◼ Napovedovanje umorov v seriji Igra prestolov z uporabo analize omrežij

Jaka Stavanja, Matej Klemen, Lovro Šubelj
University of Ljubljana, Faculty of Computer and Information Science, Večna pot 113, 1000 Ljubljana
js8927@student.uni-lj.si, mk3141@student.uni-lj.si, lovro.subelj@fri.uni-lj.si

Izvleček

Serija Igra prestolov ima veliko oboževalcev prav zaradi dramatičnih smrti pomembnih likov. V našem delu želimo na podlagi podatkov o preteklih umorih in ubojih v seriji ter dodatnih metapodatkih o likih napovedati kdo ubije koga. Iz podatkov o umorih v seriji zgradimo omrežje, kjer vozlišča predstavljajo like v seriji, usmerjene povezave pa predstavljajo umore. Na zgrajenem omrežju preizkusimo različne metode za napovedovanje povezav in preverimo njihovo uspešnost. Poleg tega iz družbenega omrežja likov pridobimo dodatne značilke in jih uporabimo za napovedovanje umorov s tehnikami podatkovnega rudarjenja. Ugotovimo, da zaradi majhne velikosti podatkovne množice in nestrukturiranosti umorov, z osnovnimi metodami napovedovanja povezav umorov ne moremo napovedati. Dodajanje novih značilk in uporaba tehnik podatkovnega rudarjenja izboljša dosežene rezultate. Pokažemo, da obstaja način, ki na zgrajenem omrežju umorov doseže najboljše rezultate, vendar ni praktično uporaben. Ta način doseže najboljšo površino pod ROC krivuljo, t.j. 0.875.

Ključne besede: ekstrakcija značilk, Igra prestolov, napovedovanje povezav, analiza omrežij.

Abstract

TV series such as HBO's Game of Thrones have a high number of dedicated followers, mostly due to the dramatic murders of the most important characters. In our work, we try to predict killer and victim pairs using data on previous kills as well as additional metadata. We construct a network where two character nodes are linked if one killed the other, then use a link prediction framework to evaluate different techniques for kill predictions. Lastly, we compute various network properties on a social network of characters and use them as features in conjunction with classic data mining techniques. Due to the small size of the dataset and the somewhat random kill distribution, we cannot make accurate predictions with standard indices alone, although using them in conjunction with additional rules based on degrees has yielded results that are more reliable. The features we compute on the social network help the classic machine learning approaches; however, they do not yield very accurate predictions. The best results overall are achieved using indices that use simple degree information, the best of which result in the Area Under the ROC Curve of 0.875.

Keywords: Feature extraction, Game of Thrones, link prediction, network analysis.

1 INTRODUCTION

With the ever increasing popularity of the TV series Game of Thrones, coming mostly from it's incredible plot twists and deaths of main characters, the question arises whether we can predict those deaths from a network analysis point of view. If we are able to predict the deaths from the data, collected from previous episodes, that means that the author is very predictable, which might not be the best thing in terms of the show being entertaining. To predict the deaths, we construct a network illustrated in Figure 1, where nodes are characters in the show (along with other entities that are able to kill another character, such as a horse or a dragon) and connect two if one has murdered the other. We then try to predict whether a certain link between two nodes in the network happened (removing the link from the network beforehand). Because we are dealing with temporal data,

we also remove links that appear in the network after the link currently being predicted. We use different approaches to assign scores to the pairs, for example different link prediction indices such as the preferential attachment index (Liben-Nowell & Kleinberg, 2007) or the Adamic-Adar index
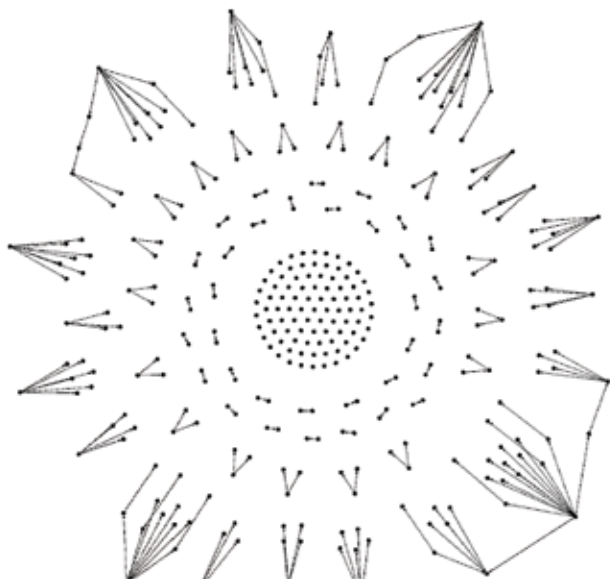

Figure 1: **Game of Thrones kills network.**

(Adamic & Adar, 2003). We also construct additional features based on various network properties and additional metadata and see how this influences the results. The network properties for the features used in machine learning approaches are then computed on a different social network of the Game of Thrones characters, where nodes representing characters are connected if the characters appear somewhat close in the original books' story.

The rest of the paper is structured as follows. In Section 2 we provide an overview of existing literature on our studied topic. In Section 3 we describe the methods that we use in our work. In Section 4 we provide the results of our work, which we then discuss in Section 5. In Section 6 we summarize the work that was done and provide some possible future improvements.

2 RELATED WORK

We use link prediction as a way to infer new links between nodes in a graph using different network properties. The field of link prediction research started evolving around trying to either generate synthetic networks or infer missing data for network-like data structures. Later on, that evolved into more of a recommendation type of approach (for example, trying to recommend friendships on social networks), based on the same principles as we use to calculate probabilities of new links. One of the most important articles on the network properties that we can use to infer new links is written by Barabási & Albert and features exploring the phenomenon of preferential attachment (Barabási & Albert, 1999). The work on this property and other founding principles for link prediction is then briefly described inside the article by the same authors (Albert & Barabási, 2002). The next explored thing is missing data (Kossinets, 2006) along with further studies on missing links and spurious networks (Guimerà & Sales-Pardo, 2009). An extensive survey on link prediction is written by Lü & Zhou (Lü & Zhou, 2011) where the whole field along with all the better-known indices to date is presented and the authors show how the classic problems in link prediction didn't use enough network properties or community structure, which was explored by Girvan & Newman (Girvan & Newman, 2002). We use some of their findings in our research. One of the most well known indices to calculate the likelihood of new links appearing in a network is the common neighbors index (Newman, 2001) (Kossinets, 2006), implying that the more common neighbors nodes have, the more likely they are to form a link. Some other popular choices include the cosine distance index (also named the Salton index) (McGill & Salton, 1983), the Jaccard index (Jaccard, 1901), the preferential attachment index (Barabási & Albert, 1999) and the Adamic-Adar index (Adamic & Adar, 2003). For our experiments, we pick some of the most used indices.

For Game of Thrones specifically, some research has already been done in terms of finding which aspects of the show resonate with the viewer count the most and how real the characters' interactions are done by modeling the show's houses with a network and exploring structural balance (Liu & Albergante, 2017). Further studies determined who has the best strategic position in the show's world (Beveridge & Shan, 2016), but the only article touching on death prediction studies (Angraal et al., 2018) for the show used Cox's proportional hazard model (Cox, 1972), which didn't explore any network structure properti-

es, but rather used a regression approach to determine which factors introduce a higher risk of mortality through time. But the killers were not taken into consideration here, we just know how likely a character is to die, thus we cannot really compare this approach with ours. This gives us a unique opportunity to try and use network properties along with metadata to better predict kills in the mythical world of the series, but instead of only predicting how likely someone is to die, we can also predict both the killers and victims, on which there has not yet been much research.

3 METHODS AND NETWORK PROPERTIES

In this section, we describe the framework used to test our link prediction methods. We also provide descriptions of some classic link prediction methods based on well-known properties of networks. Then, we introduce a new social network, described in Subsection 3.2.5, which connects characters if they appear close in the original story from the books. We compute various network properties from that network and use an automatic node embedding technique to compute features for use in traditional machine learning approach for classification.

3.1 Link prediction using the kills network

The network of kills is constructed of directed links between pairs of nodes i and j, where node i represents a character that killed the character represented by node j. The network is very small and also very sparse. It has 353 nodes and 194 links. By looking at the visualization of the network in Figure 1, we can see that most kills appear outside somewhat bigger connected components and do not seem to be attributed to hubs (i.e. high degree nodes). We can still observe a few hubs and connected components around them, where a lot of kills seem to occur, but in the majority of cases there are just two nodes involved in the kill, which on their own form a small two-node connected component. That gives us a clue as to which methods might predict kills better than others. We can construct an index that predicts links based on nodes' out degrees. Since the in-degrees in the networks could be either zero or one (meaning alive or dead) we cannot get any additional information from that besides whether a person can still kill someone or not (when they are already dead). The out-degree of a node can potentially be of use, sin-

ce people that kill a lot of people might also tend to kill more people, and those who never killed anyone might not be inclined to murder or kill. A plot of the out-degree distribution (using a logarithmic scale for the fractions of nodes) is shown in Figure 2.

In this subsection, we propose an evaluation framework and different indices that we use for the link prediction.

3.1.1 Evaluation framework for link prediction

To test how well our link prediction techniques work on our network, we construct a framework and provide a brief description of it here. It takes our prediction index function and the network as the input and outputs the Area Under the ROC Curve (AUC) value. The prediction index function assigns a score sij to every link between two nodes i and j that is being tested. A high score implies a high likelihood that the link exists in the network and a low score implies a small likelihood of the link's existence.

The core of the testing framework is the logic, which removes links from the network and then tries to predict how likely the links we have removed are to form as the network evolves through time. To use as much information as we can, we choose an episode and remove links from that episode (e.g. episode 30) and onwards from the graph. Then, we predict the links (i.e. kills) at that time using the information about kills from the previous episodes. We can then clone the original graph, remove links from the next episode in the chronology (i.e. episode 31) and predict links for that episode using the information from all the episodes before (including information from episode 30 in this example). By predicting links in this way, we are not using the data from the future to predict past links and we are using a lot more information than if we were to remove links from a certain episode onwards and just try to predict links for multiple episodes in one single iteration.
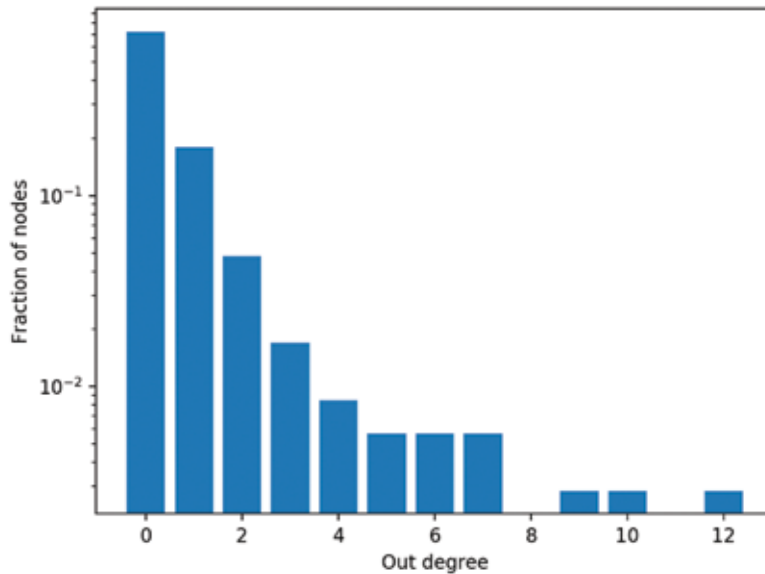
Figure 2: **Out-degree distribution for the kills network.**

The idea of the framework's core implementation is as follows:

1. Iterate through every episode in the range from 30 to 60, and at each step do:
   (a) $L_P \leftarrow$ remove node links $(i, j) \in L$ after the current episode.
   (b) Compute sij for $(i, j) \in L_P$ at the current episode time.
   (c) Save the scores to S_P.
2. $L_N \leftarrow$ randomly sample $|L_P|$ unlinked nodes $(i, j) \notin L$.
3. Compute sij for $(i, j) \in L_N$ and save scores to S_N.
4. Compute AUC by comparing the scores assigned to positive samples S_P and scores assigned to negative samples S_N. When comparing scores, we assign to b the amount of times when the score from S_P is bigger than the one from S_N, and assign to e the number of times when the two scores are equal. This is counted across a random sample of $|L_P|$ pairs. Then, we compute the AUC as $\frac{b + e/2}{|L_P|}$.

3.1.2 Alive index

We create a type of a baseline index by looking at the network's high level properties. We check if the killer has an in-degree of zero and the target has an in-degree of zero (i.e. killer is alive and target has not been killed yet). If the endpoints of a link satisfy these conditions, the value of the index is 1, otherwise it is 0.

3.1.3 Preferential attachment index

Real world networks tend to have a scale-free degree distribution due to a phenomenon known as preferential attachment (Barabási & Albert, 1999). The preferential attachment index (Liben-Nowell & Kleinberg, 2007) is defined as $s_{ij} = k_i k_j$, where k_i is the degree of node i. For our problem, we use out-degrees only, as only they provide useful information. The idea behind the index is in the preferential attachment process — nodes are more likely to connect with nodes that have a high degree, thus a link between two nodes with high degrees should be assigned a high score s_{ij}. We modify this definition slightly and define a modified preferential attachment index as $s_{ij} = k_i^{(out)} k_j^{(out)}$, where $k_i^{(out)}$ is the out-degree of node i. The logic behind that is that the more kills one has to their record, the more they are inclined to murder and vice-versa. Additionally, we include the in-degree information in a modified version of the preferential attachment index: if the source node or the target node has an in-degree larger than zero (either the killer or the victim is already dead), a very negative score ($-\infty$) is returned. Otherwise, the regular version of the index gets computed.

3.1.4 Adamic-Adar similarity index

In real world networks, links tend to appear between nodes that have a lot of common neighbors (Watts & Strogatz, 1998). But due to preferential attachment, nodes tend to connect to higher degree nodes more

likely, thus making that neighbor less useful for predicting new links between two nodes. The Adamic-Adar similarity index (Adamic & Adar, 2003) takes the high degree neighbors into account and is defined as

$$s_{ij} = \sum x \in \Gamma_i \cap \Gamma_j \frac{1}{\log k_x},$$

where Γ_i is the neighborhood of node i and k_x is the degree of node x. Similarly, as for the preferential attachment index, we also include a modified version of the Adamic-Adar index, where we take into consideration whether the killer or the victim is already dead.

3.1.5 Community index
New links in e.g. social networks tend to appear between members that are inside of the same community and only rarely between members of different communities (Girvan & Newman, 2002). We can find densely linked communities where people are killing each other the most and treat the new links in those communities as more likely to occur than the ones outside communities. So that we do not ignore links between communities, we can also count the amount of links that occur between two communities and model the inter-community densities. Let {C} be the set of communities output by the Leiden modularity optimization algorithm (Traag, Waltman, & van Eck, 2019), and ci the community of node i. Then,

$$s_{ij} = \begin{cases} m_i/\binom{n_i}{2}, & when \ c_i = c_j \\ m_{ij}/(n_i \cdot n_j), & when \ c_i \neq c_j \end{cases},$$

where ni is the number of nodes in the community of node i, mi the number of links within community of node i and mij is the number of links between communities of nodes i and j. As is the case for previous two indices, we include a modified version of the community index as well, returning a very negative score when the killer or victim are already dead.

3.2 Feature extraction for machine learning approaches
Besides using the classic index-based link prediction techniques, we also make use of a machine learning-based approach, in which we construct features from network properties and additional metadata, and use them to train a classifier. The data used for training the classifier includes properties of the kills net-

work such as PageRank scores for all characters and the basic kills from the original dataset. Computing a score like PageRank (Brin & Page, 1998) on the kills network would be pointless as the nodes have an in-degree of at most one, so the scores would be high for those who already died. We can take a different network of characters into account for this particular case. An online repository of sample networks, found at https://github.com/melaniewalsh/sample-social-network-datasets, contains a sample of the Game of Thrones social network, which creates an edge between two character nodes if they appear within a 15-word distance in the original books. Since the shows are vastly influenced by the books (at least very strongly up to episode 60 to which our kills data is collected) we can use properties from that network as well to gather some additional information. This network has 107 nodes and 352 edges, and is shown in Figure 3.

3.2.1 Standard machine learning framework
The framework we use for predicting links using machine learning is similar to the framework that is described in Section 3.1.1. We split the procedure for getting scores for test links into two parts — first we obtain the scores for positive examples and then the negative examples.

To obtain the scores for positive examples, we first choose some episode, whose kills we are currently trying to predict. These links make up our current test set. Links that appear in the episodes after the selected one are ignored, as we must not predict the past based on future events. The kills that happened prior to the selected episode make up our current training set. In addition to that, we sample the same amount of negative examples in order to make the training set balanced. The classifier is then trained on the training set and used to predict scores for chosen test examples. This is repeated for multiple chosen episodes and at the end we obtain scores for P positive examples, where P contains all the kills that have happened in the chosen episodes.
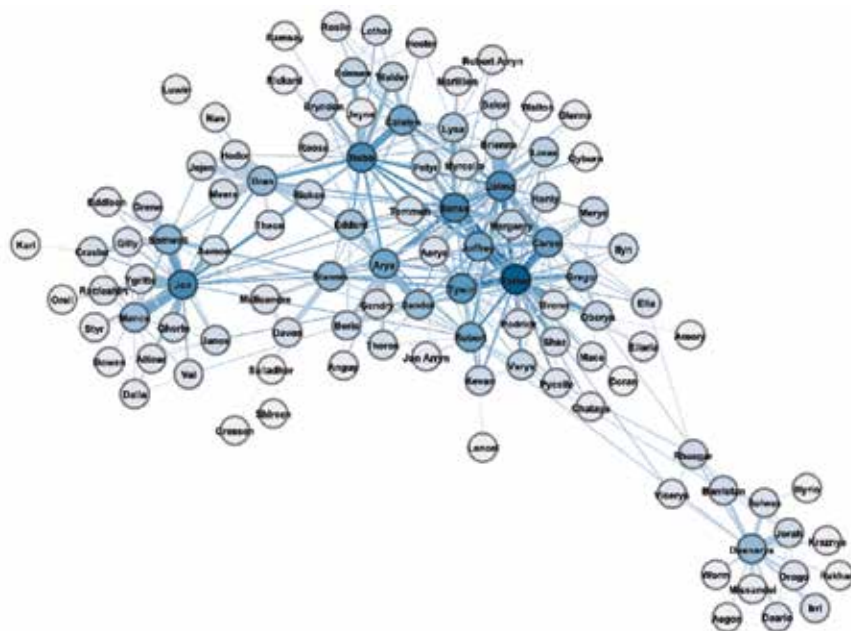
Figure 3: **Game of Thrones social network.**

For negative examples, we take all the kills from our dataset and sample the same amount of non-kills to form a training set. The test set consists of P randomly sampled non-kills from the entire dataset. These test examples are sampled in a way that they do not overlap with negative examples in the training set. For obtaining the scores, we use three different models: K-nearest neighbors (KNN), logistic regression and support vector machine (SVM) (Hastie, Tibshirani, & Friedman, 2009). The rest of the framework remains the same as the one described in Section 3.1.1.

3.2.2 PageRank

The first feature we use is the PageRank score (Brin & Page, 1998). We can calculate it to find the most important characters as they will hopefully have the highest scores. Being more important could mean two things — either you are very important and thus have a lot of security by guards and other helpers, so you are very unlikely to die, or the exact opposite. The opposite would imply that since this series is oriented around taking the power from others, you are more likely to die if you are very important, which seems more plausible since the show thrives on the sudden deaths of the more popular characters. A very low PageRank score of some character would then also mean that they are very unlikely to have their death portrayed, since the viewers and readers don't care or

have forgotten about the least important people in the story. We use two PageRank-based features, the first being *killer_pagerank* and the second *victim_pagerank*, since we are trying to predict killer and victim pairs. The top 5 scoring characters after PageRank calculations are Tyrion (0.055), Jon (0.045), Daenerys (0.041), Jaime (0.037) and Sansa (0.036). From our knowledge of the show, we can safely claim that these are some of the most, if not the most important characters, so we can then be sure that these scores make sense in terms of importance. We assign a mean of all the PageRank scores for each character that is found in the kills dataset, but is not present in the social network.

3.2.3 Betweenness centrality

Another measure that we extract from the social network as a feature is the betweenness centrality (Freeman, 1977). This score measures how many shortest paths from two different nodes go through a certain other node. That means that the higher the score, the more control over a big portion of the network a node has (i.e. is one of the nodes that can make the network split quickly). That should also yield a measure of importance — if one character wants to reduce the power of a part of a network, they can disconnect it from the biggest component by eliminating nodes with high betweenness centrality. Again, we observe that the five top scoring people are also the most important

characters in the story: Jon (0.230), Robert (0.209), Tyrion (0.198), Daenerys (0.157) and Robb (0.127). For every character from the kills dataset that is not in the social network, we use a mean of betweenness centrality scores, similarly to the PageRank scores.

3.2.4 Community detection

Simply using the house that a character belongs to as a feature could perhaps help. The intuition behind this is that there might be more kills occuring between members of different houses than between members of the same house. Since our dataset has a lot of undefined values for the houses, we can find communities in the social network using an algorithm such as the Leiden modularity optimization (Traag et al., 2019). We can assign a label of the community to each character and see if that helps us with the prediction by creating some sort of indicator between which groups the kills occur. After running the community detection algorithm on the social network, we can see that we obtain meaningful results (meaningful to people who watch the show) in terms of alliances. The network gets partitioned into five big communities. For example, we can see that Khal Drogo, Daenerys Targaryen, Aegon Targaryen and Jorah Mormont all fall into the same community, even though they do not originate from the same houses, but as we know from watching the show, they are allies. Every character from the kills dataset that is not included in the social network gets assigned to a dummy community.

3.2.5 Automatic feature extraction for machine learning approaches

In addition to handcrafting features we also try an approach using automated feature extraction, specifically node2vec (Grover & Leskovec, 2016) to obtain node and link features. We use these in place of previously handcrafted features.

For a given node, node2vec constructs a feature representation (embedding) that aims to preserve the network neighbourhood properties in a vector space of fixed dimensionality. Depending on parameters p (return parameter) and q (in-out parameter), the algorithm performs different types of biased random walks in order to represent the node's neighbourhood. Setting p to a low value encourages a search that is local to the given node, while setting q to a low value encourages a more explorative search. The

sampled neighbourhoods are then used to estimate a feature representation that maximizes the probability of observing these neighbourhoods for the given node. We obtain a link embedding by concatenating together the embeddings of source and target node. If a node has no embedding, a generic embedding for an unknown node is assigned to it. We train the embeddings on the social network, capturing character co- ocurrences. Because we are dealing with a very small dataset, we cannot afford to tune the hyperparameters reliably. Following the findings of authors of the method, we set the parameters to $p = 2$ and $q = 0.5$ since these settings are shown to bias the embeddings to capture homophily. We fix the node embedding size to 16.

4   RESULTS

For our experiments we select and remove links that are associated with kills that happened in seasons four to six. There are 114 of those, to which we add 114 randomly selected unlinked nodes and compute the AUC based on these examples. We repeat this process five times to account for the randomness in selection of negative examples and provide the mean AUC and its standard deviation. We support the AUC scores with precision and recall scores as well, since AUC only measures how well a randomly selected positive example can be distinguished from a randomly selected negative example. For the classic link prediction techniques, we classify examples with scores strictly higher than zero as positive and the others as negative. For the machine learning approach, every example with a score greater than or equal to 0.5 is classified as positive and the others as negative. We then calculate the precision as

$$precision = \frac{\#true\ positives}{\#true\ positives + \#false\ positives}$$

and recall as

$$recall = \frac{\#true\ positives}{\#true\ positives + \#false\ positives}.$$

The obtained results are shown in Table 1.

Results show that the community index achieves the best result in terms of AUC, 0.875. The best precision and recall scores are obtained using the alive index, which are 0.822 and 0.930. Other indices using the death info achieve similar results, however their precision and recall scores are different.

Tabela 1: **Results for four link prediction methods: preferential attachment index, Adamic-Adar index, community index and alive index. The table shows the mean AUC, precision and recall and their standard deviation over five runs. The symbol marks the versions of indices where a check is first performed if the killer or the victim is already dead.**

Method	AUC	Precision	Recall
preferential attachment	0,503 (0,020)	0,522 (0,113)	0,087 (0,000)
preferential attachment †	0,862 (0,015)	0,654 (0,123)	0,070 (0,000)
Adamic-Adar	0,500 (0,000)	0,000 (0,000)	0,000 (0,000)
Adamic-Adar †	0,872 (0,028)	0,000 (0,000)	0,000 (0,000)
community index	0,500 (0,000)	0,000 (0,000)	0,000 (0,000)
community index †	**0,875** (0,018)	0,000 (0,000)	0,000 (0,000)
alyve index	0,863 (0,032)	**0,822** (0,000)	**0,930** (0,000)

Table 2: **Results for three classic machine learning methods (using the basic dataset): K-nearest neighbors, logistic regression and support vector machine (SVM). The table shows the mean AUC, precision and recall and their standard deviation over five runs.**

Method	AUC	Precision	Recall
KNN	**0,632** (0,048)	0,641 (0,024)	0,453 (0,020)
Logistic regression	0,596 (0,036)	**0,797** (0,010)	0,400 (0,013)
SVM	0,556 (0,066)	0,688 (0,024)	**0,523** (0,007)

The Adamic-Adar and community index achieve 0 precision and recall because no links get classified as positive. For the community index, this is due to the network components being very disconnected, while for the Adamic-Adar this is due to the fact that nodes cannot have common neighbors as kills are only attributed to one person, meaning two killers cannot kill the same victim.

Other indices that don't use information about deaths achieve AUC scores around 0.5. This implies that they perform no better than if links were classified randomly.

The standard machine learning approaches came out to be a little bit better than random when using the base kills dataset using only the out-degrees of characters as a feature, with AUC scores ranging from 0.556 to 0.650 as shown in Table 2.

By adding network features (PageRank, betweenness and community identifier for each character) we improve the general performance of all the classifiers and obtain a better top score of 0.686 by using SVM

and the handcrafted features. The features created by node2vec yield worse AUC and precision scores and higher recall scores than the handcrafted features. All the final results for the machine learning approaches are shown in Table 3.

Table 3: **Results for three classic machine learning methods (using the augmented dataset): K-Nearest-Neighbors, Logistic regression and Support Vector Machine (SVM). The table shows the mean AUC, precision and recall and their standard deviation over five runs.**

Method	AUC	Precision	Recall
KNN	0,659 (0,035)	0,725 (0,016)	0,453 (0,004)
Logistic regression	0,658 (0,033)	**0,816** (0,016)	0,418 (0,013)
SVM	**0,686** (0,058)	0,719 (0,058)	0,456 (0,056)
KNN (node2vec)	0,650 (0,044)	0,595 (0,052)	**0,702** (0,036)
Logistic regression (node2vec)	0,640 (0,025)	0,600 (0,051)	0,684 (0,028)
SVM (node2vec)	0,605 (0,091)	0,642 (0,062)	0,632 (0,022)

5 DISCUSSION

We see that the sparsity of the network and its size (less that 400 nodes) make link prediction on such a small network very inaccurate in most cases.

Since the original network of kills does not seem to have any community-like structure, it is very hard to predict kills based on community-based link prediction methods, such as the community index. The modularity optimization algorithm finds more than 100 communities with no links between them, which means that the modeling of densities between communities does not give us any information, since the probability of a link occurring between communities is zero. That is a solid foundation for the claim that the authors have done a good job by not creating a very obvious structure of the kills, where someone would kill a lot of people from e.g. their opposing house, making it easy to predict that there will be another similar kill occurring in the following episodes which the viewers have not yet seen.

We cannot achieve good results by constructing indices that try to predict kills using out-degrees only, since the majority of nodes have a very low out-degree (as can be observed on Figure 2).

When we add the death information, we observe that the modified Adamic-Adar index performs as good

as the alive index, since it represents the same idea. We know that two nodes cannot have a common successor, since a character can only die once. They can only have one common predecessor, however that implies that both the killer and victim are already dead. By taking the death information into consideration, we automatically decide that when one character is already dead, there will be no link. That makes the common neighborhood factor in the index irrelevant, making it decide the outcome based only on whether a character is alive or not.

The indices that use the death information achieve the best results because of the way the network is constructed. Nodes either have an in-degree of zero or one (depending on whether they were already killed or not). When sampling positive examples in our framework, we remove the edge for that positive example, decreasing the in-degree of the target node from one to zero. When sampling negative examples, we do not remove any edges. However, because our original network is constructed from deaths in the series, most of the characters in our network have died at some point. Therefore, the target node of a negative example is quite likely to have an in-degree of one. These death information indices make use of the fact that when we sample a positive example, we are going to decrease the in-degree of the target to zero, and when we sample a negative example, the in-degree of the target is likely to still be one (i.e. that the target was killed by somebody else at some point). We try to account for this by adding some additional isolated nodes (corresponding to characters that did not kill anyone and did not die), but the index still performs best on the expanded network. So although the baseline index and the other indices which use death information achieve good results, they do not do so by using any structural properties of the network but rather just by abusing the way our network is constructed. The key takeaway here is that achieving an AUC score around

0.85 is not that hard. The hardest part is achieving a noticeable increase in performance over the baseline classifier. The results from the standard machine learning approaches are bad, since we only use the out degree as the basic feature from the original dataset to predict kills. When we augment it with different centrality measures and community identifiers, the performance is improved, mostly because of the fact that we do not only have one feature anymore

and because these features are not equally weighted anymore. However, there should be some added value to the features due to what the features represent, which is explained for each index in Section 3.

Among the approaches using node2vec features, the approach using KNN achieves the best results. Visualization of the link embeddings reveals why the approach performs well. Figure 4 shows embedded positive and negative links, projected onto a two-dimensional plane using t-distributed Stochastic Neighbor Embedding (t-SNE) (Maaten & Hinton, 2008). We can observe that the links are often surrounded by links of the same class in their neighbourhood, allowing KNN to correctly predict many links. The visualization also reveals one of the reasons the approaches using node2vec do not perform better: the links where at least one of the nodes does not have a "proper" embedding are embedded closely. The circle-shaped cluster in the visualization represents the links where at least one of the nodes were assigned a generic embedding for unknown nodes (due to some character being present in the social network but not in the kills network). The cluster contains very mixed classes, rendering KNN less useful. In additional experiments using node2vec features, we found that using a link embedding technique where node embeddings are averaged instead of concatenated together (i.e. ignoring direction of the link) results in a significant performance drop. The modified embeddings in combination with KNN result in a mean AUC reduced by 0.068, a mean precision reduced by 0.028 and mean recall reduced by 0.127. This aligns well with intuition since, for example, a notorious killer is more likely to kill an innocent victim than vice versa.
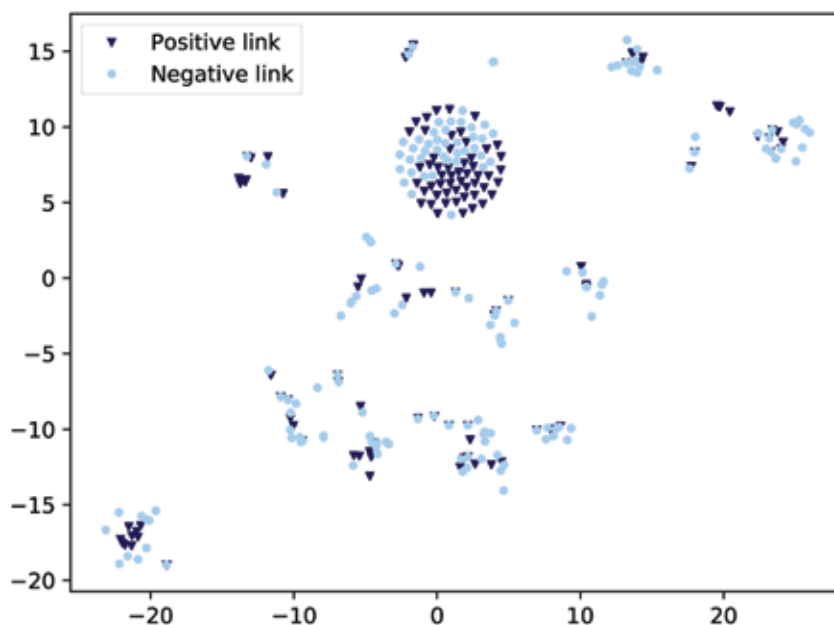
Figure 4: **Node2vec embeddings for positive and negative links in the kills network, projected onto two dimensions using t-SNE.**

6 CONCLUSION

Throughout our work we have acknowledged that the Game of Thrones kills do not have a particularly detectable pattern, since all the kills appear between two nodes that have not yet killed anyone before in most cases. But since our network is small, our test samples are even smaller and that can give deceivingly high AUC scores for indices that would potentially fail on bigger networks and in different scenarios. By using classic machine learning and link prediction techniques, we have found that, on this dataset, no index or feature works better than a simple baseline index (the alive index), which does not model some useful property, but rather abuses the way the network is formed. Most of the techniques used to predict kills in Game of Thrones gave us a fairly good AUC score, but the predictions that our approaches get right do not have a high "shock factor". For example, our classifiers might be able to predict that Jon Snow will kill a wight, but likely fails on less obvious kills. For future work, our approaches could be tested on different TV shows, books or even movies to see how predictable the kills are. Our death information indices could potentially fail on bigger networks with a bit more diverse structure (e.g. having more bigger connected components)

and that would give other more basic indices higher accuracy. We could also construct edges using some other information besides kills, e.g. who had a relationship with whom in some show or who scammed whom in a criminal series.

Source code

The source code to reproduce results presented in this paper is available at https://github.com/matejklemen/got-link-prediction.

REFERENCES

[1] Adamic, L. A., & Adar, E. (2003). Friends and neighbors on the web. *Social Networks*, 25(3), 211–230. Albert, R., & Barabási, A.-L. (2002). Statistical mechanics of complex networks. Reviews of modern physics, 74(1), 47.

[2] Angraal, S., Bhatnagar, A., Verma, S., Shergill, S., Gupta, A., & Khera, R. (2018). Risk Factors Associated with Mortality in Game of Thrones: A Longitudinal Cohort Study. *arXiv preprint arXiv:1802.04161*.

[3] Barabási, A.-L., & Albert, R. (1999). Emergence of Scaling in Random Networks. *Science, 286(5439)*, 509–512.

[4] Beveridge, A., & Shan, J. (2016). Network of thrones. *Math Horizons*, 23(4), 18–22.

[5] Brin, S., & Page, L. (1998). The Anatomy of a Large-Scale Hypertextual Web Search Engine. In *Seventh international world-wide web conference*.

[6] Cox, D. R. (1972). Regression models and life-tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34(2), 187–202.

[7] Freeman, L. C. (1977). A set of measures of centrality based on betweenness. *Sociometry*, 35–41.

[8] Girvan, M., & Newman, M. E. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12), 7821–7826.

[9] Grover, A., & Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 855–864).

[10] Guimerà, R., & Sales-Pardo, M. (2009). Missing and spurious interactions and the reconstruction of complex networks. *Proceedings of the National Academy of Sciences*, 106(52), 22073–22078.

[11] Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2nd ed.). Springer.

[12] Jaccard, P. (1901). Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bull Soc Vaudoise Sci Nat* , 37 , 547–579.

[13] Kossinets, G. (2006). Effects of missing data in social networks. *Social networks*, 28(3), 247–268.

[14] Liben-Nowell, D., & Kleinberg, J. (2007). The link-prediction problem for social networks. *Journal of the American society for Information Science and Technology* , 58(7), 1019–1031.

[15] Liu, D., & Albergante, L. (2017). Balance of thrones: a network study on "Game of Thrones" that unveils predictable popularity of the story. *arXiv preprint arXiv:1707.05213*.

[16] Lü, L., & Zhou, T. (2011). Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications*, 390(6), 1150–1170.

[17] Maaten, L. v. d., & Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov), 2579–2605.

[18] McGill, M., & Salton, G. (1983). Introduction to modern information retrieval. 1983. *McGraw-Hil, New York* . Newman, M. E. (2001). Clustering and preferential attachment in growing networks. *Physical review E*, 64(2), 025102.

[19] Traag, V. A., Waltman, L., & van Eck, N. J. (2019). From Louvain to Leiden: guaranteeing well-connected communities. *Scientific Reports*, 9(1), 5233. doi: 10.1038/s41598-019-41695-z

[20] Watts, D. J., & Strogatz, S. H. (1998, June 04). Collective dynamics of 'small-world' networks. *Nature, 393*(6684), 440–442.

.

Jaka Stavanja je študent drugega letnika magistrskega programa Računalništvo in informatika na Fakulteti za računalništvo in informatiko Univerze v Ljubljani. Zanimajo ga različne metode strojnega učenja, umetne inteligence, uporaba analize omrežij na še na tak način neraziskanih problemih, primarno pa se ukvarja z razvojem različnih spletnih rešitev.

.

Matej Klemen je študent drugega letnika magistrskega programa Računalništvo in informatika na Fakulteti za računalništvo in informatiko Univerze v Ljubljani. Zanimajo ga različna področja umetne inteligence, predvsem uporaba strojnega učenja v povezavi z obdelavo jezika. Na tem področju trenutno raziskuje uporabo vložitev v povezavi z različnimi nalogami obdelave jezika. V prostem času pomaga pri razvoju odprtokodnih knjižnic za podporo strojnemu učenju, kot je na primer Orange.

.

Lovro Šubelj je docent na Fakulteti za računalništvo in informatiko Univerze v Ljubljani. Diplomiral je leta 2008 na Fakulteti za matematiko in fiziko in Fakulteti za računalništvo in informatiko ter doktoriral leta 2013 na temo analize velikih omrežij. Je avtor ali soavtor več kot petdeset znanstvenih prispevkov in patentov ter urednik prestižnih mednarodnih znanstvenih revij. Njegovo preteklo delo je bilo izbrano kot izjemen znanstveni dosežek v Sloveniji ter predstavljeno na uglednih mednarodnih univerzah kot sta Stanford in UCSD. Sodeloval je že pri številnih uspešno zaključenih raziskovalnih in razvojnih projektih v sodelovanju s podjetji Petrol, Celtra, Optilab, Iskratel in drugimi.