10701 Spring 2022 Project: Semi-Supervised Learning

William Zhang
Department of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15289
wz2@andrew.cmu.edu

Zachary Esakof
Department of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15289
zesakof@andrew.cmu.edu

Luis Reyna
Department of Materials Science
Carnegie Mellon University
Pittsburgh, PA 15289
lreynade@andrew.cmu.edu

Thomas Tam
Heinz College
Carnegie Mellon University
Pittsburgh, PA 15289
yuchowt@andrew.cmu.edu

Introduction

In the real world, it is very easy to acquire a vast amount of unlabeled data, from either vast data troves available online or through Internet scrapers. However, labeling this data typically requires significant time and effort from numerous experts. Semi-supervised learning techniques have had considerable success in training classifiers comparable to those produced by supervised learning, despite their use of limited labeled data. Due to the high cost of employing domain experts to label data, this exciting area of research also holds high value for organizations. Furthermore, interest in this field has produced entire textbooks dedicated to semi-supervised learning (Zhu and Goldberg, 2009). Our paper assesses the feasibility of utilizing semi-supervised learning techniques in performing multi-modal classification on social media data from natural disasters. By reducing the time required for data labeling without sacrificing accuracy, we believe that applying semi-supervised techniques in this domain can produce significant value for disaster response teams.

Data

Our paper uses the CrisisMMD dataset, which is a collection of disaster-related tweets scraped from Twitter. This dataset satisfies several constraints of our problem. First, social media posts created during disasters are usually multi-modal in nature, with both textual and visual content present. Although the dataset is fully labeled, the creators do note that labeling the dataset required significant effort by multiple experts that occasionally disagreed. As such, this dataset provides a useful basis for evaluating different multi-modal semi-supervised learning techniques relative to a fully supervised baseline (Ofli et al, 2020).
Now, several properties of the dataset will be discussed that are relevant for our purposes. The dataset is gathered from tweets during several natural disasters over the past decade. Due to constraints, the dataset creators filtered and sampled all the collected tweets such that the dataset contains only multimodal data and the dataset does not contain any duplicate tweets.
The dataset contains approximately 16,000 tweets and 18,000 associated images. Note that a single tweet can have 0, 1, or multiple images associated with it. We are primarily concerned with the task of determining whether a given tweet is considered informative for purposes of humanitarian aid, which requires the tweet and image to contain evidence of an immediate need for aid. We describe our preprocessing operations in a later section.

Related Work

Cotraining

This particular CrisisMMD dataset has been utilized for a multi-modal supervised learning classifier before (Ofli et al, 2020). The prior approach takes a supervised learning approach, training a text-only classifier, an image-only classifier, and a multi-modal classifier. For the text-only classifier, the paper utilizes a pre-trained word2vec model to convert each tweet into a word-level matrix before applying a series of convolution layers, a max pooling layer, and fully connected layers to produce a final classification. The image-only classifier utilizes the VGG16 model, while adapting the final layer to this particular classification task. The paper's multi-modal model runs the combined output layer of the text model and the image model through dense layers to produce a final prediction. The paper's approach to multi-modal learning involves training the two classifiers jointly (i.e., the loss is backpropagated through both classifier stacks), with each classifier utilizing its own modality's features.

Our sketch of the co-training algorithm utilizing a text and image model is presented below (Figure 1). The specific parameters used in our co-training implementation will be discussed in Section 5 (Methods). Co-training utilizes multiple classifiers; each classifier is built over a different modality of the available data. In each iteration of co-training, we train each modality's classifier on the labeled training set, use those classifiers to select the most confident positive and negative predictions from each model's candidate set, and add those most confident predictions to the labeled training set for the next iteration. As such, co-training augments the labeled training set with the predictions of constituent classifiers with the goal of training a better classifier in the next iteration (Blum and Mitchell, 1998).

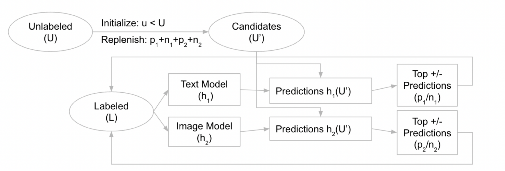

Figure 1: Cotraining Diagram

It is worth noting that the performance of co-training does depend on the classifiers being sufficient and independent. For sufficient, each classifier's feature set alone is capable of performing classification on the data set. For the independent requirement, the feature sets utilized by the classifiers should be independent. In other words, co-training can be thought of as an iterative approach to building a classifier by combining multiple classifiers that use distinct slices of the available input features.

In NLP models, sequence-aligned recurrent neural networks or convolutions are used to try and capture the temporal and contextual nature associated with words in a text (Vaswani et al, 2017). The transformer paper introduces an encoder-decoder architecture that is able to capture sequence relationships globally and has shown to perform well on translations tasks along with requiring less training time. It is also worth noting that since the transformer is structured as an encoder-decoder, we can also use the transformer to learn an embedding for a given phrase by taking the output of the encoder stack.

Graph Models

Graph-based models are a fit for semi-supervised learning problems that have an underlying graph structure. With its many connections between individuals and organizations, social media data is ripe for applying these graphical methods. Common applications are to estimate relationships between individuals, such as Korean politicians (Yoon and Park, 2014). Other common applications include user sentiment analysis, which incorporates information from the user and temporally relevant information that they post or interact with (Iyer et al, 2019). This latter example overlaps nicely with our work in that the authors built a semi-supervised graph to determine the emotions associated with a Twitter post. In our work, we also build a semi-supervised graph, but predict whether a Twitter post is informative, or helpful, in a natural or humanitarian disaster. Where many social media graphs rely

on network features such as likes, retweets, and followings (Iyer et al, 2019), our graph focuses on a post's direct content. Furthermore, we develop a multimodal graph that incorporates features from both text and image modalities.
Two fundamental assumptions are made to construct a semi-supervised mapping (Zhu and Goldberg, 2009):

- Given a labeled data point, the prediction of the mapping is close to the correct label.
- The mapping has to be smooth on the graph.

These semi-supervised graph models are constructed using a training set that consists of labeled and unlabeled points. The graph represents the data points as nodes, and the edges connecting them are characterized by weights. The weights are of great importance as they are used to store information that is used to understand relationships between pairs of points. A higher weight between two nodes therefore indicates a higher degree of similarity between two points and, in turn, between their labels. To complete the graph, labels from the labeled points are propagated to the unlabeled regions or the unlabeled points are given random labels (Zhu and Goldberg, 2009). In both cases, producing labels for unlabeled training data is the main task of training these models.

Data Preparation

The dataset is preprocessed following the baseline presented by Ofli et al. In particular, we remove stop words, non-ASCII characters, numbers, URLs, hashtag signs, and punctuation marks with white-spaces from the textual data. For the image data, we resize each image to 256x256, scale the pixel values to a numeric value between 0 and 1 before normalizing the pixel values with respect to the ImageNet dataset. Normalizing the pixels with respect to the ImageNet dataset allows us to utilize the pre-trained VGG16 model.
Furthermore, since each modality's labels for the dataset were annotated separately, there are cases where the text label for a given tweet disagrees with the image label for the same tweet. In a similar fashion to the baseline, we remove any tweets where the text and image labels disagree. This operation produces a total of 11,400 text and 12,708 images. We also note that a tweet in the dataset can contain multiple images; in these cases, we arbitrarily select the first image to achieve an equal number of text and image data points.
We use the train-test-dev split provided by the dataset. After removing conflicting labels, we are left with a 70% train set of 8066 tweets and a 15% test set of 1518 tweets. The dataset also contains an uneven split, with about twice as many tweets labeled as informative, positive class, than uninformative, the negative class.

Methods

Co-Training Model

Naive Bayes Text Model
The "bag of words" model is used to featurize each tweet's text into a vector representation based on the constituent words. The "bag of words" corpus is all words within the training data set of tweets. A naive bayes classifier is then built on top of the "bag of words" model. To handle the situation where a tweet contains a word not present in the "bag of words" model from driving the likelihood to zero, we impose a Beta(1.001, 1.9) prior and set the parameter estimates via MAP.

Alternative Transformer Text Model
In this architecture, each tweet is tokenized into a sequence of integers. As such, each tweet text is converted into a sequence of integers. The sequence of integers is passed through an embedding layer, which converts each integer into a word embedding, before being pushed a single transformer encoder layer. A transformer encoder layer augments the input embedding with a positional element before pushing it through a 2-head multi-attention layer, then a normalization with a residual connection, then a fully connected feed forward network, then outputting after normalization with a residual connection. The output of the transformer encoder layer is passed through a global average pooling to reduce to a single feature vector before a 64 neuron layer (with 0.1 dropout and ReLU

activation) before outputting through a 2 neuron layer with softmax activation.

As mentioned in the transformer architecture paper, we treat the word embedding size as a hyperparameter. The model will determine the embedding layer weights over the course of training the model through standard backpropagation using the default Keras Adam optimizer weights under binary cross-entropy loss. Similarly, we scale the number of epochs by 50 + 2 * iterations with early stopping engaged at 10 epochs of less than 1% accuracy increase.

VGG16 Image Model
The VGG16 model was a state of the art model proposed in 2016 that won the ImageNet competition in 2014. Instead of training a convolutional neural network from scratch, we decided to fine-tune the VGG16 model for our purposes, in a similar fashion to the baseline. To better fit the VGG16 model to our particular classification task, we flatten the output of the VGG16 model before passing it through a fully connected layer with 64 neurons (with ReLU activation and using 0.5 dropout) and finally through an output layer of 2 neurons with softmax activation.
Only the weights of layers after the VGG16 output are refined through classical backpropagation. We utilize the Adam optimizer with default Keras parameters with a binary cross entropy loss since we have a binary classification problem. The model is trained for at most 100 + 10 * iteration epochs with the model achieving the highest accuracy retained as the final model. This function allows more opportunities to reach an optimal model as the available training data increases over time. We stop training the model once we have encountered a run of 20 epochs where the accuracy has not increased by 1% – this early stopping allows the image model to break out of local minima.

Co-Training Model Parameters
The co-training process is also controlled by several distinct parameters that control how much the dataset evolves over iterations and the degree to which the constituent classifiers are refined. The parameters are discussed in further detail below: number of positive examples (P), number of negative examples (N), initial labeled training set (L), initial unlabeled set (U'), and number of iterations (I). Developing code to orchestrate these various parameters and datasets was another body of work pursued as part of this paper.

I - This parameter controls the number of iterations that the co-training process runs for. Under our setup, this parameter controls the number of times the constituent models are re-trained from scratch on an augmented labeled training set. The labeled training set is replenished at a rate controlled by P and N for each iteration *I*.

L - This parameter controls the size of the initial labeled training set. The initial labeled training set is a randomly selected subset of the baseline training dataset. We strip labels from the remaining training data points and use them to populate the candidate set, U'. We do not take steps to preserve the balance between the informative and noninformative labels.

U' - This parameter controls the size of the unlabeled candidate data points that each iteration considers adding to the labeled set. This unlabeled set is a randomly selected subset of the remaining unlabeled training data after selecting the initial labeled training set, L. In each iteration, the classifiers will only attempt to label U' as opposed to all the unlabeled data available. Furthermore, once the classifiers have added their most confident points to the labeled data, U' will be replenished from the larger pool of unlabeled training data. That is, the size of U' at the beginning of each iteration remains constant unless all of the unlabeled training data is exhausted.

P/N - These parameters control the maximum number of positive and negative examples, respectively, that each classifier adds from U' to the labeled set. Each classifier will mark its most confident P/N examples from the unlabeled set U' to be added to L at the end of an iteration. U' will then be replenished by $2P + 2N$ unlabeled training data points in order to remain constant between iterations.

Co-Training Training
Each constituent model is completely re-trained in each iteration on all available labeled training data. Currently, neither model is refined from the previous iteration. Each iteration, we add at most P+N examples selected by each classifier to the labeled training data. We present evaluations below where all classifier examples augment the same pool and where each classifier maintains its own training data, augmented only by the other classifier.

Multi-Modal Classification
Co-Training produces two strong classifiers. Multi-modal classification simply takes the more confident prediction from either the text or the image classifier.

Graph Harmonic Function Model

Graph Construction
We present a graph-based model that finds a harmonic mapping which is characterized by a weighted averaging property. The model builds a kernel that extracts information from both modalities and only considers k nearest neighbors to determine the labels of the unlabeled training points. To test the model, the smoothness assumption is made to find the k labeled nearest neighbors of a test point in the corresponding feature space and a majority vote approach is used to classify them.

In our model, the Gaussian kernel,$w_{ij} = e^{\|x_i - x_j\|/2*\sigma^2}$, is used to obtain the weights. This kernel captures the expected smooth relationship between a point and the points around it. More specifically, w_{ij} is large if the training points x_i and x_j are similar and it decreases radially as the Euclidean distance between x_i and x_j increases. If every point is connected to all the other points, then the graph is said to be a fully connected graph. To simplify the complexity of the model and to capture local structure of the data, only the k nearest neighbors (KNN) are considered so the graph is said to be a KNN graph.

To build the graph, two kernel matrices are obtained, one from tweet texts and one from tweet images (Figure 2). The variance parameters needed to define the Gaussian kernels are estimated by finding the average distance between all the points. The kernel matrix associated with the set of images is constructed using a function that uses the Gaussian kernel to calculate the weights w_{ij} with $i < j$ to obtain an upper triangular matrix. Subsequently, the matrix and its transpose are added and the identity matrix is subtracted from the resulting matrix. The kernel matrix associated with the text modality is obtained using the function rbf_kernel from the python library sklearn. The final kernel is defined as a linear combination of the first two kernels, the two constant values are hyperparameters that act as weights or relative importance put on text or image modality. For example, if the entries of one of the kernels are significantly smaller than the entries of the other kernel, then the parameters can be adjusted to balance out the effects of both sets of weights on the predictions. Since the graph is a KNN graph, only the k largest weights are needed so the rows of the final kernel matrix have only k nonzero entries.

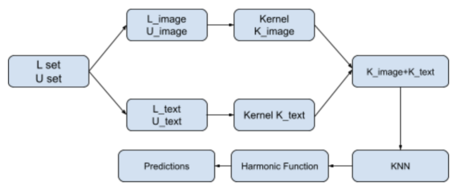

Figure 2: Graph Model Diagram

Harmonic Function
A harmonic function is defined as a function that labels the labeled data points correctly and satisfies the following weighted sum for every unlabeled data point (1). As mentioned above, this function is also expected to be smooth on the graph. The harmonic function updates the labels of the unlabeled data points based on the information obtained from the weights.

$$f(x_i) = y_i \, i = 1, ..., l$$

$$f(x_j) = \frac{\sum_{k=1}^{l+u} w_{jk} f(x_k)}{\sum_{k=1}^{l+u} w_{jk}} \, j = l+1, ..., l+u \tag{1}$$

The values assigned to the unlabeled points are updated until they converge to a value that is between zero and one. Then a threshold is used to produce discrete predictions.

There are other ways to propagate and predict labels. For example, labels can be propagated using a KNN approach without a harmonic function (Zhu and Goldberg, 2009). Also, it is possible to find a closed form of f, but to do that large matrices have to be inverted. Since finding the inverse of a large matrix is computationally expensive and this iterative approach is guaranteed to converge, only this approach is considered.

Hyperparameters
σ_{text} - The average distance between pairs of points from a random subset of the training set and the training set itself is estimated and the result is used to estimate σ_{text}. This approach is taken to reduce the complexity of the model.
σ_{images} - The approach used to estimate σ_{test} is used to estimate σ_{images}.
K - Since the weights reflect how similar two points are and are inversely proportional to the distance between them, the weights associated with the k nearest neighbors of a point are used to update its label.
α - Importance parameter for K_{text}, the kernel that is associated with the training text set. Changing the importance of K_{text} either increases or decreases the effect of the weights from the text of the tweets on the prediction.
β - Importance parameter for K_{image}, the kernel associated with the training image set. Changing the importance of K_{image} either increases or decreases the effect of the corresponding weights on the predictions.
Threshold - [0,1] - The threshold parameter produces discrete label predictions. If a prediction is greater than the threshold, then the corresponding point is labeled as informative and non-informative otherwise.

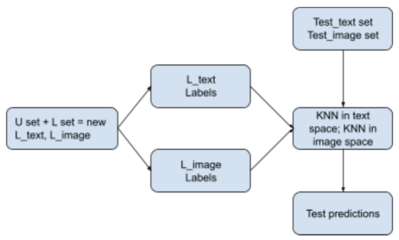

Figure 3: Graph Model Evaluation

To test the model, both modalities have to be used to classify the points from the test set (Figure 4). Since two points that are close to each other are expected to have the same label, the k nearest neighbors of every test point are found in the corresponding feature space by finding the Euclidean distance between them. As a result, two sets of nearest neighbors are obtained so, by adding the number of elements in those sets, two votes are generated. If the average of the two votes is greater than ⌊k/2⌋ then the corresponding test point is labeled as a 1 or informative and 0 or noninformative otherwise.

Results

Scenario	Model	Accuracy	Precision	Recall	F1-Score
Baseline	Text	80.8	81.0	81.0	80.9
Baseline	Image	83.3	83.1	83.3	83.2
Baseline	Text + Image	84.4	84.1	84.0	84.2

Table 1: Baseline results

Scenario	Model	Accuracy	Precision	Recall	F1-Score
Cotraining (P=6, N=6)	Text	72.2	58.1	57.1	57.6
Cotraining (P=6, N=6)	Image	79.2	66.7	68.5	67.6
Cotraining (P=6, N=6)	Text + Image	79.2	66.7	68.5	67.6
Cotraining (P=3, N=6)	Text	71.9	60.6	56.2	58.3
Cotraining (P=3, N=6)	Image	76.5	71.7	61.9	66.5
Cotraining (P=3, N=6)	Text+Image	76.5	71.7	61.9	66.5
Cotraining (I=10)	Text	71.4	47.6	57.1	51.9
Cotraining (I=10)	Image	78.5	68.1	66.5	67.3
Cotraining (I=10)	Text + Image	78.5	68.1	66.5	67.3
Cotraining (I=50)	Text	71.9	63.6	55.9	59.5
Cotraining (I=50)	Image	75.6	73.8	60.1	66.2
Cotraining (I=50)	Text + Image	75.6	73.8	60.1	66.2
Single Tran. Block (Embedding=1024)	Text	67.6	0.0	0.0	0.0
Single Tran. Block (Embedding=1024)	Image	77.5	61.8	66.5	64.1
Single Tran. Block (Embedding=1024)	Text + Image	78.1	60.6	68.3	64.2
Cotraining (Separate Labeled Sets)	Text	75.6	50.0	66.5	57.1
Cotraining (Separate Labeled Sets)	Image	78.9	55.5	73.0	63.0
Cotraining (Separate Labeled Sets)	Text + Image	78.9	55.5	73.0	63.0

Table 2: Cotraining model results.

Scenario	Model	Accuracy	Precision	Recall	F1-Score
Graph Model (k = 3, threshold = 0.5)	Text	58.1	24.6	28.7	26.5
Graph Model (k = 3, threshold = 0.7)	Text	57.4	24.1	28.7	26.2
Graph Model(k=3, threshold=0.4)	Text + Image	60.2	19.1	16.7	17.7

Table 3: Graph-based model results.

Discussion and Analysis

The results of our models and experiments are contained in Tables 1-3. Note that Table 1 has been reproduced from Ofli et al and serves as the baseline, supervised model against which we compare our results. We will begin by evaluating our cotraining results (Table 2) across various scenarios and against the baseline and then consider our graph results (Table 3). While we did not surpass this benchmark, our results provide several insights into the nature of semi-supervised learning on multimodal data that may prove useful for future research.

Beginning with cotraining (Table 2), across all of our experiments, there was not a marked change in the unimodal or multimodal accuracy. However, precision and recall varied greatly in some cases, such as the unimodal text precision after 10 iterations (47.6%) and 50 iterations (63.6%). Since our models relied on random, 10% subsets of the training data to initialize, significant variation across these random samples may introduce volatility into our results. This random sampling approach likely explains some of the variation between our cotraining scenarios.

There are several trends worth noting in regard to the size of the labeled training set. When we increased the contributions (P and N) that each cotraining model made towards the shared labeled dataset from P=N=3 to P=N=6, we noted a slight uptick in the text, image, and multimodal accuracies. This makes intuitive sense because increasing these parameters results in more data being added to the labeled set given a fixed number of iterations. With a larger labeled set, the model has more training data to leverage when making predictions against the test set. The parameters P and N were kept small in these experiments due to computational constraints, but we theorize that our results could be improved upon by setting the cotraining parameters such that training only ends once all unlabeled training points have been labeled. Another means of improving these cotraining results may lie in implementing a knock-out threshold for each iteration, whereby a certain number of low-confidence predictions are removed from the labeled data and added back into the unlabeled data. However, this would increase the number of iterations, and therefore total runtime, needed for our model to label all of the unlabeled training data.

Our results also illustrate some dangers when working with multimodal data. In particular, our cotraining multimodal results largely mirror the predictions from the image component model. This suggests that predictions from our pretrained, VGG16 image model strictly dominate those from the simpler Naive Bayes text model. Encouragingly, when we experimented with a single transformer block model, we found lower text model accuracy, but multimodal accuracy that decoupled from and outperformed the image model. Similar to the results from Ofli et al, this experiment demonstrates that multimodal learning can use insights unique to each modality and outperform the unimodal models. While involving significant orchestration to manage the various labeled, unlabeled, and candidate datasets, cotraining ultimately relies on the strength of its component models. Therefore, additional investment in these text and image models may drive our performance higher.

Turning to the results from our graph model, we see far lower values across all evaluation metrics when compared to the baseline and cotraining results. While our cotraining model relied upon standard, proven models, such as the pretrained VGG16, the graph model was constructed from first principles using a more experimental approach. Therefore, there may be many ways for future research to improve on our semi-supervised, multimodal graph model. Where cotraining relies upon the component models, graph-based models such as the one developed here depend largely on the weights between data points (Balcan and Blum et al, 2005). Our Gaussian kernel, for example, uses the Euclidean distance between two data points. Since our data has very high dimensionality, over 15,000 features due to word tokenization and image pixels, our usage of the Euclidean distance in calculating the Gaussian kernel may be a poor assumption. All else constant, as the number of dimensions increases, data suffers from a phenomenon known as the curse of dimensionality. Determining nearest neighbors in such a high dimensional space may therefore be unreliable, which would impact the integrity of our graph connections. Exploring dimensionality reduction techniques, such as principal component analysis through singular value decomposition, may yield more favorable results.

Since graph-based semi-supervised learning depends critically on the construction and quality of the graph, building graphs related to natural disaster might require expert knowledge in defining similar functions used to assign edges and weights. In our paper we have only experiment with KNN and different values of K. There are other graph construction methods, such as eNN, b-matching (Jebara, Wang and Chang, 2009), to explore local structure of the data. Our multimodal data adds to the complexity of graph construction, for example, what value of relative importance of text modality vs. image modality would yield better results, very likely requires domain knowledge.

References

[1] Avrim Blum, Tom Mitchell, “Combining Labeled and Unlabeled Data with Co-Training”, Proceedings of the eleventh annual conference on Computational learning theory. 1998.

[2] Firoj Alam, Ferda Ofli, Muhammad Imran. “Analysis of Social Media Data using Multimodal Deep Learning for Disaster Response”, International AAAI Conference on Web and Social Media (ICWSM), 2020.

[3] 10701 Introduction to Machine Learning HW 2 Spring 2022 Carnegie Mellon University.

[4] Tomas Mikolov, Kai Chen, Greg S. Corrado, Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. International Conference on Learning Representations. 2013.

[5] `https://forums.fast.ai/t/why-do-we-normalize-with-imagenet-stats-when-fine-`tuning/86411/6

[6] Karen Simonyan, Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Visual Recognition”. ICLR 2015.

[7] Xiaojin Zhu, Andrew Goldberg. “Introduction to Semi-Supervised Learning”. Synthesis Lectures on Artificial Intelligence and Machine Learning, Vol. 3, No. 1 , Pages 1-130 (2009).

[8] Rahul Radhakrishnan Iyer, Jing Chen, Haonan Sun, Keyang Xu. “A Heterogeneous Graphical Model to Understand User-Level Sentiments in Social Media”. Arxiv (2019).

[9] Yoon, H.Y., Park, H.W. “Strategies affecting Twitter-based networking pattern of South Korean politicians: social network analysis and exponential random graph model”. Qual Quant 48, 409–423 (2014).

[10] Ashish Vaswani and Noam Shazeer and Niki Parmar and Jakob Uszkoreit and Llion Jones and Aidan N. Gomez and Lukasz Kaiser and Illia Polosukhin. “Attention Is All You Need”. Arxiv (2017).

[11] Zheng-Jun Zha, Tao Mei, Jingdong Wang, Zengfu Wang and Xian-Sheng Hua, "Graph-based semi-supervised learning with multi-label," 2008 IEEE International Conference on Multimedia and Expo, 2008, pp. 1321-1324, doi: 10.1109/ICME.2008.4607686.

[12] Tony Jebara, Jun Wang and Shih-Fu Chang, “Graph Construction and b-Matching for Semi-Supervised Learning”, International Conference on Machine Learning (2009).