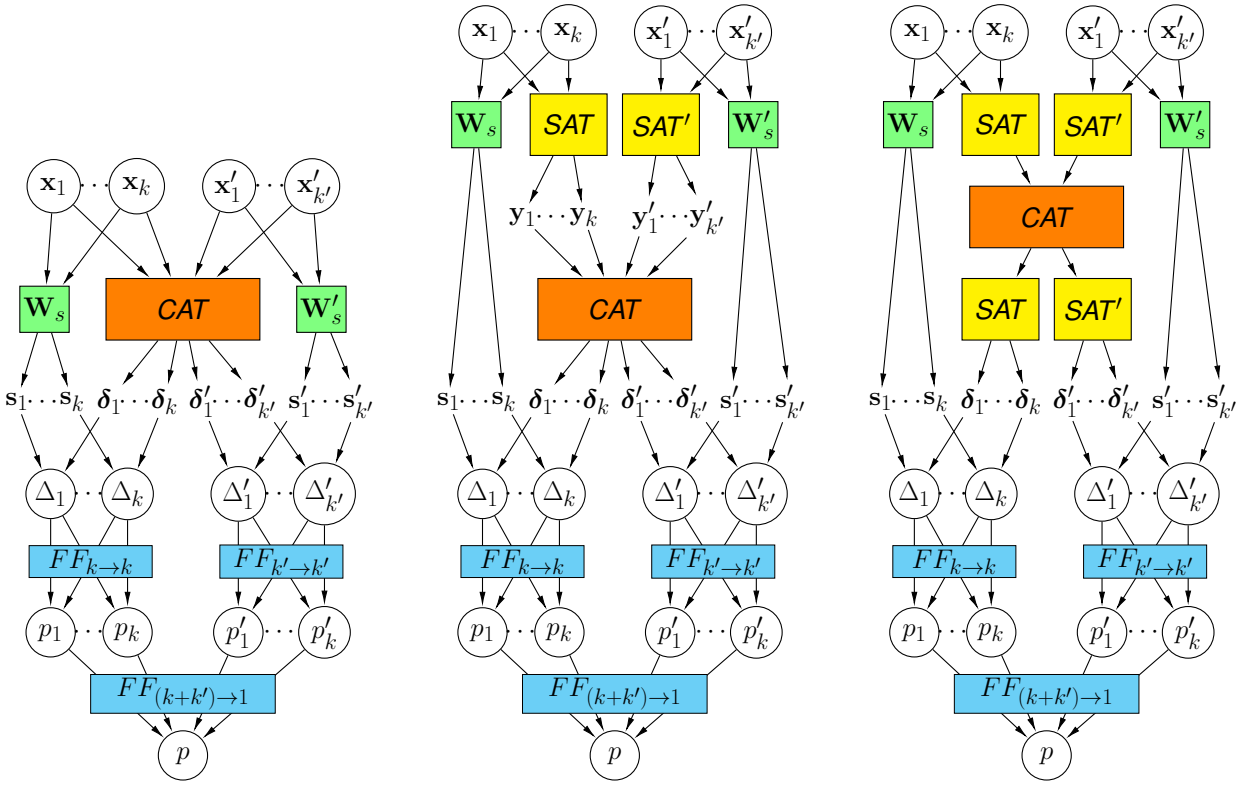
Finally, we have new embeddings $\boldsymbol{\delta}_1, \boldsymbol{\delta}_2, \ldots, \boldsymbol{\delta}_k$ of left nodes and $\boldsymbol{\delta}'_1, \boldsymbol{\delta}'_2, \ldots, \boldsymbol{\delta}'_{k'}$ of right nodes, all d_V-dimensional vectors.

8.5 The *CATSETMAT* Architecture

Akin to Hyper-SAGNN, we define a neural network architecture called *CATSETMAT* (**C**ross **At**tention for **Set Mat**ching) that has both self- as well as cross-attention layers. *CATSETMAT* uses structures defined in Section 8.4.3 that are derived from the self-attention layers in Hyper-SAGNN. A detailed network architecture is depicted in Figure 8.2. And as explained before, we use the architecture to solve the set matching prediction (SMP) problem which in turn solves the bipartite hyperedge prediction (BHP) problem – a problem we had termed bipartite hyperedge set matching prediction (BHSMP).

Basically, it commences with the vector representations of two sets of hyperedges – one left ($\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k\}$), and the other right ($\{\boldsymbol{x}'_1, \boldsymbol{x}'_2, \ldots, \boldsymbol{x}'_{k'}\}$). These inputs are fed into two Hyper-SAGNN-like self-attention (SAT) blocks SAT and SAT' separately, which have their respective parameter sets $\boldsymbol{W}_Q^{(\mathit{SAT})}, \boldsymbol{W}_K^{(\mathit{SAT})}, \boldsymbol{W}_V^{(\mathit{SAT})}$ and $\boldsymbol{W}_Q^{(\mathit{SAT}')}, \boldsymbol{W}_K^{(\mathit{SAT}')}, \boldsymbol{W}_V^{(\mathit{SAT}')}$). The SAT-blocks give out revised embeddings $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_k$ and $\boldsymbol{y}'_1, \boldsymbol{y}'_2, \ldots, \boldsymbol{y}'_{k'}$. Although this applies one round of attention to each of the left and right hyperedges, no information about the cross-connections between them have been learned by our model.

Here is where the cross-attention (CAT) block – CAT – comes into picture, not-to-mention, with its own set of parameters $\boldsymbol{W}_Q^{(\mathit{CAT})}$, $\boldsymbol{W}_K^{(\mathit{CAT})}$, $\boldsymbol{W}_V^{(\mathit{CAT})}$, $\boldsymbol{W}_Q'^{(\mathit{CAT})}$, $\boldsymbol{W}_K'^{(\mathit{CAT})}$, $\boldsymbol{W}_V'^{(\mathit{CAT})}$. The CAT block takes the revised embeddings through equations (8.9)–(8.11) and returns fresh embedding vectors

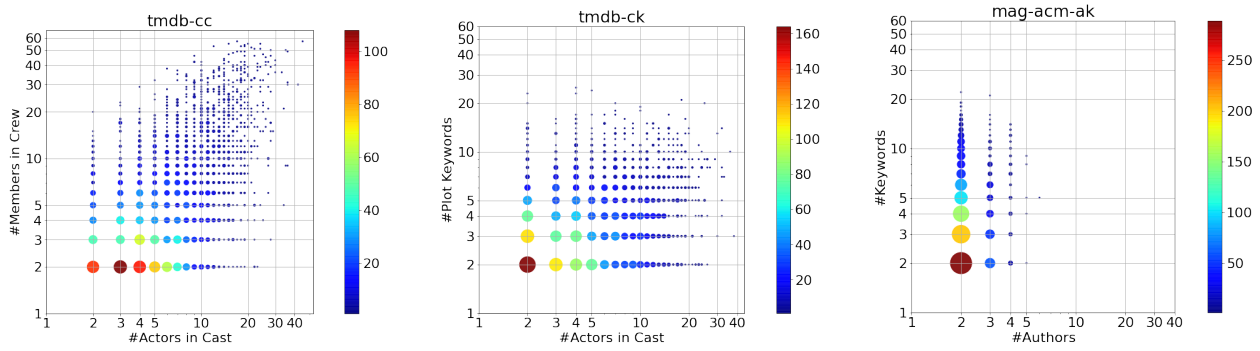


(a) CATSETMAT-X (b) CATSETMAT-SX (c) CATSETMAT-SXS

Figure 8.2: Neural network architectures of the three variants of the CATSETMAT algorithm that we propose. (a) Only one cross-attention layer (X). (b) An additional pair of self-attention layers (S) before cross-attention (SX). (c) Another additional pair of self-attention layers after cross-attention (SXS).

Table 8.1: The list of bipartite hypergraph datasets introduced in this thesis, along with their vital statistics: # left nodes $|V|$, # right nodes $|V'|$, # left hyperedges $|\mathcal{F}|$, # right hyperedges $|\mathcal{F}'|$, and # bipartite hyperedges $|\mathcal{B}|$.

Dataset	Left nodes ($\mathcal{V}$)	Right nodes ($\mathcal{V}'$)	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{V}'\rvert$	$\lvert\mathcal{B}\rvert$	$\lvert\mathcal{F}\rvert$	$\lvert\mathcal{F}'\rvert$
`tmdb-cc`	Cast (actors)	Crew (other members)	4,556	3,802	2,825	2,824	2,744
`tmdb-ck`	Cast (actors)	Plot keywords	3,156	1,256	2,669	2,656	2,621
`mag-acm-ak`	Authors	Keywords	1,059	2,338	1,388	847	1,379



(a) `tmdb-cc` (b) `tmdb-ck` (c) `mag-acm-ak`

Figure 8.3: Size distributions of the left and right hyperedges in the bipartite hypergraph for the three datasets. On X-axis is the size of the left hyperedges, and on Y-axis, that of right hyperedges.

$\boldsymbol{\delta}_1, \boldsymbol{\delta}_2, \ldots, \boldsymbol{\delta}_k$ and $\boldsymbol{\delta}'_1, \boldsymbol{\delta}'_2, \ldots, \boldsymbol{\delta}'_{k'}$ – vectors we call *dynamic embeddings*, just as Zhang, et al. [133] do for Hyper-SAGNN. In parallel is a layer of static-weights $\boldsymbol{W}_s, \boldsymbol{W}'_s \in \mathbb{R}^{d \times d_s}$ that simply transforms the original left vectors $\boldsymbol{x}_i$ and right vectors $\boldsymbol{x}'_{i'}$ into *static embeddings* $\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_k$ and $\boldsymbol{s}'_1, \boldsymbol{s}'_2, \ldots, \boldsymbol{s}'_{k'}$ defined by $\boldsymbol{s}_i := \tanh\left(\boldsymbol{W}_s^T \boldsymbol{x}_i\right)$ and $\boldsymbol{s}'_{i'} := \tanh\left(\boldsymbol{W}_s'^T \boldsymbol{x}'_{i'}\right)$ respectively, where $1 \leq i \leq k$ and $1 \leq i' \leq k'$. Following Zhang et al. [133], we compare the static and dynamic embeddings using the Hadamard square operator, resulting in vectors $\Delta_1, \Delta_2, \ldots, \Delta_k$ and $\Delta'_1, \Delta'_2, \ldots, \Delta'_{k'}$ defined as $\Delta_i := (\boldsymbol{\delta}_i - \boldsymbol{s}_i)^{\circ 2}$ and $\Delta'_{i'} := (\boldsymbol{\delta}'_{i'} - \boldsymbol{s}'_{i'})^{\circ 2}$. Finally, two positional feed-forward layers [133] $FF_{k \to k}$ and $FF_{k' \to k'}$ followed by a consolidating layer $FF_{(k+k' \to 1)}$ computes the probability p that the set of nodes match or not.

We use altogether three variants of `CATSETMAT`: `CATSETMAT-X` (only *CAT*), `CATSETMAT-SX` (a pair of *SAT* followed by a *CAT*), and `CATSETMAT-SXS` (`CATSETMAT-SX` with another pair of *SAT* modules following *CAT*). Architecture diagrams for all the three variants have been shown in Figure 8.2.

8.6 Bipartite Hypergraph Datasets

Refer Table 8.1 for details about datasets.

8.6.1 Data Description

All datasets we have prepared are available on the following link:

https://www.dropbox.com/s/cxykmi37695jlcw/catsetmat_data.zip?dl=0

Let us discuss the datasets one by one.

8.6.1.1 TMDB Cast-Crew (tmdb-cc)

We take a subset of movies from *The Movie Database* (*TMDB*; https://www.themoviedb.org/) available as a *Kaggle* (https://www.kaggle.com/) dataset named *TMDB 5000 Movie Dataset* (https://www.kaggle.com/tmdb/tmdb-movie-metadata). The node sets considered here are those of *movie-actors* and the *movie-crew* (set of other important people involved in a movie, such as directors, producers, *etc.*). Every bipartite hyperedge contains a set of actors (the *cast*; the left nodes) and a set of *crew*-members (the right nodes) who participated in at least one movie.

8.6.1.2 TMDb Cast-Keywords (tmdb-ck)

TMDB also gives information about *plot keywords* that hint at the content of a movie. A bipartite hyperedge would contain a set of actors (the *cast*; the left nodes) and a set of *keywords* (the right nodes) related to least one movie.

8.6.1.3 MAG ACM Authors-Keywords (mag-acm-ak)

This is a bibliographic network from *Microsoft Academic Graph* (*MAG*) [99], from which we choose those publications that appeared in conferences and journals associated with *Association for Computing Machinery* (*ACM*; https://www.acm.org/). The two node sets used are those of *authors* (the left node set) and (research-based) *keywords* (the right node set), and each bipartite hyperedge contains a set of co-authors and a set of keywords at least one of their papers has been tagged with.
Note: The raw data for MAG is not available directly and it's a long process to collect portions of it. However, there used to be an online source that had official links to download the whole MAG dataset. We have an earlier version of the data downloaded with us, and this processing was made on that dataset. Nevertheless, the prepared data is available via the Dropbox link given earlier.

8.6.2 Data Preparation

8.6.2.1 Processing Data

We start with a raw dataset consisting of multiple entities, of which we choose two. Then we fetch bipartite hyperedges, which are merely events occurring at a given point of time (e.g., a movie getting released, or a paper getting published, etc.); this gives us sets of higher-order co-occurrences of the two entities we chose. Next, we apply a couple of filters to refine the data:

1. **Time filter**: We filter the data into events occurring at time $t \in [t_{min}, t_{max}]$.
2. **Occurrence frequency filter**: We only keep those nodes in the node-sets V and V' that have their occurrence count within a particular range $[o_{min}, o_{max}]$.
3. **Size filter**: We only keep those bipartite hyperedges $F \cup F'$ that have left and right hyperedge sizes (individually) in the range $|F|, |F'| \in [s_{min}, s_{max}]$.

Please note that the filters are applied sequentially, and the prepared data is stored as a two-column (left hyperedge, right hyperedge) CSV file.

8.6.2.2 Negative Sampling

For the BHP problem, each bipartite hyperedge $F \cup F' \in \mathcal{B}$ is considered to be a positive example ($\in \mathcal{P}$), and each non-hyperedge $F \cup F' \in \hat{\mathcal{B}} := \mathbb{B}(\mathcal{F}, \mathcal{F}') \setminus \mathcal{B}$, a negative class sample. But since the negative patterns are too many, we avoid the problem of class imbalance by *negative class sampling*. For this, we first pick two random hyperedges: one from left, $F \in \mathcal{F}$ and the other from right, $F' \in \mathcal{F}'$, and tag the pair as a negative class pattern ($\in \mathcal{N}$) if they aren't connected via a bipartite hyperedge (*i.e.*, if $F \cup F' \notin \mathcal{B}$). *We use a negative-to-positive ratio of* 5:1.

8.6.2.3 Train-Test Split

We split the bipartite hyperedges and sampled bipartite non-hyperedges (sampled as mentioned in Section 8.6.2.2 above) into train and test data using a 80:20 train-test split ratio. For `mag-acm-ak`, we create validation data as well, using 60:20:20 as train-validation-test proportions, and perform hyperparameter tuning on the validation data, as described in Section 8.8.1.5. For each dataset-algorithm combination, we perform *five* experiments, performing a separate split and a separate negative sampling for each. This also introduces a standard deviation in our results, which we have reported with the results.

8.7 Related Work

As far as usual hypergraphs are concerned, they have not been studied as much as graphs (some earlier works are are [135, 64, 17]). Of some recent approaches [11, 107, 132, 7, 31, 46, 51, 82, 123, 122, 8, 133] for hyperedge prediction and hypergraph embedding, the most recent one, *viz.*, Hyper-SAGNN [133] performs the best. It uses a self-attention based model wherein each hyperedge is handled separately. Our approach is strongly based on theirs, and the whole concept of cross-attention (and a combination thereof with self-attention) extends their framework to set matching as well.

It is to be noted that left and right hyperedges in a bipartite hypergraph are basically sets of nodes; hence machine learning techniques to handle sets become relevant here. Deep set embedding calls

for permutation-invariant neural networks, and there has been a considerable amount of work on this topic [111, 128, 112, 62, 134, 73, 118], of which we use FSPool [134], a sort-pooling based technique, as one of our baselines. Moreover, deep set-to-set matching has been performed on image data [88], but it does not apply to our problem since it uses a single universal set, as opposed to two disjoint ones in the case of a bipartite hypergraph.

Bipartite networks are so essential that Guillaume et al. [40] have argued for an underlying bipartite structure in *all* networks. Of late, neural network techniques for usual graphs have become widely recognized [38, 55, 42, 110] (ref. Wu et al. [120] for a survey). But the same is not true for deep bipartite networks. The *CATSETMAT* architecture we propose inherits attention from GAT [110] and weakly relates to BGNN [43]. ***Furthermore, to the best of our knowledge, there is no work that explores a bipartite hypergraph from the perspective of network science***, and all of them [140, 5, 44] belong to the domain of discrete mathematics.

8.8 Experiments

We perform bipartite hyperedge prediction experiments on some real-world datasets. Apart from the baselines described in Section 8.8.1, we use three versions of `CATSETMAT`: `CATSETMAT-X` (only *CAT*), `CATSETMAT-SX` (a pair of *SAT* followed by a *CAT*), and `CATSETMAT-SXS` (`CATSETMAT-SX` with another pair of *SAT* modules following *CAT*), which have been depicted in Figure 8.2.

8.8.1 Baselines

As discussed in Section 8.7, we prepare four classes of baselines: node2vec-based, bipartite-graph-based, set-embedding-based and hyperedge-prediction-based, a brief and detailed explanation of each of them have been given below.

8.8.1.1 Brief Overview

1. **node2vec-based**:
   `n2v-cross-mean`, `n2v-full-mean`, `n2v-cross-min`, `n2v-full-min` (We describe these baselines in detail in the supplementary material).

2. **Bipartite-graph-based**: Since our hypergraph could be interpreted as a bipartite "graph" (Observation 8.1), we prepare some baselines on bipartite versions of two popular link prediction algorithms [66]: *Common Neighbor (CN)* [79] and *Adamic Adar (AA)* [2]

3. **Set-embedding-based** (**FSPool**): For this, we first convert the node embeddings obtained from node2vec [38] (with dimension 16, since it showed best results) of each of the left and right hyperedges into set-embeddings using FSPool [134], and then learn a simple classifier.

4. **Hyperedge-prediction-based** (**Hyper-SAGNN**): We use the existing best performing algorithm Hyper-SAGNN [133] for hyperedge prediction, wherein we interpret our bipartite hypergraph as a usual one as per Observation 8.1. We use the same experiment settings as Zhang et al. [133].

8.8.1.2 Detailed Description: Node2vec-based

For these set of baselines, we convert the bipartite hypergraph into a usual graph by expanding each bipartite hyperedge $b = F \cup F' \in \mathcal{B}$ into a set of three types of edges:

- *Cross-edges* $\mathcal{E}_\times(b) := F \times F'$
- *Left self-edges* $\mathcal{E}_\circ(b) := \{e \subseteq F : |e| = 2\}$
- *Right self-edges* $\mathcal{E}'_\circ(b) := \{e \subseteq F' : |e| = 2\}$

Next, we embed each node v into its node2vec [38] features $X(v)$ and find cosine similarity between the incident nodes of each edge. Finally, we either take a mean or a min of that score and call it a prediction score. We use the same settings used in Hyper-SAGNN [133] for their node2vec based baselines.

1. `n2v-cross-mean`: For node2vec-cross-mean, we take $(v, v') \in \mathcal{E}_\times(b)$ and define the score to be:
$$n2v_{cross,mean}(b) := \frac{1}{|\mathcal{E}_\times(b)|} \sum_{(v,v')\in\mathcal{E}_\times(b)} X(v)^T X(v'). \tag{8.12}$$
2. `n2v-full-mean`:
$$n2v_{full,mean}(b) := \frac{1}{|\mathcal{E}(b)|} \sum_{(u,v)\in\mathcal{E}(b)} X(u)^T X(v), \tag{8.13}$$
where $\mathcal{E}(b) := \mathcal{E}_\times(b) \cup \mathcal{E}_\circ(b) \cup \mathcal{E}'_\circ(b)$ denotes the full set of edges.
3. `n2v-cross-min`:
$$n2v_{cross,min}(b) := \min_{(v,v')\in\mathcal{E}_\times(b)} X(v)^T X(v'). \tag{8.14}$$
4. `n2v-full-min`:
$$n2v_{full,min}(b) := \min_{(u,v)\in\mathcal{E}(b)} X(u)^T X(v). \tag{8.15}$$

8.8.1.3 Detailed Description: Bipartite-graph-based

To create these baselines, we first convert our bipartite hypergraph $\mathcal{H} = (V \cup V', \mathcal{F} \cup \mathcal{F}', \mathcal{B})$ into an induced bipartite graph $\mathcal{G} = (V \cup V', \mathcal{E})$ by defining $\mathcal{E}$ as:

$$\mathcal{E} := \{(v, v') \in V \times V' \mid \exists b \in \mathcal{B} \text{ s.t. } \{v, v'\} \subseteq b\},$$

and then compute the bipartite versions of the Common Neighbor (CN) and the Adamic Adar (AA) scores as follows:

$$CN(v, v') = \frac{LCN(v, v') + RCN(v, v')}{2}, \text{ and } AA(v, v') = \frac{LAA(v, v') + RAA(v, v')}{2},$$

where "L" and "R" stand for left and right scores respectively. If Γ denotes the neighbors of a node (or union of neighbors of a set of nodes), we can define the individual scores as (note that left nodes have neighbors in the right and vice-versa):

$$LCN'(v, v') = |(\Gamma(\Gamma(v)) \setminus \{v\}) \cap (\Gamma(v') \setminus \{v\})|$$

$$RCN(v, v') = |(\Gamma(\Gamma(v')) \setminus \{v'\}) \cap (\Gamma(v) \setminus \{v'\})|$$

$$LAA'(v, v') = \sum_{w \in (\Gamma(\Gamma(v)) \setminus \{v\}) \cap (\Gamma(v') \setminus \{v\})} \frac{1}{\log(1 + |\Gamma(w)|)}$$

$$RAA(v, v') = \sum_{w' \in (\Gamma(\Gamma(v')) \setminus \{v'\}) \cap (\Gamma(v) \setminus \{v'\})} \frac{1}{\log(1 + |\Gamma(w'|)}$$

We thus have CN and AA scores for each left-right vertex pair. Using this, we find the similarity scores for a left-right hyperedge pair (F, F') as:

$$\textsc{Algo}(F, F') = \textsc{Aggregate}(\{\textsc{Algo}(v, v') \mid (v, v') \in F \times F'\}), \text{ and}$$

where we use three choices for the AGGREGATE function and two for ALGO, thereby forming the six baselines:

1. `bipartite-AA-min`: AGGREGATE=MINIMUM; ALGO=AA
2. `bipartite-CN-min`: AGGREGATE=MINIMUM; ALGO=CN
3. `bipartite-AA-max`: AGGREGATE=MAXIMUM; ALGO=AA
4. `bipartite-CN-max`: AGGREGATE=MAXIMUM; ALGO=CN
5. `bipartite-AA-avg`: AGGREGATE=AVERAGE; ALGO=AA
6. `bipartite-CN-avg`: AGGREGATE=AVERAGE; ALGO=CN

8.8.1.4 Hyperparameter Settings

There are no parameters for CN and AA scores for the bipartite graph based baselines. For the node2vec-based baselines, we use embedding dimension as 64, and other node2vec parameters are

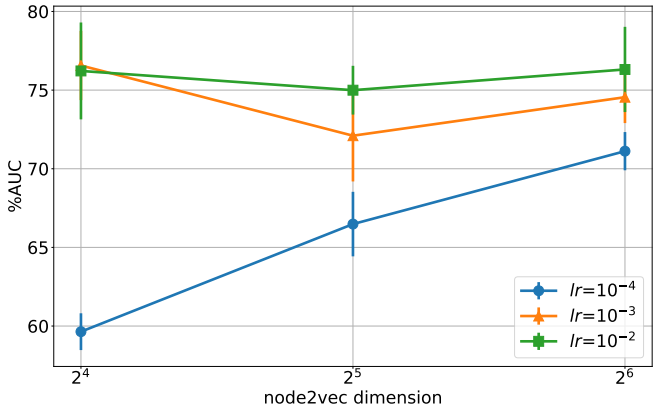


Figure 8.4: Tuning two hyperparameters on the dataset `mag-acm-ak` for `CATSETMAT`: node2vec dimension dim and learning rate lr.

taken from Hyper-SAGNN. For FSPool and HyperSAGNN, we take the latent dimension as 16 and other parameters are as per Hyper-SAGNN. We tried this with other dimensions as well, but the best results were on this configuration.

8.8.1.5 Hyperparameter Tuning for `CATSETMAT`

We fix some hyperparameters as per Hyper-SAGNN, and for the embedding dimension and the learning rate, we perform a hyperparameter tuning, as shown in Figure 8.4, and then pick them to be 16 and 0.001 respectively. We use the Adam optimization algorithm for our tasks. The initial embeddings that we use in our model are from node2vec (an alternating random walk on nodes and hyperedges respectively), just as Hyper-SAGNN does.

8.9 Results and Discussion

The results for hyperedge prediction on a few datasets have been listed in Table 8.2, where each baseline class (see Section 8.8.1) has been separated by a horizontal line. For each algorithm, and each dataset, we report the %AUC test scores for the SMBHP problem (see Definition 8.5). In each case, we see that our algorithm *CATSETMAT* performs the best unanimously. More observations are described in the following sections (Sections 8.9.2–8.9.3). An explanation of the poor performance of some state-of-the-art algorithms has been provided in Section 8.9.4.

8.9.1 Running times

While the bipartite algorithms (first six) took within seconds (10-20 seconds) to complete, the node2vec ones (next four) ran for around 5-10 mins per dataset (the node-to-vec embeddings were pre-stored

Table 8.2: Results (%AUC) for the bipartite hyperedge prediction problem

Algorithm	tmdb-cc	tmdb-ck	mag-acm-ak
bipartite-AA-min	43.9987 ± 0.9787	37.6217 ± 1.2686	49.9388 ± 1.1913
bipartite-CN-min	43.8015 ± 0.9613	37.5078 ± 1.2589	49.8958 ± 1.1743
bipartite-AA-max	53.0463 ± 0.4581	49.4547 ± 0.5887	63.5835 ± 0.5314
bipartite-CN-max	52.5955 ± 0.4971	49.3627 ± 0.5697	62.0365 ± 0.6099
bipartite-AA-avg	54.6746 ± 0.9042	46.9037 ± 0.7630	$\mathit{64.5778 \pm 1.6425}$
bipartite-CN-avg	54.0634 ± 0.9388	46.7163 ± 0.7498	63.3550 ± 1.7340
n2v-cross-mean	77.4467 ± 1.1838	$\mathit{63.5867 \pm 0.4989}$	59.2567 ± 2.0266
n2v-full-mean	$\mathit{77.6833 \pm 1.5083}$	62.8900 ± 0.1606	59.0933 ± 2.2257
n2v-cross-min	52.4433 ± 1.5320	51.9333 ± 1.2777	32.8867 ± 1.1585
n2v-full-min	53.9433 ± 1.0404	52.8267 ± 1.1521	31.7933 ± 1.6376
FSPool	63.8305 ± 0.8315	50.2596 ± 1.3653	63.4810 ± 1.6438
Hyper-SAGNN	55.6071 ± 2.9784	47.0969 ± 1.3497	63.0752 ± 13.0197
CATSETMAT-X	83.2761 ± 2.2315	74.6650 ± 2.1525	67.6457 ± 3.0129
CATSETMAT-SX	86.5461 ± 3.3937	$\mathbf{84.0697 \pm 0.9814}$	$\mathbf{76.6221 \pm 0.9676}$
CATSETMAT-SXS	$\mathbf{88.2177 \pm 0.8268}$	81.9037 ± 2.3348	75.6302 ± 3.0833

and used, so experiment not conducted multiple times). On the other hand, FSPool took 10 mins, 10 mins, and 7 mins respectively for the three datasets per experiment. Hyper-SAGNN takes a total of 12-15 mins for the first two datasets and around 10 mins for the mag-acm-ak (again, per experiment). Finally, our algorithms took around 25-30 mins per experiment for the first two datasets, and around 15 mins for the last one. For a total of five experiments, each of our algorithms took a total of ~ 375 minutes (~ 6.3 hours) to run.

8.9.2 Performance of Baselines

Although bipartite link prediction algorithms (bipartite-AA-min–bipartite-CN-avg) perform poorly with AUCs ranging from 43–55% and 37–50% for the first two datasets, for mag-acm-ak, some of them perform considerably better with AUCs going up to 64.5% as well. These algorithms heavily depend on the interconnectivity between the left and right hyperedges, and it seems to be the best in the mag-acm-ak dataset. The next set of algorithms – the four node2vec-based ones – perform much better than the foregoing group, but still lag behind *CATSETMAT* by huge margins. Surprisingly, the bipartite graph based algorithms outperform the node2vec ones for the last dataset. This only shows that performances vary dramatically from dataset to dataset.

The next two algorithms – FSPool and Hyper-SAGNN, though being cutting-edge deep learning approaches in their respective fields (*viz.*, set and hyperedge embeddings respectively), fail to capture the

desired bipartite hypergraph structure. The main reason for this, as has been explained in Section 8.9.4, is that they struggle to learn a consistent representation of nodes since the same set of nodes get involved in bipartite hyperedges (positive class) as well as bipartite non-hyperedges (negative class) simultaneously. Moreover, `Hyper-SAGNN` also shows quite a high amount of deviation (13%), which is undesirable of any machine learning algorithm. But this only shows the huge dependence of `Hyper-SAGNN` on the data preparation process (*i.e.*, negative-sampling and train-test split) that we perform five times. More about this has been discussed in Section 8.9.4.

8.9.3 Performance of CATSETMAT

Finally, the last three algorithms – the ones we have proposed in this chapter – are the best performing ones among the lot. For `tmdb-cc`, we see that our best method is 13.56% better than the group-wise next best (*i.e.*, `n2v-full-mean`), which has an AUC of only 77.68%. Another of our methods (`CATSETMAT-SXS`) performs the best for `tmdb-ck`, with a much higher improvement of 33.06% as compared to `n2v-cross-mean`, the group-wise next best. For both these datasets, we obtain quite high AUC values of around 84–88%, but the same is not true for `mag-acm-ak`, for which the best performance is limited to 76.62%, albeit given by our algorithm `CATSETMAT-SX`. Although this is a much lower score as compared to the other two datasets, but it is 18.65% higher than `bipartite-AA-avg`, which is the best performing baseline. Barring our algorithms `CATSETMAT-X`, `CATSETMAT-SX`, and `CATSETMAT-SXS`, no other algorithm consistently performs well on all the three datasets. While at least one dataset deemed problematic for each of the baselines, these three algorithms remained consistent for all datasets. Moreover, no baseline touched the 80% AUC mark except the `CATSETMAT`-based algorithms. We credit this success to the cross-attention paradigm, which focuses on the cross-links more than on the left and right hyperedges, thereby avoiding the positive-negative dilemma as explained in Section 8.9.4.

Although the *CAT* module (see Figure 8.2) is our main contribution in this chapter, adding more *SAT* layers is expected to model the individual left- and right-hyperedges as well, since it might capture some inner domain-specific relational structure in each of them. We can clearly see this effect being illustrated in Table 8.2, wherein adding only one pair of *SAT* modules before *CAT* (in `CATSETMAT-SX`) boosts the performances of a purely *CAT*-based algorithm `CATSETMAT-X` by upto 3.92%, 12.59%, and 13.27% for the three datasets respectively. While the addition of another pair of *SAT* modules (`CATSETMAT-SXS`) shows the potential of improving the AUC performance by an additional 1.93% for `tmdb-cc`, but same is not true for the `tmdb-ck` and `mag-acm-ak`, where the performance instead drops by 2.57% and 1.31% respectively. This effect could be attributed to over-fitting due to an increase in model parameters. As a result of this, we conclude that addition of more number of *SAT* modules would not further boost the performance, and so, it should be limited to a single pair of *SAT*

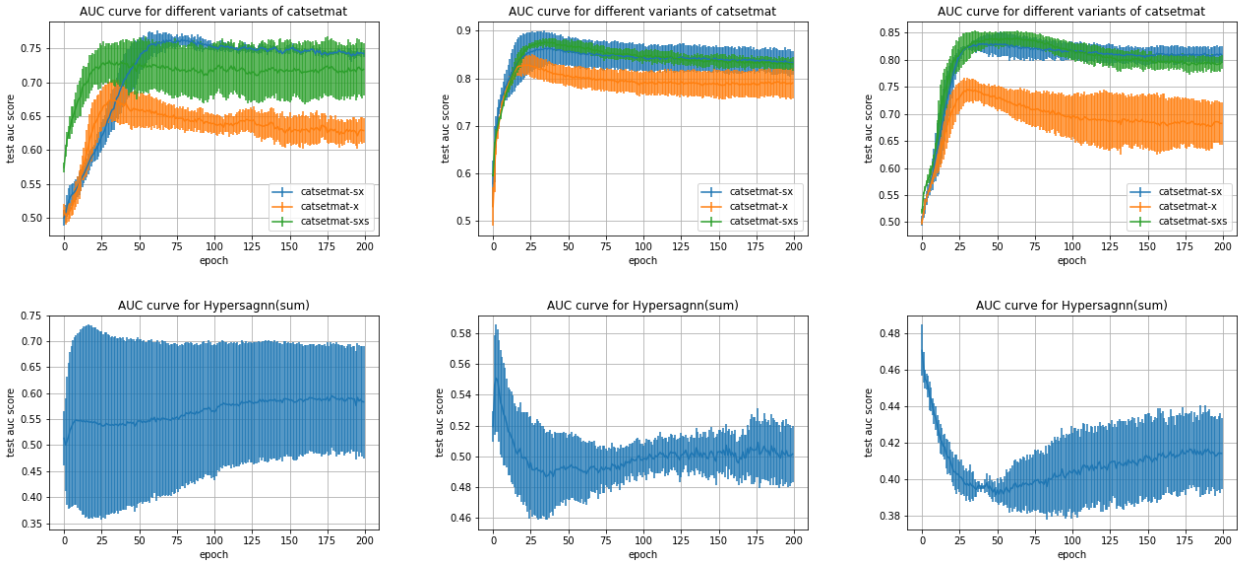


Figure 8.5: Epoch-by-epoch AUC performance curves (mean and standard-deviation values)

modules before *CAT*, as done in `CATSETMAT-SX`. The same could be seen from a t-test performed at $5 + 5 - 2 = 8$ degrees of freedom. When `CATSETMAT-X` is compared with `CATSETMAT-SX`, we get p values of 14.6%, 0.005%, and 0.05% respectively for datasets `tmdb-cc`, `tmdb-ck`, and `mag-acm-ak`, which shows a significant improvement when one extra self-attention layer is added. While on the other hand, we get p values of 36.7%, 12.6%, and 55.6% respectively when `CATSETMAT-SX` and `CATSETMAT-SXS` are compared, showing that this rise/fall[1] is not so significant.

8.9.4 The Positive-Negative Dilemma

It is clear how our algorithm works well, and how the baselines fail to capture the bipartiteness for hyperlink prediction. Table 8.5 shows learning curves for two algorithms on the three datasets. We could see that unlike *CATSETMAT*, the other HyperSAGNN does not seem to converge, and keeps oscillating around 0.5 AUC.

We reason for this behavior as follows. A careful look at the definition of a per-fixed bipartite hypergraph (Definition 8.1) would reveal a straightforward yet important fact: ***that left and right sets of hyperedges are fixed in advance***. What makes this factor interesting is another definition: that of bipartite non-hyperedges (again, Definition 8.1) or the negative class, which ensures that ***each negative sample would be formed from the same fixed left and right hyperedges that formed samples from the positive class***.

[1]The first dataset shows a rise, and the rest two show a fall; nevertheless they are all statistically insignificant.

To better denote these concepts, we know that $\mathcal{F}$ and $\mathcal{F}'$ represent the sets of left and right hyperedges, which are fixed even before defining bipartite relations $\mathcal{B}$ (the positive class), let alone defining bipartite non-relations $\hat{\mathcal{B}}$ (the negative class). That is, a typical positive sample looks like $F \cup F'$, where the choice of F and F' is not arbitrary, since they belong to the fixed sets $\mathcal{F}$ and $\mathcal{F}'$ respectively. What is striking to observe is that, a negative sample would also look like $\hat{F} \cup \hat{F}'$, where, again, $\hat{F} \in \mathcal{F}$ and $\hat{F}' \in \mathcal{F}'$. In other words, there is nothing that stops a left hyperedge F that involves in forming a positive sample from getting involved in forming a negative sample as well. The same could be said for a right hyperedge F' also. As a result, we observe that the positive and the negative samples are built from mostly (if not entirely) the same left and right hyperedges.

When an algorithm that treats the entire hyperedge $F \cup F'$ as a whole is deployed to embed it, it keeps getting confused, as to how to train the parameters to account for both classes simultaneously. Thus, it does not settle for a stable output for both F as well as F', owing to F and F''s simultaneous association with both the classes. This is what algorithms like `Hyper-SAGNN` and `FSPool` suffer from, when it comes to a per-fixed bipartite hypergraph. If we, however, change our negative sampling method to a sized-random [81] technique, we essentially change the per-fixed hypergraph to a simple hypergraph.

Surprisingly enough, the result becomes totally different! `Hyper-SAGNN` now reports a test AUC of a whopping 98% on `mag-acm-ak`, a dataset that was the toughest to handle as per Table 8.2. It is clear from the foregoing arguments that there exists a positive-negative dilemma in usual hyperedge embedding approaches as far as bipartite hypergraphs are concerned. The unusually higher variance of 13% observed in Table 8.2 for `Hyper-SAGNN` on `mag-acm-ak` could be easily attributed to this dilemma. There could be many more variants of negative sampling techniques for the bipartite hyperlink prediction problem. For now, we have only looked at the per-fixed version of the non-bipartite-hyperedges. More negative sampling methods would give different perspectives of bipartite hypergraphs and their evolution.

8.10 Conclusion

A bipartite hypergraph is a peculiar data structure almost never studied in network analysis. We formalize almost all notions of this structure with standard notations used for graphs and usual hypergraphs, and introduce the bipartite hyperedge prediction problem. However, since we were able to establish an equivalence between this problem and set-matching, we could propose a solution for the latter and use it for the former. We could successfully posit that focusing on cross-attention plays a significant role in capturing bipartite relations well. When evaluated using AUC scores, the performances of all versions of our algorithms were much higher than expected. An important insight that we were able establish was the existence of a positive-negative dilemma in existing cutting-edge

algorithms like Hyper-SAGNN designed for the very job of hyperedge prediction.

Chapter 9

Parts to Whole: Putting it all Together

"Vanilla Ice Cream $\neq$ *Cold* $+$ *Sweet* $+$ *Vanilla Aroma* $+$ *Softness* $+$ *Yellow."*

$\sim$ `Daniel Katz`

9.1 Summary

9.1.1 What did we cover?

We started with merely *touching* the field of hypergraphs – *i.e.*, by ***measuring and exploiting its effect on its clique-induced versions*** and problems therein. In particular, we were interested in the ***link prediction problem and a handful of seemingly strong heuristics*** therefor that perform very well in networks.

Then we moved closer, in that we ***studied the effect negative sampling has on hyperlink prediction***, and also aimed to provide some ***concrete negative sampling algorithms for better evaluation of hyperlink prediction techniques***. Moving further, we decided to ***tackle the hyperlink prediction problem by developing strong insights from real-world hypergraphs***, *esp.* from hypergraphs studied by Benson et al. [11]. Inspired by the success therein, we moved a huge step forward and focused on ***changing the information-flow model in hypergraphs and exploiting this to perform hyperlink prediction via a neural network***.

In parallel, we had been working on the idea of ***capturing the dynamics governing a bipartite hypergraph*** – a concept that we had only seen in a handful of works on discrete mathematics, and had potential in representing higher-order relations of a bipartite variety. More specifically, we decided to focus on a special kind of bipartite hypergraph – a ***"per-fixed"*** one – wherein the set of left and right hyperedges remains fixed beforehand. ***Predicting the formation of bipartite hyperlinks using a neural network model*** was the ultimate goal of this exercise.

9.1.2 What ideas did we test/explore?

While studying pairwise links under the effect of hypergraphs, we were interested in the ***role hyperedges play in the exceptionally high performance of link prediction heuristics***. Particularly, we had conjectured that ***since hyperedges induce cliques (or regions of high density) in the resulting graph, they would create a favourable environment for neighborhood based heuristics*** such as common-neighbors, Adamic Adar, etc. Also, since we expected a strong influence of higher-order relations in the structure of induced graphs, we ideated to ***utilize hypergraph-based node similarity scores for link prediction instead***, thereby taking our analysis to a conclusive end.

While exploring hyperlink prediction, we had seen how insignificant the negative sampling step in its literature was – contrary to that in the link prediction problem. We guessed that the ***impact of negative sampling would be high, since the performance of well-established hyperlink prediction algorithms heavily varied*** as we sampled the non-hyperlinks differently. Further, we thought of ***rigorously testing the idea that hyperedges in the future mainly form from dense regions in the past (the clique-closure hypothesis, CCH)***, and then using this to improve upon existing solutions to the hyperlink prediction problem outright. We also got interested in the idea that ***flow of information in a hypergraph*** should not be as straightforward as they seem to be – *i.e.*, from nodes to hyperedges and vice versa, and ***smaller sub-groups of nodes should also play explicit roles in the process***. We were interested in testing this idea ***using hyperlink prediction as a downstream task***.

As far as bipartite higher-order relations are concerned, we were *not* interested in a *simple* bipartite hypergraph, and focused on the per-fixed one instead. We could mentally ***link the problem of "predicting bipartite hyperlinks" to that of "set matching" in collections of sets***, but did not find models that solved the latter problem. So, we thought of ***improving upon the current state-of-the-art in hyperlink prediction – HyperSAGNN – and build a deep learning framework to handle set matching for sets-of-sets***, so as to ***use the same solution for bipartite hyperlink prediction*** as well.

9.1.3 What solutions/insights did we contribute towards?

We saw that ***link prediction on clique-induced graphs gets heavily affected by their underlying hypergraph structure***. Numerically speaking, we noted that all the five datasets had an ***AUC adjustment factor*** – a metric whose deviation from unity is proportional to the said effect – ***greater than one*** for all link prediction heuristics (except for Jaccard Coefficient, but only on one dataset); in some cases, we even observed this factor shoot up to 1.9. To correct for this bias, we ***suggested a readjustment to the AUC performance***. Moreover, in order to exploit the effect hyperedges have on graphs, we provided ***rigorous mathematical structures that could transmit a link prediction heuristic from graphs to hypergraphs, and observed that AUC performances increased significantly*** when hypergraph-based

node similarity heuristics were used along with graph-based ones.

On the negative-sampling end, we were able to successfully show that it is an extremely important step, since ***hyperlink prediction algorithms from the literature exhibited random relative performance on the same datasets as we went from one negative sampling approach to another***; in this process, we also established ***two benchmark algorithms for negative sampling: MNS and CNS***. As for our hypothesis that ***hyperedges in a hypergraph are formed from cliques and near-cliques (CCH), we had strong evidence in terms of our hypothesis test results, which clearly confirmed our conjecture***. ***Using it straightaway for hyperlink prediction gave significant improvements on the task***, and further reinforced the veracity of CCH. Further, that a sub-higher-order based paradigm would improve hyperlink prediction motivated us for sure, its computational intractability did create barriers. But we were successfully able to not only ***come up with a sub-optimal heuristic `T2C2` to capture sub-hyperedges more practically***, but also ***reason for the fact that the approach was better***. Moreover, the ***improvement of hyperlink prediction results using SHONeNs further established our sub-higher-order information flow model***.

For bipartite hypergraphs, we could ***successfully establish the notions of per-fixed bipartite hypergraphs, set-matching for set-of-sets, and the application of the latter for bipartite hyperlink prediction***. It was ***surprising to observe that cutting-edge techniques for hypergraphs and set-embedding such as HyperSAGNN and FSPool respectively failed miserably in capturing the notion of bipartite hyperedges***, and hence could not discriminate them against bipartite non-hyperedges. We could establish that this was due to the ***intermingling of the self-attention parameters***, which couldn't heed the fixed left and right hyperedges in the per-fixed bipartite hypergraph we were processing. A segregation of parameters by creating a different attention block for connections across left and right node-sets (*i.e.*, cross-connections) ***boosted the performance of the task at hand by huge factors***. We formulated a ***cross-attention based neural network architecture – CATSETMAT – that could handle such bipartite hyperlinks better***. ***Bipartite hypergraph datasets*** we have prepared and thus used for this task form a novel contribution in their own right.

9.2 A Roadmap to the Future

That "higher-order relations skew link prediction on graphs" has much bigger consequences than can be discussed within the scope of this thesis. Firstly, the broader question remains: "*Why do higher-order relations even affect link prediction*? *And how can one even correct it in general*"? There is no simple answer to these questions at this point. We believe that the fundamental reason why higher-order relations have this effect is that there is no unambiguous way to model higher-order relations in simple graphs. Any approach to convert hypergraphs into simple graphs either loses information or adds bias or both. We also hypothesize that the *effect of higher-order relations is not limited to link prediction*

and they affect several other problems pertaining to networks as well — these too constitute the main directions for future research. We would also like to *use the functional-formulation we established to convert graph-based similarity scores to hypergraphs for global, random-walk based measures such as Katz, PageRank, etc.* as well as for *link prediction on directed, signed, and heterogeneous networks*.

In extensions to hyperlink prediction, we would like to *extract more concepts from baselines such as Bayesian Sets and Spectral Hypergraph Clustering (baselines for C3MM) and incorporate them into C3MM*, to hopefully improve further. The sub-higher-order formulation we have proposed has only been used for hyperlink prediction in this thesis. It opens pathways to multiple different directions as well: *extending SHONeNs to generic sets (for, say set embeddings), to directed- and knowledge-hypergraphs, and for applications such as clustering and node classification*. The theoretical analyses we perform has dug-up promising insights, which could be used to more rigorously analyze the domain of hypergraphs and their evolution. Also, we would aim to *work with more variety of networks: heterogeneous hypergraphs, directed hypergraphs, and weighted hypergraphs, and on specific applications* as well.

The behavior of bipartite hypergraphs is next to unknown among machine learning researchers. We have hopefully shed some light on one aspect of their application on a problem having real-world implications — the bipartite hyperedge prediction problem. *But a lot of problems remain unanswered*. For instance, we still have a hazy picture of *why other baselines do not perform well* (although we have given reasoning for the same). Also, *a variety of unusual higher-order relations still remain to be modeled: e.g., a k-partite hypergraph*. Another huge area that needs attention is the *preparation and analysis of more and more bipartite hypergraph real-world datasets*. We leave these problems as future work.

9.3 Our Vision

Higher-order relations span across an ocean of concepts waiting to be picked up by network scientists. The ultimate struggle of every research – *truth-finding* – seems easier when seen from a toned-down version of the environment, *e.g.*, from the perspective of pairwise relations or graphs in the case of networks, but we cannot reject the reality that nature does not necessarily follow pairwiseness. This is exemplified by each chapter in this thesis. Nevertheless, pairwiseness is helpful for the extremely useful mathematical analyses (owing to advances in graph theory) and to address practical concerns such as computational complexity. But as researchers working on the intersection of science and engineering, we ought be foresighted. There was a time when data analysis was performed on kilobytes of data, and today, we are playing in terabytes; similar could be said about computing power. It should not be hard to see that the future holds more and more resources and better and better algorithms for optimal processing of data, to cater in-turn to the advancement of artificial intelligence.

9. PARTS TO WHOLE: PUTTING IT ALL TOGETHER

Summarizing the foregoing statements, we mean to posit that ***the sheer computational infeasibility of higher-order relations or hypergraphs should not hinder research therein, lest we miss the joy of uncovering truer models seldom paid heed to in the past***. For example, the field of Gestalt theory had in the far past been extensively utilized by researchers such as Marvin Minsky [75], Satosi Watanabe [116], *et al*. who were working on the foundational aspects of pattern recognition. They in their formulations have questioned the very pairwiseness that narrows the scope of pattern representation itself, and further extends its impact to the notions of similarity, clustering, classification, *etc*. It has been half a century since, and we see only traces of higher-order-ness in a handful of researches happening on hypergraphs, simplicial complexes, multi-way data analysis, *etc*.

Through this thesis, we feel that we have made some insightful revelations about the working of hypergraphs as real-world networks. The seemingly gigantic perception of the power set of a set of vertices sure forces one to resort to pairs of vertices instead, but one needs to push the boundaries of graph theory and start tackling hypergraphs, for a graph by definition is also a hypergraph but the opposite is not true. With the advent of deep learning, *explainability* has become a huge issue, and though research on the same is slowly gaining pace, unexplained yet excellently performing deep learning models are way ahead in the race. Pieces of research such as *shortcut learning* [34] and *attention is not explanation* [47][1] forces one to believe that unless intervened, with time, a typical machine learning model's performance is going to be far more important than its explanation.

Our encounter with deep learning hypergraph models have shown us that to explain why a deep learning model works well – which it sure does – is not an easy task. In fact, we were able to learn and contribute better via Chapters 3–6 than we did through the other two chapters that involved deep learning, albeit we made our best attempts to explain why our latter models SHONeN and CATSETMAT work. Moreover, what we understand by the superlative performance of these models on hypergraphs is that careful curation of subtle elements of higher-order relations (namely, *sub-hyperedges* and *cross-connections* respectively) gives a better performance than the deep learning model having to figure out the subtleties themselves. Thus, this very exercise – of hunting for insights into the governing principles of hypergraphs – could hit two birds with one stone: learn a better performing model, and explain it via the insights gained *before* building it. Surprisingly enough, our guess is that one does not even need to study a hypergraph if a powerful enough deep learning model is employed to predict hyperedges from a graph itself — only it won't explain why it did what it did.

That said, we are hopeful that more and more interest would approach the intersection of hypergraphs and machine learning. One might very well be interested in mere link prediction, which seems to be a purely graph-based exercise. But Chapters 3 and 4 vouch against this argument and clearly establish that higher-order relations should not be ignored. Similarly, if the aim itself is to predict

[1] There exists a counter work – *attention is not not explanation* [119] – as well!

whether a group of entities would come together as a higher-order collaboration – the problem of hyperlink prediction – there are only a handful of approaches available in the literature. Moreover, slight changes made to the existing models based on strong formulations (such as *clique-closure* or *sub-higher-order* or *cross-attention*) enhance the performance of these models not only for unipartite hypergraphs (Chapters 6 and 7), but for bipartite hypergraphs as well (Chapter 8).

In summary, we envision a future where higher-order relations are invested into to such an extent, that instead of graph research driving research on hypergraphs (as is the case with graph Laplacian followed by hypergraph Laplacian, graph neural network followed by hypergraph neural network, graph convolution followed by hypergraph convolution, graph GANs followed by hypergraph GANs, and graph attention followed by hypergraph attention), the chronological order of the interest undergoes a reversal.

References

[1] Mansurul A. Bhuiyan, Mahmudur Rahman, Mahmuda Rahman, and Mohammad Al Hasan. GUISE: Uniform Sampling of Graphlets for Large Graph Analysis. In *2012 IEEE ICDM*, pages 91–100, Dec 2012. 58, 65

[2] Lada A Adamic and Eytan Adar. Friends and Neighbors on the Web. *Social Networks*, 25(3): 211–230, 2003. 20, 21, 32, 39, 47, 109, 120

[3] Sameer Agarwal, Kristin Branson, and Serge Belongie. Higher Order Learning with Graphs. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 17–24, New York, NY, USA, 2006. ACM. ISBN 1-59593-383-2. doi: 10.1145/1143844.1143847. URL http://doi.acm.org/10.1145/1143844.1143847. 1, 4, 6, 14, 22, 25, 64, 80, 86, 87, 88, 99

[4] Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, and Mohammed Zaki. Link Prediction using Supervised Learning. In *SDM06: Workshop on Link Analysis, Counter-terrorism and Security*, 2006. 17, 38, 58, 65

[5] Chidambaram Annamalai. Finding Perfect Matchings in Bipartite Hypergraphs. In *Proceedings of the Twenty-seventh Annual ACM-SIAM Symposium on Discrete algorithms*, pages 1814–1823. SIAM, 2016. 7, 120

[6] David Malet Armstrong. *Nominalism and Realism: Volume 1: Universals and Scientific Realism*. Universals and Scientific Realism. Cambridge University Press, 1978. ISBN 9780521280334. URL https://books.google.co.in/books?id=aAg6AAAAIAAJ. 1

[7] Song Bai, Feihu Zhang, and Philip HS Torr. Hypergraph Convolution and Hypergraph Attention. *arXiv preprint arXiv:1901.08150*, 2019. 87, 98, 99, 119

[8] Sambaran Bandyopadhyay, Kishalay Das, and M Narasimha Murty. Line Hypergraph Convolution Network: Applying Graph Convolution for Hypergraphs. *arXiv preprint arXiv:2002.03392*, 2020. 119

[9] Juan I Fuxman Bass, Alos Diallo, Justin Nelson, Juan M Soto, Chad L Myers, and Albertha JM Walhout. Using Networks to Measure Similarity between Genes: Association Index Selection. *Nature Methods*, 10(12):1169, 2013. 36

[10] Austin R Benson, David F Gleich, and Jure Leskovec. Higher-order organization of complex networks. *Science*, 353(6295):163–166, 2016. 4

[11] Austin R Benson, Rediet Abebe, Michael T Schaub, Ali Jadbabaie, and Jon Kleinberg. Simplicial Closure and Higher-order Link Prediction. *Proc. National Academy of Sciences*, 115(48): E11221–E11230, 2018. 5, 12, 32, 33, 48, 56, 58, 64, 65, 71, 77, 80, 85, 86, 99, 100, 119, 129

[12] Austin R Benson, Ravi Kumar, and Andrew Tomkins. Sequences of sets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1148–1157, 2018. 12

[13] Claude Berge. *Graphs and Hypergraphs*. Translated by Edward Minieka. North-Holland Publishing Company, 1973. ISBN 9780444103994. URL https://books.google.co.in/books?id=X32GlVfqXjsC. 4

[14] Claude Berge. *Hypergraphs: Combinatorics of Finite Sets*, volume 45 of *North-Holland Mathematical Library*. Elsevier Science, 1984. ISBN 9780080880235. URL https://books.google.co.in/books?id=jEyfse-EKf8C. 4

[15] Catherine A Bliss, Morgan R Frank, Christopher M Danforth, and Peter Sheridan Dodds. An evolutionary algorithm approach to link prediction in dynamic social networks. *Journal of Computational Science*, 5(5):750–764, 2014. 32

[16] Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. NetGAN: Generating Graphs via Random Walks. *arXiv preprint arXiv:1803.00816*, 2018. 99

[17] Phillip Bonacich, Annie Cody Holdren, and Michael Johnston. Hyper-edges and Multidimensional Centrality. *Social Networks*, 26(3):189–203, 2004. 109, 119

[18] Alain Bretto. *Hypergraph Theory: An Introduction*. Mathematical Engineering. Springer International Publishing, 2013. ISBN 9783319000800. URL https://books.google.co.in/books?id=lb5DAAAAQBAJ. 4

[19] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16: 321–357, 2002. 57

[20] Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016. 49

[21] Sara Cohen and Aviv Zohar. An Axiomatic Approach to Link Prediction. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015. 21

[22] Harold SM Coxeter. Self-dual Configurations and Regular Graphs. *Bulletin of the American Mathematical Society*, 56(5):413–455, 1950. 6

[23] A. Davis, B. B. Gardner, and M. R. Gardner. *Deep South: A Social Anthropological Study of Caste and Class.* University of Chicago Press, Chicago, IL, US, 1941. 56

[24] Jesse Davis and Mark Goadrich. The Relationship between Precision-Recall and ROC Curves. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 233–240. ACM, 2006. 49

[25] Javier De Las Rivas and Celia Fontanillo. Protein–Protein Interactions Essentials: Key Concepts to Building and Analyzing Interactome Networks. *PLoS Computational Biology*, 6(6), 2010. 2

[26] Hongbo Deng, Jiawei Han, Michael R Lyu, and Irwin King. Modeling and Exploiting Heterogeneous Bibliographic Networks for Expertise Ranking. In *Proceedings of the 12th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 71–80, 2012. 109

[27] Arthur Conan Doyle. *The Complete Sherlock Holmes*. Garden City, N.Y.: Doubleday & Co., 1930. 3

[28] Herbert Edelsbrunner and John Harer. *Computational Topology: An Introduction*. American Mathematical Soc., 2010. 23

[29] Willis D. Ellis, editor. *A Source Book of Gestalt Psychology*. Kegan Paul, Trench, Trubner & Company, 1938. doi: 10.1037/11496-000. URL https://psycnet.apa.org/record/2007-10344-000. 3

[30] Tom Fawcett. An Introduction to ROC Analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006. 28

[31] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph Neural Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3558–3565, 2019. 87, 98, 100, 119

[32] Francois Fouss, Alain Pirotte, Jean-Michel Renders, and Marco Saerens. Random-walk Computation of Similarities between Nodes of a Graph with Application to Collaborative Recommendation. *IEEE Trans. on Knowledge and Data Engineering*, 19(3):355–369, 2007. 37

[33] Dario Garcia Gasulla, Claudio Ulises Cortés García, Eduard Ayguadé Parra, and Jesús José Labarta Mancho. Evaluating Link Prediction on Large Graphs. In *Artificial Intelligence Research and Development: Proceedings of the 18th International Conference of the Catalan Association for Artificial Intelligence*, pages 90–99. IOS Press, 2015. 56

[34] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut Learning in Deep Neural Networks. *arXiv preprint arXiv:2004.07780*, 2020. 133

[35] Zoubin Ghahramani and Katherine A Heller. Bayesian Sets. In Y. Weiss, B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 435–442. MIT Press, 2006. URL http://papers.nips.cc/paper/2817-bayesian-sets.pdf. 66, 80

[36] Robert W Ghrist. *Elementary Applied Topology*, volume 1. Createspace Seattle, 2014. 23

[37] Liang Gou, Xiaolong Zhang, Hung-Hsuan Chen, Jung-Hyun Kim, and C Lee Giles. Social Network Document Ranking. In *Proceedings of the 10th Annual Joint Conference on Digital Libraries*, pages 313–322, 2010. 36

[38] Aditya Grover and Jure Leskovec. Node2Vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 855–864, 2016. 99, 100, 101, 120, 121

[39] Huan Gui, Jialu Liu, Fangbo Tao, Meng Jiang, Brandon Norick, and Jiawei Han. Large-scale Embedding Learning in Heterogeneous Event Data. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 907–912. IEEE, 2016. 109

[40] Jean-Loup Guillaume and Matthieu Latapy. Bipartite Structure of all Complex Networks. *Information Processing Letters*, 90(5):215–221, 2004. 120

[41] Raf Guns. Link Prediction. In *Measuring Scholarly Impact*, pages 35–55. Springer, 2014. 14, 32, 40, 47

[42] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017. 120

[43] Chaoyang He, Tian Xie, Yu Rong, Wenbing Huang, Yanfang Li, Junzhou Huang, Xiang Ren, and Cyrus Shahabi. Bipartite Graph Neural Networks for Efficient Node Representation Learning. *arXiv preprint arXiv:1906.11994*, 2019. 120

[44] Olga Heismann. *The Hypergraph Assignment Problem*. PhD thesis, Technische Universität Berlin, 2014. 120

[45] Peter D Hoff, Adrian E Raftery, and Mark S Handcock. Latent Space Approaches to Social Network Analysis. *Journal of the American Statistical Association*, 97(460):1090–1098, 2002. doi: 10.1198/016214502388618906. URL https://doi.org/10.1198/016214502388618906. 24

[46] Jie Huang, Chuan Chen, Fanghua Ye, Jiajing Wu, Zibin Zheng, and Guohui Ling. Hyper2vec: Biased Random Walk for Hyper-network Embedding. In *International Conference on Database Systems for Advanced Applications*, pages 273–277. Springer, 2019. 119

[47] Sarthak Jain and Byron C Wallace. Attention is not Explanation. In *Proc. 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, pages 3543–3556, 2019. 133

[48] Johnson Jeffrey. *Hypernetworks in the Science of Complex Systems*, volume 3. World Scientific, 2013. 4

[49] Glen Jeh and Jennifer Widom. SimRank: a Measure of Structural-context Similarity. In *Proceedings of the eighth ACM SIGKDD International Conference on Knowledge discovery and data mining*, pages 538–543, 2002. 32

[50] Hawoong Jeong, Zoltan Néda, and Albert-László Barabási. Measuring Preferential Attachment in Evolving Networks. *EPL (Europhysics Letters)*, 61(4):567, 2003. 32

[51] Jianwen Jiang, Yuxuan Wei, Yifan Feng, Jingxuan Cao, and Yue Gao. Dynamic Hypergraph Neural Networks. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2635–2641, 2019. 119

[52] N. Kashtan, S. Itzkovitz, R. Milo, and U. Alon. Efficient Sampling Algorithm for Estimating Subgraph Concentrations and Detecting Network Motifs. *Bioinformatics*, 20(11):1746–1758, July 2004. ISSN 1367-4803. doi: 10.1093/bioinformatics/bth163. 58, 62, 65

[53] Daniel Katz. Gestalt psychology. *Methuen & Co. LTD.: London*, 1950. 3

[54] Leo Katz. A New Status Index Derived from Sociometric Analysis. *Psychometrika*, 18(1): 39–43, Mar 1953. ISSN 1860-0980. doi: 10.1007/BF02289026. URL https://doi.org/10.1007/BF02289026. 39, 47, 66, 80

[55] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. *CoRR*, abs/1609.02907, 2016. URL http://arxiv.org/abs/1609.02907. 87, 99, 120

[56] Maksim Kitsak, Fragkiskos Papadopoulos, and Dmitri Krioukov. Latent geometry of bipartite networks. *Physical Review E*, 95(3):032309, 2017. 109

[57] Thomas Klimpel. How is a hypergraph different from a bipartite graph? Computer Science Stack Exchange, 2013. https://cs.stackexchange.com/q/12769 (version: 2017-04-13). 6

[58] Kurt Koffka, editor. *Principles of Gestalt Psychology*. Harcourt, Brace, 1935. URL https://psycnet.apa.org/record/1935-03991-000. 3

[59] Da Kuang, Chris Ding, and Haesun Park. Symmetric Nonnegative Matrix Factorization for Graph Clustering. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, pages 106–117. SIAM, 2012. 78

[60] Jérôme Kunegis, Ernesto W De Luca, and Sahin Albayrak. The Link Prediction Problem in Bipartite Networks. In *International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems*, pages 380–389. Springer, 2010. 80

[61] Eungju Kwon, Jongwoo Kim, Nojeong Heo, and Sanggil Kang. Personalized Recommendation System using Level of Cosine Similarity of Emotion Word from Social Network. *Journal of Information Technology and Architecture*, 9(3):333–344, 2012. 36

[62] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks. In *International Conference on Machine Learning*, pages 3744–3753, 2019. 120

[63] Steven M Lehar. *The World in your Head: A Gestalt View of the Mechanism of Conscious Experience*. Psychology Press, 2003. 6

[64] Dong Li, Zhiming Xu, Sheng Li, and Xin Sun. Link Prediction in Social Networks based on Hypergraph. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 41–42. ACM, 2013. 39, 47, 119

[65] Pan Li and Olgica Milenkovic. Inhomogeneous hypergraph clustering with applications. In *Advances in Neural Information Processing Systems*, pages 2308–2318, 2017. 4

[66] David Liben-Nowell and Jon Kleinberg. The Link Prediction Problem for Social Networks. In *Proc. 12th Int. Conf. on Information and Knowledge Management*, CIKM '03, pages 556–559, New York, NY, USA, 2003. ACM. ISBN 1-58113-723-0. doi: 10.1145/956863.956972. 5, 14, 17, 20, 21, 32, 38, 39, 47, 49, 56, 65, 78, 80, 100, 120

[67] Ryan N Lichtenwalter, Jake T Lussier, and Nitesh V Chawla. New Perspectives and Methods in Link Prediction. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 243–252. ACM, 2010. 56, 57, 58, 63, 65

[68] Ryan Lichtnwalter and Nitesh V Chawla. Link Prediction: Fair and Effective Evaluation. In *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*, pages 376–383. IEEE Computer Society, 2012. 56, 57

[69] Linyuan Lü and Tao Zhou. Link Prediction in Complex Networks: A Survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150–1170, 2011. 56

[70] Zhengdong Lu, Berkant Savas, Wei Tang, and Inderjit S Dhillon. Supervised Link Prediction using Multiple Sources. In *2010 IEEE International Conference on Data Mining*, pages 923–928. IEEE, 2010. 47

[71] Fraser MacBride. Relations. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Winter 2016 edition, 2016. 2

[72] Víctor Martínez, Fernando Berzal, and Juan-Carlos Cubero. A Survey of Link Prediction in Complex Networks. *ACM Comput. Surv.*, 49(4):69:1–69:33, December 2016. ISSN 0360-0300. doi: 10.1145/3012704. URL http://doi.acm.org/10.1145/3012704. 21, 32, 39, 47, 56

[73] Changping Meng, Jiasen Yang, Bruno Ribeiro, and Jennifer Neville. HATS: A Hierarchical Sequence-Attention Framework for Inductive Set-of-Sets Embeddings. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 783–792, 2019. 120

[74] Aditya Krishna Menon and Charles Elkan. Link prediction via matrix factorization. In *Joint european conference on machine learning and knowledge discovery in databases*, pages 437–452. Springer, 2011. 32

[75] Marvin Minsky. A Framework for Representing Knowledge. In P. Winston, editor, *The Psychology of Computer Vision*. McGraw-Hill, 1975. (*Reprinted from "MIT-AI Laboratory Memo, June 1974,* URL http://hdl.handle.net/1721.1/6089."). 3, 133

[76] M. E. J. Newman. Coauthorship Networks and Patterns of Scientific Collaboration. *Proceedings of the National Academy of Sciences*, 101(suppl 1):5200–5205, 2004. ISSN 0027-8424. doi: 10.1073/pnas.0307545100. URL https://www.pnas.org/content/101/suppl_1/5200. 39

[77] Mark Ed Newman, Albert-László Ed Barabási, and Duncan J Watts. *The Structure and Dynamics of Networks*. Princeton university press, 2006. 108

[78] Mark EJ Newman. Clustering and Preferential Attachment in Growing Networks. *Physical review E*, 64(2):025102, 2001. 21, 32, 38, 47, 66, 78

[79] Mark EJ Newman. The Structure and Function of Complex Networks. *SIAM review*, 45(2): 167–256, 2003. 108, 120

[80] Federica Parisi, Guido Caldarelli, and Tiziano Squartini. Entropy-based approach to missing-links prediction. *Applied Network Science*, 3(1):17, 2018. 32

[81] Prasanna Patil, Govind Sharma, and M. Narasimha Murty. Negative Sampling for Hyperlink Prediction in Networks. In Hady W. Lauw, Raymond Chi-Wing Wong, Alexandros Ntoulas, Ee-Peng Lim, See-Kiong Ng, and Sinno Jialin Pan, editors, *Advances in Knowledge Discovery and Data Mining*, pages 607–619, Cham, 2020. Springer International Publishing. ISBN 978-3-030-47436-2. 100, 127

[82] Josh Payne. Deep Hyperedges: A Framework for Transductive and Inductive Learning on Hypergraphs. *arXiv preprint arXiv:1910.02633*, 2019. 87, 99, 119

[83] Ratha Pech, Dong Hao, Liming Pan, Hong Cheng, and Tao Zhou. Link prediction via matrix completion. *EPL (Europhysics Letters)*, 117(3):38002, 2017. 32

[84] Heinrich Reitberger. Leopold Vietoris (1891-2002). *Notices of the American Mathematical Society*, 49(10):1232–1236, 2002. 23

[85] Steffen Rendle. Factorization Machines with libFM. *ACM Trans. Intell. Syst. Technol.*, 3 (3):57:1–57:22, May 2012. ISSN 2157-6904. doi: 10.1145/2168752.2168771. URL http://doi.acm.org/10.1145/2168752.2168771. 66, 80

[86] Christian P Robert and George Casella. The Metropolis-Hastings Algorithm. In *Monte Carlo Statistical Methods*, pages 231–283. Springer, 1999. 63

[87] Erich Rome. Simulating Perceptual Clustering by Gestalt Principles. In *25th Workshop of the Australian Association for Pattern Recognition*, pages 191–198. OAGM/AAPR, 2001. 3

[88] Yuki Saito, Takuma Nakamura, Hirotaka Hachiya, and Kenji Fukumizu. Deep Set-to-Set Matching and Learning. *arXiv preprint arXiv:1910.09972*, 2019. 120

[89] Purnamrita Sarkar, Deepayan Chakrabarti, and Andrew W Moore. Theoretical Justification of popular Link Prediction Heuristics. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011. 21, 24

[90] Tanveerulhuq Shaik, Vadlamani Ravi, and Kalyanmoy Deb. Evolutionary multi-objective optimization algorithm for community detection in complex social networks. *SN Computer Science*, 2(1):1–25, 2020. 32

[91] Claude Elwood Shannon. A Mathematical Theory of Communication. *Bell System Technical journal*, 27(3):379–423, 1948. 48

[92] Ankit Sharma, Terrence J Moore, Ananthram Swami, and Jaideep Srivastava. Weighted Simplicial Complex: A Novel Approach for Predicting Small Group Evolution. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 511–523. Springer, 2017. 99

[93] Ankit Sharma, Shafiq Joty, Himanshu Kharkwal, and Jaideep Srivastava. Hyperedge2vec: Distributed Representations for Hyperedges, 2018. URL https://openreview.net/forum?id=rJ5C67-C-. 99

[94] Govind Sharma, Prasanna Patil, and M Narasimha Murty. C3MM: Clique-Closure based Hyperlink Prediction. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI)*. AAAI Press, 2020. URL https://link.springer.com/chapter/10.1007/978-3-030-47436-2_46.

[95] Govind Sharma, Prasanna Patil, and M Narasimha Murty. SHONENs: Sub-higher-order Neural Networks for Hyperedges. In *2020 International Conference on Data Mining (ICDM)*. IEEE, 2020. Submitted.

[96] Govind Sharma, Aditya Challa, Paarth Gupta, and M Narasimha Murty. Higher Order Relations Skew Link Prediction in Graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. Sumbitted.

[97] Govind Sharma, Swyam Singh, V Susheela Devi, and M Narasimha Murty. The CAT SET on the MAT: Cross Attention for Set Matching in Bipartite Hypergraphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. Sumbitted.

[98] David Shore (creator), Hugh Laurie (actor), Daniel Sackheim (director), Matt Witten, Peter Blake (writers), et al. "Cursed" (Season 1, Episode 13). *House M.D. (TV Series)*, March 1, 2005 (first aired). URL https://www.imdb.com/title/tt0606015/. 108

[99] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June Hsu, and Kuansan Wang. An Overview of Microsoft Academic Service (MAS) and Applications. In *Proceedings of the 24th International Conference on World Wide Web*, pages 243–246, 2015. 118

[100] M. Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 2012. ISBN 9781285401065. URL https://books.google.co.in/books?id=1aMKAAAAQBAJ. 13

[101] Yizhou Sun and Jiawei Han. Mining Heterogeneous Information Networks: Principles and Methodologies. *Synthesis Lectures on Data Mining and Knowledge Discovery*, 3(2):1–159, 2012. 109

[102] Yizhou Sun, Rick Barber, Manish Gupta, Charu C Aggarwal, and Jiawei Han. Co-author Relationship Prediction in Heterogeneous Bibliographic Networks. In *2011 International Conference on Advances in Social Networks Analysis and Mining*, pages 121–128. IEEE, 2011. 36

[103] Fei Tan, Yongxiang Xia, and Boyao Zhu. Link Prediction in Complex Networks: a Mutual Information Perspective. *PloS one*, 9(9):e107056, 2014. 48

[104] Mursel Tasgin, Amac Herdagdelen, and Haluk Bingol. Community detection in complex networks using genetic algorithms. *arXiv preprint arXiv:0711.0491*, 2007. 32

[105] Amanda L Traud, Peter J Mucha, and Mason A Porter. Social Structure of Facebook Networks. *Physica A: Statistical Mechanics and its Applications*, 391(16):4165–4180, 2012. 109

[106] Charalampos E Tsourakakis, Jakub Pachocki, and Michael Mitzenmacher. Scalable motif-aware graph clustering. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1451–1460, 2017. 4

[107] Ke Tu, Peng Cui, Xiao Wang, Fei Wang, and Wenwu Zhu. Structural Deep Embedding for Hyper-networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 12, 99, 119

[108] Kathryn Turnbull, Simón Lunagómez, Christopher Nemeth, and Edoardo Airoldi. Latent Space Modelling of Hypergraph Data, 2019. 22, 23

[109] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017. 97

[110] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph Attention Networks. *arXiv preprint arXiv:1710.10903*, 2017. 99, 120

[111] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order Matters: Sequence to Sequence for Sets. *arXiv preprint arXiv:1511.06391*, 2015. 120

[112] Edward Wagstaff, Fabian Fuchs, Martin Engelcke, Ingmar Posner, and Michael A. Osborne. On the Limitations of Representing Functions on Sets. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6487–6494, Long Beach, California, USA, 09–15 Jun 2019. PMLR. 120

[113] Peng Wang, BaoWen Xu, YuRong Wu, and XiaoYu Zhou. Link Prediction in Social Networks: The State-of-the-art. *Science China Information Sciences*, 58(1):1–38, Jan 2015. ISSN 1869-1919. doi: 10.1007/s11432-014-5237-y. 21, 47

[114] Tingli Wang and Guoqiong Liao. A Review of Link Prediction in Social Networks. In *2014 International Conference on Management of e-Commerce and e-Government (ICMeCG)*, pages 147–150. IEEE, 2014. 56

[115] Yueyang Wang, Ziheng Duan, Binbing Liao, Fei Wu, and Yueting Zhuang. Heterogeneous Attributed Network Embedding with Graph Convolutional Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 10061–10062, 2019. 109

[116] Satosi Watanabe. *Pattern Recognition: Human and Mechanical*. John Wiley & Sons, Inc., USA, 1985. ISBN 0471808156. 3, 133

[117] Eric W Weisstein. Levi Graph. *https://mathworld. wolfram. com/*, 2000. URL https://mathworld.wolfram.com/LeviGraph.html. 6

[118] Chris Wendler, Markus Püschel, and Dan Alistarh. Powerset Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, pages 927–938, 2019. 120

[119] Sarah Wiegreffe and Yuval Pinter. Attention is not Explanation. 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)*, pages 11–20, 2019. 133

[120] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020. 120

[121] Ye Xu, Dan Rockmore, and Adam M Kleinbaum. Hyperlink Prediction in Hypernetworks using Latent Social Features. In *International Conference on Discovery Science*, pages 324–339. Springer, 2013. 5, 12, 55, 56, 58, 64, 71, 80, 99

[122] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. HyperGCN: A New Method For Training Graph Convolutional Networks on Hypergraphs. In *Advances in Neural Information Processing Systems*, pages 1509–1520, 2019. 87, 99, 119

[123] Naganand Yadati, Vikram Nitin, Madhav Nimishakavi, Prateek Yadav, Anand Louis, and Partha Talukdar. Link Prediction in Hypergraphs using Graph Convolutional Networks. *Openreview Blind Submission by ICLR*, 2019. URL https://openreview.net/forum?id=ryeaZhRqFm. 99, 119

[124] Rawan I Yaghi, Hossam Faris, Ibrahim Aljarah, Al-Zoubi Ala'M, Ali Asghar Heidari, and Seyedali Mirjalili. Link prediction using evolutionary neural network models. In *Evolutionary Machine Learning Techniques*, pages 85–111. Springer, 2020. 32

[125] Dingqi Yang, Bingqing Qu, Jie Yang, and Philippe Cudre-Mauroux. Revisiting User Mobility and Social Relationships in LBSNs: A Hypergraph Embedding Approach. In *The World Wide Web Conference*, WWW '19, pages 2147–2157, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-6674-8. doi: 10.1145/3308558.3313635. URL http://doi.acm.org/10.1145/3308558.3313635. 109

[126] Yang Yang, Ryan N Lichtenwalter, and Nitesh V Chawla. Evaluating Link Prediction Methods. *Knowledge and Information Systems*, 45(3):751–782, 2015. 56, 57, 58

[127] Lin Yao, Luning Wang, Lv Pan, and Kai Yao. Link prediction based on common-neighbors for dynamic social network. *Procedia Computer Science*, 83:82–89, 2016. 32

[128] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep Sets. In *Advances in Neural Information Processing Systems*, pages 3391–3401, 2017. 120

[129] C. T. Zahn. Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters. *IEEE Transactions on Computers*, C-20(1):68–86, 1971. 3

[130] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*, pages 5165–5175, 2018. 32

[131] Muhan Zhang, Zhicheng Cui, Tolutola Oyetunde, Yinjie Tang, and Yixin Chen. Recovering Metabolic Networks using A Novel Hyperlink Prediction Method. *arXiv preprint arXiv:1610.06941*, 2016. 56, 64, 80

[132] Muhan Zhang, Zhicheng Cui, Shali Jiang, and Yixin Chen. Beyond Link Prediction: Predicting Hyperlinks in Adjacency Space. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 5, 55, 56, 58, 64, 66, 71, 76, 77, 79, 80, 81, 99, 119

[133] Ruochi Zhang, Yuesong Zou, and Jian Ma. Hyper-SAGNN: A Self-attention based Graph Neural Network for Hypergraphs. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=ryeHuJBtPH. 5, 6, 12, 55, 87, 97, 99, 101, 113, 117, 119, 121

[134] Yan Zhang, Jonathon Hare, and Adam Prügel-Bennett. FSPool: Learning Set Representations with Featurewise Sort Pooling. *arXiv preprint arXiv:1906.02795*, 2019. 120

[135] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning with Hypergraphs: Clustering, Classification, and Embedding. In *Advances in Neural Information Processing Systems*, pages 1601–1608, 2007. 64, 80, 87, 99, 119

[136] Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. Predicting Missing Links via Local Information. *The European Physical Journal B*, 71(4):623–630, 2009. 32

[137] Yu Zhu, Ziyu Guan, Shulong Tan, Haifeng Liu, Deng Cai, and Xiaofei He. Heterogeneous hypergraph embedding for document recommendation. *Neurocomputing*, 216:150–162, 2016. 109

[138] Albert L. Zobrist and William B. Thompson. Building a distance function for gestalt grouping. *IEEE Transactions on Computers*, 100(7):718–728, 1975. 3

[139] Afra Zomorodian. Fast Construction of the Vietoris-Rips complex. *Computers & Graphics*, 34 (3):263–271, 2010. 23

[140] Inna Zverovich and Igor Zverovich. Bipartite Bihypergraphs: A Survey and New Results. *Discrete Mathematics*, 306(8-9):801–811, 2006. 7, 109, 120