One Graph to Rule them All: Using NLP and Graph Neural Networks to analyse Tolkien's Legendarium

Vincenzo Perri[1,*,†], Lisi Qarkaxhija[2,*,†], Albin Zehe[3,*,†], Andreas Hotho[3,*] and Ingo Scholtes[2,1]

[1]Data Analytics Group, Department of Informatics(IfI), Universität Zürich, CH-8050 Zürich, Switzerland
[2]Chair of Informatics XV - Machine Learning for Complex Networks, Center for Artificial Intelligence and Data Science (CAIDAS), Julius-Maximilians-Universität Würzburg, D-97074 Würzburg, Germany
[3]Chair of Informatics X - Data Science, Center for Artificial Intelligence and Data Science (CAIDAS), Julius-Maximilians-Universität Würzburg, D-97074 Würzburg, Germany

Abstract

Natural Language Processing and Machine Learning have considerably advanced Computational Literary Studies. Similarly, the construction of co-occurrence networks of literary characters, and their analysis using methods from social network analysis and network science, have provided insights into the micro- and macro-level structure of literary texts. Combining these perspectives, in this work we study character networks extracted from a text corpus of J.R.R. Tolkien's Legendarium. We show that this perspective helps us to analyse and visualise the narrative style that characterises Tolkien's works. Addressing character classification, embedding and co-occurrence prediction, we further investigate the advantages of state-of-the-art Graph Neural Networks over a popular word embedding method. Our results highlight the large potential of graph learning in Computational Literary Studies.

Keywords

computational literary studies, character networks, network analysis, graph neural networks, NLP

1. Motivation and Background

Computational Literary Studies (CLS) have recently taken advantage of the latest developments in Natural Language Processing (NLP), with deep learning techniques bringing major improvements for tasks relevant to literature analysis: Examples include named entity recognition [28], anaphora and coreference resolution [44], sentiment analysis [12], scene detection [54] or genre classification [52]. Apart from NLP, machine learning has recently shown great potential in data with complex *relational* structure that can be represented as *graph* or *network* $G = (V, E)$, consisting of nodes $u, v, \ldots \in V$ and links $(u, v) \in E$. In CLS, this abstraction is frequently used to study *character networks*, i.e. graphs where nodes represent literary characters and links represent relationships such as, e.g., their co-occurrence in a sentence or scene,

CHR 2022: Computational Humanities Research Conference, December 12 – 14, 2022, Antwerp, Belgium
*Corresponding author.
†These authors contributed equally.
perri@ifi.uzh.ch (V. Perri); lisi.qarkaxhija@uni-wuerzburg.de (L. Qarkaxhija); zehe@informatik.uni-wuerzburg.de (A. Zehe); hotho@informatik.uni-wuerzburg.de (A. Hotho); ingo.scholtes@uni-wuerzburg.de (I. Scholtes)
0000-0002-9472-0783 (A. Zehe); 0000-0002-0483-5772 (A. Hotho); 0000-0003-2253-0216 (I. Scholtes)

or dialogue interactions [23]. Building on this abstraction, several works in CLS used (social) network analysis to study the narrative structure of literary works [32, 47]: Considering 19th century novels, Elson, McKeown, and Dames [11] analysed macroscopic properties of conversation networks, i.e. fictional characters engaging in dialogue. Beveridge and Shan [7] applied centrality measures and community detection to a network of characters that occur within close proximity in the text of *A Storm of Swords*, the third novel in George R.R. Martin's series *Song of Ice of Fire*. Using the same network extraction, Bonato, D'Angelo, Elenberg, Gleich, and Hou [8] studied statistical properties of character networks in three popular novels. Several authors applied centrality measures to identify important characters, e.g. in works by Shakespeare [53], *Alice in Wonderland* [1], or the first novel of the *Harry Potter* series [42]. In a recent work, Agarwal, Vijay, et al. [2] used character networks to facilitate the automated classification of literary genres. Using J.R.R. Tolkien's *The Lord of the Rings*, which we also consider in our manuscript, Ribeiro, Vosgerau, Andruchiw, and Pinto [38] applied social network analysis to its network of characters. Li, Zhang, Tan, and Li [27] studied small-world and scale-free properties of character co-occurrence networks in movie scripts, among them the movie adaptation of *The Lord of the Rings*.

Existing studies of character networks mainly used methods from (social) network analysis to gain insights into the narrative structure of literary texts. At the same time, recent advances in *geometric deep learning* [9] and *Graph Neural Networks* (GNNs) provide new ways to apply deep learning to graph-structured data, which creates interesting opportunities for the study of character networks. A major advantage of GNNs over other network analysis or machine learning techniques is their ability to leverage relational patterns in the topology of a graph, while at the same time incorporating additional node or link features. This facilitates unsupervised and (semi-)supervised learning tasks at the node, link, or graph level. Examples include graph representation learning [34, 17], community detection [10], node and link classification [21, 49], link prediction [55], or graph classification [56]. To the best of our knowledge, no works have combined recent advances in (i) natural language processing, e.g. to recognize entities, resolve coreferences, extract meaningful character networks, or generate word embeddings, and (ii) graph neural networks, e.g. to compute latent space representations of fictional characters or address node- and link-level learning tasks. In CLS, the combination of these paradigms can help with several tasks, e.g. (semi-)supervised character classification, semantic analyses of character relationships, comparisons of character constellations in different works, or automated genre classification. Addressing this gap, we combine NLP, network analysis and graph learning to analyse J.R.R. Tolkien's Legendarium, namely *The Silmarillion*, *The Hobbit*, and the three volumes of *The Lord of the Rings*. Our contributions are:

- We apply entity recognition and coreference resolution to detect and disambiguate characters in Tolkien's Legendarium. We report key statistics like character mentions via pronouns, nominal mentions or explicit references and compare them across different works in the Legendarium. We use sequential character mentions to generate narrative charts of the five considered works, which highlight Tolkien's *interlaced narrative style*.
- We extract *character networks* for each work in our corpus, i.e. graphs $G = (V, E)$ where nodes $v \in V$ capture characters in the Legendarium, while undirected edges (v, w) represent the co-occurrence of two characters in the same sentence. Apart from generating character networks for the five works in our corpus, we generate a *single* network that

captures all characters in the works that constitute Tolkien's Legendarium. We use this to perform a macroscopic, graph-based characterisation of Tolkien's works.
- We address the question to what extent graph learning techniques can leverage the topology of automatically extracted character networks, and which advantages they provide over word embeddings that just consider word-context pairs. To this end, we evaluate the performance of different methods for the (i) latent space representation of characters, (ii) (semi-)supervised classification of characters, and (iii) prediction of character co-occurrences. The results confirm that the inclusion of topological information from character networks considerably improves the performance of these tasks.

Analysing a single graph representation of multiple literary works unfolding in the same fictional universe, our work demonstrates the potential of graph neural networks for computational literary studies. To facilitate the use of our data and methods in the (digital) study of Tolkien's works[1], we make the results of our entity recognition, coreference resolution and character network extraction available. To foster the application of our methods to other corpora, we also provide a set of jupyter notebooks that reproduce our findings.[2]

2. Text Corpus and Data Processing

We consider the English full text of *The Lord of the Rings* (consisting of the volumes *The Fellowship of the Ring*, *The Two Towers* and *The Return of the King*, each split into two books), *The Hobbit* and *The Silmarillion*. We used the python-based NLP pipeline BookNLP[3] to extract linguistic and literary features. BookNLP uses the NLP service spaCy [20] to perform tokenisation, that is, splitting text into a list of words and special characters (like punctuation) and to split text into sentences. This first step in the NLP pipeline is the basis for all further processing steps.

Entity Recognition and Coreference Resolution Entity recognition refers to the task of detecting all references to *entities* (e.g., characters, location) in a text corpus. These references can either be explicitly named references (e.g., "Bilbo Baggins", "Smaug"), noun phrases (e.g., "the hobbit", "the dragon") or pronouns (e.g., "she", "they"). BookNLP uses an entity annotation model that has been trained on a large annotated data set [5] to identify named entities, noun phrases as well as pronoun references. After these references have been detected, in a next step coreference resolution can be applied, which is a very hard task in general [45] and is especially hard in the context of literary texts due to the high variation of references used and the very long texts [40, 22]. Confirming this view, our initial analyses revealed that the performance of BookNLP's coreference resolution, which was trained on a data set of annotated coreferences [4] was not satisfactory when applying it to our corpus. We thus decided to focus on named references, and resolve these using a set of simple manually-created disambiguation rules (e.g., "Sam" → "Sam Gamgee", "Peregrin" → "Pippin").[4] Although this approach may yield

[1] https://digitaltolkien.com/.
[2] https://github.com/LSX-UniWue/one-graph-to-rule-them-all.
[3] https://github.com/booknlp/booknlp.
[4] The full list of disambiguation rules can be found in our repository.

a low recall (i.e. there are many unidentified coreferences since pronouns and noun phrases are not considered), we find that this coreference resolution yields high precision (i.e. almost all resolved coreferences that we inspected manually were correct). We found this approach preferable over a "full" coreference resolution for two reasons: First, considering our focus on character networks, a coreference resolution with high recall but lower precision would give rise to many spurious character co-occurrences that would harm our analyses of graph learning techniques. Second, our corpus of Tolkien's Legendarium is special in the sense that it has a large number of named references, which give rise to rich character networks despite limiting our view to named references.

Extraction of Character Co-Occurrences After finding references to characters, we next extract co-occurrences of pairs of characters that can be used to build character networks. While the co-occurrence of characters does not necessarily imply an interaction between them, due to its simplicity it is a frequently used approach to construct character networks in CLS [7, 23, 8]. After evaluating different strategies (cf. Appendix A), we decided to extract each co-occurrence of two characters in the same sentence.

Descriptive Statistics of Text Corpus In Table 1, we provide key summary statistics that characterize the different works in our corpus. Apart from differences in terms of tokens, we find striking differences between character mentions in the five texts: *The Silmarillion* uses considerably fewer pronoun mentions than the other four works, while using more explicit references by name. We attribute this to the compressed writing style of *The Silmarillion*, which rather resembles a fictional historical record that lists character names and locations, and gives a chronology of events compared to the more conventional prose style of *The Hobbit* and *The Lord of the Rings*.

Table 1
Summary statistics of the text corpus used in our analysis.

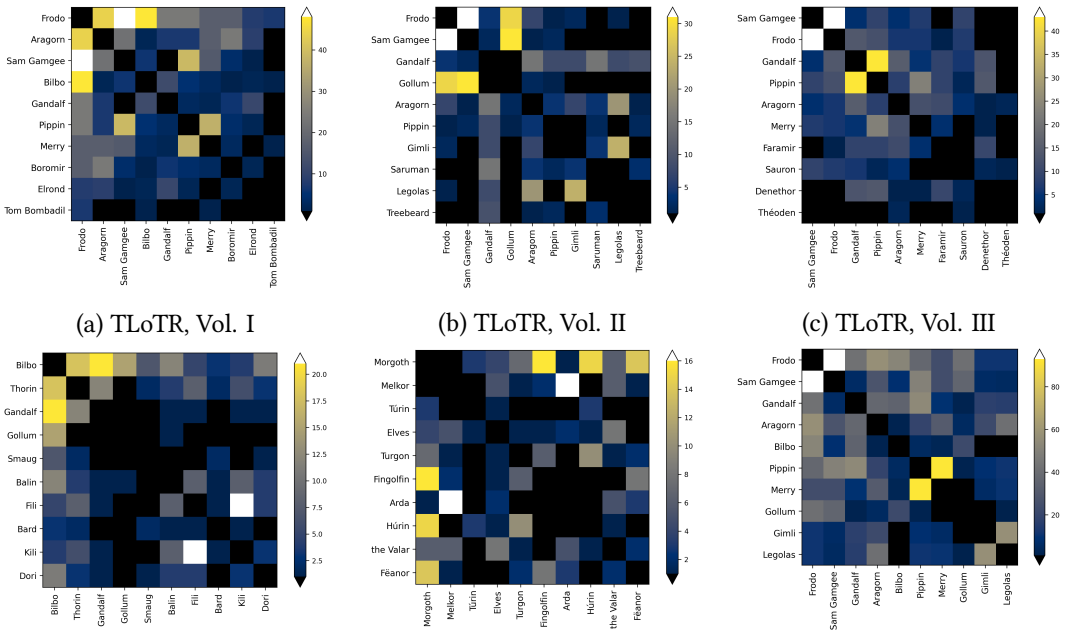
	Hobbit	TLotR Vol. I	TLotR Vol. II	TLotR Vol. III	Silmarillion
#tokens	113,235	224,156	189,539	163,191	147,833
#character mentions	15,477	31,770	27,576	24,766	26,468
% Pronouns	56.19	54.11	56.27	51.65	37.48
% Nominal mentions	27.7	22.76	22.3	25.93	29.15
% Explicit references	16.11	23.13	21.42	22.42	33.38
Character Co-occurrences	431	886	615	844	1,674

We calculate the number of character co-occurrences for each work in our corpus, which we indicate in Table 1 and visualise in Figure 1a – Figure 1f. The overall number of co-occurrences is influenced by the number of explicit references (cf. Table 1), since we only extract co-occurrences if both characters are mentioned by name in the same sentence.

Character-based Visualization of Narrative Structure We finally use the character mentions in conjunction with the chapter in which they occurred to automatically produce a narrative chart for the three volumes of *The Lord of the Rings*. To avoid occurrences of characters

that are only mentioned while not being present, we excluded mentions of characters within dialogues, as detected by BookNLP. The narrative charts for the three volumes of *The Lord of the Rings* are shown in the three panels of fig. 2. The columns within each panel represent chapters. To ease the presentation, we focus on a selected set of main characters shown in the rows. The color of each row-column cell represents the fraction of the number of times a characters is mentioned in a given chapter, relative to the number of mentions of other key characters presented in the narrative chart.

While it is easy to generate those charts, we can use them to identify major plot lines and reveal a narrative structure that is characteristic for *The Lord of the Rings*: In volume I, we see that Frodo and the Hobbits maintain a chief role throughout book I and II, i.e. both parts of volume I of *The Lord of the Rings*. A shift is visible for the second half (book II), which coincides with the Hobbits' arrival in Rivendell. In volume II, we see a clear transition in narrative structure that is due to Tolkien's adoption of an *interlacing* narrative style [51]. The first half of volume II (book III) focuses on the main characters Aragorn, Legolas, and Gimli, and their attempt to rescue Merry and Pippin from the Uruk-Hai. Leaping back in time, the second half of volume III (book IV) focuses on the journey of Frodo, Sam, Gollum and their encounter with Faramir, which coincides with a brief absence of Gollum as he hides from the rangers. Following the original titles suggested for the six books that constitute the three volumes of *The Lord of the Rings* [41], these interlaces can be called *The Treason of Isengard* and *The Journey of the Ring-bearers*. In volume III, we see a similar but less marked separation



(a) TLoTR, Vol. I (b) TLoTR, Vol. II (c) TLoTR, Vol. III

(d) *The Hobbit* (e) *The Silmarillion* (f) *Legendarium*

Figure 1: Number of co-occurrences between the ten most frequently mentioned characters in the five works of our corpus (a) – (e) and the whole Legendarium (f).

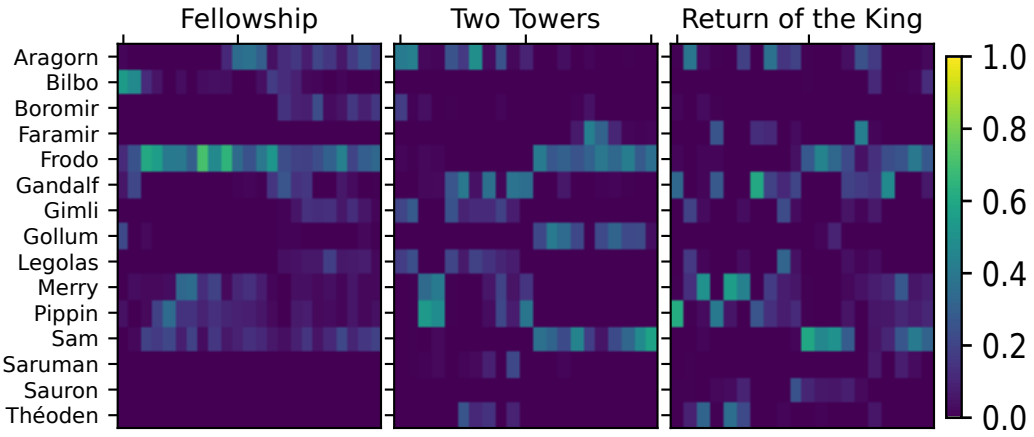


Figure 2: Narrative charts for the three volumes of *The Lord of the Rings*. Rows indicate key characters, while columns represent chapters. Colours capture the relative frequency of character mentions relative to other key characters captured in the charts. In Vol. I (left panel) Frodo and the Hobbits maintain a central role across the whole volume. For Vol. II and III (middle and right panel) clear transitions in the narrative structure are visible, which are due to an *interlacing narrative style*.

between two interlaced plot lines: The first half (book V) focuses on the War of the Ring in Gondor, while book VI continues the story of Frodo's and Sam's journey to Mount Doom, followed by the Scouring of the Shire, which explains Merry's and Saruman's reappearance in the last part. The original titles referring to those interlaces are *The War of the Ring* and *The End of the Third Age*. The non-linear narrative style expressed in the narrative charts in Figure 2 is common in early medieval literature [26]. Tolkien likely adopted it due to his familiarity with medieval literature, to increase suspense and to achieve a sense of historicity [3]. Since these charts capture the narrative structure that we would expect, we can assume that our character extraction, even though using only limited coreference resolution, captures the overall occurrences of key characters rather well. In Appendix B we include additional narrative charts for *The Hobbit* and *The Silmarillion*.

3. Analysis of Character Co-occurrence Networks

Building on Section 2 we now use character co-occurrences to construct *character networks* for J.R.R. Tolkien's Legendarium. A *graph* or *network* is defined as tuple $G = (V, E)$, where $u, v, ... \in V$ represent nodes and $(u, v) \in E \subseteq V \times V$ is a set of directed or undirected links. For our analysis, we model characters as nodes V and their co-occurrences within a sentence as undirected links E, i.e. $(u, v) \in E \implies (v, u) \in E$ for all $u, v \in V$. We further add link weights $w : E \to \mathbb{N}$, i.e. we assign a value $w((u, v))$ to each link (u, v) that counts the co-occurrences of u and v. We adopt this definition to construct character networks for each of the four works in our corpus. A particularly interesting aspect of our corpus is that all works refer to a single Legendarium, with frequent cross-reference, and thus, overlaps in terms of characters. We use this to construct a *single* network of characters across all works, which we call *Legendarium Graph*. Except for a single disconnected pair of nodes that we removed, the resulting graph has a single connected

component with 238 nodes, which enables a macroscopic analysis of Tolkien's Legendarium. Figure 3 shows a visualisation of the character network that has been generated using the force-directed layout algorithm by Fruchterman and Reingold [14]. This algorithm simulates attractive forces between connected nodes (and repulsive forces between all nodes) such that their positions in the stable state of the resulting dynamical system highlight patterns in the topology.

Table 2
Key network metrics for the character networks in our corpus, where *Legendarium* refers to the graph spanning *all* works.

	nodes	edges	density	mean degree	diam.	avg. shortest path
The Hobbit	30	119	0.14	8.0	4	1.9
The Lord of the Rings Vol. I	65	217	0.052	6.68	5	2.31
The Lord of the Rings Vol. II	55	169	0.057	6.13	5	2.4
The Lord of the Rings Vol. III	45	159	0.08	7.07	5	2.26
The Silmarillion	136	803	0.044	11.81	5	2.41
Legendarium	238	1233	0.023	10.48	6	2.84

Network Size, Density, and Mean Degree Modelling character networks enables us to compute metrics from social network analysis, which we report in Table 2. The first columns of Table 2 show the number of nodes $n = |V|$ and links $m = |E|$ and the density $\rho = \frac{m}{n(n-1)}$ of each character network, where the latter captures which fraction of possible links actually occurred. We find that *The Hobbit* and *The Silmarillion* have the smallest and largest number of nodes respectively. *The Hobbit* has a higher link density, which is likely because the plot is focused on a small number of strongly interacting characters. For *The Lord of the Rings* we see that the number of nodes for Vol. III is smaller than for the previous two volumes, while the density is larger. This reflects the fact that the last volume is strongly focused on interactions between small groups of characters, e.g. Frodo, Sam and Gollum. The degree of a node $v \in V$ is defined as the number of other nodes to which it is connected by links, i.e. $d(v) := |\{u : (u, v) \in E\}|$. The fourth column in Table 2 reports the mean node degree of characters, where larger mean degrees are associated with higher link density.

Shortest paths, Diameter and Betweenness Centrality An important feature of (social) networks is the structure of shortest paths between nodes, which provide a topological notion of pair-wise distances [33]. We calculate shortest paths between all pairs of nodes and report the average shortest path length across all character pairs. We further calculate the diameter, i.e. the maximum shortest path length between any pair of nodes. The results in Table 2 show that, on the one hand, the average shortest path length is associated with the number of nodes (smallest average shortest path length for the smallest networks *The Hobbit* and *The Lord of the Rings* Vol. III). On the other hand, it is influenced by mean degree and link density, which explains why *The Lord of the Rings* Vol. III and *The Silmarillion* have similar values despite the latter having more than twice as many nodes. The shortest paths between characters allows us

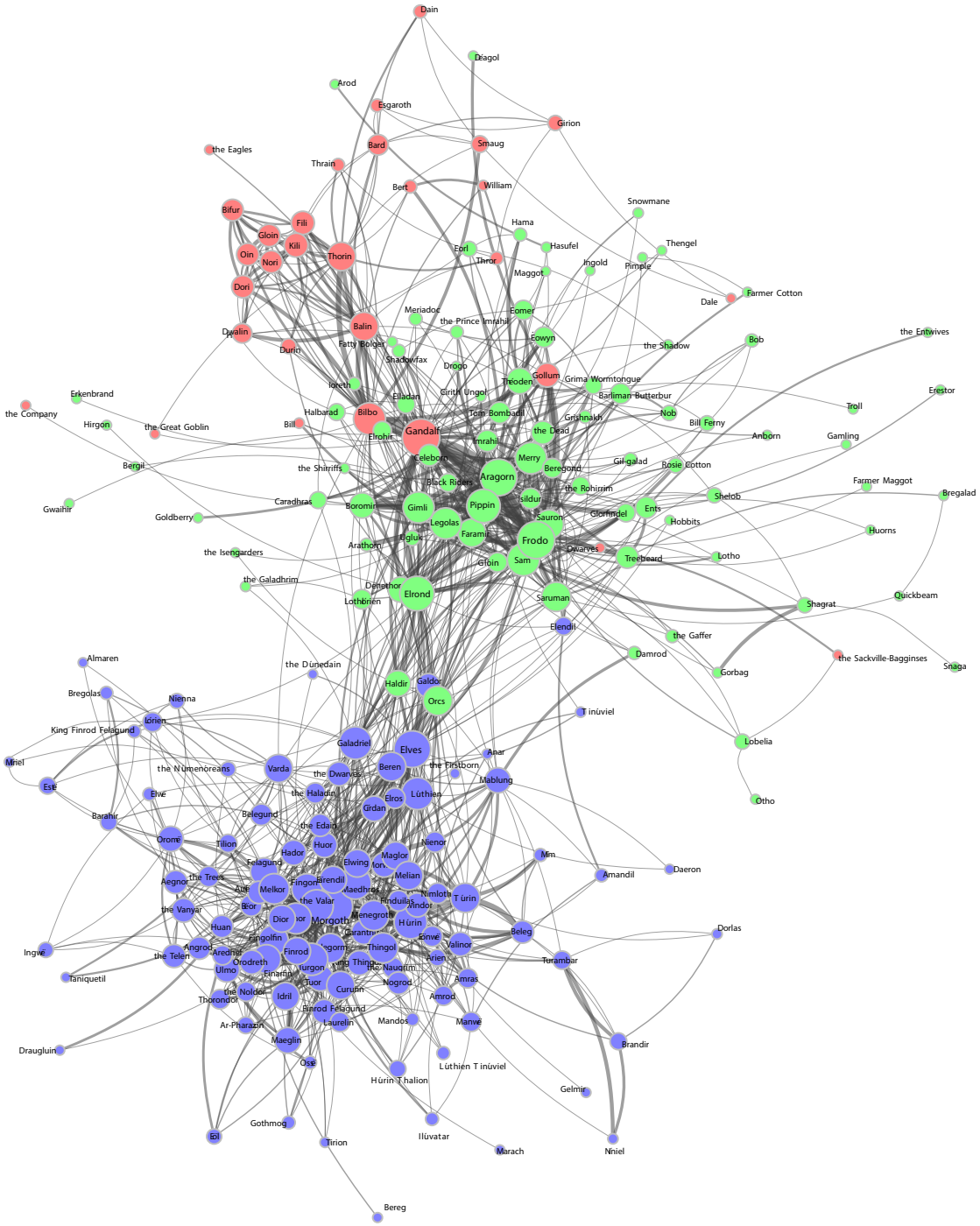


Figure 3: Force-directed visualisation of automatically extracted character network for Tolkien's Legendarium with 238 nodes and 1233 links. Node colours indicate the works in which a character occurs most frequently (red: *The Hobbit*, green: *The Lord of the Rings*, blue: *The Silmarillion*). Node sizes are logarithmically scaled based on node degrees, edge thickness are logarithmically scaled based on edge weights, i.e. number of character co-occurrences. Visualisation was created using the Open Source network analysis and visualisation library `pathpy` [18]. This character networks facilitates a macroscopic, graph-based analysis that considers intertextual aspects in J.R.R. Tolkien's Legendarium.

Table 3
Ranking of characters in Tolkien's *Legendarium* based on degree and betweenness centrality.

	degree	betweennes
1	Frodo	Elves
2	Sam	Frodo
3	Gandalf	Gandalf
4	Aragorn	Bilbo
5	Pippin	Galadriel
6	Merry	Aragorn
7	Morgoth	Morgoth
8	Bilbo	Elrond
9	Legolas	Lúthien
10	Gimli	Orcs
11	Elrond	Sam
12	Gollum	Saruman
13	Elves	Pippin
14	Túrin	Merry
15	Húrin	Húrin

to define a path-based notion of centrality. The *betweenness centrality* [50] of node v captures the number of shortest paths between any pair of nodes that pass through v, i.e. nodes have higher betweenness centrality if they are important for the "communication" between other nodes. The second column of table 3 reports characters with the highest betweenness centrality for the single *Legendarium* graph. We note the high betweenness centrality *Galadriel*, who despite an ephemeral appearance in *The Lord of the Rings* and absence in *The Hobbit* is one of few characters that link the narrative across different ages in Tolkien's mythology.

4. Application of Graph Neural Networks

We now turn our attention to the application of state-of-the-art graph learning techniques to the character network of Tolkien's Legendarium, which has been introduced and characterised in the previous section. Our analysis is focused on our guiding research question outlined in Section 1, i.e. what additional information we can draw from the topology of the character network, compared to an application of standard machine learning to a word embedding technique. A major hurdle for the application of machine learning to character networks is that standard techniques like, e.g., logistic regression, support vector machines, or neural networks require features in a continuous vector space. Their application to discrete objects like graphs typically requires a two-step procedure that consists of (i) a representation learning or embedding step to extract vectorial features of nodes and/or links, and (ii) a downstream application of machine learning to those features. This approach is limited by the fact that graphs with complex topologies are fundamentally non-Euclidean objects [9], which limits our ability to find a generic vector space representation that is suitable for different learning tasks.

Addressing this limitations, recent works in the field of *Geometric Deep Learning* [9] and

Graph Learning have generalized deep learning to graph-structured data. Among those works, Graph Neural Networks (GNNs) [43, 16, 39, 15] have developed into a particularly successful paradigm. A major advantage of GNNs over other network analysis or machine learning techniques is their ability to capture both relational patterns in the topology of a graph as well as additional (vectorial) features of nodes or links. A common concept of GNNs is their use of hidden layers with *neural message passing*, i.e. nodes repeatedly exchange and update feature vectors with neighbouring nodes thus incorporating information from their neighbourhood. The (hidden) features generated in this way can be used to address learning tasks by means of a perceptron model with a non-linear activation function. The gradient-based optimization of GNNs can be thought of as an implicit way to generate a topology- and feature-aware latent space representation of a graph that facilitates node-, link- or graph-level learning tasks [19].

Moving beyond the social network analysis techniques applied in Section 3, in the following we apply two state-of-the-art GNN architectures to the character network of Tolkien's Legendarium. We use them to address three unsupervised and supervised learning tasks: (i) representation learning, i.e. finding a meaningful latent space embedding of characters, (ii) (semi-)supervised node classification, i.e. assigning characters to different works in the Legendarium, and (iii) link prediction, i.e. using a subset of links in the graph to predict missing links in a holdout set.

Latent Space Representation of Characters in Tolkien's Legendarium Representation learning is a common challenge both in natural language processing and graph learning. In NLP, word or text embedding techniques are commonly used to generate vector space representations that can then be used to apply downstream machine learning techniques to literary texts. A popular word embedding technique is `word2vec` [31, 30], which we use as a baseline in our analysis that only uses the text corpus while being agnostic to the topology of the character network. We specifically use the `SkipGram` architecture to train a neural network with a single hidden layer with d neurons that captures the context of words in the text corpus, i.e. a concatenation of all works in our corpus. The weights of neurons in the hidden layer are then interpreted as positions of words in a d-dimensional latent vector space. For our analysis, we used the `word2vec` implementation in the package `gensim` with default parameters, i.e. we use a latent space with $d = 300$ dimensions.

Apart from this standard NLP approach to generate latent space embeddings, we apply two graph representation learning techniques to generate character embeddings based on the topology of the character network: The first approach, Laplacian Eigenmaps, uses eigenvectors corresponding to the leading eigenvalues of the Laplacian matrix of a graph [6]. This can be seen as a factorization of an eigenmatrix [37] that yields a representation of nodes in a d-dimensional latent space, where d is chosen to be smaller than the number of nodes. To determine a reasonable choice for the number of dimensions d, we performed an experiment in which we evaluated the average performance of a logistic regression model for node classification (see detailed description below) for different dimensions d of the latent space. The results of this experiment are shown in Figure 4. As expected, we observe tendency to underfit for a very small ($d < 10$) number of dimensions, while the performance saturates as we increase d beyond a "reasonable" value that allows to capture the topological patterns in the network. This

analysis informed our choice of $d = 20$ for the subsequent experiments. As second approach we adopt the popular graph representation learning method `node2vec`, which applies the `Skip-Gram` architecture to sequences of nodes generated by a biased second-order random walk (i.e. a walk with one-step memory) in a graph [17]. We again use $d = 20$ hidden dimensions to make it comparable with the previous method. We note that choosing a value of $d = 20$ for the graph representation learning techniques, which is substantially smaller than the default value of $d = 300$ for `word2vec`, is justified because the number of nodes in the Legendarium graph ($n = 238$) is much smaller than the vocabulary used by `word2vec` ($n = 18,430$).

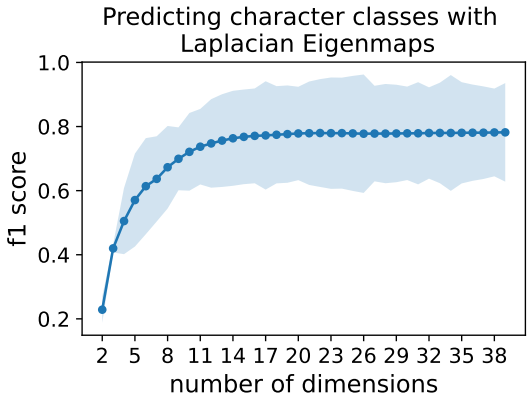


Figure 4: Performance of node classification (y-axis) for a Laplacian Eigenmap graph embedding of characters in Tolkien's Legendarium (with subsequent logistic regression) using different numbers of dimensions (x-axis).The shaded area indicates the standard deviation of the f1-score.

We finally adopt two Graph Neural Networks, namely Graph Convolutional Networks (GCNs) [21] and Graph Attention Networks (GATs) [49]. Both of these deep graph learning techniques use a variant of neural message passing, the main difference being that GATs use a learnable attention mechanism that can place different weights on nearby nodes [57]. We trained both architectures to address the graph learning tasks, i.e. node classification and link prediction, outlined below, using a hidden layer with $d = 20$ neurons. We then use the hidden layer activations of both architectures to infer a representation of characters in a 20-dimensional latent space. To exclusively focus on the graph *topology*, for our experiments we treated the network as an unweighted graph. Additional results for experiments with weighted graphs are included in Appendix D.

Leveraging the ability of GNNs to consider additional node attributes, we compare three different approaches: First, we use GNNs without additional node features, which we emulate by initializing the message passing layer with a one-hot-encoding (OHE) of characters. In other words, for a network with n nodes, each node $i = 0, \dots, n-1$ we assign a "dummy feature" vector $f_i \in \mathbb{R}^n$ defined as:

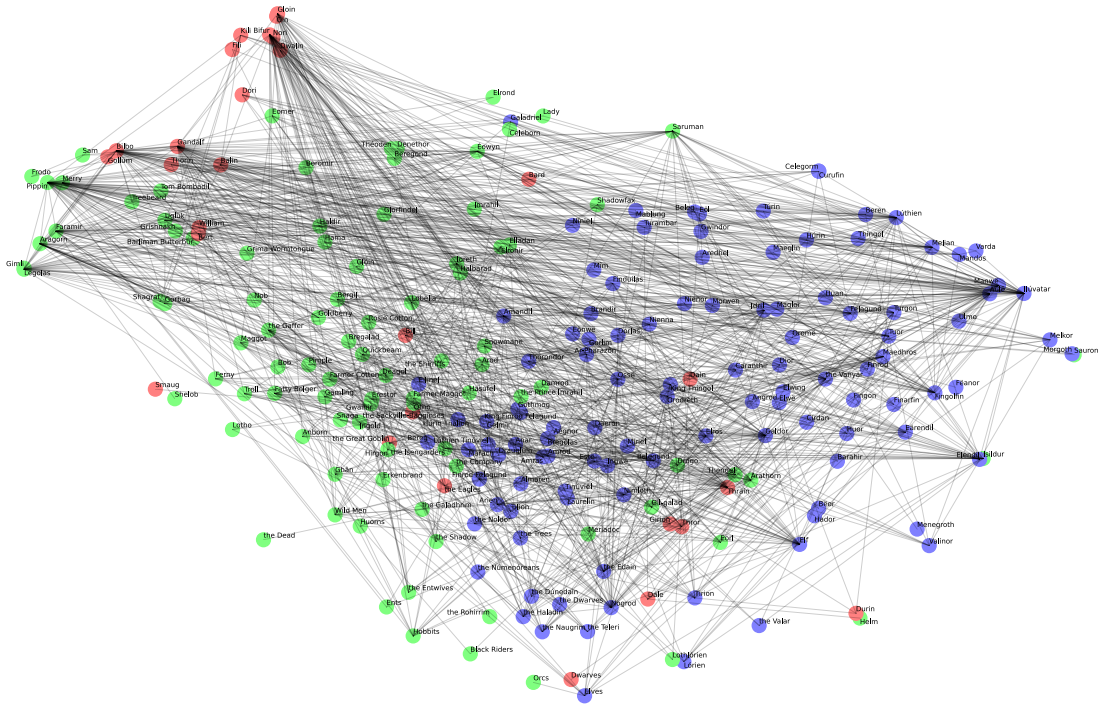
$$f_i = (\underbrace{0, \dots, 0}_{i \text{ times}}, 1, \underbrace{0, \dots, 0}_{n-i-1 \text{ times}})$$

Second, we use the node embeddings generated by `node2vec` as additional node features that are used in the message passing layers. Third, we assign the `word2vec` as additional node

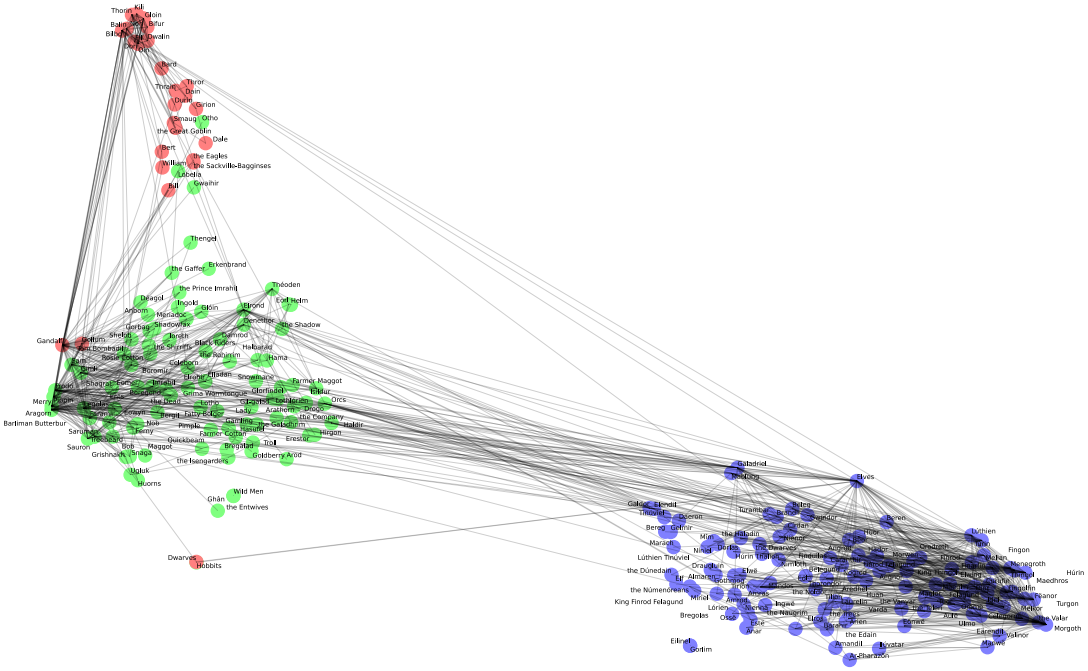
features thus combining NLP and graph neural networks. In Figure 5 we illustrate two latent space representations of characters generated by (a) `word2vec` and (b) the combination of GCN with `word2vec` features. For both figures, we used t-SNE [48] to reduce the latent space embedding to two dimensions, nodes are coloured according to the works in which the corresponding characters occur most frequently. A comparison of the two embeddings clearly highlights the advantage of graph learning over a mere application of `word2vec`: Different from the word embedding, the combination of GCN with `word2vec` generates a latent space representation that captures the distinction of characters across different works in Tolkien's Legendarium. We argue that this visualisation highlights the additional information that graph neural networks can leverage from the topology of character networks, as opposed to mere word-context pair statistics. In the following, we more thoroughly investigate this interesting aspect in two clearly defined learning tasks.

Predicting Character Classes We use the methods outlined above to address a supervised node classification problem in the *Legendarium Graph*, i.e. the single character network capturing all works in Tolkien's Legendarium. We assign three labels to nodes that correspond to the work (i.e. *The Silmarillion*, *The Hobbit*, and *The Lord of the Rings*), in which the corresponding character is most prominent. We extract these labels automatically as $\text{argmax}_{book} \text{count}(c,book)/\sum_{c \in C} \text{count}(c,book)$, where $\text{count}(c, book)$ is the number of mentions of character c in *book* and C are all characters. We verify the labels manually, finding them to be reasonable in all cases. The resulting three classes contain 113 (*The Silmarillion*), 30 (*The Hobbit*) and 99 (*The Lord of the Rings*) characters, i.e. there is class imbalance that we address by using the macro-averaged f1-score, precision and recall. We highlight that the information on the different works is withheld from all methods, i.e. for the word embedding and character network construction we concatenate all works to a single text corpus. We train our models on a training set of labelled characters and use them to predict the unknown classes of unlabelled characters in a test data set. As a baseline that does not utilise the topology of the character network, we first train a logistic regression model using the embeddings generated by `word2vec`. Similarly, we train a logistic regression model on the embeddings generated by Laplacian Eigenmaps and `node2vec`. For `node2vec` we use three different sets of hyper-parameters p and w, where for $p = q = 1$ `node2vec` is equivalent to the graph embedding technique DeepWalk [34]. We trained the model for 200 epochs. We finally train the GCN and GAT model either using one-hot encoding (OHE) or assigning additional node features generated by the word and graph embeddings as explained above. For both we used two message passing layers and we trained the models for 5000 epochs using an Adam optimizer with learning rate $lr = 0.0001$. We evaluate all models using a 10−fold cross-validation. Average results with standard deviation are shown in Table 4.

Interestingly, we find that the performance of character classification based on the word embedding `word2vec` is worse than *any* of the graph learning techniques, thus highlighting the added value of the graph perspective. The graph embedding technique `node2vec`, which uses a SkipGram architecture to embed nodes, clearly outperforms the graph-agnostic `word2vec`, despite both using the same logistic regression model for the class prediction. Moreover, we find that a simple application of Laplacian Eigenmaps, i.e. a mathematically principled matrix



(a) Latent space embedding of characters obtained by applying the word embedding `word2vec` to the whole text corpus; edges represent character co-occurrences in the text.



(b) Graph Convolutional Network using `word2vec` character embeddings as additional node features

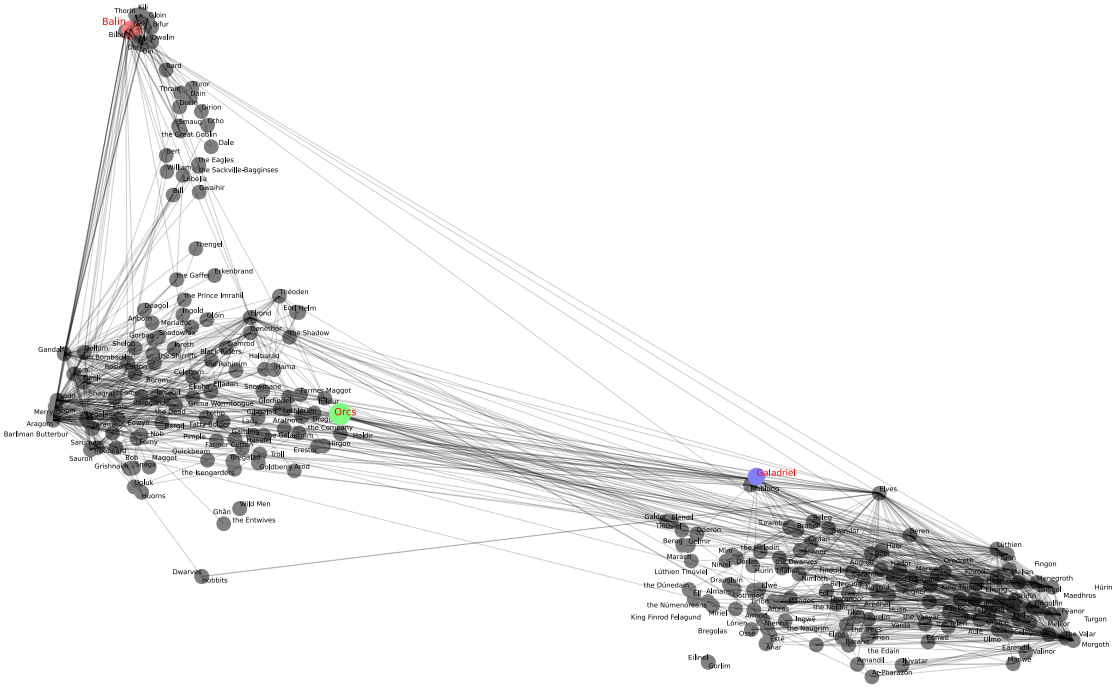
Figure 5: Comparison of two latent space embeddings of characters in Tolkien's Legendarium using `word2vec` (a) and the hidden layer activations of a Graph Convolutional Network, where nodes carry additional features that correspond to `word2vec` embeddings (b). Node colours indicate the work in which characters appear most frequently (red: *The Hobbit*, green: *The Lord of the Rings*, blue: *The Silmarillion*). Two-dimensional visualisations were generated using the dimensionality reduction technique t-SNE [48].

Table 4
Performance of word embedding (word2vec), graph representation learning (Laplacian Eigenmap and node2vec), and graph neural networks (GCN and GAT) in supervised character classification and link prediction for a character network capturing Tolkien's Legendarium. For the word and graph embedding techniques, a logistic regression model was used for classification and link prediction.

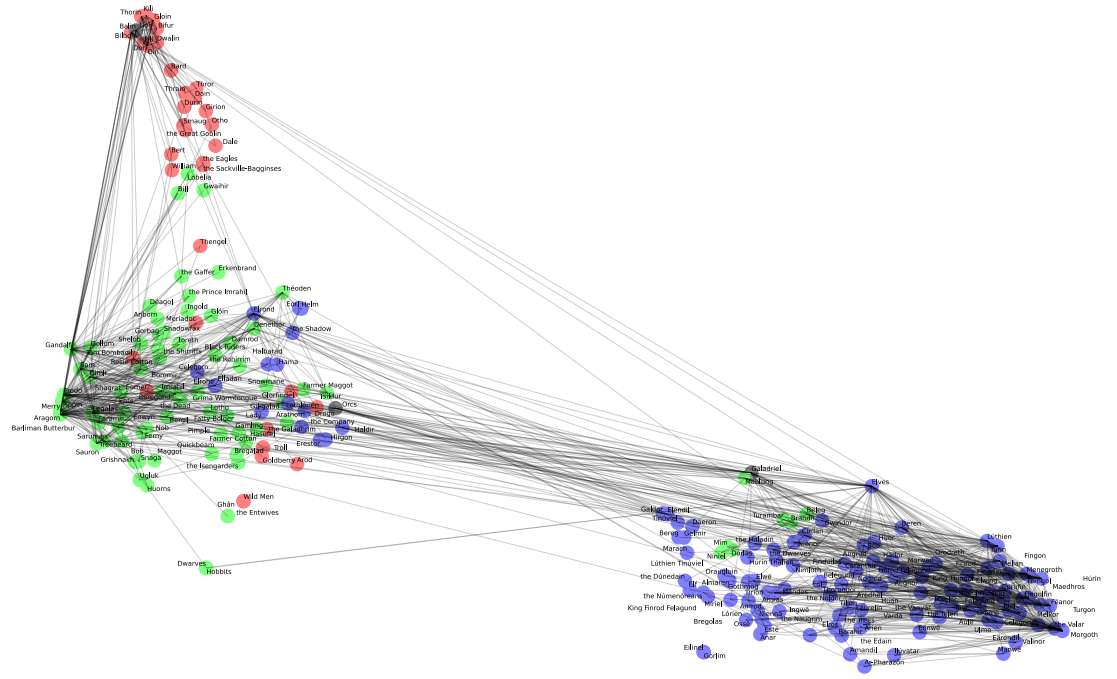
	Character classification			**Link Prediction**
	F1-score	Precision	Recall	ROC/AUC
word2vec^{300}	79.45 ± 0.12	80.67 ± 0.13	80.08 ± 0.12	78.16 ± 0.79
Laplacian Eigenmap^{20} (LE)	81.52 ± 0.14	86.59 ± 0.09	84.13 ± 0.14	64.03 ± 2.28
node2vec$^{20}{}_{p=1,q=4}$	82.10 ± 0.12	84.54 ± 0.12	84.85 ± 0.11	79.06 ± 0.01
node2vec$^{20}{}_{p=4,q=1}$	81.75 ± 0.12	81.69 ± 0.12	84.51 ± 0.12	80.71 ± 0.01
node2vec$^{20}{}_{p=1,q=1}$	87.07 ± 0.13	89.10 ± 0.12	87.30 ± 0.13	80.15 ± 0.02
GCN$^{20}{}_{\text{LE}}$	71.89 ± 0.20	72.40 ± 0.22	73.24 ± 0.18	**88.53 ± 1.25**
GCN$^{20}{}_{\text{node2vec}_{p=1,q=1}}$	92.17 ± 0.10	**96.69 ± 0.04**	91.08 ± 0.11	88.03 ± 1.43
GCN$^{20}{}_{\text{OHE}}$	90.55 ± 0.13	92.71 ± 0.12	89.95 ± 0.13	80.78 ± 1.16
GCN$^{20}{}_{\text{word2vec}}$	**92.32 ± 0.10**	95.27 ± 0.08	**91.42 ± 0.10**	81.67 ± 1.43
GAT$^{20}{}_{\text{LE}}$	53.60 ± 0.08	51.72 ± 0.05	57.78 ± 0.08	79.41 ± 1.83
GAT$^{20}{}_{\text{node2vec}_{p=1,q=1}}$	91.18 ± 0.09	95.57 ± 0.04	90.37 ± 0.10	86.23 ± 0.99
GAT$^{20}{}_{\text{OHE}}$	83.52 ± 0.14	85.59 ± 0.15	84.68 ± 0.14	80.52 ± 1.76
GAT$^{20}{}_{\text{word2vec}}$	89.41 ± 0.07	93.98 ± 0.08	89.13 ± 0.10	82.88 ± 2.08

decomposition, yields comparable node classification performance than the neural network-based node2vec embedding. The best precision is achieved for a Graph Convolutional Network (GCN) with additional node features generated by node2vec, while the best f1-score and recall are achieved for GCN with additional word2vec embeddings. We attribute this to the fact that the combination of word embeddings and graph neural networks integrates two complementary sources of information, thus highlighting the advantages of methods that leverage both NLP and graph learning.

In Figure 6 we further demonstrate the ability of Graph Neural Networks to perform a semi-supervised classification, i.e. their ability to accurately predict classes based on a very small number of labelled examples. Figure 6(a) shows the training network with three randomly coloured characters, one for each ground-truth class. Nodes in the test set are shown in grey and node positions are chosen based on the latent space representation shown in Figure 5(b). We use these three labelled characters to train a GCN with additional word2vec embeddings as node features. We then use the trained model to predict the character classes in the test set. The predicted classes are shown as coloured nodes in Figure 6(b), where the three training nodes are shown in grey. A visual comparison of Figure 6(b) and 5 (b) allows to evaluate the prediction against the ground truth. Despite the very sparse labelled examples, and thanks to its use of the graph topology of the unlabelled nodes in the training set, the GCN model is able to accurately predict character classes, reaching an f1-score of $\approx$ 79.7%, a precision of $\approx$ 78.6% and a recall of $\approx$ 82.6%. Remarkably, this shows that the combination of a GCN model with word2vec node features yields a higher f1-score and recall with only three labelled examples than a word embedding alone, even when all characters in the training set are labelled.



(a) Training network with three labelled characters *Galadriel*, *Orcs*, and *Balin* (shown as coloured nodes).



(b) Character classes predicted by Graph Convolutional Network (GCN) using `word2vec` character embeddings as additional node features.

Figure 6: Illustration of semi-supervised character classification, where the training set contains only three randomly chosen labelled characters. Node colours indicate ground truth classes (a) and class predictions by Graph Convolutional Network (GCN) (b) (red: *The Hobbit*, green: *The Lord of the Rings*, blue: *The Silmarillion*). Two-dimensional visualisations were generated using the dimensionality reduction technique t-SNE [48]. Node positions correspond to Figure 5 (b), which can be used for a comparison with ground truth character classes. Remarkably, a GCN model with additional `word2vec` features achieves an f1-score of $\approx 79.7\%$ with an associated precision of $\approx 78.6\%$ and recall of $\approx 82.6\%$.

Predicting Character Interactions We finally address link prediction, which refers to the task of predicting "missing" links in a graph, i.e. links that are either "missing" in incomplete data or that likely form in the future. Link prediction is a well-studied graph learning problem, with important applications in social network analysis and recommender systems [29]. In the context of character networks, it is relevant because it could be used to alleviate the low recall of the rigid, sentence-based character network extraction that we employed in Section 3.

Adopting a supervised approach, we split the edges of the *Legendarium Graph* in a training and set set, where we withhold 10 % of the edges during training to test our model. We use `word2vec`, Laplacian Eigenmaps and `node2vec` to generate embeddings $f_v \in \mathbb{R}^d$ of characters. For `word2vec` embeddings are generated using the full text corpus. For the graph embedding techniques Laplacian Eigenmaps and `node2vec` we only use links in the training set, potentially putting them at a disadvantage in terms of training data. We use the resulting feature vectors to calculate the element-wise (Hadamard) product $f_v \circ f_w \in \mathbb{R}^d$ for character pairs v, w, which yields d-dimensional features for all character pairs. We then use features of positive instances (i.e. pairs v, w connected by a link (v, w) in the training set) and negative instances (pairs v, w not connected by a link in the training set) to train a (binary) logistic regression classifier and use the trained model to predict links in the test set. We use negative sampling to mitigate the imbalance between negative and positive instances in the training set. For the two GNN architectures we adopt the common approach to add a *decoding* step that computes the Hadamard product of node features after the last message passing layer. We use a Binary Cross Entropy with Logits Loss function and train the models for 15000 epochs using an Adam Optimizer with learning rate 0.001.

We use the Receiver-Operating Characteristic (ROC) to evaluate our models, i.e. we compute ROC curves that give the true and false positive rates across all discrimination thresholds. An example (and explanation) is included in Appendix E. We compute the Area Under Curve (AUC) of ROC curves within the unit square, which range from 0 to 1. 0.5 corresponds to the performance of a random classifier, values < 0.5 indicate worse and values > 0.5 better than random performance. We again evaluate all models using a 10−fold cross-validation. Table 4 reports average results and standard deviations of the AUC for all models. With the exception of Laplacian Eigenmaps, we find that graph methods generally perform better than `word2vec`. We further observe that GNNs perform considerably better than `node2vec`, where the best performance is achieved when coupling GCN with Laplacian Eigenmaps or `node2vec`.

5. Conclusion

In summary, we used natural language processing techniques like named entity recognition and coreference resolution to construct a single character network from a corpus of works that constitute J.R.R. Tolkien's Legendarium. Apart from characterising the network based on social network analysis, we adopt state-of-the-art graph learning techniques to (i) generate latent space embeddings of characters, (ii) automatically classify characters based on the work to which they belong, and (iii) predict character co-occurrences. For all three tasks, we find a significant advantage of Graph Neural Networks (GNNs) over a common word embedding technique and we find that a combination of both yields the best performance. Our approach

to construct a single graph for multiple literary texts could be interesting to analyze other corpora of works with overlapping characters (e.g. mythology, historical novels, etc.). We further believe that our results on the application of GNNs to address a link prediction task have interesting implications for computational literary studies. Considering the difficulty of coreference resolution, and the low coverage of the resulting character networks that we observed in our experiments, we expect that link prediction could potentially be used as an approach to address the low recall observed for the sentence-based co-occurrence networks. In future work, we will further consider the modelling of character co-occurrences as *temporal character co-occurrence networks*, where the temporal ordering of sentences determines the "time stamps" of edges in the resulting dynamic graph. This promises the application of recently developed higher-order graph modelling, visualization, and learning techniques [24, 35, 46, 25, 36], which capture patterns in the temporal ordering of edges and can thus provide insights beyond the mere graph topology.

Acknowledgements

Vincenzo Perri and Ingo Scholtes acknowledge support by the Swiss National Science Foundation, grant 176938.

References

[1] A. Agarwal, A. Corvalan, J. Jensen, and O. Rambow. "Social network analysis of alice in wonderland". In: *Proceedings of the NAACL-HLT 2012 Workshop on computational linguistics for literature*. 2012, pp. 88–96.

[2] D. Agarwal, D. Vijay, et al. "Genre Classification using Character Networks". In: *2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS)*. Ieee. 2021, pp. 216–222.

[3] E. E. Auger. "The Lord of the Rings' interlace: Tolkien's narrative and Lee's illustrations". In: *Journal of the Fantastic in the Arts* 19.1 (2008), p. 70.

[4] D. Bamman, O. Lewke, and A. Mansoor. "An annotated dataset of coreference in English literature". In: *arXiv preprint arXiv:1912.01140* (2019).

[5] D. Bamman, S. Popat, and S. Shen. "An annotated dataset of literary entities". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 2138–2144.

[6] M. Belkin and P. Niyogi. "Laplacian eigenmaps for dimensionality reduction and data representation". In: *Neural computation* 15.6 (2003), pp. 1373–1396.

[7] A. Beveridge and J. Shan. "Network of thrones". In: *Math Horizons* 23.4 (2016), pp. 18–22.

[8] A. Bonato, D. R. D'Angelo, E. R. Elenberg, D. F. Gleich, and Y. Hou. "Mining and modeling character networks". In: *International workshop on algorithms and models for the web-graph*. Springer. 2016, pp. 100–114.

[9] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. “Geometric deep learning: going beyond euclidean data”. In: *IEEE Signal Processing Magazine* 34.4 (2017), pp. 18–42.

[10] J. Bruna and X. Li. “Community detection with graph neural networks”. In: *stat* 1050 (2017), p. 27.

[11] D. K. Elson, K. McKeown, and N. J. Dames. “Extracting social networks from literary fiction”. In: (2010).

[12] R. Feldman. “Techniques and applications for sentiment analysis”. In: *Communications of the ACM* 56.4 (2013), pp. 82–89.

[13] M. Fey and J. E. Lenssen. “Fast graph representation learning with PyTorch Geometric”. In: *arXiv preprint arXiv:1903.02428* (2019).

[14] T. M. Fruchterman and E. M. Reingold. “Graph drawing by force-directed placement”. In: *Software: Practice and experience* 21.11 (1991), pp. 1129–1164.

[15] C. Gallicchio and A. Micheli. “Graph Echo State Networks”. In: 2010, pp. 1–8. DOI: 10.1109/ijcnn.2010.5596796.

[16] M. Gori, G. Monfardini, and F. Scarselli. “A new model for learning in graph domains”. In: *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* 2 (2005), 729–734 vol. 2.

[17] A. Grover and J. Leskovec. “node2vec: Scalable feature learning for networks”. In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 2016, pp. 855–864.

[18] J. Hackl, I. Scholtes, L. V. Petrović, V. Perri, L. Verginer, and C. Gote. “Analysis and Visualisation of Time Series Data on Networks with Pathpy”. In: *Companion Proceedings of the Web Conference 2021*. Www ’21. Ljubljana, Slovenia: Association for Computing Machinery, 2021, pp. 530–532. DOI: 10.1145/3442442.3452052. URL: https://doi.org/10.1145/3442442.3452052.

[19] W. L. Hamilton. *Graph Representation Learning*. Vol. 14. 3. Morgan & Claypool Publishers, 2020, pp. 1–159.

[20] M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd. “spaCy: Industrial-strength Natural Language Processing in Python”. In: (2020). DOI: 10.5281/zenodo.1212303.

[21] T. N. Kipf and M. Welling. “Semi-Supervised Classification with Graph Convolutional Networks”. In: *Proceedings of the 5th International Conference on Learning Representations* (Iclr). Iclr ’17. Palais des Congrès Neptune, Toulon, France, 2017. URL: https://openreview.net/forum?id=SJU4ayYgl.

[22] M. Krug. “Techniques for the Automatic Extraction of Character Networks in German Historic Novels”. doctoralthesis. Universität Würzburg, 2020. DOI: 10.25972/opus-20918.

[23] V. Labatut and X. Bost. “Extraction and analysis of fictional character networks: A survey”. In: *ACM Computing Surveys (CSUR)* 52.5 (2019), pp. 1–40.

[24] R. Lambiotte, M. Rosvall, and I. Scholtes. “From networks to optimal higher-order models of complex systems”. In: *Nature physics* 15.4 (2019), pp. 313–320.

[25] T. LaRock, I. Scholtes, and T. Eliassi-Rad. “Sequential motifs in observed walks”. In: *Journal of Complex Networks* 10.5 (2022), cnac036.

[26] J. Leyerle. “The interlace structure of Beowulf”. In: *University of Toronto Quarterly* 37.1 (1967), pp. 1–17.

[27] J. Li, C. Zhang, H. Tan, and C. Li. “Complex Networks of Characters in Fictional Novels”. In: *2019 IEEE/ACIS 18th International Conference on Computer and Information Science (ICIS)*. 2019, pp. 417–420. DOI: 10.1109/icis46139.2019.8940174.

[28] J. Li, A. Sun, J. Han, and C. Li. “A survey on deep learning for named entity recognition”. In: *IEEE Transactions on Knowledge and Data Engineering* 34.1 (2020), pp. 50–70.

[29] L. Lü and T. Zhou. “Link prediction in complex networks: A survey”. In: *Physica A: statistical mechanics and its applications* 390.6 (2011), pp. 1150–1170.

[30] T. Mikolov, K. Chen, G. Corrado, and J. Dean. *Efficient Estimation of Word Representations in Vector Space*. 2013. URL: http://arxiv.org/abs/1301.3781.

[31] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. *Distributed Representations of Words and Phrases and their Compositionality*. 2013. URL: http://arxiv.org/abs/1310.4546.

[32] F. Moretti. *Network theory, plot analysis. Literary Lab Pamphlet 2*. 2011.

[33] M. Newman. *Networks*. Oxford university press, 2018.

[34] B. Perozzi, R. Al-Rfou, and S. Skiena. “Deepwalk: Online learning of social representations”. In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2014, pp. 701–710.

[35] V. Perri and I. Scholtes. “HOTVis: Higher-order time-aware visualisation of dynamic graphs”. In: *International Symposium on Graph Drawing and Network Visualization*. Springer. 2020, pp. 99–114.

[36] L. Qarkaxhija, V. Perri, and I. Scholtes. “De Bruijn goes Neural: Causality-Aware Graph Neural Networks for Time Series Data on Dynamic Graphs”. In: *arXiv preprint arXiv:2209.08311* (2022).

[37] J. Qiu, Y. Dong, H. Ma, J. Li, C. Wang, K. Wang, and J. Tang. “Netsmf: Large-scale network embedding as sparse matrix factorization”. In: *The World Wide Web Conference*. 2019, pp. 1509–1520.

[38] M. Ribeiro, R. Vosgerau, M. Andruchiw, and S. Pinto. “The complex social network from The Lord of The Rings”. In: *Revista Brasileira de Ensino de Física* (2016).

[39] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. “The Graph Neural Network Model”. In: *Trans. Neur. Netw.* 20.1 (2009), pp. 61–80. DOI: 10.1109/tnn.2008.2005605. URL: https://doi.org/10.1109/TNN.2008.2005605.

[40] F. Schröder, H. O. Hatzel, and C. Biemann. “Neural end-to-end coreference resolution for German in different domains”. In: *Proceedings of the 17th Conference on Natural Language Processing (KONVENS 2021)*. 2021, pp. 170–181. URL: https://aclanthology.org/2021.konvens-1.15/.

[41] T. Shippey. *The road to Middle-earth: how JRR Tolkien created a new mythology*. Hmh, 2014.

[42] A. C. Sparavigna. “On social networks in plays and novels”. In: *International Journal of Sciences* 2.10 (2013).

[43] A. Sperduti and A. Starita. “Supervised Neural Networks for the Classification of Structures”. In: *Trans. Neur. Netw.* 8.3 (1997), pp. 714–735. DOI: 10.1109/72.572108. URL: https://doi.org/10.1109/72.572108.

[44] R. Sukthanker, S. Poria, E. Cambria, and R. Thirunavukarasu. “Anaphora and coreference resolution: A review”. In: *Information Fusion* 59 (2020), pp. 139–162.

[45] R. Sukthanker, S. Poria, E. Cambria, and R. Thirunavukarasu. “Anaphora and coreference resolution: A review”. In: *Information Fusion* 59 (2020), pp. 139–162. DOI: https://doi.org/10.1016/j.inffus.2020.01.010.

[46] L. Torres, A. S. Blevins, D. Bassett, and T. Eliassi-Rad. “The why, how, and when of representations for complex systems”. In: *SIAM Review* 63.3 (2021), pp. 435–485.

[47] P. Trilcke. “Social network analysis (SNA) als Methode einer textempirischen Literaturwissenschaft”. In: *Empirie in der Literaturwissenschaft*. Brill mentis, 2013, pp. 201–247.

[48] L. Van der Maaten and G. Hinton. “Visualizing data using t-SNE.” In: *Journal of machine learning research* 9.11 (2008).

[49] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. “Graph Attention Networks”. In: *6th International Conference on Learning Representations* (2017).

[50] S. Wasserman, K. Faust, et al. “Social network analysis: Methods and applications”. In: (1994).

[51] R. C. West. “The Interlace Structure of The Lord of the Rings”. In: *A Tolkien Compass* 2 (1975), pp. 75–91.

[52] J. Worsham and J. Kalita. “Genre Identification and the Compositional Effect of Genre in Literature”. In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, 2018, pp. 1963–1973. URL: https://aclanthology.org/C18-1167.

[53] M. C. Yavuz. “Analyses of character networks in dramatic works by using graphs”. In: *2020 7th International Conference on Behavioural and Social Computing (BESC)*. Ieee. 2020, pp. 1–4.

[54] A. Zehe, L. Konle, L. K. Dümpelmann, E. Gius, A. Hotho, F. Jannidis, L. Kaufmann, M. Krug, F. Puppe, N. Reiter, A. Schreiber, and N. Wiedmer. "Detecting Scenes in Fiction: A new Segmentation Task". In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Online: Association for Computational Linguistics, 2021, pp. 3167–3177. DOI: 10.18653/v1/2021.eacl-main.276. URL: https://aclanthology.org/2021.eacl-main.276.

[55] M. Zhang and Y. Chen. "Link prediction based on graph neural networks". In: *Advances in neural information processing systems* 31 (2018).

[56] M. Zhang, Z. Cui, M. Neumann, and Y. Chen. "An End-to-End Deep Learning Architecture for Graph Classification". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*. Aaai'18/iaai'18/eaai'18. New Orleans, Louisiana, USA: AAAI Press, 2018.

[57] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. "Graph neural networks: A review of methods and applications". In: *AI Open* 1 (2020), pp. 57–81.

A. Alternative Approaches to Detect Character Co-Occurrences

As an alternative to the sentence-based co-occurrence approach described in Section 2, we also evaluated two other approaches in initial experiments: detection based on (i) the parse tree and (ii) a sliding text window. For (i), we marked all sentences as an "interaction" between two characters if they had the same head word in the parse tree. This is the most strict version of our character network construction, as it only captures explicit interactions (e.g., "Frodo saw Sam"). Due to the small number of detected interactions, we discarded this strategy. On the other hand, (ii) is more lenient than our final sentence-based approach, only requiring two characters to be mentioned within a window of a fixed number of characters (letters). This approach of extracting character co-occurrences within a sliding text window has been adopted in a number of prior works [7, 8]. It allows to capture interactions between characters that are not mentioned in the same sentence, but introduces the risk of detecting a large number of spurious interactions. In our experiments, we chose a window size of 2000 characters, with the additional restriction that chapter borders may not be crossed. Aiming for character networks that maintain a balance between recall (i.e. detecting all meaningful character links) and precision (i.e. limiting the number of spurious links), we decided to use the sentence-based interaction detection.

B. Narrative Charts for *The Hobbit* and *The Silmarillion*

Complementing the results in Section 2, in Figure 7 we present two additional narrative charts that we generated for *The Hobbit* and *The Silmarillion*.

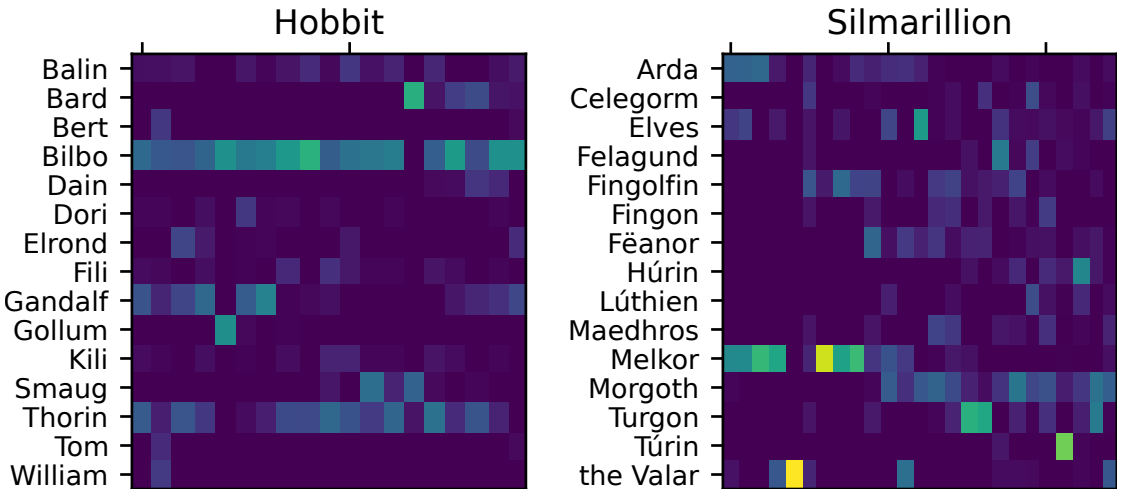


Figure 7: Narrative charts for *The Hobbit* (left) and *The Silmarillion* (right)

C. Visualisation of Additional Latent Space Embeddings

In Figure 8 and Figure 9 we include additional visual representations of latent space embeddings of characters obtained by those methods that have not been included in the main text. For all methods, we employed t-SNE [48] to reduce the dimensionality of the latent space embeddings to two dimensions.

D. Node Classification and Link Prediction with Weighted GNNs

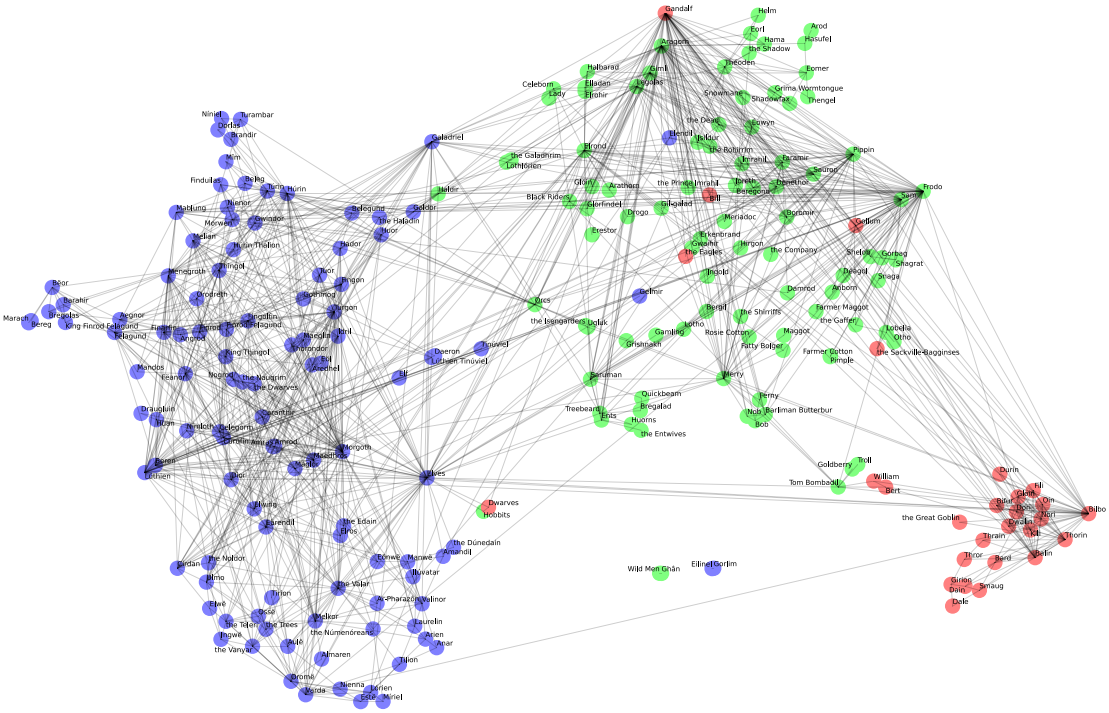
Complementing the results discussed in the main article, in Table 5 we include additional results for which we applied Graph Neural Networks to *weighted* graphs, i.e. different from Table 4 we consider a character network with link weights $w((v, w))$ that capture the number of co-occurrences of two characters v, w. Due to time constraints, we performed this analysis only for the best performing Graph Neural Network, i.e. GCN. These additional results suggests that, at least for our corpus, the inclusion of link weights does not significantly improve the performance of models for character classification and link prediction.

Table 5
Additional results for node classification in the *Legendarium* graph where links carry weights that capture the number of character co-occurrences .

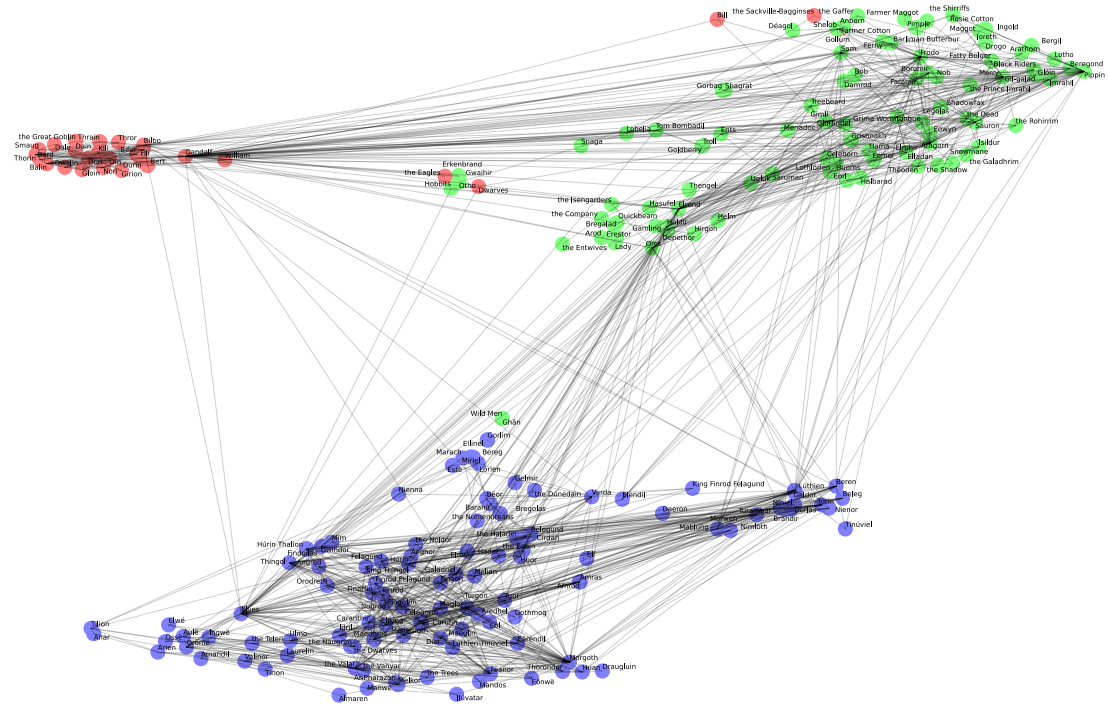
	Character classification			**Link Prediction**
	F1-score	Precision	Recall	ROC/AUC
$GCN^{20}{}_{LE}$	60.85 ± 0.12	60.84 ± 0.16	63.40 ± 0.10	79.41 ± 1.83
$GCN^{20}{}_{node2vec_{p=1,q=1}}$	92.14 ± 0.11	**95.87 ± 0.07**	91.28 ± 0.11	**86.23 ± 0.99**
$GCN^{20}{}_{OHE}$	87.92 ± 0.10	93.02 ± 0.08	86.36 ± 0.11	80.52 ± 1.76
$GCN^{20}{}_{word2vec}$	**92.96 ± 0.10**	95.67 ± 0.07	**92.03 ± 0.10**	82.88 ± 2.08

E. Explanation of ROC/AUC Evaluation of Link Prediction

In Section 4 we use the Area Under Curve (AUC) of a Receiver-Operating Characteristic (ROC) curve to evaluate the performance of our models in link prediction, which is a common approach to evaluate the diagnostic quality of binary classifiers in information retrieval and machine learning. A key advantage of this approach is that it enables us to evaluate the performance of a binary classifier across all possible discrimination thresholds, which can be adapted to tune the sensitivity/specificity of the prediction depending on application requirements. To assist the reader to follow this evaluation approach, below we explain one exemplary ROC curve obtained for a link prediction in the *Legendarium* graph using the a `node2vec` embedding of characters and a logistic regression model. To generate this curve, we first consider the prediction scores (i.e. in the case of logistic regression the positive class probability) assigned to each node pair in the test set, where a link is predicted whenever the score is above a given discrimination threshold ϵ. For each value of ϵ we can now calculate the true and false positive

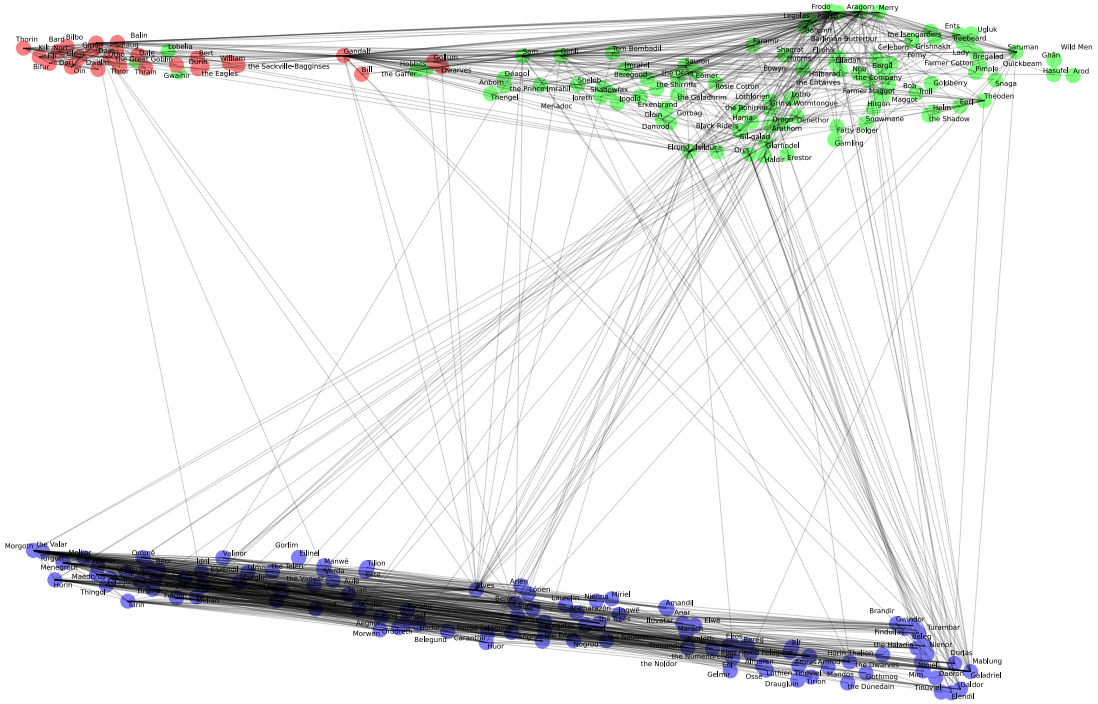


(a) node2vec

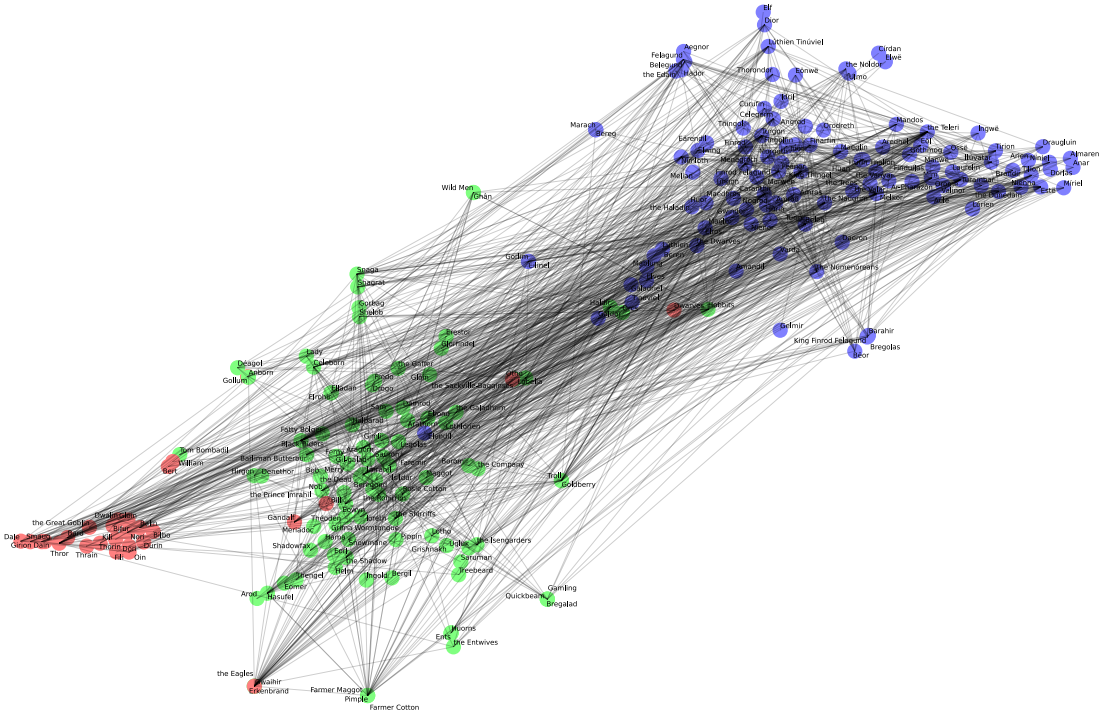


(b) Graph Attention Networks with additional word2vec features

Figure 8: Latent space embedding of characters in Tolkien's Legendarium using node2vec (a) and Graph Attention Networks with additional node2vec features (b).



(a) Graph Convolutional Networks with one-hot-encoding (OHE)



(b) Laplacian Eigenmap

Figure 9: Latent space embedding of characters in Tolkien's Legendarium using GCN with one-hot encoding (OHE) (a) and Laplacian Eigenmap (b).

rate (TPR and FPR), i.e. the fraction of those predicted links for which the prediction is correct and the fraction of unconnected node pairs for which a link is predicted errorneously. A sweep over all possible discrimination thresholds ϵ now yields a ROC curve in the unit square. In Figure 10 we show the ROC curve of a logistic regression model using node2vec features. A classifier that perfectly classifies all instances in the data will assume initial values of FPR=0 and TPR=0 only for the maximal discrimination threshold of $\epsilon = 1$, where all instances are assigned to the negative class. For any ϵ smaller than the maximum and larger than the minimum value of zero, a perfect classifier correctly predicts all instances, which yields TPR=1 and FPR=0. For the minimum value of $\epsilon = 0$, the classifier necessarily predicts the positive class for all instances, which yields TPR=1 and FPR=1. We thus find that the ROC curve of a perfect classifier follows the left and upper border of the unit square, which yields an Area Under Curve (AUC) of one. Conversely, the ROC curve of a classifier that consistently predicts the opposite of the true class follows the bottom and right border of the unit square, which yields an Area Under Curve (AUC) of zero. For a classifier with no diagnostic ability, the FPR and TPR are expected to increase equally as we lower the discrimination threshold ϵ, i.e. the ROC curve follows the so-called *diagonal of no-discrimination* (see red dashed line in Figure 10) and the Area Under Curve is expected to be close to 0.5.

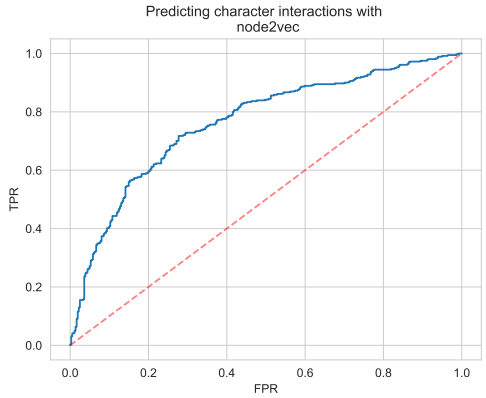


Figure 10: Exemplary ROC curve for link prediction using node2vec embedding and logistic regression.

F. Evaluation of Link Prediction for Individual Works

In the main text we present and discuss the performance of different techniques to predict links in a single character network that spans Tolkien's Legendarium. Apart from this analysis, we additionally evaluated link prediction in character co-occurrence networks that have been generated for the three works of our corpus, i.e. *The Silmarillion*, *The Hobbit*, and *The Lord of the Rings* separately. For the sake of completeness, we include these results in Table 6 below. Like in Appendix D, we applied the Graph Convolutional Network (GCN) model on a weighted

graph, while Graph Attention Networks (GAT) were applied to an unweighted graph.[5]

Table 6
Additional results for link prediction in character co-occurrence networks constructed for individual works in our corpus.

	The Silmarillion ROC/AUC	*The Hobbit* ROC/AUC	*The Lord of the Rings* ROC/AUC
$GCN^{20}{}_{LE}$	80.38 ± 1.97	71.87 ± 5.94	76.25 ± 2.05
$GCN^{20}{}_{node2vec_{p=1,q=1}}$	81.74 ± 2.3	63.86 ± 5.57	76.61 ± 2.87
$GCN^{20}{}_{OHE}$	74.78 ± 1.9	70.22 ± 8.76	63.15 ± 4.37
$GCN^{20}{}_{word2vec}$	76.82 ± 2.68	70.23 ± 7.15	67.42 ± 2.77
$GAT^{20}{}_{LE}$	68.99 ± 3.46	76.88 ± 4.53	61.3 ± 7.38
$GAT^{20}{}_{node2vec_{p=1,q=1}}$	79.37 ± 2.13	71.21 ± 6.53	71.92 ± 2.69
$GAT^{20}{}_{OHE}$	70.88 ± 3.12	71.52 ± 8.14	63.53 ± 3.68
$GAT^{20}{}_{word2vec}$	73.72 ± 4.13	72.29 ± 7.78	64.52 ± 4.29
$GCN^{20}{}_{LE}$ (weighted)	80.4 ± 1.44	69.84 ± 7.23	81.31 ± 1.96
$GCN^{20}{}_{node2vec_{p=1,q=1}}$ (weighted)	82.7 ± 1.92	68.09 ± 7.42	76.46 ± 3.56
$GCN^{20}{}_{OHE}$ (weighted)	74.78 ± 3.3	67.25 ± 9.41	63.15 ± 3.12
$GCN^{20}{}_{word2vec}$ (weighted)	78.8 ± 1.8	62.97 ± 22.35	64.53 ± 1.79

[5]We resorted to an unweighted graph due to a supposed implementation error in the weighted GAT implementation in version 2.0.4 of the graph learning library `pytorch-geometric` [13].