**Method One**Diagram

Description automatically generated **– Sudo Document Training**

A corpus of documents with existing truth labels are used as training data. Each document is used to create a sudo document containing a concatenation of the abstracts of each entity instance within the document. The features are extracted from each of the training sudo documents and passed into a learning algorithm. The learning algorithm is used by a classifier to generate predictions on standalone-unlabelled entity abstracts. The output of the classifier is the probability prediction of whether the abstract input is sensitive or not.

*Diagram

Description automatically generated***Method Two – Lone Abstract Training**

A set of distinct entities and their abstracts are retrieved from a labelled corpus of documents. Entities are assigned the truth values of the most sensitive document they appear in. For example, if an entity appears in a document labelled as sensitive – then the entity abstract is labelled sensitive also. If an entity appears in no sensitive documents – then the entity abstract is labelled as not sensitive. Features are extracted from each entity abstract, with a learning algorithm utilising k-fold cross-validation to make sensitivity predictions for each entity.

**One-Hot vs TF-IDF Feature Extraction**

When extracting features from entity abstracts or sudo documents I was most comfortable with one-hot encoding however I recognised that TF-IDF was a viable option in this scenario. One-hot encoding involves representing a document as a vector of binary values of whether each word in the vocabulary appears in a given document or not. TF-IDF encoding utilises the frequency of documents of which a given word does not appear in (IDF) as well as the frequency of each word in a given document (DF) to produce a sparse vector representation of each word in a given document. One-hot encoding does not take into account frequency of words due to the binary nature of the vector representation, whereas TF-IDF encoding presents a far more effective way of representing the frequency and relevancy of words in a given document. Given that capturing the relevance of certain words in entity abstracts was crucial, I felt TF-IDF encoding was more suitable in this scenario.

**Down Sampling Training Data**

In the process of experimenting with both classification methods, I noticed that the results were appearing heavily skewed towards predicting unsensitive labels. Upon inspecting class labels within the training data, unsensitive documents were far more frequent than documents labelled sensitive. Assuming this was the cause of the skewed predictions, I applied down sampling to the training data. Down sampling is the process of randomly sampling observations from the more frequent class label such that the number of observations used in the training data is equal for each class label. By randomly sampling an equal number of entities assigned an unsensitive label as the frequency of entities that were labelled sensitive, the training data resulted in far more intuitive predictions from the trained model.

**Results**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1** | **F2** | **BAC** |
| **Sudo Document** | 0.2305 | 0.8436 | 0.3621 | 0.5507 | 0.5796 |
| **Lone Abstract** | 0.3285 | 0.0478 | 0.0835 | 0.0577 | 0.5120 |

**Choice of Classifier**

|  |  |
| --- | --- |
|  | **BAC** |
| **Stochastic Gradient Descent** | 0.5796 |
| **Multi-Layer Perceptron** | 0.5147 |
| **Multinomial Naïve Bayes** | 0.5009 |
| **Linear SVC** | 0.5050 |

**Conclusion**