Hello, welcome to my presentation. I'm Kurt Routley, a Master's Student from Simon Fraser University. I'll be discussing my paper co-authored with Oliver Schulte, "Aggregating Predictions vs. Aggregating Features for Relational Classification."

(click) I'll start by describing our relational database setup. (click) Our setup contains tables for entities as well as relationship tables, which define a relationship of entities joined together. In this example from the National Hockey League, which made up one of our datasets, we have entities Match, Team, and Player. Each entity is assigned a unique identification number to facilitate table joins. These entities are linked together through the relationship table Game, where each Match has two Teams with 20 Players on each team. (click) The Game table facilitates our link-based classification problem, where Game is our target table and we try to predict the target attribute, or class, Result. For example, we know how Daniel Sedin, Rostislav Olesz and Cory Stillman performed in Match 33 and the result of their performance, but we may want to predict what the result was based on Henrik Sedin's performance. The first three players form a relational neighborhood for Henrik Sedin and are used to assist the prediction.

(click) To prepare our dataset for classification from the database tables, we form an extended database table by joining all entity (click) and relationship tables into a large table of feature vectors. (click) Continuing our example from the National Hockey League, we have the Game relationship table, which contains data for each NHL Game, joined with the Player entity table, which contains a Player's last season statistics to form the large feature table.

Once we have this large feature table, we can begin relational classification. (click) Now we begin our tale of two approaches: Aggregating Predictions, and Aggregation Features. (click) Looking at aggregating predictions first, we start out with a feature vector of each relational neighbor. This is a single vector containing the unique IDs for each entity, as well as all the features for each entity. (click) We then classify each feature vector using classifiers such as logistic regression or support vector machines, and obtain a prediction or score for each feature vector. (click) These predictions are then aggregated over all links in the relational neighborhood into a final classification score using an aggregate operator, also called a combining rule. This final classification prediction is used to classify the feature vectors.

(click) The second approach is to take all feature vectors, which are links in the relational neighborhood, and aggregate them using a variety of aggregate functions into a single aggregated feature vector. (click) This aggregated feature vector is run through a classifier and gives the final classification score for the relational neighborhood. This process is also called propositionalization. Our paper compares the classification results of aggregating predictions versus aggregating features.

(click) Some aggregate functions are shared by both feature aggregation and score aggregation, while others are specific to the aggregation approach. The reason that some functions are shared and others are specific are that some aggregate functions can work as both a feature and a predictor, while others can only act as a feature or a predictor. For example, you can take the Average number of goals and also the average of all predictions, so Average is a shared function. The sum of goals is an appropriate

feature, but the sum of predictions doesn't make much sense. In our experiments, Average, Maximum, Minimum, Midrange, and the Geometric Mean are used in both processes. Sum, Standard Deviation, and Degree are features specific to feature aggregation. The Noisy-Or is used as a combining rule during Score aggregation.

(click) Continuing with the National Hockey League example, an aggregated data table will preserve the target attribute, while aggregating the other features. By aggregating over a match and team, we aggregate the features of the players on a team in a match. Some aggregated features formed as a result are the sum of goals, average penalty minutes, and the maximum number of goals scored in the previous season.

(click) Now we'll outline some of the advantages and disadvantages of each approach, as well as proposed remedies to some of the disadvantages. We'll look first at feature aggregation. The advantages of feature aggregation are that learning time is much faster than score aggregation, and there is less memory required. One disadvantage is that aggregating features loses the distribution information. We can remedy this by using multiple aggregate functions and preserve the first two moments of the distribution. Another disadvantage is that feature aggregation ignores degree disparity, that is, when some relational neighborhoods are larger than others. By adding the degree of a neighborhood as a feature, we can give the classifier information on this degree disparity.

The advantage of score aggregation on the other hand is that it keeps the full distribution information, as each feature vector is preserved in its original form. We see that degree disparity also affects score aggregation, and a classifier will overweight instances with many links in its relational neighborhood. A solution to this is to reweight instances based on the size of its relational neighborhood. The results we will report have the proposed remedies for both approaches implemented.

(click) We will now look at how aggregating the dataset transforms the size of the dataset. For the IMDb and Financial datasets, there is a significant reduction in the number of rows after aggregating. We also observe in the Premier League dataset that the number of columns also increases significantly during feature aggregation. The main point is that aggregation decreases the number of rows, but also increases the number of columns. (click) Graphically, we can observe the increase in the number of columns, and the reduction of the original data rows to the new number of aggregated rows. We see that IMDb and Financial had their rows reduced to under 1% of their original row count, and we observe that datasets with higher degree disparity have a higher row compression ratio. The Premier League dataset's column count increased by almost 700%.

(click) Next, we will see how degree disparity is present in these datasets. The sports datasets for the National Hockey League, Premier League, and National Basketball Association have little to no degree disparity, as the number of players on a team in a match is typically fixed. For the other two datasets, there is a considerable amount of degree disparity. In IMDb, we see the number of ratings for each movie varies wildly, with one movie having as little as 1 rating, and another having as many as 3,427 reviews. We also observe that the standard deviation is larger than the average number of reviews,

showing a large data skew. Degree disparity can also be observed in the Financial dataset, although not to the same extreme.

(click) We can also analyze the correlations between features and the target class before and after aggregation. In all cases, we see that aggregation makes the correlation stronger, measured here using Weka's information gain. In many cases, there was almost a 5-fold increase in the information gain of the feature using the average aggregator. In the case of IMDb, the average revenue of the director was the strongest predictor, but this feature was not aggregated, as the aggregation was done over the reviews for each movie. In light of this, the information gain still increases for the director's average revenue, indicating feature aggregation can help to weed out unnecessary features.

(click) Now we'll examine each of our datasets individually. Note that the row and column counts for each dataset are for the original datasets prior to aggregation. Our largest dataset was IMDb, where the target feature was highBoxOffice, which indicated whether or not a movie has made more than $10 million US dollars in the box office. Some predictive features for this dataset were the average revenue of the director, the language of the movie, and the age of the user writing a review. For the Financial dataset, we tried to predict whether or not an account had a loan based on the transactions. Some predictive features were the average salary of the district where the account is, the payment frequency of the account, and the remittance of a transaction under the account. For all of the real-world sports datasets, we used the result of a team in a match as the target feature, and used player features such as goals scored within the match and goals scored in the last season. We split the NHL dataset into two datasets, one using only player performance statistics from the previous season, and another adding player performance statistics from the match. Using only last season statistics for the NHL didn't achieve much higher than 55%, so gamblers may want to avoid this.

(click) Now we can examine the results of each approach. Once again, these are the results for both approaches with our proposed remedies implemented for some of the disadvantages mentioned previously. Feature aggregation consistently achieved over 85% classification accuracy in all datasets, with the exception of the NHL dataset using only last season statistics, where the features weren't very informative. (click) Comparing against score aggregation, we see that quite often using the best combining rule in score aggregation still performs worse than feature aggregation. Looking at the classification accuracy of score aggregation, we observe there is no uniformly best combining rule, but averaging type operators, such as average, midrange, and geometric mean tended to perform well in comparison to the other extremal aggregators, such as maximum, minimum, and noisy-or. (click) The training and testing datasets used an 80%/20% split for all of our experiments, and the classification accuracy and F1-measure were recorded for every classification run.

(click) Graphically, we see that feature aggregation is the best method a posteriori for every dataset on both classification accuracy and the F1-measure. The Financial dataset had statistically significant results, with feature aggregation outperforming score aggregation by 12.5%. The National Basketball Association had a smaller dataset, and match PlusMinus was an informative feature, so almost all classification runs achieved 100% classification accuracy with both approaches. Again, we see that even if the best score aggregator is chosen, it still performs worse than feature aggregation.

(click) Examining the learning times of both approaches, we see that feature aggregation can save resources and has a much faster learning time for IMDb and Financial datasets. For the Premier League dataset, the number of columns increased substantially after aggregation in comparison to the other datasets, and this caused feature aggregation to perform slower.

(click) In conclusion, we compared two approaches for link-based classification, aggregating features or propositionalization, and aggregating predictions or using combining rules. We found aggregating features performs better on all datasets by examining the classification accuracy and F1-measure of each approach. There was no generally best combining rule for predictions, although averaging rules were more consistent than extremal rules. Degree disparity affected both approaches, but this was remedied by reweighting instances and adding the degree as a feature. A hybrid approach of aggregating features and aggregating scores is a promising area for future work.

(click) Thank you.