Classification of hierarchical text using geometric deep learning: the case of clinical trials corpus

Sohrab Ferdowsi *[1,2], Nikolay Borissov[3,4], Julien Knafou[1], Poorya Amini[3,4], and Douglas Teodoro[2,1]

[1]*HES-SO, Geneva, Switzerland*
[2]*University of Geneva, Switzerland*
[3]*Risklick AG, Bern, Switzerland*
[4]*Clinical Trials Unit, Bern, Switzerland*

Abstract

We consider the hierarchical representation of documents as graphs and use geometric deep learning to classify them into different categories. While graph neural networks can efficiently handle the variable structure of hierarchical documents using the permutation invariant message passing operations, we show that we can gain extra performance improvements using our proposed selective graph pooling operation that arises from the fact that some parts of the hierarchy are invariable across different documents. We applied our model to classify clinical trial (CT) protocols into completed and terminated categories. We use bag-of-words based, as well as pre-trained transformer-based embeddings to featurize the graph nodes, achieving f1-scores $\simeq 0.85$ on a publicly available large scale CT registry of around 360K protocols. We further demonstrate how the selective pooling can add insights into the CT termination status prediction. We make the source code and dataset splits accessible.

1 Introduction

The safety and efficacy evaluation of medications and clinical interventions is performed using clinical trials (CT's) (Plenge, 2016). Prior to their implementation, CT protocols are carefully designed, detailing important aspects of the study, including the number of enrolled patients, their inclusion and exclusion criteria, and the expected outcome, as required by healthcare authorities (Turner, 2020). Regrettably, a large fraction of CT's terminate before reaching a study conclusion (Fogel, 2018). This is linked directly to delays in providing treatment for the world diseases and to significant excess financial costs (DiMasi et al., 2016).

CT protocols are often modelled and represented using tree-like or more generally graph-like structures, such as XML, JSON and DOM (Benson and Grieve, 2021). These models use a set of nodes $\mathcal{V}$ representing sections connected by a set of relations $\mathcal{E}$ of type *part of* to encode nested information. The information encoded by a given section of a CT is then the recursive combination of the information encoded by its subsections. As an example, Fig. 1 depicts a simplified CT protocol. Without explicit encoding, a flat-structured text feature extractor would ignore these inter-dependencies. To best consider the inter-connected nature of different elements of a CT protocol, it is thus necessary to take its hierarchical structure into account.

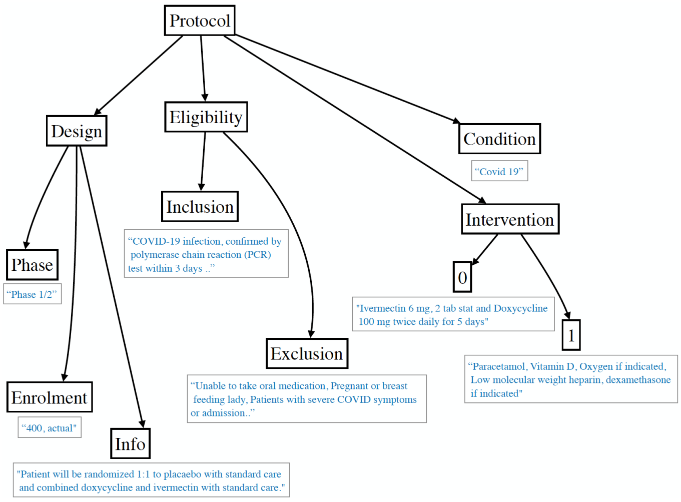

Protocol
Design
Eligibility
Condition
"Covid 19"
Inclusion
Intervention
"COVID-19 infection, confirmed by polymerase chain reaction (PCR) test within 3 days .."
Phase
"Phase 1/2"
0
"Ivermectin 6 mg, 2 tab stat and Doxycycline 100 mg twice daily for 5 days"
1
Exclusion
"Paracetamol, Vitamin D, Oxygen if indicated, Low molecular weight heparin, dexamethasone if indicated"
Enrolment
"Unable to take oral medication, Pregnant or breast feeding lady, Patients with severe COVID symptoms or admission.."
"400, actual"
Info
"Patient will be randomized 1:1 to placebo with standard care and combined doxycycline and ivermectin with standard care."

Figure 1: Schematic view on the simplified tree structure of a typical clinical trial protocol from ClinicalTrials.gov. Leaf nodes contain free text. Top parent nodes are fixed across trees, but children nodes can contain variable structure.

[*] Correspondence: `sohrab.ferdowsi@hesge.ch` and `douglas.teodoro@unige.ch`

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 608–618
November 7–11, 2021. ©2021 Association for Computational Linguistics

The systematic way to consider the structural information of data is the promising paradigm of graph neural networks (GNN's) and more generally geometric deep learning (Bronstein et al., 2017). Geometric deep learning aims to provide a joint representation of different component features, along with their topology. A key element in most GNN models is the *message passing* algorithm between nodes (Gilmer et al., 2017), which aggregates the features of nodes based on their neighborhood connectivity. Focusing on the structure of a CT protocol, which is essentially a tree with free-text on its leaf nodes, the parent nodes initialized with zero vectors will aggregate feature vectors of the children nodes during message passing iterations, as depicted in Fig. 2.

The power of GNN's lies on their ability to aggregate topology and features when node labeling is arbitrary, i.e., when there is no canonical way of labeling the nodes. Famous examples of this use case are in molecular graphs, e.g., (Gilmer et al., 2017), when graph nodes consist of atoms, for which no natural ordering is meaningful. The message passing between nodes, followed by a set pooling operation (typically averaging) after node-level representation learning will then provide a global representation for the whole graph, making them suitable for downstream tasks like regression (Wang et al., 2019). As for the hierarchical text data like CT protocols, however, this is not exactly the case. While the leaf nodes may have arbitrary structure and hence can benefit from general recipes of GNN's, the parent nodes are typically fixed across all graphs and do not have to undergo global pooling. Therefore, as we will show next, a selective pooling that keeps this structure intact is preferred to the permutation invariant pooling.

This paper has the following *contributions*:

- This is the first effort to use GNN's on hierarchical text data with free text. While the works of (Shang et al., 2019b; Choi et al., 2017) use GNN's on one-hot-encoded nodes of medical codes, our node features consist of embeddings of free text, as we discuss in sec. 3. Beyond the example of CT data considered in this paper, this approach can be useful for other hierarchical text data, e.g., like in scientific publications or trademark and patent-related texts.
- For the hierarchical text data with a combination of fixed and variable node structures, we propose "selective pooling" that benefits from the GNN's node-embedding power, as well as the a priori knowledge of the fixed part of the structure.
- We present the first deep learning-based approach to CT termination status prediction. The recent work of (Elkin and Zhu, 2021) uses hand-crafted features and reports AUC $\simeq 0.73$. On a similar experimental setting our results reach AUC $\simeq 0.93$ without feature engineering.

We furthermore provide practical insights and recipes as how to make a bag-of-words (BOW) vectorizer around 50 times faster than a pre-trained BERT model (Devlin et al., 2018) in evaluation run time with a loss of F1-score of classification only around 2%.

The rest of the paper is organized as follows. In section 2, we describe the related works for the different elements to our paper, i.e., the general problem of text classification, graph-based deep learning, the use of graphs in text, as well as the use of machine learning in the study of CT's. Section 3 discusses our proposed solutions in terms of how we featurize the texts and the way we represent hierarchical texts as graphs suitable for graph-based deep learning. Section 4 details how we prepare the CT corpus for classification, the baseline and the proposed methods used and a discussion of the classification results of each of these methods, as well as an effort for explainability of the graph-based representation. The paper is finally concluded in section 5.

2 Related work

2.1 Text classification

An extensively studied problem at the core of many NLP applications is the text classification problem, which assigns categorical labels to textual data. Apart from the classification algorithm, feature representation for text is a crucial step of text classification. Classical approaches represent text using the BOW representation of tokens, which disregard the sequential nature of text and essentially provide histogram-like information of tokens within the corpus. A next generation of methods use word embedding techniques, like the word2vec (Mikolov et al., 2013a,b) and GloVe (Pennington et al., 2014), which furthermore consider the neighborhood relation of words and can benefit from external corpora for training the representations. Another line of work focuses on the sequential structure of tokens within text and uses deep learning architectures like the CNN's, as in (Kim, 2014), or RNN's, as e.g., in (Tai et al., 2015) to capture semantic information.

The state-of-the-art paradigm for a wide range of language understanding tasks, including text classification, is language modeling using transformers (Vaswani et al., 2017). Most notably BERT (Devlin et al., 2018) and its subsequent works, like RoBERTa (Liu et al., 2019) and XLNet (Yang et al., 2019), provide state-of-the-art results using different pre-training strategies based on the transformer architecture. These structures benefit from the self-attention mechanism that both provides better sequence modeling capabilities with longer range focus, as well as parallel processing capabilities to fully exploit GPU and TPU processing capacities. The very high expressive power of these networks makes them capable of benefiting from very large corpora trained on different areas and languages providing valuable domain knowledge to the downstream tasks. However, an important shortcoming of the original transformer structures is their inability to process long texts, due to their quadratic complexity w.r.t. the sequence length. Moreover, they require substantial hardware requirements also at the inference time making them not applicable to certain scenarios.

The issue with quadratic complexity of transformers, however, is being actively studied and a multitude of solutions exist to date, such as the Linear transformer of (Katharopoulos et al., 2020), the Efficient attention (Shen et al., 2021), the Linformer (Wang et al., 2020), the Longformer (Beltagy et al., 2020), or the Reformer (Kitaev et al., 2019), among others. While they provide the promise of linear complexity with satisfactory performance, as tested by benchmarks like in (Tay et al., 2020), on the pragmatic side, there still does not exist many pre-trained models available, especially for particular domains like biomedical.

2.2 Graph neural networks

While most deep learning architectures operate on Euclidean grids with fixed structure, GNN's attempt at the generalization of deep learning concepts to graph structured data using symmetry and invariance.

Consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}; \mathcal{X})$ with node sets $\mathcal{V} = \{v_1, \cdots, v_{|\mathcal{V}|}\}$, a set of edges $\mathcal{E}$ consisting of pairs of nodes (u_i, v_i), which denote the existence of an edge between the two nodes $u_i, v_i \in \mathcal{V}$, as well as a set of features $\mathcal{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_{|\mathcal{V}|}\}$ associated to each of the nodes.

While many applications consider finding useful representations within a graph, the graph classification/regression problems consider finding a global representation $\mathbf{z}_j$ for every given $\mathcal{G}_j \in \{\mathcal{G}_1, \cdots, \mathcal{G}_N\}$. This should incorporate topology information from $\mathcal{E}_j$, as well as feature information $\mathcal{X}_j$. Note that in general, there does not exist a canonical ordering of nodes within a graph, i.e., within the same graph, nodes can be re-labeled without any semantic implications and no one-to-one correspondence necessarily exists between nodes across different graphs.

The standard approach to tackle this permutation ambiguity is the message passing between nodes, as e.g., in (Gilmer et al., 2017), where nodes $v \in \mathcal{N}(u) = \{v \in \mathcal{V} | (v, u) \in \mathcal{E}\}$ in the immediate neighborhood of u send a "message" using an "aggregation" operation on their features, which is then used to "update" the features of $\mathbf{u}$, as:

$$\mathbf{x}_u^{[l+1]} = \mathbb{U}\left[\mathbf{x}_u^{[l]}; \mathbb{A}\left\{\mathbf{x}_v^{[l]}, \forall v \in \mathcal{N}(u)\right\}\right], \qquad (1)$$

where super-scripts $1, \cdots, l, \cdots, L$ refer to the fact that this operation is carried out L times. Starting from $\mathbf{x}^{[1]} = \mathbf{x} \in \mathcal{X}$, $\mathbb{A}\{\cdots\}$ and $\mathbb{U}[\cdot, \cdot]$ are generic differentiable aggregate and update operations, respectively. Famous examples of these operations are the Graph Convolutional Networks (GCN) from (Kipf and Welling, 2016), or the Graph Attention Network (GAT) from (Veličković et al., 2017), among many others.

At the end of L iterations of message passing, each $\mathbf{x}_v^{[L]}, v \in \mathcal{V}$ has aggregated features from its L-hop neighbors, so that both topology and feature information have been taken into account. These aggregated features should then follow a global "pooling" stage $\mathbb{P}_G\{\cdots\}$, where a final representation $\mathbf{z}_{\mathcal{G}_j}$ is derived for the whole graph, as:

$$\mathbf{z}_{\mathcal{G}_j} = \mathbb{P}_G\{\mathbf{x}_v^{[L]}, v \in \mathcal{V}\}, \qquad (2)$$

which is usually taken to be simply an averaging operation. This final representation is then treated as an input feature to a generic classifier, usually a differentiable MLP. The learnable parameters of the GNN, i.e., those from the aggregation and update for each layer, as well as the final MLP are then jointly updated with back-propagation using usual techniques of deep learning on mini-batches of training examples $\{(\mathcal{G}_1, y_1), \cdots, (\mathcal{G}_{|\mathcal{V}|}, y_{|\mathcal{V}|})\}$, with y_j being the label associated to $\mathcal{G}_j$ for the exemplar case of graph classification.

2.2.1 GNN's for text

A new line of work, e.g., (Yao et al., 2019; Zhang and Zhang, 2020; Ding et al., 2020), tries to represent textual data as graphs, where the graphical structure is built from co-occurrence of words, either in a corpus level and hence constructing a very big graph for the whole corpus, or in a document level where a separate graph is constructed for each document. Text classification will then be carried out using GNN's as node classification and graph classification problems, for the first and second cases, respectively. This is fundamentally different from our case, where the graph structure is not constructed from text, but the text itself is structured hierarchically, as in a CT protocol.

Another line of work, as in (Shang et al., 2019b; Choi et al., 2017; Shang et al., 2019a) considers the Electronic Health Records (EHR) data as graphs and uses GNN's to integrate them within healthcare data for solving different tasks. For their case, however, the node features consist of one-hot encoding fixed ontologies, and do not contain free text like in our case.

2.3 Machine learning efforts on CT understanding

There has been few works in the literature reporting data-driven methods to assess the termination status of CT's. The work of (Follett et al., 2019) uses a simple text mining approach to identify keywords associated to CT termination of the ClinicalTrials.gov (CTGov) data and uses random forests to classify the risk of termination. The work of (Geletta et al., 2019) uses Latent Dirichlet Allocation to find risk-relevant topics and uses the topic probabilities for risk classification using random forests.

The recent work of (Elkin and Zhu, 2021) poses the problem as a classification of CT's into "completed" and "terminated" categories. They use feature engineering to feed a set of hand-crafted features into different off-the-shelf classical classifiers. However, even their ensemble methods do not provide satisfactory results. They furthermore perform traditional feature selection and ranking strategies to identify top keywords associated to CT termination.

3 Proposed method

3.1 Text featurization

As a baseline approach to get vectorized representations for free text, we use a BOW-based representation followed by TF-IDF re-weighting, as well as random projections. We also use state-of-the-art pre-trained transformers to improve performance. We next describe these approaches.

3.1.1 Bag-of-words

After standard pre-processing of text (lower-casing, removal of special characters and punctuations, ..), we construct a BOW-based vectorized representation for each protocol, disregarding the hierarchical structures. This is then followed by TF-IDF to re-weight tokens based on their relative importance.

Concretely, for a set of tokens $\mathcal{W} = \{w_1, \cdots, w_i, \cdots, w_{|\mathcal{W}|}\}$, a CT protocol $1 \leq j \leq N$ is represented by $\mathbf{x}_j = [x_{j1}, \cdots, x_{ji}, \cdots, x_{j|\mathcal{W}|}]^T$, where x_{ji} counts the number of occurrences of w_i in the j^{th} protocol. TF-IDF re-weights the i^{th} element of these vectors as

$$\tilde{x}_{ji} = \left(\mathbf{x}_j^T \mathbf{1}_{|\mathcal{W}|}\right) \log\left(\frac{N}{||\mathbf{x}(i)||_0}\right) x_{ji}, \qquad (3)$$

where $\mathbf{1}_{|\mathcal{W}|}$ is an all-ones vector of size $|\mathcal{W}|$, the ℓ_0 norm $||\cdot||_0$ counts the number of non-zero elements of a vector and $\mathbf{x}(i)$ is the i^{th} row of the matrix $\mathrm{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_j, \cdots, \mathbf{x}_N]$.

An important difficulty with BOW-based representations is the dimensionality $|\mathcal{W}|$, which can be as high as even 10^6. Feeding this to a model with learnable parameters has a very high chance of over-fitting, as well as a very high computational complexity for matrix-vector operations.

BOW-based representations, however, benefit from very high degrees of sparsity. A classical result from the domain of compressive sensing (Candes and Tao, 2005) suggests that a high dimensional sparse vector $\tilde{\mathbf{x}}$ can be projected to lower dimensions using a random matrix $\mathrm{A} \in \Re^{d \times |\mathcal{W}|}$ as $\hat{\mathbf{x}} = \mathrm{A}\tilde{\mathbf{x}}$, virtually without any loss of information. Provided that the sparsity is high enough, one can aim for $d \ll |\mathcal{W}|$. Furthermore, it has been shown (Li et al., 2006) that the random projection matrix itself can be chosen to also be sparse. This is very beneficial in practice, since both $\tilde{\mathbf{x}}$ and A can be stored and multiplied in sparse matrix format, e.g. using numerical packages like SciPy (Virtanen et al., 2020).

3.1.2 Pre-trained language models

A major drawback of the BOW-based representations is that they totally disregard context and the sequential nature of text, since they only provide a histogram-based statistic of token counts. As discussed earlier in sec. 2.1, the state-of-the-art solution to language modeling is based on the transformer architecture (Vaswani et al., 2017), most notably as in (Devlin et al., 2018), for which a large number of models pre-trained on very large-scale corpora exist.

While one gets better performance by further fine-tuning transformers on the downstream task at hand, the computational requirements, most notably their GPU memory consumption, makes the fine-tuning step very expensive for certain tasks. In the case of CT protocols, there is usually more than 100 nodes for each CT, making this step particularly difficult. We therefore suffice only with fixed embeddings from pre-trained models.

To embed a piece of text using transformers into a vectorial representation, we use mean-pooling that considers the attention mask for each token into account, as suggested, e.g., in (Reimers and Gurevych, 2019).

3.2 Graph representation of hierarchical text

Hierarchical text usually comes with a tree structure, where free text appears in the leaf nodes. Compared to the setup of sec. 2.2 for general graphs, the difference that this brings to the message passing is that the neighborhood $\mathcal{N}(u)$ of node u reduces simply to the set of its children nodes $\mathcal{C}(u)$. Furthermore, the non-leaf nodes $\{u \in \mathcal{V} | \mathcal{C}(u) \neq \emptyset\}$ will be initialized with zero features, and will aggregate features from their children during iterations. Fig. 2 summarizes the graph representation steps.

3.2.1 Selective pooling

Since the general structure of CT protocols is invariable across different CT examples, one can do better than the general pooling strategy of Eq. 2.

Consider a set of nodes $\bar{\mathcal{V}} \in \mathcal{V}$, for which the enumeration is preserved across all $\mathcal{G}_1, \cdots, \mathcal{G}_N$. To benefit from this invariance, one can consider a simple "selective pooling" as

$$\mathbb{P}_S\big[\mathbf{x}_v^{[L]}, v \in \bar{\mathcal{V}}\big] = \Big\|_{v \in \bar{\mathcal{V}}} \mathbf{x}_v^{[L]}, \tag{4}$$

where $\|$ denotes the concatenation of vectors. To also benefit from the global pooling of GNN's, one can consider the final graph representation as a simple concatenation of global and selective pooling as:

$$\mathbf{z}_{\mathcal{G}_j} = \mathbb{P}_G\big\{\mathbf{x}_v^{[L]}, v \in \mathcal{V}\big\} \Big\| \mathbb{P}_S\big[\mathbf{x}_v^{[L]}, v \in \bar{\mathcal{V}}\big]. \tag{5}$$

Note that while $\mathbb{P}_G\{\cdots\}$ is a set operation, $\mathbb{P}_S[\cdots]$ is essentially a list operation and hence the order of the elements should be kept uniform across all graphs.

4 Experiments[1]

4.1 Data preparation

Healthcare authorities of different countries have different requirements for the publication of the CT protocols. There are 17 CT registries as identified by the WHO[2], where the largest and most complete one is that of *ClinicalTrials.gov* (CTGov). This repository is publicly available to download[3] and is the one we use in this paper. The CTGov corpus is updated daily, and the snapshot used in our experiments (downloaded on 10th of December 2020) contains 360,497 CT protocols. Similar to the work of (Elkin and Zhu, 2021), we include only interventional studies (i.e., we exclude observational studies as they have a different nature). Furthermore, we exclude the studies with *recruiting* status (as their outcome is not clear yet), and only consider protocols whose overall status is either *completed* (74% of our subset), which was used to assign the *"completed"* class, or *terminated* (9%), *withdrawn* (5%) and *suspended* (1%), which are grouped collectively into the *"terminated"* class, as these 3 categories have similar risk outcome in practice. This resulting set contains 188,915 protocols, which we split into train, validation and test sets with ratios of 70%, 15% and 15%, respectively.[4]

While numerous criteria can be considered to judge a CT as risky (e.g., whether they achieved

[1] Source code is available at https://github.com/sssohrab/ct-classification-graphs.

[2] https://www.who.int/clinical-trials-registry-platform/network

[3] https://ClinicalTrials.gov/AllAPIJSON.zip

[4] The database, as well as exact splits are available at *zenodo* (Ferdowsi et al., 2021).

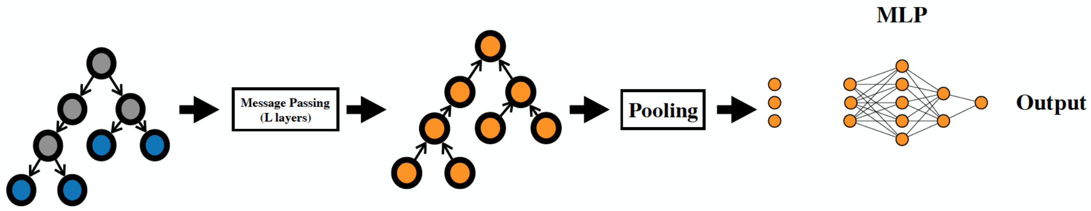


Figure 2: A hierarchical structure with extracted feature vectors from free text in leaf nodes is passed to a generic message passing algorithm. The resulting graph with aggregated features is pooled and passed to an MLP to produce the predicted class label.

FDA approval, whether they had reported safety issues, ..), as a basic label-assignment strategy, we consider the "*completed*" CT's as low-risk, and the "*terminated*" CT's as risky ones. This leaves us with a proportion of $\simeq (74\%, 26\%)$ for low-risk and high risk classes, respectively.

4.2 Baseline methods

We report the classification results based on the following methods.

The work of (Elkin and Zhu, 2021) described earlier uses a snapshot of the CTGov collection that is downloaded in 2019, to which we do not have access. Furthermore, their labeling strategy slightly differs from ours, as they only consider "completed" and "terminated" status. Nevertheless, these are very minor differences and the results are still comparable. They report F1-scores $\simeq 0.33$, but only considering the positive class, hence no micro-macro weighting to be included in table 1.

Fast-text (Joulin et al., 2017) is a very efficient library for text classification providing a strong baseline. We used the same pre-processing steps as our BOW-based methods (lower-casing, removal of punctuations and special characters) for tokenization. We trained the model and used the auto-tuning functionality on the validation split using the standard hyper-parameter sets.

BOW-W denotes the standard bag-of-words using a token size of $|\mathcal{W}|$ followed by TF-IDF re-weighting as described earlier in sec. 3.1. Note that if the resultant vector representation is fed directly to a classifier with learned parameters, the chance of over-fitting increases with $|\mathcal{W}|$, forcing to chose small vocabulary sizes.

BOW-W-RPd addresses this issue with Random Projections, as described in sec. 3.1. As motivated earlier, along with the sparsity of the BOW representations, the projection matrix itself can furthermore be chosen to be highly sparse. In our experiments, we chose $|\mathcal{W}| = 500,000$, $d = 768$ (to be comparable with transformers), and for each row, we set a sparsity of 0.01 using magnitude thresholding, i.e., only $5,000$ non-zero elements, followed by normalization to unit-norm. This allowed us to significantly speedup the calculations (as well as memory), hence not suffering from the very slow run-time of packages like Gensim (`models.rpmodel`) (Rehurek and Sojka, 2011) with our simple SciPy sparse package. As an example, to vectorize 1000 CT protocols, it takes around 7 sec, which is roughly 50 times faster than encoding them with a BERT model (in evaluation mode) on a GPU.

PubMedBERT-pretrain-768 refers to the base-BERT pre-trained on PubMed abstracts and PubMedCentral full-texts as introduced in (Gu et al., 2020). We do not fine-tune the weights on our classification task and use the model in evaluation mode only. We use the interface provided by the Transformer's library (Wolf et al., 2020), and use the PyTorch framework (Paszke et al., 2019) in all our experiments. After the embedding vectors are calculated, we use the exact same network as the BOW counterpart (but with different hyper-parameter sets).

Flat-1 refers to the case where we take the tree structure of the CT protocol and simply flatten it into 1 field. We then vectorize this field using the above methods and feed it to a classifier network. As for the classifier, we use a 3-layer MLP with a low-rank decomposition of the first linear layer, along with the standard deep learning recipes (batch-normalization, dropout, ..) and use the Adam optimizer with standard hyper-parameters. At each mini-batch, we re-weight the importance of each CT sample to the cross-entropy loss based on the class priors, such that the classes become virtually balanced. We keep this loss function (weighted-BCE) the same across all experiments.

Flat-9 summarizes each CT into 9 vectors, each corresponding to one major parent node of the tree. This is to avoid shrinking all information into one vector and keeps some of the original tree structure. The 9 chosen fields are "sponsor-collaborator", "oversight", "description", "condition", "design", "arms-intervention", "outcomes", "eligibility" and "contacts-location" modules of CT-Gov protocols. When non-existing in some protocols, we assign them all-zero vectors. We feed the resulting 9-channel input tensor to a network with a shared initial small MLP head that independently processes each channel and then concatenates the results and passes it through another small MLP. Except for the concatenation part, this is essentially equivalent to the network used in flat-1.

GCN-global uses 3 layers of the standard graph convolutional block of (Kipf and Welling, 2016) with hidden dimension of 200 and a global average pooling as in Eq. 2. This is then followed by a standard MLP to produce the final output. We use the GCN implementation as provided by the PyTorch-Geometric framework (Fey and Lenssen, 2019) and use the data-loading functionalities therein to handle our GNN experiments.

GCN-selective-9 uses the selective pooling that we introduced in Eqs. 4 and 5. To keep the dimensionalities comparable with the global pooling, we chose the output dimension of the third GCN as 20. When the 9 fields, plus the global pooling are concatenated, this will amount to 200, same as in the GCN-global above. In order to see the effect of graph-based modeling, these 9 fields are chosen to be the same as in the Flat-9 method above.

4.3 Classification results

Table 1 summarizes the classification results on the test set of our collection based on the standard precision, recall, F1-score macro, F1-score micro, as well as the area under the ROC curve metrics.

The following observations can be made from the classification results:

- We notice that taking the hierarchical structure of the CT protocols into account is crucial for classification. The flat-9 models significantly outperform those of flat-1.

- Increasing the vocabulary size of BOW tokens significantly improves performance. The random projections, as well as the sparsification of the projector matrix are very effective tricks to make this practical.

- The use of transformer-based embeddings invariably improves performance w.r.t. the BOW. This, however, comes at the price of slower run-times, around 50 times slower than the BOW counterpart starting from raw text to the embedding.

- GNN-based modeling of CT protocols provides a net increase of performance w.r.t. the flat structures. As a very straightforward example for comparison, because of the linear nature of BOW (disregarding the TF-IDF), the embedded features of each of the fields of the BOW-flat-9 are the summation of their corresponding children nodes of BOW-GCN-selective-9 before starting the message passing. During the algorithm's iterations, the message passing takes the hierarchy into account and pools the information much more effectively than a simple summation, resulting into a superior final performance.

- The selective pooling proposed in this paper, which incorporates knowledge of the document structure, as well as the message passing of GNN's over the leaf nodes increases performance w.r.t. the global pooling, which benefits only from the latter. This is particularly useful for explainability analyses as we will see next.

4.4 Explainability

While attracting a lot of recent attention from deep learning research communities, the explainability of graph neural network models are less explored and hence less developed compared to the grid-structured data like flat text or images (Yuan et al., 2020). This is in part due to the lack of locality information which arises from the inherent permutation ambiguity of nodes that we discussed earlier. So the extension of the explainability techniques developed for grid data is not trivial, e.g., due to the non-differentiable nature of the graph adjacency matrix.

As a useful workaround to bypass these issues, the selective pooling proposed in this paper can readily use basic gradient-based techniques used in other domains. As an example, we investigate the norm of the gradient of the selectively-pooled nodes w.r.t. the output, i.e.:

$$\alpha_v = \Big|\Big| \frac{\partial y^c}{\partial \mathbf{x}_v^{[L]}} \Big|\Big|_2, \quad v \in \bar{\mathcal{V}} \qquad (6)$$

Table 1: Performance comparison for models described in sec. 4.2.

Method	Precision	Recall	F1-score		AUC
			macro	micro	
(Elkin and Zhu, 2021)	-	-	-	-	0.7281
Fast-text (Joulin et al., 2017)	0.8489	0.7205	0.7531	0.8402	0.8456
BOW-500000-RP768-flat-1	0.6145	0.6300	0.6146	0.6453	0.7034
PubMedBERT-pretrain-768-flat-1	0.6512	0.6763	0.6260	0.6346	0.7246
BOW-1000-flat-9	0.6489	0.6713	0.6488	0.6346	0.7369
BOW-500000-RP768-flat-9	0.7572	0.7793	0.7652	0.7906	0.8701
PubMedBERT-pretrain-768-flat-9	0.8144	0.8144	0.8144	0.8419	0.8911
BOW-500000-RP768-GCN-global	0.8185	0.8233	0.8208	0.8462	0.9116
PubMedBERT-pretrain-768-GCN-global	0.8426	0.8503	0.8463	0.8675	0.8881
BOW-500000-RP768-GCN-selective-9	0.8419	0.8337	0.8376	0.8632	0.9082
PubMedBERT-pretrain-768-GCN-selective-9	0.8454	0.8519	0.8485	0.8697	0.9267

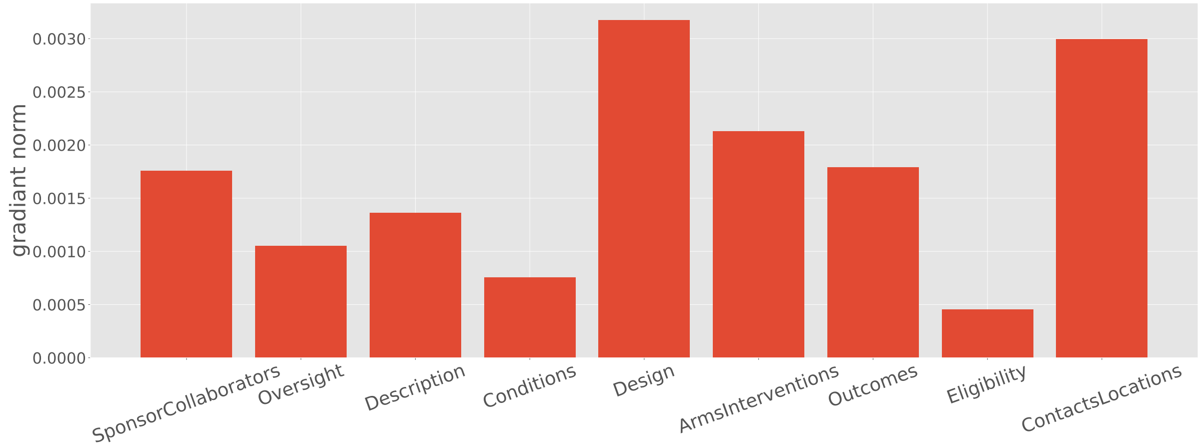


Figure 3: Gradient norms of 9 CT protocol fields w.r.t. the class outputs averaged over 1,000 not-completed CT's.

Fig. 3 sketches the average values for 1,000 CT's classified as high-risk by the *BOW-500000-RP768-GCN-selective-9* model above.

Fig. 4 shows the t-SNE (Van der Maaten and Hinton, 2008) visualizations of the same 9 fields ($\mathbf{x}_v^{[L]} \in \Re^{20}$) for the two classes.

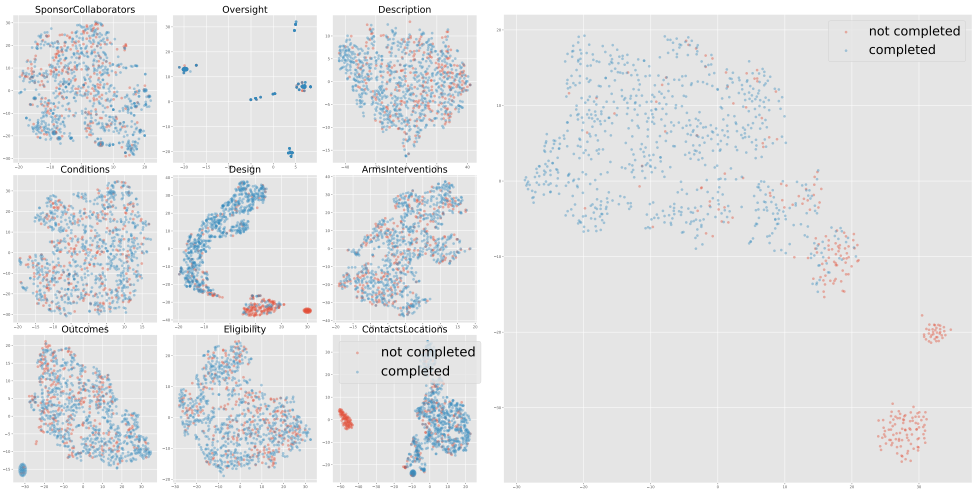
The two above figures confirm the source of risk in CT protocols to be the "Design" and "ContactsLocation" fields. This is in accordance with studies like (Fogel, 2018), which identify the main sources of CT failure as lack of recruitment, that appears under the Design module of the protocol, and problems related to funding, which can be associated with the locations in which the trials are carried out.

5 Conclusions

We used Graph Neural Networks to represent the structure, as well as the extracted features from free text within hierarchical documents. Our use case was the classification of interventional clinical trials into two different risk categories based on their protocols. On a publicly available corpus of around 360K protocols, we showed that the use of GNN's provides a net increase in performance, compared to structure-agnostic baselines. To better incorporate the power of GNN's into the invariable a priori known template, we proposed selective pooling to boost the performance of global pooling. Furthermore, we showed that this approach provides straightforward solutions for explainability, where we demonstrated some consistency between gradient activities of protocol fields within our model to known factors of risk from clinical trials research literature.

Acknowledgments

This work is funded by the Swiss Innovation Agency, Innosuisse under the project with funding number 41013.1 IP-ICT.



(a) Per-field

(b) Global

Figure 4: Visualization using t-SNE of the (a) selective pooling of the 9 fields (b) the layer after pooling. The selective pooling reveals the fields usually associated to risk.

References

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Tim Benson and Grahame Grieve. 2021. Uml, xml and json. In *Principles of Health Interoperability*, pages 399–426. Springer.

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42.

Emmanuel J Candes and Terence Tao. 2005. Decoding by linear programming. *IEEE transactions on information theory*, 51(12):4203–4215.

Edward Choi, Mohammad Taha Bahadori, Le Song, Walter F Stewart, and Jimeng Sun. 2017. Gram: graph-based attention model for healthcare representation learning. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 787–795.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Joseph A DiMasi, Henry G Grabowski, and Ronald W Hansen. 2016. Innovation in the pharmaceutical industry: new estimates of r&d costs. *Journal of health economics*, 47:20–33.

Kaize Ding, Jianling Wang, Jundong Li, Dingcheng Li, and Huan Liu. 2020. Be more with less: Hypergraph attention networks for inductive text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4927–4936.

Magdalyn E Elkin and Xingquan Zhu. 2021. Predictive modeling of clinical trial terminations using feature engineering and embedding learning. *Scientific reports*, 11(1):1–12.

Sohrab Ferdowsi, Nikolay Borissov, Julien Knafou, Poorya Amini, and Douglas Teodoro. 2021. Classification of hierarchical text using geometric deep learning: the case of clinical trials corpus.

Matthias Fey and Jan E. Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

David B Fogel. 2018. Factors associated with clinical trials that fail and opportunities for improving the likelihood of success: a review. *Contemporary clinical trials communications*, 11:156–164.

Lendie Follett, Simon Geletta, and Marcia Laugerman. 2019. Quantifying risk associated with clinical trial termination: a text mining approach. *Information Processing & Management*, 56(3):516–525.

S Geletta, L Follett, and MR Laugerman. 2019. Latent dirichlet allocation in predicting clinical trial failures.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pages 1263–1272. PMLR.

Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2020. Domain-specific language model pretraining for biomedical natural language processing.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics.

A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2019. Reformer: The efficient transformer. In *International Conference on Learning Representations*.

Ping Li, Trevor J Hastie, and Kenneth W Church. 2006. Very sparse random projections. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 287–296.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Robert M Plenge. 2016. Disciplined approach to drug discovery and early development. *Science translational medicine*, 8(349):349ps15–349ps15.

Radim Rehurek and Petr Sojka. 2011. Gensim–python framework for vector space modelling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, 3(2).

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3973–3983.

Junyuan Shang, Tengfei Ma, Cao Xiao, and Jimeng Sun. 2019a. Pre-training of graph augmented transformers for medication recommendation. *arXiv preprint arXiv:1906.00346*.

Junyuan Shang, Cao Xiao, Tengfei Ma, Hongyan Li, and Jimeng Sun. 2019b. Gamenet: Graph augmented memory networks for recommending medication combination. In *proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1126–1133.

Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. 2021. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3531–3539.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. 2020. Long range arena: A benchmark for efficient transformers. In *International Conference on Learning Representations*.

J Rick Turner. 2020. New fda guidance on general clinical trial conduct in the era of covid-19. *Therapeutic Innovation & Regulatory Science*, 54:723–724.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

Xiaofeng Wang, Zhen Li, Mingjian Jiang, Shuang Wang, Shugang Zhang, and Zhiqiang Wei. 2019. Molecule property prediction based on spatial graph embedding. *Journal of Chemical Information and Modeling*, 59(9):3817–3828. PMID: 31438677.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7370–7377.

Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. 2020. Explainability in graph neural networks: A taxonomic survey. *arXiv preprint arXiv:2012.15445*.

Haopeng Zhang and Jiawei Zhang. 2020. Text graph transformer for document classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8322–8327.