
# INF554

Sakula HYS & Ghada BEN SLAMA
sakula.hys@polytechnique.edu
ghada.ben-slama@polytechnique.edu
(Les Cupidons)

# 1 INTRODUCTION

To adress the problem of extractive summarization, we propose two different approaches. While one is based solely on the text property of the data, the second one uses the graph properties given between the sentences of the whole data. While the pure LSTM approach scored higher at first, with an F1-score of 0.535, eventually the graph approach combined to LTSM was the best we could get, with a slightly better score of 0.557.

# 2 FEATURE SELECTION / EXTRACTION

We have two types of features: one in a JSON file and another in a TXT file. The JSON file provides textual data and information about the speaker, while the TXT file represents the graph relationships between the textual data.
Our approach was based on constructing a graph with nodes representing concatenated text data and speaker information. The edges of the graph signify connectivity, and the edge attributes denote the type of relationship between the data. This construction enables the computation of attention coefficients for training.
The process [3] of preprocessing of the data involves several sequential steps to enhance the quality and uniformity of textual data. Initially, we remove angle brackets from each sentence, and we eliminate punctuation and digits. We use then tokenization using the NLTK library to break down sentences into individual words. Stopwords which is a already prepared list in NLTK library represting the unuseful words, are then removed from the list of tokens. We do then lemmatization process which refines words to their base forms, which means that we consider the word 'use' and 'useful' as the same word. The final step involves filtering the lemmatized words, retaining those present in a predefined list of allowed words or excluding non-alphabetic terms.we convert then all words to lowercase.This allows to preprocess the text in a way to train our model on a clean text and to avoid having outliers.

We identified a substantial class imbalance in the dataset, where 80% of instances had a label of 0, and only 20% had a label of 1. This imbalance introduced a noticeable bias in the models, especially evident when employing the Graph Convolutional Network (GCN). Even when experimenting with the Graph Attention Network (GAT) model and suboptimal parameter choices, the models consistently leaned towards predicting 0. In order to tackle this problem, we chose a good pos_weight parameter in the BCELoss which is explained in the second section.
To leverage information from diverse sources within our dataset, we implemented distinct embedding strategies tailored to the nature of the data. For the text data, we employed BERT (Bidirectional Encoder Representations from Transformers) for embedding. BERT is well-suited

for understanding context and relationships within textual information.
For the speaker-related data, we opted for one-hot encoding as the embedding technique. One-hot encoding is a binary representation where each speaker is uniquely represented by a binary vector, ensuring that the model can effectively distinguish between different speakers.
Furthermore, recognizing the importance of capturing relationships between text data, we have to do embedding of edge attributes using the one-hot encoding.

# 3 MODEL CHOICE, TUNING AND COMPARISON

The decision to utilize a Graph Neural Network was driven by the need to effectively exploit the graph data inherent in our dataset. This was particularly advantageous for our problemn where the data provided involves text and graph data. However, our data did not align well with linearity, making SVM and any other linear model less suitable for our specific problem.
An attempt was made to implement a Graph Convolutional Network (GCN) to exclusively leverage the graph structure along with text data and edges (without edge attributes). Unfortunately, the GCN consistently predicted 0, suggesting potential challenges or limitations in the model's ability to learn from the given features. This outcome prompted further investigation into parameter tuning and model architecture adjustments.
We believe that models like XGBoost have the potential to deliver good results. However, not incorporating graph data might be suboptimal for our specific problem. Given that our dataset includes both text and graph data, neglecting the graph structure could result in a loss of valuable information. Therefore, we have chosen to focus on graph neural models such as Graph Attention Networks (GAT).
In order to test a model applied only on text data, we used LTSM model and we found good predictions with F1 score 0.535, so we decided to make a merge between GAT and LTSM and have some new neural network with 2 layers of GAT and 1 layer of LTSM which gave a score of 0.556.

## 3.1 LTSM Model

Long Short Term Memory is a type of recurrent neural network architecture that allows to learn long-range word dependencies in a text. This type of model is effective for text calssification and summarization, since a contextual analysis of each element can be done.
The main idea of this model is that when computing the output of an element, the input is partially made of the output data from the previous element.
As a sub-part of RNN, an LSTM cell looks similar to an RNN cell. The architecture is composed of 3 parts : a first part where it is decided how much of past information is kept, a second part

where new information is added, and a last part that passes the aggregated information onto the next cell. LSTM network gets its name from the fact that there is two type of memory lines throughout the entire process. A short term memory line (hidden state), that is only depending on the previous cell, and a long term memory line (cell state), that gathered information from all the previous cells.

## 3.2 GAT Model

Graph Attention Network (GAT) is a type of neural network architecture designed for processing graph-structured data. The key idea behind GAT[2] is to enable nodes in a graph to selectively attend to their neighbors' information, allowing the model to focus on relevant nodes when making predictions.

GAT employs a self-attention mechanism to assign attention coefficients to neighboring nodes based on their feature representations. The attention coefficients are computed using a shared learnable attention mechanism.[1]

Each node is represented by an initial feature vector $x_i^0$

The attention coefficients: $\alpha_{i,j} = \frac{\exp\left(\mathbf{a}^\top \text{LeakyReLU}(\mathbf{W}[\mathbf{x}_i \,\|\, \mathbf{x}_j])\right)}{\sum_{k \in \mathcal{N}(i)\cup\{i\}} \exp\left(\mathbf{a}^\top \text{LeakyReLU}(\mathbf{W}[\mathbf{x}_i \,\|\, \mathbf{x}_k])\right)}$.

Where $a$ is learnable vector and $W$ is learnable matrix and $||$ represents the concatenation. Then, the feature vector becomes: $x_i^{l+1} = \sigma(\sum_{j \in \mathcal{N}(i)} \alpha_{ij}^l W^l x_j^l)$ with $\sigma$ is the activation function. The distinction between GAT and GCN as explored in the lab, lies in their treatment of neighboring nodes. GCN treats all neighbors equally by utilizing the adjacency matrix, while GAT assigns importance to specific node neighbors over others.
This concept is crucial in our task because we have edge attributes that provide valuable information about the relationships between nodes, and our goal is to leverage this information for more effective modeling.

## 3.3 Our Model

Our model is a combination of LTSM and GAT model, using 2 layers of GAT and 1 layer of LTSM.

To prevent overfitting, we set the drop parameter to 0.2. In light of the imbalanced class distribution, we addressed this disparity by choosing the pos_weight parameter in the Binary Cross-Entropy (BCE) loss function. The purpose behind selecting a pos_weight greater than 1 was to assign more significance to the positive class (1) during model training, compensating for its lower representation in the dataset. Specifically, we set the pos_weight to the inverse of the class distribution, calculated as 0.8/0.2. The parameters for the hidden_layers were chosen through experimentation to achieve the maximum F1 score.

# REFERENCES

[1] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks?, 2022.

[2] https://github.com/pyg team/pytorch$_{g}eometric/blob/master/examples/gat.py.Gat$.

[3] https://github.com/tkeldenich/NLP$_{P}reprocessing/blob/main/NLP_{P}reprocessing.ipynb.Nlp_{p}reprocessi$