Task 1

**1.1**

This sentiment analysis method works by starting with pre-processing the input sentences to limit the number of extra features generated in the vectorising stage. Pre-processing starts with setting all words to lowercase and removing any punctuation or numeric characters so only letters remain. We also noticed that the abbreviation ‘mn’ was commonly used in place of ‘million’, so we also made that substitution to make the results more readable. We also use a WordNet Lemmatiser, to eradicate any additional columns created when vectorising, with a part-of-speech tag encoded in the lemmatiser to give it more information on the context of the word. We finish by removing any stopwords from the sentences so that our model does not get attracted to low-level information.

The strengths in our text pre-processing stage lie in using a lemmatiser as opposed to a stemmer to perform a reduction on our words. The provision of part of speech tags to give context to the mapping allows for better, more accurate results since we have real, dictionary words to build our model on. This pre-processing is limited by some text normalisations such as setting all words to lowercase since this can remove some sentence structure. For example, one could consider a word in capitals to provide a greater emphasis on the meaning of a sentence. Further, the time it takes to process the statements is another limitation since lemmatising is more rigorous, and it takes longer to perform these substitutions.

To generate features to feed into our model, we use different techniques to generate types of features: TF-IDF Vectorisation features, to reflect the importance of words in a sentence, and calculating the number of positive and negative words in a sentence. Using TF-IDF vectorisation as opposed to a simple Count Vectoriser was chosen because it incorporated a frequency element into the counts to give our model a better chance of detecting which words are most important in describing the sentiment of a sentence. Further, we decided to set the hyperparameter ngram\_range to (1, 2) so that it created features for both unigrams and bigrams. This meant we were able to capture the influence of a negative-positive word, such as *not good*, on the sentiment of our sentence – a big strength in this approach.

However, the choice to encode negatives at this point does have its limitations. If we consider the phrase ‘this is not going to have a good effect’, the ‘not’ extends beyond the word ‘going’ to affect the adjective ‘good’. In its current state our pre-processing, would not this into account which is the major limitation in this stage of our pre-processing.

We chose to utilise two more features: to provide the model with the number of positive and negative words in the sentence. Using Vader’s sentiment scoring to deduce which words were positive and which were negative, we felt this was a useful feature to use as it was a direct metric on the sentiment of the sentence’s words ignoring and underlying understanding. This is where both the approach’s strengths and limitations lie. Since it is a direct word scoring, we believe it will give our model a good understanding of the sentence’s contents; however, limitations are that Vader’s word dictionary is not exhaustive, so we cannot produce a score for each word, and it is just a count so will not provide any deeper weighting to more negative or positive words.

**1.2**

To train and test our model effectively, we split our data using a 60/20/20 percentage split into three datasets, train, test, and validation, so that we could train the model, tune the hyperparameters and test our results without data leakage between stages to avoid bias is our results. We then pre-processed each dataset as discussed above and, during the vectorisation stage, ensured that we used the fit\_transform method on the training dataset and only used the transform method on the test and validation sets.

Once our features have been created, we will be using sklearn’s Logistic Regression Classifier to label our texts. This algorithm is chosen due to its robust nature, since we will be dealing with a lot of uncommon observations, and its low prediction and training time when dealing with such a large set of features.

Using the ROC AUC Score as our central performance metric, we created a baseline Logistic Regression model and found severe overfitting with train and validation ROC AUC Scores of 0.9931 and 0.08550 respectively. A good start, but a model that needed some hyperparameter tuning. Utilising Optuna’s SearchCV framework we focussed on tuning ‘C’ between the values of 0.0001 and 1 as this was the key regularisation parameter to reduce to model’s overfitting. We used ten cross-validation folds in this hyperparameter search to confirm our tuning was performing effectively. This search suggested a value of 0.9987 (6 decimal places) for our ‘C’ hyperparameter, which after careful evaluation resulted in train and validation ROC AUC Scores of 0.8572 and 0.8292 respectively. These are much better results since there is a lower difference between our train and test scores, telling us that we had reduced the amount of overfitting in our model whilst simultaneously keeping our performance high.

**1.3**

Text

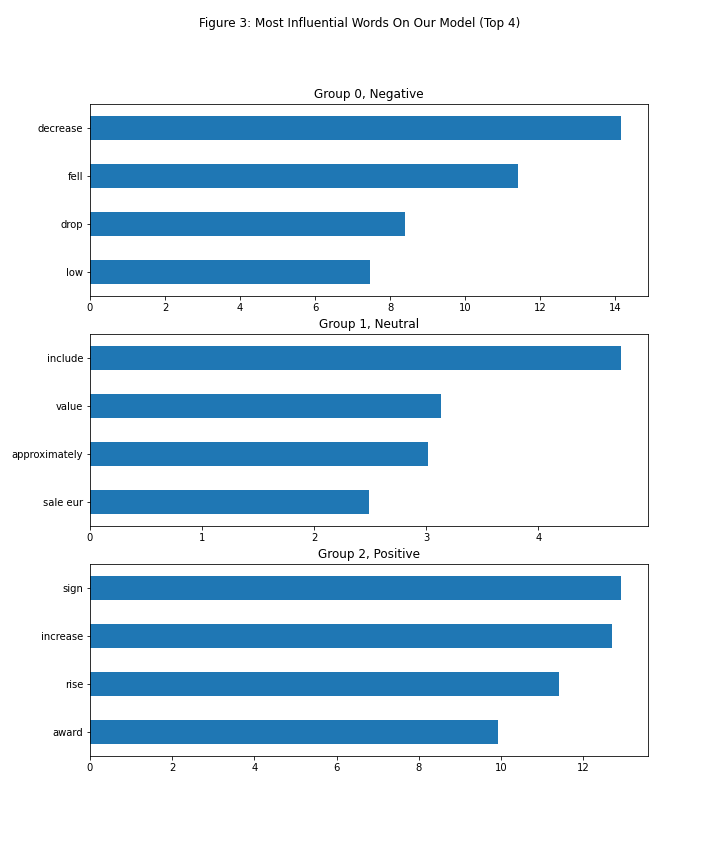
Description automatically generatedTo effectively evaluate our model, we calculated metrics on our test set. Since this was unseen data, it would give us a good understanding of how our model will truly perform. We used four performance metrics in conjunction: ROC AUC Score, Accuracy, Precision and Recall. ROC AUC Score refers to the area under the ROC Curve which measures the level of separability between classes and is useful in giving an overall representation of how well a model performs. Something to be careful of when quoting ROC AUC Scores in evaluation is that it depends entirely on the order of the probabilities, not the values themselves. The implication is that it is not useful for model comparison, just intrinsic model performance. We use Accuracy to give us another view on an aggregate level and can be defined as the proportion of true predictions – quite an intuitive metric. However, we need to be careful of reading too deeply into this value since it can easily misrepresent what is going on. With a model that simply predicted the most common class, one can get a very high accuracy score if there is a class imbalance – something we do have in our dataset as Neutral sentences are significantly the most common. Finally, Precision and Recall are closely linked and can be looked at class-by-class or on an overall level; Precision is the proportion of predicted classes that are true, and Recall is the proportion of actual classes that were correctly predicted. However, due to how these values are defined, there is a trade-off to be made between which type of error one wants to prioritise. Further, used together, these metrics completely ignore the impact of true negative predictions for each class. Now, this completely depends on the situation, but the implication of this is that these metrics do not Graphical user interface, chart

Description automatically generateduse all the information available and so do not give the evaluator the complete picture.

To give an initial view of how our model classifies sentences, the ROC AUC Scores give us a good measure of how our models are performing. As we can see in Figure 1, our model performs well in terms of both performance and robustness; each dataset evaluates to a score between 0.86 and 0.82, giving us high values, and ones which show the model performs consistently across different datasets.

Further, we used a confusion matrix in Figure 2 to provide a deeper look at how the model is classifying our sentences. As we can see below, Neutral sentences are classified most successfully, Negative sentences least successfully and Positive sentences in between. Now, the main thing that stands out is the large number of correctly classified Neutral sentences, supported by a high class-based Recall Score of 0.91. This tells us that if a sentence is Neutral, it is likely to be correctly classified. However, looking at Positive and Negative sentences, we see that this is not the case. In both cases, more sentences were incorrectly classified as Neutral than correctly classified with Positive and Negative classifications evaluating at a Recall Score of 0.41 and 0.49 respectively. Contrastingly, the Precision for each class is in the range of 0.75 and 0.68 telling us that, in the majority, the predicted class tends to be correct – a strength of our model.

It is believed this is due to the imbalance in the data whereby most sentences are Neutral sentences. This can be seen since the class-based Recall and Precision Scores, are directly correlated with the class imbalance that is present in the initial data. To improve our method and combat this, it would be prudent to use some re-sampling techniques such as Random Oversampling, SMOTE Oversampling or Random Undersampling. This will leave us with a more balanced dataset to train our results on and give our model a better chance of learning how to discern Neutral sentences from Positive and Negative sentences. Moreover, changing the algorithm used for our model may give us better results since the Logistic Regression algorithm is limited to a linear decision boundary which could have had a negative influence on our misclassifications.



Finally, we used Figure 3, which shows the largest four word-vectorised feature coefficients for each class, to ascertain which words were having the most impact on classifying our sentences. We note that coefficients can be used since all word-vectorised features are normalised. This was very interesting since the words that the model found most useful are words innately relevant to each class. Moreover, it is useful to see that, for Neutral sentences specifically, the bigram ‘sale eur’ ranked as one of the most influential words. This highlights that setting our ngram\_range to include bigrams was helpful for our model to understand sentence structure. Finally, it was interesting to note that the largest coefficients of the Neutral class were far lower than the coefficients of the Positive and Negative classes. This could suggest that, for Neutral sentences, decisions were made based on a broader range of words; whereas, for Positive and Negative classes, specific words were more important in determining the decision made. The implication of this is that our model may have missed more nuanced word influences and numerous words coming together to suggest a particular sentiment as opposed to being defined by a specific word.

Developments based on Figure 3, would be to introduce some additional pre-processing techniques such that structure across the sentence can be seen on specific words. This could be done by adding prefixes or extending the ngram\_range to include longer terms whilst also raising the minimum frequency requirements that we would deem acceptable to include a feature into our model.

Task 2

**2.1**

To create our named entity recognition tagger, we pre-processed our sentences to get word features to put into sklearn-crfsuite’s CRF algorithm. Its main strengths lie in the large range of features we have to calculate such as word-specific features and features that give us information on the previous and subsequent word. Further, because we are using this CRF algorithm, we can introduce regularisation to control the impact these features have to make the model as robust as possible.

The specific features we have opted for can be split into three groups: features for the current word the model is evaluating, features for the previous word and features for the subsequent word. In every three cases, the features calculated are the same for each respective word thus I shall only cover the calculations we make, but every calculation is relevant to each of the three words! Firstly, we include a ‘bias’ feature to allow us to shift the algorithm’s activation function to optimise learning. We also have several word-specific features such as the word itself (set to lowercase), the first three letters of the word, the last three letters of the word and the last two letters of the word. These features are so that we can capture any prefixes or suffixes that the word may contain as our model may find it useful in determining what type of word it is – the suffix of ‘tagging’ highlights this word as a verb so is unlikely to be an entity. Further, we included five Boolean features depicting if the word is uppercase, a digit, in title case, punctuation or in nltk’s stopword list. These will be useful since casing contributes heavily to a human’s entity recognition processing so our model should recognise these connections. Moreover, the stopword indicator should help the model differentiate between entities and the O tag as we should not find entities in the stopword list. To conclude our feature list present for the current, previous, and subsequent words, we include the part of speech tag for the word as it will give the model information on how the word fits into the sentence structure, providing the model with another layer of information. In the cases that the current word is the start or end of the string, we mark this word with a beginning of string or end of string identifier since the current word will not have any previous or subsequent words respectively to analyse.

The tagging scheme we use to label entities in this dataset is the IO (inside-outside) scheme whereby O indicates the word belongs to no entity and an I- prefix signifies the tag is part of an entity chuck. The limitation of this scheme is that it cannot correctly identify consecutive entities of the same type. But looking at the unique labels in our train data, we find that we only have O tags and tags with I- prefixes suggesting that this tagging scheme is the most appropriate for our model.

**2.2**

Bringing these pre-processing ideas together, we start by dividing our dataset into train, test and validation datasets using a 60/20/20 split so that we have suitable data to train our model, validate its performance and test the results. We create the features for each word in the train, test, and validation so that we have three datasets. This comes in the form of a list of lists, one for each sentence, with dictionaries as its elements, containing each word’s features. We do not need to manipulate the labels in any way as they are already in the desired form.

To train our sklearn-crfsuite, CRF model, the framework works similarly to routine sklearn classes such that we initiate the model and then execute a fit method with the train data as arguments. We then evaluate our fitted model using the F1 Score as our metric. Sklearn-crfsuite has versions of some performance metrics such that they work with vectorised data in a layered format. We calculate an F1 Score for both the train and validation datasets to give us an idea of how well the model performs and how robust the model is. We look for high F1 Score values for good performance and we look for close train and validation values to show the robustness and show that the model performs similarly across seen and unseen data. Based on the values we see, we adjust the CRF hyperparameters, c1 and c2, to try and improve the performance and robustness of our model.

**2.3**

As mentioned, we chose the F1 Score as our central metric to evaluate the performance of our model. However, we will also be using accuracy, precision and recall scores to give us a fuller picture of how our model is performing as they contain direct information on different classification outcomes such as False Positives, False Negatives and True Positives. We have chosen the F1 Score as our central performance metric since it is the harmonic mean of the precision and recall scores. Thus, it helps tune both metrics simultaneously and so captures many results-oriented statistics in one. To help improve the validity of our F1, precision and recall scores, we remove O tags from our calculations to help adjust for the data imbalance.

The first limitation of these metrics is that they omit the impact of successfully not-misclassifying. So, if this model was used for a context whereby misclassification has detrimental impacts on the user, then these metrics would not instil confidence. Additionally, it is important to be aware that an imbalanced dataset would have a deceptive impact on accuracy score. Moreover, in our NER-tagger context, one cannot specify which labels to look at. With rare classification events, it gives a very skewed idea of errors that are being made. Finally, precision and recall Scores both have the limitation that each only provides half of the information and so we only get the full picture if we use them in conjunction.

Chart, bar chart

Description automatically generatedTo start, we can quote the performance metrics for each dataset to give us information on how well it has been fit and how well it performs. As we can see in Figure 1, we can see that accuracy is consistent across each dataset which gives us a positive, overall view of the classification robustness across datasets. However, looking at the F1 scores, we can see that the post-tuned model is slightly overfitted since our training and validation sets have a 0.06 gap (0.9106 and 0.8547); however, this is the closest we could get the values in tuning whilst maintaining a suitable performance. We see a similar, overfit pattern for our precision and recall scores. However, in terms of intrinsic performance, we need to look at the performance metrics for our test dataset since it is our unseen sample. In all cases, these performance metrics are lower than the train and validation scores. We have F1, precision and recall scores of 0.6958, 0.8576 and 0.6101 respectively which tells us that our model is not perfect, but, since we have excluded O tags from these calculations, this is particularly positive considering the imbalance in data. Recall shows a particular weakness in our performance metrics, telling us that our model is not effective at identifying entities. Moreover, looking at the model’s classification report, we can see which Calendar

Description automatically generatedentities are particularly harming our aggregate score. Our model is worst at identifying organisations and miscellaneous items with recall scores of 0.25 and 0.14 respectively. A possible source of this is the lack of observations for these categories – there are only seven miscellaneous entities in our test set – as our model has not been given the chance to learn about these entity patterns. However, an equally likely cause of this weakness is that we do not have suitable features to help the model identify these types of entities. In the future, to boost our Organisation and Location success, we make use of a gazetteer to allow us to look up particular words and check if they are included in the organisation and location dictionaries.

Chart, bar chart

Description automatically generatedDigging deeper into these test metrics, this idea is supported by the confusion matrix found in Figure 2. Our model is most successful at correctly giving words O tags. It is also very successful at recognising people. With precision and recall scores of 1.0 and 0.84 respectively, we see that there are few cases of misclassification. Looking at the two sides of the confusion matrix, we notice that if our model had given a predicted label other than O, this label is likely to be correct, whereas, except for the person tag, an entity is more likely to be given an O tag than correctly classified. This tells us that our model is not good enough at understanding the difference between something that is not an entity (O) and something that is. But when it does classify a word as an entity, it is likely to be a correct classification. To counter this, expanding the features inputted into the model would be very useful as it would give the model more information to work with in determining the meaning. As we can see in Figure 3, a bar chart showing the 10 largest features that impacted the model’s decision making, the word in question appears as an impactful feature frequently. This reinforces the idea that we need to expand our feature selection because the model is missing out on a lot of underlying knowledge. One of the few cases that it seems to have picked up on underlying word structure is the case where the word is in the title case, so unlikely to be tagged with O (seen by the large negative weight on this feature). As a final note, the bias feature having such a high weight towards the O tag highlights why we are getting such low recall scores: there are too few important features, so the model will just default to the O tag.

**2.4**

To apply our NER tagger to the Financial Phrasebank dataset, we need to use our pre-processing functions to generate the required features for our model. Since the Financial Phrasebank is split into two sets, we concatenate the train and test sets together and then run the entire data through our pre-processing functions. After doing this we can generate predictions on our model to classify each word. Since the Financial Phrasebank has sentiment scores attached to each sentence, we can easily compute a sentiment score for each entity that the model finds. We do this by mapping negative sentiment to a score of -1, positive sentiment to a score of 1 and neutral sentiment to 0. This means, that once the model is run and we have generated predictions, we can find the overall sentiment of a particular entity using simple addition.

Since our NER Tagger uses the inside-outside scheme, the tags that we can extract are on a word-by-word basis i.e., Goldman Sachs is tagged as an organisation, but each word is tagged as I-ORG separately. Thus, we process the predictions such that if there are consecutive entity tags, then we merge them. So, we can accurately analyse the sentiment for organisations with our tagging limitations.

As we can see in Figure 4, we see that there is a greater range of positive sentiment scores than negative sentiment scores. This is probably due to the data imbalance in the Financial Phrasebook Dataset – there are over double the number of positive sentences than negative sentences in this dataset. What also jumps out in these results is ‘bank’/ ‘Bank’ appearing in both lists and ‘Rapala VMC Corporation STOCK EXCHANGES RELEASE October’ including more information that we could wish. Unfortunately, both issues are probably due to our pre-processing and our model does not understand the difference between capitalisations – something that could be reduced by more entities for the model to learn from and features that look at the word connections more closely.

Figure 4: Table to show the five most positive and negative organisations in the Financial Phrasebank Dataset as predicted by our NER tagger model.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Five Most Positive Organisations | |  | Five Most Negative Organisations | |
| Organisation | Overall Sentiment Score |  | Organisation | Overall Sentiment Score |
| Bank | 5 |  | Raute Corporation | -1 |
| Suominen Corporation | 4 |  | Marimekko Corporation | -1 |
| Nokia Siemens Networks | 2 |  | Rapala VMC Corporation STOCK EXCHANGES RELEASE October | -1 |
| Incap Corporation of Finland | 1 |  | Burrill Life Sciences Capital Fund | -1 |
| KONE Corporation | 1 |  | bank | -1 |

Task 3

**3.1**

There are three types of visualisations used in tasks 1.3 and 2.3: a vertical bar chart, a horizontal bar chart and a confusion matrix in the structure of a heatmap. With regards to the chosen marks, we used bars to show the magnitude of our metrics since, when aligned, it is easier for the human eye to make a comparison instead of a point or line.

We also used a lot of different motion channels to display our information. Starting with colour, we had two situations depending on the number of dimensions we wanted to display. Firstly, for all figures in task 1.3 and figure 2.3.2, we used black and white as our display colours. This is because black and white are the most contrasting colours we could use which is important. After all, contrast helps enhance both identity and magnitude channels: high contrast highlights a mark’s location and its size in comparison to other marks. Moreover, with regards to our heatmap confusion matrices, we use colour to highlight large values which naturally draws the human eye towards it – using colour as a pop-out mechanism allows for efficient processing as, at-a-glance, the user can determine what the visualisation message is. In instances where we do need colour, such as for Figure 2.3.1, we use contrasting colours as opposed to a gradient design because we do not need colour to infer order, we use colour to help differentiate between different datasets.

We also utilise size as a motion channel in Figures 1.3.1, 1.3.3, 2.3.1 and 2.3.3 with bar charts. We use the size of the bars to represent the quantitative attribute for each categorical attribute that we have. This is because we have one quantitative attribute that we are interested in and the size of bars as a motion channel is very effective for comparing the size of two values since we can utilise the proximity of the bars to highlight large or small values. Moreover, for our bar charts, it is more important to compare the quantitative attributes for each categorical attribute than to understand exactly what the figures are. This is because, to evaluate our model, we require performance metrics to be high and close together across datasets which can effectively be achieved using size as a motion channel.

We decided to opt for a horizontal bar chart for Figures 1.3.3 and 2.3.3 since the categorical attributes to evaluate were words that needed to be clear. Thus, having the categorical attribute along the y-axis and using a horizontal motion channel meant that the overall readability of the visualisation improved.

**3.2**

As mentioned in 3.1, there are three types of visualisations we use. We will start by discussing the heatmap confusion matrices. We use colour as a pop-out mechanism to highlight our primary attribute. The first thing that the user queries is the mark that has the darkest colour followed by the other marks, as the colours get lighter. After deciding where the most important marks are, the user then looks are the labels for these marks to determine the relationship that the important marks are representing.

With regards to our vertical bar chart in Figures 1.3.1, the first visual query that the user performs is identifying the largest bar in the visualisation since it is definitively larger than the others. The user then queries both other bars since they are very similar in size and then performs a tertiary query to determine which bar is larger. This tertiary query takes the longest, but the focus of the visualisation is to display that the metrics are similar in size. Figure 2.3.1 works slightly differently in that most marks are large so the first visual query that is made is to identify the smallest bars in the visualisation: F1 and recall scores for the test dataset. The mark representing the F1 score is the first to be processed followed by the recall mark since it is markedly smaller than the other values, but not unnoticeable. The user then queries the precision marks since they are the next set of marks that are contrasting. The accuracy marks are the next visual query that is made since there is very little size contrast so does not draw the eye as much. The user then queries to determine the labels for the interesting parts of the plot to understand the relationship that is being shown.

Finally, the first query that is made when observing the horizontal bar chart in Figure 1.3.3 is the three largest bars in the neutral plot since they are central and come together to be a large target for the user. The next query is made towards the largest marks for the negative and positive plots (‘decrease’ and ‘rise’). This is because they are the largest contrasting marks in comparison to the other marks in each plot. The user then queries the labels for each bar to understand which words are being shown to be the most important. For Figure 2.3.3, the first query that is made is at the top, with the largest marks displaying the most influential features since they stand out due to size being an effective motion channel. The user then queries the only negative bar, at the bottom of the plot, since it is in the other direction to the other marks so inherently stands out. The next query that is made is to observe the colours in the plot to obtain the next level of information. The final query is to determine which features are the most influential.

Task 4

**4.1**

To show if there was a link between an author’s number of hours of writing per day and their total output, we decided to use a scatter plot to show this relationship. This approach was chosen because we utilise many motion channels effectively to create this visualisation. Since we want to display the relationship between two variables, we employ the use of both position motion channels to represent each observation as a point. This is because, it is easy to compare observations and, altogether, we can see if there is a link between these two variables. Thus, our visualisation satisfies **Munzer’s [1]** second level of visualisation validation, Task Abstraction, by displaying the relationship between the two variables.

Moreover, we also added a linear line of best fit to summarise the data and to give the user a marker of if there was a link between the two variables at-a-glance. We chose a linear model to accurately display the link between these two variables since we could easily ground the line to cross the y-axis at zero – it makes sense that zero hours of writing would lead to zero pages written. We also show the relationship with clarity because a straight line is very easy to understand – providing information to determine if there is a link or not, but no more information than what is required. This is important because it will lower the time taken for a human to recognise any link, and so further adheres to **Munzer’s [1]** third level of visualisation validation, Visual Encoding.

To visualise if there is a link between the total number of book sales and the average book rating, we chose to use a scatter plot to display these two variables. For similar reasons to the first visualisation, since we wanted to show the relationship between two variables, the most effective way to achieve this is to

With regards to both plots we have mentioned, we have also used colour motion channels in two ways to highlight techniques in this visualisation. Our marks have been made black as this colour is the most noticeable against a white background. We have chosen this channel for the marks since they are the most important attribute in terms of informing the user on the relationship between the two variables. Since the line of best fit is also very useful for the user, we colour it red to also highlight this further. Red was chosen because it is contrasting with both black and white and so it has high levels of noticeability against both the background and marks.

Again, for both plots mentioned, we included a Tooltip to provide additional, mark-specific information that would not be essential in determining a link between the two variables but would be useful to the user if they wanted to have a deeper look at any specific variables. The importance of this is so that the user does not leave the visualisation with any further questions regarding the link.

**4.1** In about two pages, write a short description of the visualization techniques you used and a justification for your choices. You should refer to the principles of info vis, relevant aspects of human perception and cognition, and the scientific literature where appropriate.

**4.2.** Using appropriate levels and types of validation (as in Chapter 4 of Munzner and the lectures from week 2), assess the quality of your visualization by making appropriate measurements and observations of the other students in your group (the groups will be defined separately) in an analytic task using your visualisation. The lab class on 25th April will be dedicated to this activity, so you will need a complete visualization by then. Your report on this should be no more than one page. (10 marks).